On-Chip Network Design Considerations for Compute Accelerators

Ali Bakhoda
University of British Columbia
Department of Electrical and Computer Engineering
Vancouver, BC, Canada
bakhoda@ece.ubc.ca

John Kim
KAIST
Department of Computer Science
Daejeon, Korea
jjk12@kaist.edu

Tor M. Aamodt
University of British Columbia
Department of Electrical and Computer Engineering
Vancouver, BC, Canada
aamodt@ece.ubc.ca

ABSTRACT

There has been little work investigating the overall performance impact of on-chip communication in manycore compute accelerators. In this paper we evaluate performance of a GPU-like compute accelerator running CUDA workloads and consisting of compute nodes, interconnection network and the graphics DRAM memory system using detailed cycle-level simulation. First, we study performance of a baseline architecture employing a scalable mesh network. We then propose several microarchitectural techniques to exploit the communication characteristics of these applications while providing a cost-effective (i.e., low area) on-chip network. Instead of increasing costly bisection bandwidth, we increase the the number of injection ports at the memory controller router nodes to increase *terminal* bandwidth at the *few* nodes. In addition, we propose a novel "checkerboard" on-chip network which alternates between conventional, *full*-routers and *half*-routers with limited connectivity. This network is enabled by limited communication of the many-to-few traffic pattern. We describe a minimal routing algorithm for the checkerboard network that does not increase the hop count.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Multiprocessors – Interconnection architectures

General Terms

Design, Performance

1. INTRODUCTION

Manycore compute accelerators present interconnect traffic that is different from that seen in conventional multi-core architectures. In this work, we explore the on-chip interconnection network design space for compute accelerators with the goal of finding cost-effective on-chip network designs for future manycore accelerator architectures (which may be GPUs, or otherwise).

An important aspect of this work is the use of real applications and detailed simulation of the overall compute accelerator including on-chip network, compute cores, and DRAM memory system. Relying on fixed synthetic traffic patterns or trace-driven evaluations results in *open-loop* simulations as the network does not impact the network traffic injected [1]. We evaluate the *closed-loop* system behavior of on-chip networks on 24 CUDA applications and evaluate the impact of on-chip network design on overall system performance – and not just on the network-only performance metrics such as network latency.

PACT'10, September 11–15, 2010, Vienna, Austria.
ACM 978-1-4503-0178-7/10/09.

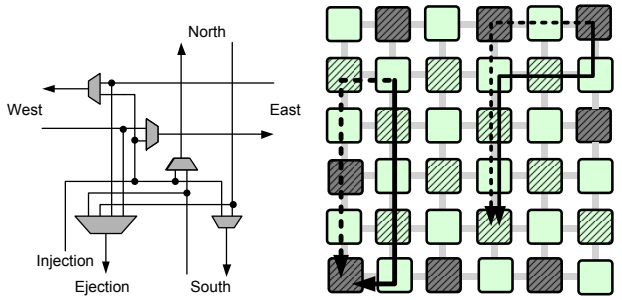
2. NETWORK CHARACTERISTICS

We analyze networks characteristics of compute accelerators using GPGPU-Sim [2] combined with Booksim 2.0 [1]. Our baseline is a 6×6 2D mesh with a 5-cycle per hop delay (a 4-cycle router pipeline and a 1-cycle channel delay) and a 16-byte channel bandwidth. The mesh connects 28 compute cores and 8 memory controllers (MC).

Router Latency and Bisection Bandwidth: An aggressive 1-cycle router *does* decrease NoC latency compared to the baseline 4-cycle router but this reduced latency results in modest speedups. On the other hand, doubling the bisection bandwidth results in substantial performance improvements. Unfortunately doubling the channel width results in a quadratic increase in router area which makes it very costly to build. Therefore, we should aim for cost-effective techniques that improve network throughput instead of investing chip resources in aggressive router architectures to reduce network latency.

Many-to-few Traffic Pattern: The compute accelerator architectures we study present the network with a *many-to-few* traffic – with *many* compute nodes communicating with a *few* memory controllers. Additionally, the traffic sent from compute cores to MCs consists of either read requests (small packets) or, less frequently, write requests (large packets) while the traffic from MCs to compute cores only consists of read-replies (large packets). This creates an imbalance in injection rates at the compute cores compared to the MCs. Higher injection rates of memory response data returning from the MCs creates hot-spots in the reply network which can stall the MC.

Network Coupling: Memory access latency is composed of three dependant components – NoC traversal to reach the MC (request network), memory access, and another NoC traversal to reach the compute core (reply network). One component can impact others and create a *coupling* effect. For example, if the reply network cannot fully absorb the traffic injected by an MC due to a hot-spot, the MC will stop accepting requests from the request network as its buffers fill up and this in turn affects the request network. We find



(a) Connectivity of a half-router (normal routers have complete connectivity)

(b) Routing. Dashed lines are examples of XY routes prevented by half-routers(hatched); alternate feasible routes are solid.

Figure 1: Checkerboard mesh On-chip Network

that bottlenecks in the reply network can impact the request network and decrease DRAM efficiency.

3. DESIGNING COST EFFECTIVE NOCS

In this section we leverage the insights from Section 2 to design more cost-effective NoCs for compute accelerator applications.

Memory Controller Placement: The high injection rate of MCs can create hot-spots in the baseline network in which the MCs are placed in neighbouring locations on top and bottom of the chip similar to Tilera and Intel's 80-core chip. A staggered memory placement like the one used in [2] enables higher injection (and network utilization) from the MC to improve overall performance. This MC placement is specially helpful when the MC injection rates are high.

Channel Slicing: Taking advantage of the quadratic area dependency on channel bandwidth, the NoC area can be reduced using channel slicing [1] and creating a *double* network, each with half the channel bandwidth of the original *single* network. Simulations show that a double mesh exceeds performance of a single mesh with the same bisection bandwidth while significantly reducing the total router area.

Multiple Injection ports for MC Routers: To help reduce the bottleneck at the *few* nodes with many-to-few traffic patterns, we propose a simple change to the routers attached to the few MC nodes which consists of adding additional injection ports from the MC and creating a *multi-port* router microarchitecture. This additional port does not increase any network channel bandwidth but instead, increases the *terminal* bandwidth by providing more injection bandwidth from the MC nodes. Speedup is possible due to the *many-to-few* traffic pattern inherent in our workloads. By increasing the terminal bandwidth through additional injection ports, speedups of up to 25% can be achieved while creating a very small area overhead, approximately 1% increase in network area. Increasing the number of injection ports beyond 2 results in minimal additional performance increase.

Checkerboard Mesh: The limited communication pattern of the many-to-few traffic pattern (no all-to-all communication) provides opportunity for optimization. We propose a *checkerboard* NoC that exploits this traffic pattern to reduce the network area. Figure 1(b) shows a 6×6 configuration of the checkerboard network where each router in the network alternates between a *full*-router shown with a solid shaded square and a *half*-router drawn with hatching. The full-router provides full connectivity between all five ports while a half-router limits connectivity as packets can not change dimension within a router (see Figure 1(a)). Half-routers significantly reduce the router area, i.e., a full-router requires a 5×5 crossbar while the half-router only requires four 2×1 crossbars (two for each dimension) and one 4×1 crossbar for the ejection port.

However, the checkerboard presents a communication limitation, i.e., it is impossible to route from a full-router to another full-router which is an odd number of columns or rows away. By constraining the location of the MC nodes to half-routers, this limitation does not become a problem since compute cores do not communicate with each other directly. The half-routers can communicate with all other nodes in the system, but simple DOR routing cannot be used.

Routing Algorithm: Assuming a baseline XY dimension-ordered routing, XY routing cannot deliver the packet in the proposed checkerboard network for the following two traffic patterns: (1) routing from a full-router to a half-router which is an odd number of columns away and not in the same row and (2) routing from a half-router to a router which is an even number of columns away and not in the same row. If YX routing is used as the baseline routing algorithm, similar routing restrictions exist as well.

For case (1), since a packet cannot "turn" at a half-router, YX routing is used instead of XY routing (e.g. see the packet starting in the upper left corner in Figure 1(b)). For case (2) a packet needs to "turn" using YX routing followed by XY routing as shown by the packet starting in the top right router in Figure 1(b). A random, intermediate full-router is selected within the minimum quadrant containing the source and destination that is not on the same row as the source and is not an odd number of columns away from source. The packet is routed minimally using YX to this intermediate router and then, routed using XY to its destination.

Flow Control: Two virtual channels are needed to avoid circular dependencies and routing deadlock: one for the YX routing and another for XY routing. While the checkerboard network performs *minimal* routing and routes with minimal hop count, it results in a significant reduction in area. Detailed area estimations show half-routers only occupy roughly half the area of a full-router.

4. SUMMARY

We make three observations regarding GPU-like compute accelerators: network bandwidth is more important than latency, many-to-few traffic introduces imbalances in resource utilization and the coupling between request and reply networks can reduce performance. Based on these observations several cost-effective solutions can be employed to improve performance: staggered MC placement, channel slicing, multiple injection ports at MC routers and checkerboard mesh.

5. REFERENCES

[1] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[2] Ali Bakhoda, George L. Yuan, Wilson W. L. Fung, Henry Wong, and Tor M. Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *ISPASS-2009*, pages 163–174, April 2009.