Microprocessors and Microsystems 42 (2016) 24–36

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Analysis of network-on-chip topologies for cost-efficient chip multiprocessors

Marta Ortín-Obón[a,*], Darío Suárez-Gracia[b], María Villarroya-Gaudó[a], Cruz Izu[c], Víctor Viñals-Yúfera[a]

[a] Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, María de Luna 1, 50018, Zaragoza, Spain
[b] Qualcomm Research Silicon Valley, California, US
[c] Department of Computer Science, University of Adelaide, South Australia, 5005, Australia

ARTICLE INFO

Article history:
Received 8 May 2015
Revised 22 December 2015
Accepted 13 January 2016
Available online 1 February 2016

Keywords:
Interconnection networks
Chip multiprocessor
Topology
Mesh
Torus
Ring

ABSTRACT

As chip multiprocessors accommodate a growing number of cores, they demand interconnection networks that simultaneously provide low latency, high bandwidth, and low power. Our goal is to provide a comprehensive study of the interactions between the interconnection network and the memory hierarchy to enable a better co-design of both components. We explore the implications of the interconnect choice on overall performance by comparing the behaviour of three topologies (mesh, torus, and ring) and their concentrated versions. Simply choosing the concentrated mesh over the ring improves performance by over 40% in a 64-core chip.

The key strength of this work is the holistic analysis of the network-on-chip and the memory hierarchy. Experiments are carried out with a full-system simulator that carefully models the processors (single and multithreaded), memory hierarchy, and interconnection network, and executes realistic parallel and multiprogrammed workloads. We corroborate conclusions from several previous works: network diameter is critical, the concentrated mesh offers the best area-energy-delay trade-off, and traffic is very light and highly unbalanced. We also provide interesting insights about application-specific features that are hidden when studying only average results. We include a fairness analysis for multiprogrammed applications, and refute the idea of the memory controller placement greatly affecting performance.

1. Introduction

Nowadays, a single chip may contain multiple processors and a significant amount of memory. A popular trend consists of interconnecting several nodes, each of them with a core and one or more levels of private and/or shared cache memories. Nodes communicate through an interconnection network that allows them to exchange coherence messages and cache blocks, and has a major impact on overall performance, energy consumption, and area. We focus on general purpose CMPs, where both high-performance and low-power are required in equal shares.

Only a few works study the interconnect by modelling in detail the processors, memory hierarchy, and interconnection network. However, those analysis are often performed with synthetic traffic or application traces that do not entirely capture the behaviour of a real execution [6,10,25,30]. This work simulates both parallel and multiprogrammed workloads with real applications, carefully modelling all the components above-mentioned. This allows us to study the effect of the interconnection network configuration on the whole system and the real interactions between the memory subsystem and the interconnect. We revisit the comparison of several topologies with our detailed simulation framework to update the results, validate or refute previous conclusions, and complete them with further analysis. We present an analysis of three topologies with varying degrees of complexity, performance, power, and area: mesh, torus, and ring. We model CMPs with 16 and 64 single-threaded cores, including a configuration with 16 4-threaded cores, and explore the effect of modifying the location and number of memory controllers. Our goal is to draw meaningful conclusions on the studied network configurations and study the details, pointing out the best choice from an integrated performance, area, and energy standpoint.

The rest of this document is organized as follows: Section 2 presents the related work; Section 3 describes the CMP architecture and the interconnection network configuration;

* Corresponding author. Tel.: +34 876555341.
E-mail addresses: ortin.marta@unizar.es, ortin.marta@gmail.com (M. Ortín-Obón), dario@unizar.es (D. Suárez-Gracia), mvg@unizar.es (M. Villarroya-Gaudó), cruz@cs.adelaide.edu.au (C. Izu), victor@unizar.es (V. Viñals-Yúfera).

http://dx.doi.org/10.1016/j.micpro.2016.01.005
0141-9331/

Section 4 introduces the methodology followed in this work; Section 5 describes the qualitative analysis of the topologies; Section 6 explains our simulation results, and Section 7 concludes the paper.

2. Related work

Several publications have highlighted the impact of the network on performance, energy, and chip area. However, only a few papers focus on the comparison of interconnection network configurations. Balfour and Dally present an analysis of how different topologies affect performance, area, and energy efficiency [6]. However, they do not model the memory subsystem, only use synthetic traffic patterns, and do not consider simple topologies like the ring. Gilabert *et al.* focus on physical synthesis of several networks, but do not simulate real applications or systems larger than 16 cores [16]. Villanueva *et al.* highlight the importance of a comprehensive simulation framework and present results of the execution of real parallel applications and its close relationship with cache behaviour [41]. Sanchez *et al.* explore the implications of interconnection network design for CMPs [36]. We complement their results including a simple topology (ring), multiprogrammed workloads, traffic distribution analysis, the effect of memory controller placement, and the influence of the network topology on fairness.

Many papers propose alternatives to conventional router architectures, topologies, and flow control methods on isolation. However, they do not consider the impact on the overall system and back up the results with network-only simulations of synthetic traffic and traces. Carara *et al.* revisit circuit-switching which, as opposed to packet-switching, allows to reduce buffer size, and guarantees throughput and latency [10]; Walter *et al.* try to avoid hotspots on systems on chip by implementing a distributed access regulation technique that fairly allocates resources for certain modules [42]; Mishra *et al.* propose an heterogeneous on-chip interconnect that allocates more resources for routers suffering higher traffic but they only get good results with a mesh topology [33]; Koibuchi *et al.* detect that adding random links to a ring topology results in big performance gains, although they only experiment with a network simulator [25]. All these studies either do not model the whole system, do not include a significant variety of real workloads, or do not experiment with different topologies. Also, most of them only include network-related metrics and fail to report on overall performance, or elaborate conclusions based on IPC (instructions per cycle), which has been reported to be unsuitable for parallel applications [47].

Another approach consists on designing the network considering the behaviour of the memory subsystem and the coherence protocol. Yoon *et al.* propose an architecture with parallel physical networks with narrower links and smaller routers that eliminates virtual channels [45]. Seiculescu *et al.* propose to use two dedicated networks: one for requests and one for replies [37]. Lodde *et al.* introduce a smaller network for invalidation messages, but only test their design with memory access traces [30]. Agarwal *et al.* propose embedding small in-network coherence filters inside on-chip routers to dynamically track sharing patterns and eliminate broadcast messages [5]. These studies try to improve the performance of the most commonly used networks, but do not venture with less conventional topologies. Also, they only experiment with a maximum of 16 cores. Krishna *et al.* propose a system to improve the frequent one-to-many and many-to-one communication patterns by forking and aggregating packets to avoid the increment in traffic as the number of nodes increases [26]. Bezerra *et al.* try to reduce traffic by statically mapping memory blocks to physical locations on the chip that are close to cores that access them [8]. The last two proposals are only evaluated with a typical mesh topology.

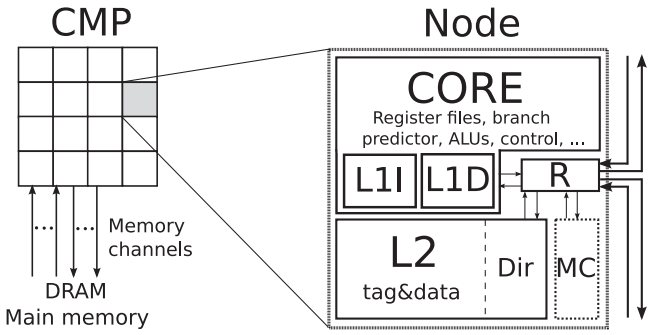

Fig. 1. Block diagram including a chip and the components of a tile. MC stands for memory controller, R is the router, and Dir is the directory, which is included in the L2 cache. This example router has two input and two output ports connected to other neighbouring tiles.

Table 1
Main characteristics of the CMP.

Cores	16 single and multithreaded cores, and 64 single-threaded cores, Ultrasparc III Plus, in order, 1 instruction/cycle and thread, 2 GHz frequency
Coherence protocol	Directory-based, MESI, directory distributed among L2 cache banks
Consistency model	Sequential
Private L1 cache	32KB data and instruction caches, 4-way set associative, 2-cycle hit access time, 64B line size, pseudo-LRU replacement policy
Shared L2 cache	Physically distributed, 1 bank/tile, 1MB per bank, 16-way set associative, 64B line size Pseudo-LRU replacement policy, inclusive, interleaved by line address 7-cycle hit access time
Memory	4 memory controllers, distributed in the edges of the chip, (both for 16 and 64-core architectures), 160-cycle latency Section 6.7 considers different number and location of memory controllers

3. CMP architecture framework

This section presents the modelled CMP architecture and a detailed description of all the interconnection network configurations.

3.1. General system architecture

Our study focuses on homogeneous CMPs. The system is composed of several tiles connected by an interconnection network. Each tile has a core with a private first level cache (L1) split into data and instructions and a bank of the shared second level cache (L2), both connected to the router. In the initial setting, four tiles in the edges of the chip also include a memory controller. Fig. 1 depicts the block diagram of the chip and a tile with memory controller. It also includes the connections between the elements in the tile and the router. Table 1 summarizes the key parameters of the system. To model the architecture we based our design on other systems with similar characteristics, both from academia [7,37,46] and industry (Tilera's TILE*Pro*64 [40], Intel Xeon Phi [20], and Intel 48-core processor [19]). To size our L2 cache (which is our last level cache) we have taken a configuration very frequently used in academia [1,2,22] that is also a nice compromise among the sizes of shared last level caches in high and low-end commercial platforms. For example, the AMD Opteron processor has a shared L3 cache of 6MB for 6 cores [11]; IBM Power8 has 8 to 12 cores with 8 threads per core, and includes an L3 cache with 64 to 96MB, as well as an L4 cache with 32 to 64MB [18]; Intel Xeon