Dragon Cache Coherence Protocol: Design and Simulation

February 15, 2026

1 Introduction

The goal was to simulate the Dragon cache coherence protocol in order to provide a clear and interactive visualization of its behavior in a shared-memory multiprocessor system. The simulator models the essential hardware components involved in maintaining cache coherence, including processors, private caches, coherence controllers, a shared bus, and main memory. Through step-by-step execution, the project aims to illustrate how read and write operations trigger state transitions and bus transactions.

A graphical user interface was implemented using the `raylib` library. The interface visually presents the system components, the protocol state machine, and a logging mechanism that records events occurring during each simulation cycle.

This implementation is minimalistic and focuses on demonstrating the core concepts of the Dragon cache coherence protocol. It does not attempt to model all timing details or hardware-level optimizations found in real-world processors. Instead, it serves as a bare-bones educational tool designed to make the fundamental mechanisms of cache coherence easier to understand and observe.

2 Short guideline

Starting the program opens a window similar to the one shown in Figure 1. The left side of the window displays the main hardware components involved in the simulation, while the right side shows the state diagram of the Dragon cache coherence protocol. The simulated system consists of main memory, a shared bus, and four *nodes*. Each node contains a processor, a cache memory, and a coherence controller. The coherence controller includes a write-back buffer, a snooper, an outstanding transaction table, and the finite state machine implementing the cache coherence protocol.

Before starting the simulation, the user must enter the processor instructions using the keyboard. The commands should follow the format shown in the lower part of the screen:

```
P[0-3] R/W addr ; P[0-3] R/W addr ...
```

An example of a valid command sequence is:

```
P0 R 0x4 ; P1 W 0x8 ; P0 W 0x8
```

After entering the desired sequence of instructions, the user can press the `'ENTER'` key to begin the simulation. Once the simulation starts, additional instructions cannot be added unless the program is restarted.

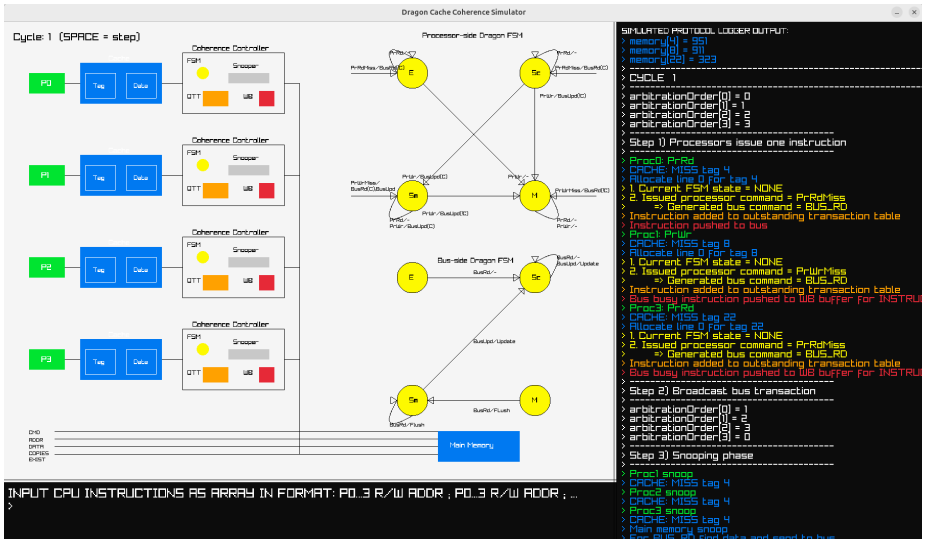

Figure 1: Program window

At this point, the instructions have been loaded and the main memory is initialized with data values corresponding to the addresses specified in the command array. One simulation cycle corresponds to the execution of a single processor instruction. By pressing the `'SPACE'` key, the user advances the simulation by one cycle. All events that occur during that cycle are recorded and displayed on the right side of the window in the section labeled as the logger.