ProtoSLICC: Automated Synthesis of Gem5-based Cache Coherence Controllers

Nicolo Carpentieri Anatole Lefort David Schall Pramod Bhatotia

System Research Group, TU Munich
email: {nicolo.carpentieri, anatole.lefort, david.schall, pramod.bhatotia}@tum.de

Cache coherence protocols lie at the heart of memory consistency, ensuring that shared data remains coherent across cores, caches, and memory. However, designing these protocols is a notoriously challenging, labor-intensive, and error-prone task due to the sheer diversity of data sharing patterns, multi-level cache hierarchies, and races between concurrent transactions. While there are efforts to verify their correctness [1], there is still no systematic or modular way to synthesize these protocols. Yet, as modern architectures evolve (incorporating distributed shared memory, processors with multiple levels of cache), correctness alone is no longer sufficient. Understanding a protocol's performance becomes equally crucial. Architects must turn to detailed simulation environments, such as gem5, to assess a protocol's performance and to explore design trade-offs. Gem5's *Ruby* memory system provides a flexible framework to model custom cache coherence protocols through its domain-specific language called SLICC (Specification Language for Implementing Cache Coherence), which developers use to write protocol specifications. Unfortunately, writing and maintaining protocols in SLICC is a labor-intensive and error-prone process. The SLICC descriptions are not modular: each new protocol typically requires starting from scratch or extensive rewrites of controller logic. Even adding a new level of cache in the system can necessitate redesigning portions of the protocol.

In this work, we present ProtoSLICC, a tool that extends the capabilities of ProtoGen by generating SLICC coherence controllers for gem5. The tool accepts two primary inputs: (1) a high-level protocol specification, defined in terms of stable states and atomic transitions, and (2) a set of parameters that define the system's topology, such as cache hierarchy, number of cores, etc. Our generator, based on ProtoGen's frontend, first parses the protocol specifications and constructs the intermediate representation (IR) of the protocol's finite state machine (FSM). ProtoGen ensures the concurrency and correctness of these FSMs, while ProtoSLICC extends this workflow through a dedicated SLICC backend. This backend translates the protocol's intermediate representation (IR) into gem5-compatible cache and directory controllers, dynamically incorporating the system topology parameters to generate implementations tailored to the target architectural environment. An additional novelty lies in ProtoSLICC's parser, which analyzes the input topology and automatically infers FSMs for auxiliary controllers (e.g., LLC, multi-level private caches, L1 instruction cache) directly from the initial protocol specifications. ProtoSLICC can handle both flat coherence hierarchies (single-level directories or monolithic coherence domains) and more complex NUMA-style or hierarchical systems with multiple coherence domains bridged together. By automating the protocol generation, we enable rapid exploration of design trade-offs and facilitate the integration of new protocols into gem5.

We aim to use the gem5 workshop to present our implementation and show first results. We will highlight the challenges encountered and future research opportunities. Finally, we hope to trigger a discussion and collect feedback to improve our design.

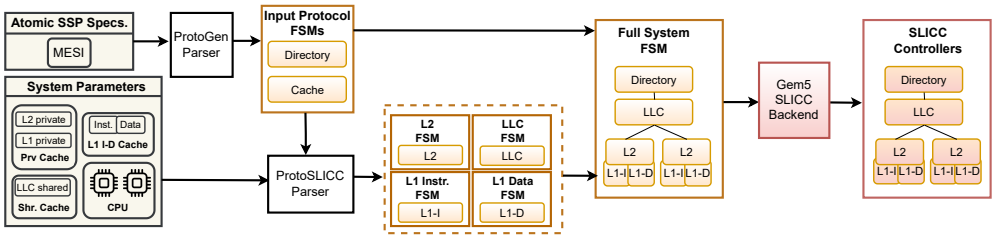

Figure 1: ProtoSLICC workflow to synthesize controllers for a directory-based MESI protocol, with private L1 instruction/data caches, private L2, and shared LLC.

[1] Nicolai Oswald, Vijay Nagarajan, and Daniel J. Sorin. “ProtoGen: Automatically Generating Directory Cache Coherence Protocols from Atomic Specifications”. In: *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 2018, pp. 247–260.