On-Chip Networks for Modern Large-Scale Chips

N

George Michelogiannakis
Lawrence Berkeley National Laboratory, USA

INTRODUCTION

In recent years, research in computer architecture has experienced a substantial shift in focus. Technology scaling has slowed down, which led to scaling difficulties for large uniprocessors (Olukotun, 2007). Clock frequency in large uniprocessors cannot be increased any more at the same rate as in the past due to excessive power dissipation. Moreover, only a limited amount of parallelism can be extracted from a single typical instruction stream using conventional superscalar instruction techniques (Postiff, 1999; Kusic, 2005). Therefore, to satisfy the increasing demands for computation, research has focused on parallel computing.

Parallel computing has been rapidly developed in recent years due to technology scaling which allows us to place hundreds or thousands of processing cores and their cache blocks in a single die called a chip multiprocessor (CMP). This has led to the development of *on-chip network*s (Dally, 2001) to enable communication between large numbers of processing, cache or memory controller blocks efficiently. On-chip networks are a rapidly developing field due to their impact in system performance and cost (Kumar, 2005; Sanchez, 2010).

The importance of on-chip networks is anticipated to rise in modern and future chip multiprocessors. The Intel 80-core Teraflop chip, manufactured in 2007, attributes a quarter of overall power consumption to the on-chip network (Hoskote, 2007). The MIT RAW attributes a higher percentage for the on-chip network power, reaching up to 36% (Taylor, 2004). The same studies also show the impact of on-chip networks to application execution time, primarily because of communication latency and throughput characteristics. More recent designs incorporate a few hundred of cores on-chip, such as the NVIDIA Fermi with 512 graphics processing cores, and AMD Fusion with four general processing cores and 408 graphics cores. To make matters worse, it is projected that with 2018 technology, communication will in fact require more energy than computation, even in the case of floating point computation in the on-chip environment (Shalf, 2010). In addition, the amount of communication will increase to satisfy future communication demands, especially since CMPs are expected to expand to 2048 processing cores within the decade. Therefore, making on-chip network communication efficient is critical, especially to satisfy future demands.

This article provides an overview of key design features and current state of the art of on-chip networks. This article also offers a discussion of current limitations and opportunities for future work in on-chip networks, focusing on flow control and datapath width for the CMP environment, as well as co-designing the network with the rest of the system. Finally, this article will also focus on the choice of flow control in CMPs.

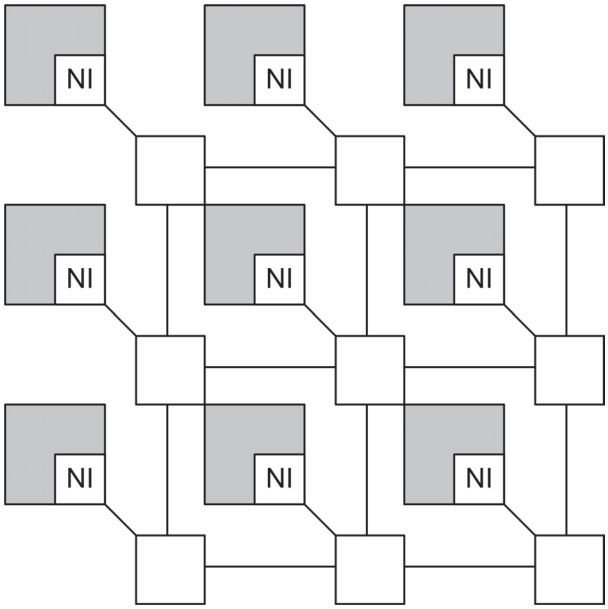
BACKGROUND

On-chip networks are composed of a set of routers interconnected by point-to-point links. The manner routers are connected to each other and thus the layout of the network is specified by the topology. While numerous topologies have been proposed, the most widely used topology is the 2D mesh due to its simplicity and modularity. In the 2D mesh, each router is connected only to its neighbors. Each router is also connected with a local processing, cache or other block through a network interface. In a N-by-N 2D mesh the average number of hops is approximately $1+\sqrt{N}$ while the maximum number of hops is $2\sqrt{N}$. A 3x3 2D mesh is shown in Figure 1.

A variety of other topologies have been proposed. Some topologies use express channels, which connect routers to far away routers in order to skip unnecessary hops. For example, the flattened butterfly topology

DOI: 10.4018/978-1-4666-5888-2.ch616

Figure 1. In a 2D mesh each router is connected only to its neighbors



(Kim, 2008) connects each router to every other router in the same row and column. This way and with minimal routing, a maximum of only four hops are needed (one per dimension) while the average number of hops is 3.5.

Routing in the network can be deterministic, oblivious or adaptive. With deterministic routing, the same path is always used for a given source—destination pair. Oblivious routing provides path diversity by randomly choosing among a set of eligible paths. Finally, adaptive routing makes choices based on current network state.

Since messages to be transported over the on-chip network can be large, messages are broken into packets. The maximum packet size is pre-defined. A packet is an independent entity that contains all necessary information to reach its destination, such as the destination ID. Because packets can themselves be substantially larger than the datapath width of the network, packets are divided into flow control digits (flits). Flits may consist of multiple—but usually one—physical digits (phits); the size of phits equals the datapath width. With this organization, packets are transferred across the narrower channels over several cycles, incurring a serialization latency that equals the number of cycles to transmit the tail (last) flit after submitting the head (first) flit, in the absence of backpressure. Head flits carry the destination and other control information for the whole packet. Other flits merely contain a packet identifier such that the network knows the packet they are a part of.

State-of-the-art on-chip networks today use input-buffered flow control with virtual channels (VCs) (Dally, 1992). With this flow control, routers use a buffer per input where flits can wait until their output becomes available. VCs are used to define classes of traffic such that contention in one VC does not affect another VC. With VCs, input buffers contain a FIFO per VC such that flits in a VC do not block flits in another VC. VCs are predominantly used to improve performance, provide quality-of-service guarantees, and prevent deadlocks by dividing traffic into classes such that cyclic dependencies due to routing decisions or packet type dependencies (such as request—reply dependencies) are eliminated (Bjerregaard, 2006; Dally, 2003).

To prevent buffer overflow, credits are used. Credits are tokens which represent a free buffer slot at a specific router input buffer and VC. When a flit departs a buffer to proceed to its output, a credit is generated for the VC that the flit was in. The credit is then transmitted to the upstream router. The upstream router keeps track of how many credits it has to the downstream router. The upstream router can only send a flit to the downstream router if it has at least one credit for the flit's VC. When the flit is sent, the credit is consumed because the free buffer slot the credit represents will be occupied by the transmitted flit. With credit flow control, buffer size must be at least twice the propagation delay between routers to avoid stalling due to credit propagation delay in situations without contention.