Using a Tiled Architecture to Process Data from High-Bandwidth, Optical Interfaces

Justin Teller and Füsün Özgüner

Department of Electrical and
Computer Engineering

The Ohio State University

2015 Neil Avenue

Columbus, Ohio 43210-1272, USA

Email: {tellerj}{ozguner}@ece.osu.edu

Robert Ewing

US Air Force Research Laboratory

Wright-Patterson Air Force Base

Dayton, Ohio 45433, USA

Email: Robert.Ewing@wpafb.af.mil

Abstract—In this paper, we propose a *physical process* inspired routing that routes data from a high-bandwidth data port to multiple processing tiles. Our routing is named Magnetic Based Routing (MBR); data is “attracted” towards the processing tiles that need the data and is “repulsed” away from other data flows.

Optical data interconnects are beginning to become more important in processing technology. However, processing the large data bandwidths from optical interconnects can be difficult, especially if there exist real-time requirements. An explicitly parallel, tiled architecture [3], [4], such as the TRIPS processor [1], [2], [5], is a promising architecture to process large amounts of data in a streaming manner. However, the raw processing power of these next-generation architectures is difficult to realize when coupled with large bandwidth data-ports. We propose Magnetic Based Routing (MBR) to route data through a tiled processing architecture. MBR models magnetic attraction and repulsion to route data through an on-chip network. An overview of how multiple data-flows are routed using MBR is shown in Figure 1. In Figure 1, a large bandwidth data port (possibly driven by an optical interconnect) generates data-flows whose destinations are data processing tasks in different locations on the chip. Our routing has four strengths: first, different data-flows repel each other, reducing the contention on the on-chip network, and MBR allows a tiled architecture to take advantage of its inherent parallelism when processing. Additionally, MBR is a mostly distributed routing algorithm which makes it scalable as we increase the number of tiles in the architecture. Finally, with only small additions, the MBR is robust, and is able to quickly recover from multiple hardware faults.

In this paper, we assume that the processing tiles are connected with a nearest-neighbor mesh network, as in the TRIPS architecture [2], [1]. MBR is implemented based on two concepts based on the magnetic physical process: “attraction” and “repulsion.” A data-flow’s sink initiates the attraction for the particular data-flow, and the information of this attraction is propagated through the network to all processing tiles. Each data-flow registers an attraction value with each processing

tile, where the attraction value is calculated by following an inverse square of the tile’s distance from the data-flow’s sink.

As data flows through the network, each data-flow registers a repulsion value to the tiles adjacent to its current path. The repulsion value discourages other data-flows from sharing the same communication links, avoiding contention on the on-chip network. This allows the tiled architecture to more efficiently utilize its processing resources. Repulsion values propagate only a few hops, as data-flows can exist in close proximity without contending for resources. Additionally, a data-flow sink registers repulsion to its adjacent tiles as well. The repulsion value at each node subtract from the total attraction value.

When routing, each packet in a data-flow uses only local information to determine the direction it is routed next. At each hop, a packet routes to the current tile’s neighbor that currently has the largest attraction value. In this way, routing using MBR is completely distributed, and increasing the number of tiles in the architecture has no impact on the routing overhead. The overhead to setup the initial attraction values across all tiles is amortized over the entire data transfer, so for streaming-type applications and large data transfers, this overhead is negligible.

The routing is also completely dynamic. When new data-flows are introduced to the architecture (or old data-flows finish), the attraction and repulsion values at each tile are updated. Then, as each data-flow chooses new paths that maximize the attraction value in each hop. In this way, MBR refines the data path with each hop, finding a global near-optimal routing solution.

Finally, MBR can be extended to support fault tolerant routing. When a fault is detected, the tiles adjacent to the fault assume that routing in the faulted direction has an infinite repulsion value. This can effectively route around faults, finding good routes for all data-flows.

For future work, we plan to implement the MBR routing algorithm on a tiled architecture. We expect a major hurdle to deadlock handling and avoidance when routing for large numbers of simultaneous data-flows, but previously proposed

Fig. 1. Overview of MBR routing data from a high bandwidth data-port to two data processing tasks. Communication network is not illustrated.

distributed routing algorithms give clues on how handle these issues.

REFERENCES

- [1] D. Burger, S. Keckler, K. McKinley, M. Dahlin, L. John, C. Lin, C. Moore, J. Burrill, R. McDonald, W. Yoder, and the TRIPS Team. Scaling to the End of Silicon with EDGE Architectures. *IEEE Computer*, pages 44–55, July 2004.
- [2] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore. Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture. In *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA'03)*, pages 422–433, San Diego, California, June 2003.
- [3] J. Teller, R. Ewing, and F. Özgüner. What are the Building Blocks of a Nanoprocessor Architecture? In *Proceedings of the International Midwest Symposium on Circuits and Systems (MWSCAS'05)*, Cincinnati, Ohio, August 2005.
- [4] J. Teller, F. Özgüner, and R. Ewing. The Morphable Nanoprocessor Architecture: Reconfiguration at Runtime. In *Proceedings of the International Midwest Symposium on Circuits and Systems (MWSCAS'06)*, San Juan, Puerto Rico, August 2006.
- [5] J. Teller, F. Özgüner, and R. Ewing. Scheduling Reconfiguration at Runtime on the TRIPS Processor. In *Proceedings of the Parallel and Distributed Processing Symposium, (IPDPS 2008)*, Miami, Florida, April 2008.