Project 5: Coherence Simulation

1 Overview

In this project, you will model different coherence protocols.

2 Logistics

This is a partner project. You are permitted to share code only with your partner. You may use C or C++ for this assignment, and you should only rely on standard libraries.

You are extending the coherence component to support the remaining states declared in enum coherence_scheme within coher_internal.h: MSI, MESI, MOESI, and MESIF.

3 Simulator Specifications

You will be creating your own copy of the coherence component. This component currently supports the MI protocol. You will need to modify coherence.c to introduce your own callbacks for cache and bus actions for the other protocols. You can add those callbacks to the protocol.c or create additional source files.

The model is that the cache manages the address to line mapping, while the coherence component manages the permissions and status of a line. On each access, the cache will query the coherence component via `uint8_t permReq(uint8_t is_read, uint64_t addr, int processorNum)`, which returns true if the access has permissions and false if it is delayed. Any delayed accesses will later complete after receiving the data via a bus action. This functionality has support for different types of requests; however, this is currently ignored. One future support would be for the cache to notify the coherence component that it needs to evict a cache line.

The coherence component can also receive notifications from the interconnection network as bus actions, `uint8_t busReq(bus_req_type reqType, uint64_t addr, int processorNum)`. The coherence component may transition to a different state, it may respond with the data, indicate sharing of the cache line, or have received the requested data. Currently, the model is just that a single cache receives the data for a request; however, there is an optimization here that is beyond the scope of this assignment.

In both request APIs, the coherence component may transition to a different coherence state. As shown in the starter code, all of these states are already being managed.

Please note that there is a separate, not default, not reference cache component that needs to be used for this assignment. It has several simplifying assumptions as well as support for multiple processors. You are welcome, but not required for the assignment to add this support to your own cache component.