Efficient Broadcasting Procedures for Constrained Reconfigurable Meshes

O. Diessel *H. ElGindy* *L. Wetherall*

Department of Computer Science and Software Engineering
The University of Newcastle
Callaghan NSW 2308
Australia

{*odiessel,hossam,lwether*}*@cs.newcastle.edu.au*

Abstract

Broadcast operations on reconfigurable meshes that only use column or row buses of known lengths are easily simulated by constrained reconfigurable meshes that restrict the lengths of bus components to a practical number of bus segments in a single cycle. Frequently, however, reconfigurable mesh algorithms make use of arbitrarily shaped and sized non-branching buses. This paper presents an optimal single source and an efficient multiple source broadcasting procedure for arbitrary linear buses under the constrained reconfigurable mesh model.

1 Introduction

Massively parallel architectures with reconfigurable communication networks have recently been proposed, investigated and prototyped [1, 8]. A common feature of such architectures is the ability of their processors to set local switches such that interprocessor wires are connected to form communication buses. A packet of data can then be broadcast in constant time on a bus component independent of its size or length. Investigation of bus delays has indicated that the broadcast delay is small, but that it cannot be correctly modelled by a constant that is independent of the bus size [5].

Beresford-Smith et al [2] proposed the *k-constrained reconfigurable mesh* to model propagation delays more realistically. In this model, propagation delay on a segment of the bus that connects two neighbouring processors is modelled by a constant and only algorithms that configure the bus system into components with sizes bounded by k segments are permitted. Beresford-Smith et al [2], and more recently ElGindy [3], presented algorithms for sorting, planar convex hull and sparse matrix multiplication for k-constrained reconfigurable meshes. The nature of the sorting and the convex hull problems allowed for the design of algorithms that need to broadcast on horizontal or vertical bus components of length k only [2]. *River broadcasting* was proposed in [3] to allow for the efficient implementation of multiple broadcasts on arbitrarily long linear bus components. However, each component was restricted to form either a vertical or a horizontal bus.

In this paper we address the problem of performing broadcast operations on linear bus components of arbitrary shape and length, which is a reoccurring operation in algorithms designed for the reconfigurable mesh architecture [6]. A detailed description of the k-constrained reconfigurable mesh model and permitted broadcast operations is given in the following section. In section 3, we present a procedure for broadcasting data from a single source on linear bus components. We discuss the implementation of an efficient procedure for performing multiple broadcasts on linear bus components in section 4, and conclude with general remarks and future research directions.

2 The Model

A reconfigurable mesh of size $m \times n$, consists of m rows and n columns of processing elements arranged in a grid, as in Figure 1. Each processor is connected to its immediate neighbours to the north, south, east and west, when present, and has four similarly labeled I/O ports through which it can communicate with its neighbours.

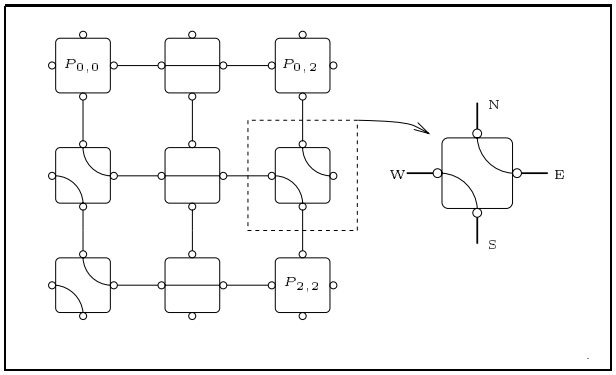

Figure 1: A reconfigurable mesh of size 3×3.

Each PE has control over a local set of short–circuit switches that allow the four I/O ports to be connected together in any combination. The 15

possible connection configurations are depicted in Figure 2.

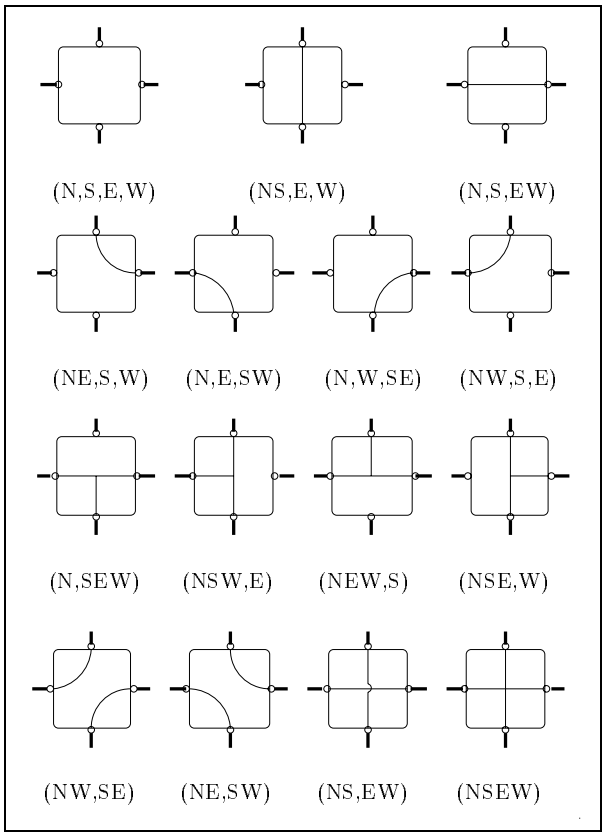

Figure 2: Reconfigurable mesh connection configurations.

The processors operate synchronously, in one machine cycle performing an arithmetic, logic or control operation, setting a connection configuration, and sending (receiving) a datum to (from) each I/O port. Each processor possesses a constant number of $\Omega(\log mn)$–bit word registers allowing it to identify itself. Processors are numbered from $P_{0,0}$ in the north–western corner, to $P_{m-1,n-1}$ in the south–eastern corner.

When a connection is set, signals received by a port are simultaneously available to any port connected to it. For example, if processors connect their northern and southern I/O ports by closing the appropriate switches as in the configuration (NS, E, W), data "broadcast" onto the "column bus" can be read by all of the processors in a column. The model allows concurrent reading from a bus, requires exclusive writing to a bus, and usually assumes a constant time communication delay on arbitrarily long connected bus components. Unfortunately, this constant propagation delay model is infeasible for a number of reasons. Due to the finite resistance and capacitance per unit wire length, signals need to be regenerated to ensure accurate detection, and the time to broadcast a signal along the wire is proportional to the square of its length. The speed of light and the clock frequency of the machine also limit the number of processors that can be reached by a signal in one cycle. To account for these limits, Beresford-Smith et al [2] modelled the propagation delay on a bus segment connecting adjoining processors as a constant, and proposed a *k–constrained* reconfigurable mesh model that limits the length of connected bus components to at most k bus segments.

A reconfigurable mesh algorithm that uses buses of arbitrary sizes can be simulated on the k–constrained model, where signals propagate at least k bus segments in one machine cycle. In the following sections we present protocols for the efficient simulation of both single and multiple source broadcasts on arbitrarily shaped linear buses.

Throughout this paper we make the following assumptions:

1. Signals are regenerated at every processor and are guaranteed to travel at least k bus segments in a clock cycle, where $k \geq 2$.

2. Each node may, within one cycle, communicate with its immediate neighbours in the bus component to exchange the values that arrived on their bus ports in order to determine whether or not they received a signal, with or without corruption as a result of conflict.

3. If multiple nodes write the same value to a bus component, then it will be received without error by all nodes connected by the component.

4. A seperate network is used for the distribution of instructions to the processing elements. Such control networks are common in existing massively parallel machines such as the MasPar [7] and CM-5 [4].

3 Single source broadcast

A re-occuring sequence of operations in reconfigurable mesh algorithms is to partition the bus system of the reconfigurable mesh by forming disjoint bus components and to broadcast a data item or a control message in each component. In this section we sketch an efficient procedure for implementing the broadcast operation in algorithms that permit exactly one processor, called *speaker*, to write to each component. Our procedure runs in $O(\frac{l}{k} + \frac{n}{k})$ cycles, where l is the length of the longest formed bus component and the size of the mesh is $n \times n$. This is the best possible in the worst case.

Our procedure for simulating a broadcast operation on a k-constrained reconfigurable mesh proceeds in three stages. The first is an initialisation stage during which the two terminal processors in each bus component are identified, and non-terminal processors set their switches to form a linear bus. The second stage performs the broadcast

operation on the individul bus components, and terminates with the arrival of the message from the speaker at the terminal processors. This stage proceeds in $\frac{l}{k}$ steps. During each step a message travels at least k bus segments. At each step, each processor communicates with its neighbours to identify the last processor, called *repeater*, to receive the message, and the repeater re-transmits the message. In the third, and final, stage terminal processors cooperate in establishing the completion of the broadcasting operation. This is performed through a two level process that confirms the arrival of messages at terminal processors in each row and then collects the confirmation in the left-most column of the k-constrained reconfigurable mesh. The idea is that the rightmost processor in each row attempts to broadcast a message. Non-terminal nodes set their switches to (N, S, EW). Terminal nodes initially set their switches to (N, S, E, W), and then change to (N, S, EW) upon arrival of the message from the speaker in their own bus component. A similar procedure is performed on the left-most column with each processor acting as a terminal processor, and processor $P_{n,0}$ writing a special control message, TERMINATE, on its northern port. The broadcast operation is considered complete upon the arrival of the TERMINATE message at processor $P_{0,0}$.

The first stage requires a constant number of steps, while the second stage requires $\frac{l}{k}$ steps, and the third stage requires $\frac{n}{k}$ steps to complete. We schedule the individual steps in the second and third stages to proceed in alternating cycles. This is essential since the length of the longest bus component, l, is not known in advance. Therefore the total running time of the procedure is $O(\frac{l}{k} + \frac{n}{k})$ steps in the worst case as previously stated.

4 Multiple source broadcasts

In this section we sketch an efficient procedure for implementing the broadcast operation when an arbitrary number of processors, including none, are permitted to write to each bus component. The main idea is to employ the *river-broadcasting* procedure introduced in [3] to broadcast a number of messages, originating from different processors, on a linear bus in a chosen direction. The algorithm runs in time $O(m + \frac{l}{k} + \log k + \frac{n}{k})$, where m is the number of speakers, l is the length of the longest bus component, and the size of the mesh is $n \times n$. Broadcasting the messages to all processors on the linear bus can then be accomplished through two river broadcasting operations.

For the sparse matrix multiplication problem in [3], choosing a direction to perform the river broadcast was simple. The remainder of this section is devoted to the description of a general procedure to assign a direction to the various bus components. Our solution for each bus component proceeds as follows:

1. Each of the two terminal nodes writes a unique message on the bus. The nodes ensure uniqueness by incorporating their own label in the message. As in the second stage of the single source broadcast procedure, the two messages travel at least k bus segments in each cycle and each node communicates with its neighbours to check the status of the message. A conflict is identified when two adjacent nodes, say p and q, detect that one of them has received a corrupt value.

2. When a conflict is detected, the node with the larger label q becomes idle. The node with the smaller label p breaks the bus and transmits both messages to both sides. The transmission in both directions is necessary since p cannot identify the source of the message that caused the conflict to occur. The signal on one side will reach the terminal node and will be ignored, and the other will result in a conflict.

3. Both processors, now recognizing the source of the conflict, send a special control message CONFLICT that notifies all nodes between them to start a conflict resolution process. This propagation will be completed in one cycle.

4. The objective of the conflict resolution process is to get the two nodes causing the conflict to exchange their messages. The message with the smaller label is then allowed to proceed to a terminal node on the bus component, and the other is discarded, thus setting an orientation for the bus component. Processors reach the above objective in a recursive fashion as follows:

   - initialize all nodes as active.
   - an active node whose label is neither locally minimal nor locally maximal becomes inactive and connects its ports to short circuit the bus component locally.
   - an active node whose label is a local maximum exchanges the labels of the two adjacent local minima, then becomes inactive and connects its ports to short circuit the bus component locally.

   The process continues until the two nodes causing the conflict exchange their labels.

   Note that the above broadcast operations are feasible since the two nodes that recognized the conflict are separated by less than k bus segments.

The above procedure runs in $O(\frac{l}{k}+\log k)$ steps in each bus component, and thus in $O(\frac{l}{k}+\log k+\frac{n}{k})$ steps for the whole k-constrained reconfigurable mesh, where the $\frac{n}{k}$ term represents the time needed to establish the termination of the bus orientation process. Therefore we conclude that multiple broadcasting can be completed in $O(m+\frac{l}{k}+\log k+\frac{n}{k})$ steps, where m is the largest number of speakers in any bus component.

5 Concluding Remarks

In section 3, we presented an efficient algorithm to simulate the broadcasting by one speaker in each linear bus component on a k-constrained reconfigurable mesh. The algorithm completes the broadcast operation in $O(\frac{l}{k}+\frac{n}{k})$ cycles, which is optimal in the worst case. The algorithm is easy to modify for tree-shaped bus components. However, the time for simulating a broadcast can not easily be established in arbirtary tree-shaped buses.

The next stage in our investigation is to develop a signal propagation model for tree-shaped buses in k-constrained reconfigurable meshes, and to study the complexity of identifying the best schedule to broadcast a message in such bus components. Performing multiple broadcasts on a tree-shaped bus component is another problem to be investigated.

Acknowledgements

This research is supported by a grant from the Australian Research Council.

References

[1] H.H. Alnuweiri, M. Alimuddin, and H. Aljunaidi, "Switch models and reconfigurable networks: Tutorial and partial survey," *Proceedings 1st Workshop on Reconfigurable Architectures*, Cancun, Mexico, April 1994.

[2] B. Beresford-Smith, O. Diessel, and H. ElGindy, "Optimal algorithms for constrained meshes," submitted to *Journal of Parallel and Distributed Computing*. A complete version appeared in *Proceedings Australian Computer Science Conference*, Adelaide, Australia, February 1995.

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

[4] C.E. Leiserson at al, "The network architecture of the connection machine CM-5," *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, San Diego, CA, 1992, pp. 272 – 285.

[5] M.Maresca and H.Li, "Connection autonomy in SIMD computers: A VLSI implementation," *Journal of Parallel and Distributed Computing*, **7:2**, October 1989, pp. 302 – 320.

[6] R. Miller et al, "Parallel computations on reconfigurable meshes," *IEEE Transactions on Computers* **42:6**, June 1993, pp. 678 – 692.

[7] J.R. Nickolls, "The design of the MasPar MP-1: A cost effective massively parallel computer," in *IEEE Digest of Papers*, IEEE Computer Society Press, 1990, pp. 25 – 28.

[8] R.K. Thiruchelvan, J.L. Trahan and R. Vaidyanathan, "On the power of segmentation and fusing buses," *Proceedings 7th International Parallel Processing Symposium*, Newport Beach, California, April 1993, pp. 79 – 83.