RaDCPED: A General Cache Coherence Protocol Processor Development Framework in Bluespec

Alfred Man Cheuk Ng
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
Email: mcn02@mit.edu

Abstract

We present a framework for the Rapid Development of Cache-Coherence Protocol Engine (RaDCPED) in Bluespec. The framework is highly parameterized and general, which allows designers to rapidly develop protocol engines for a large class of protocols. Therefore, designers can compare different designs instead of freezing the design prematurely in the development process. Since Bluespec can generate either C or Verilog RTL, RaDCPED is capable of generating designs for software simulations, FPGA-based simulations and even ASIC synthesis.

The RaDCPED Framework

Every shared memory system, whether it is a Distributed Shared Memory (DSM) system or a Chip Multi-Processors (CMP) system, can be distributed into nodes [1], [2]. We developed the RaDCPED framework to generalize the designs of the node. Figure 1 gives an overview to the generalization. As can be seen, each node can be divided into five modules: a Data Memory, three network controllers (Networkup, Networkpeer and Networklow) and a Protocol Processor. Each module deals with different aspects of the implementation of a cache coherence protocol. Caches and Shared Memories can be implemented by the Data Memory, which is responsible for data storage; the three network controllers are responsible for network message passings from different directions; and the Protocol Processor carries out actions according to the specifications of the cache coherence protocol by coordinating other modules in the node. Not every design requires all five modules. Modules shown in white are optional.

Implementation in Bluespec

RaDCPED was implemented in Bluespec. Each module consists of a standard interface and a variable implementation body. An interface defines a group of methods which capture the communications between modules efficiently [3]. The interfaces of Bluespec embeds both data and control paths, which allows the compiler to automatically generates the required control logics for implementations with different characteristics for the same interface. This approach allows designs to be highly parameterized and modular. For example, different caches, having the same interface, can be generated by the same piece of code which parameterizes the cache size, the cache line’s width, the associativity and the cache replacement policy.

Apart from the strengths mentioned above, Bluespec speeds up the development process by generating designs for both software simulation and hardware synthesis: In traditional hardware design process, simulation and synthesis are written in different languages. Normally, simulations are written in high level software languages like C/C++ or Java; while hardware is done from RTL level hardware languages like Verilog or VHDL. Therefore, designers are required to write the code twice for a single design. Moreover, the semantic gap between the simulation language and the synthesis language makes it difficult for designers to prove that the two sources represent the same design. On the other hand, designs in Bluespec can avoid these limitations.

Example Design of RaDCPED

We used RaDCPED to implement a testing system. The system employs a protocol for the Commit-Reconcile Fences (CRF) memory model [4]. An advantage of the CRF protocol over other cache coherence protocol is the possibility of avoiding the communication overhead of false shadings because the CRF explicitly separates the data synchronizations from other memory operations. This allows the system to maintain copies of a cache line at different sites at the same time.
time even when they are modifying the data of different addresses of the same cache line.

Future Work

In the future, we plan to use RaDCPED to implement more protocols and speeds up the simulations via an FPGA. We also plan to integrate RaDCPED and UNUM [6], which is a general microprocessor framework, so that shared memory systems with different microprocessor architectures can be generated rapidly.

References