Memory-aware NoC Exploration and Design

Nikil Dutt
Center for Embedded Computer Systems
Donald Bren School of Information and Computer Sciences
University of California, Irvine, CA 92697-3435
dutt@uci.edu
http://www.cecs.uci.edu/~dutt

Memory-aware NoC design methodology

In the past decade, tremendous progress has been made in NoC research, spanning architectures, protocols and tools. In addition to a large number of academic and research projects, we are now seeing several commercial realizations of NoC-based chip designs. With chip capacities going well beyond the billion transistor mark, on one hand large amounts of the die are occupied by memory resources and on the other hand many complex applications being mapped to these chips are also memory-intensive. In such instances, memories dominate all the axes of traditional design constraints, including, but not limited to performance, area (cost), and power/energy. Furthermore, the move towards sub-nanometer technologies elevates another critical design consideration: process variability and thermal sensitivity, which in turn critically affect the reliability of memories as well. All of these trends make the case for a *memory-aware NoC design methodology*.

Traditional NoC design flow

In a traditional NoC design methodology flow, applications are first analyzed to determine the essential characteristics of data traffic, thereby establishing the communication requirements for the applications. These communication requirements are typically represented as variants of a Communication Throughput Graph (CTG), where nodes represent computational cores and edges specify the direction and amount of communication between the cores. NoC synthesis algorithms are then applied to these communication throughput graphs to perform topology synthesis, mapping/binding of physical cores to CTG nodes, and communication/protocol synthesis to establish communication paths, reserve routing resources, and optimize key NoC architectural parameters, including link widths, buffer sizes, core frequencies, etc. [1].

Memory-Aware NoC design flow

Although the traditional NoC design flow implicitly captures memory requirements through a "*communication-exposed*" design paradigm – initially through the CTG and subsequently through the synthesis process – memories themselves are not treated as first-class objects during NoC synthesis. Given the importance of memory issues in NoC design, there is a critical need to incorporate memory decisions both early in the analysis phase, as well as concurrently during NoC synthesis. A *memory-aware NoC design flow* needs to facilitate early memory decisions (e.g., partitioning, hierarchies, memory access architectures), as well as memory customization concurrently with the traditional synthesis steps. Towards this end, a "*memory-exposed*" design paradigm needs to be developed, leading to a number of new and interesting research problems that will need to be coherently stitched into customized NoC design flows, depending on the application, NoC architectural template, and overall design goals and constraints. Some sample problems and issues are summarized below.

o *Combining programming models for complex NoCs*
Many researchers are advocating "memory-exposed" and "communication-exposed" programming models for classes of applications that process large volumes of data, typically in a streaming fashion. While such programming models are eminently suited for streaming applications, complex SoC platforms are likely to support a variety of applications exhibiting heterogeneity in their data access patterns and volumes, as well as in control and interrupt-driven behaviors. Thus a memory-aware NoC methodology should support the use of diverse programming models suited for a range of behaviors; the traffic patterns and constraints will then need to be succinctly captured in a unifying internal representation (e.g., a variant of the CTG). Memory-aware NoC synthesis tools can then be developed to handle such heterogeneity in programming models.

o *Rethinking the cache coherence paradigm*
A direct consequence of allowing heterogeneous programming models is that the resulting memory architectures are themselves going to be heterogeneous: caches will be only be part of the memory architectural landscape, and a diverse, customized memory architecture can be deployed for the NoC. Thus cache coherence may only be a small part of the problem: rather, NoC design flows should focus on *memory coherence* between diverse memory structures, including hybrids that combine shared memory with message passing; software controlled scratchpads, stream buffers, CAMs etc.

o *Application restructuring within NoC design flow*
Many decades of prior research in parallel processing have highlighted the hard-to-quantify, but crucial relationship between application restructuring and both coarse- and fine-grained parallelism. Regardless of whether this is done automatically or manually, each of these application variants results in distinct memory and communication

requirements. While the problem has been studied to some extent for homogeneous, massively parallel architectures, many NoCs will need to deal with a variety of heterogeneous cores, which in turn require new methods and heuristics for exploring source-level restructuring to generate a larger space of memory and communication alternatives. Thus explicit efforts are needed in software support for application/source-level memory bandwidth exploitation and exploration.

o Hybrid NoC and bus-based architectures

With the relentless integration of complex application functionality, NoC platforms will increasingly require a holistic combination of guaranteed throughput, together with best effort data delivery. This will require methodologies and tools that allow seamless integration of NoC [1] and bus-based communication architectures [2].

o Application-Memory-Communication Codesign

The traditional NoC synthesis approach assumes that the number of memories, and the memory requirements are fixed before NoC synthesis. However, there is a need to incorporate the memory requirements during the process of NoC synthesis. Data reuse graphs [3] are a promising mechanism to combine a number of different communication graphs that differ in their memory configurations. An application-memory-communication codesign process can be developed using notions similar to data reuse graphs.

o PGP (Pretty Good Platforms): rethinking phase coupling, physical coupling and optimality

An important and overarching goal of a design methodology is to realize PGPs (Pretty Good Platforms) quickly and reliably. Thus we need to revisit the notions of phase- and physical-coupling, as well as of optimality. Similar to many previous incarnations of hardware synthesis (logic, behavioral) and software compilation, we recognize that various phases of NoC synthesis are inherently coupled with each other. Many approaches have been developed to solve this phase-coupling problem. Furthermore, research has also investigated the coupling of physical design information into early NoC synthesis, as well as combinations of static analysis and dynamic profiling for performance evaluation and validation. When memory issues are added to this mix, the notion of "optimality" necessarily becomes a local view. Instead, the design methodology should support the notion of "exploration surfaces" that are composed from pareto-optimal subsets of the complex design constraints. NoC architects can then navigate the memory-aware design space to focus on the most promising regions for design refinement and further synthesis/optimization. Thus although no guarantees can be made on the convexity of these design spaces (and the resulting decisions may not be "optimal") , NoC architects can quickly explore and generate PGPs.

o Guaranteeing real-time constraints

Much of the existing body of work on NoCs has not focused on guaranteeing real-time constraints (either as intervals or deadlines), but instead has focused on throughput-driven timing constraints. However, we will increasingly see combinations of the need for real-time predictability, in addition to throughput constraints. Thus new methods and tools are required to facilitate simultaneous satisfaction of both classes of timing constraints.

o Aggressive error-aware design methodology

With the move towards ever-finer device geometries, we will see tremendous variations in error profiles based on process variations and temperature sensitivity, particularly in the memories that will occupy a large amount of the chip's real estate. NoCs, by their very nature are error-tolerant and may mitigate some of these problems. However, in these situations, we can use errors to our advantage: many classes of emerging embedded applications (e.g., communications and multimedia) are inherently error-tolerant, allowing for a lower Quality-of-Service (QoS) at the expense of higher errors. By intentionally introducing errors in the system, we may release some design "slack" in another dimension, allowing for more aggressive optimization of another design constraint [4]. NoC design methodologies should investigate such "error-aware" opportunities for aggressively trading off errors for gains in other constraint dimensions.

o Application-specific NoC design flows

NoC architectural styles will see large variations in their primary constituents: cores, interconnects and memories. Thus there is both an opportunity, as well as a need for developing customized design flows for each application, that will tune the processor, interconnect and memory parameters to generate PGPs with a range of performance, cost, and power/energy profiles.

Many more challenges and opportunities will appear as we move towards a memory-aware NoC design flow. Last, but not the least, are the additional validation and verification challenges that appear once we make memories a "first class" design concern.

References

[1] G. De Micheli and L. Benini, *Networks on Chips*, Morgan Kaufmann, San Francisco, CA, 2006.

[2] S. Pasricha and N. Dutt, *On-Chip Communication Architectures: System on Chip Interconnect*, Morgan Kaufmann, San Francisco, CA, 2008.

[3] I. Issenin and N. Dutt, "Data Reuse Driven Memory and NoC Co-Synthesis," *Proc. 2007 International Embedded Systems Symposium (IESS 2007)*, May 2007.

[4] N. Dutt, F. Kurdahi, A. Eltawil and S. Nassif "Cross-Layer Approaches to Designing Reliable Systems using Unreliable Chips", All-day tutorial, *ASPDAC 2008*, January 2008.