Design Document
Cache Coherence Simulator

Aditya Gupta 2015CSB1003
Love Mehta 2014CSB1018

1 Introduction

When we wish to design caches for a multiprocessor system we need to take care of cache coherence among the caches and the main memory. For maintaining cache coherence we define different protocols for communication amongst the caches which are MSI (Modified-Shared-Invalid), MESI (Modified-Exclusive-Shared-Invalid) and MOESI (Modified-Owned-Exclusive-Shared-Invalid).

1.1 Objectives

The main objectives of this project is to simulate different cache coherence protocols for gaining statistics about the use of cache and an insight into the working of the different cache coherence protocols in a detailed manner. We can list them in detail as follows:

- Gaining statistics about the cache usage - number of flushes, hits, misses, evictions, cache sharing.
- Also statistics about the main memory usage - amount of main memory used.
- Details about state transitions in the protocol, which cache line is in which state during and after each memory access.
- Statistics about bus usage such as number of bus requests, requesting caches, most frequent bus requests' information.
- Provide an educational means for learning the working of caches' coherence.
- Statistics about cycles required for each operation - bus requests, snooping, memory access and total program execution.

1.2 Scope

The scope of the given project is limited to the following points:

- The project only intends to study cache coherence based on the address not on the memory values.
- It simplifies cycles required for each operation to few hypothetical simple values.
- It does not take into account other complications regarding actual practical system, i.e. it simulates a simpler complexity system without any practical considerations.

1.3 Functionality

Following are the intended functionalities of the given project:

- Take an input in format of list of memory accesses and simulate each accesses in a linear manner providing relevant statistical information along with.
- Provide any particular information about the cache or bus at the end based on user request or by default.
- Finally giving all relevant statistics about the simulation performed.

2 Implementation Details

We wish to have the following modules:

2.1 Cache Set

Data Structures

- Cache Line - structure containing address (corresponding to main memory), tag, set and state information.
- List of Cache Lines.

Functions

- Add a line, Remove a line, Checking existence of a line, eviction of a line from the set.
- Getting number of lines currently in the set and whether set is empty or full.

2.2 Cache

Data Structures

- ID of the cache (ordinal of the processor attached to).
- Lists of Cache sets.

Functions

- Add a line, Remove a line, Checking existence of a line, returning state of a line or assigning a state to a line.
- Handling read & write requests (performing required operations depending upon the existence and if present then state of line).
- Handling bus requests.
- Performing an atomic operation consisting of one or many of the above operations.

2.3 Bus

Data Structures

- Bus request - structure consisting of an operation for a particular address.
- A lock for accessing the bus.

Functions

- Lock accessing and releasing.
- Performing an atomic operation of broadcasting and receiving the messages, also performing relevant operations.

2.4 Simulator

Data Structures

- Instances of Caches and bus.

Functions

- Reading input file and simulating(pushing) the request on the corresponding cache. Note that the requests will be fulfilled later in an atomic order through the use of the bus.
- Maintaining statistics through internal variables of the instances and displaying them (depending upon client requests).

3 Testing

Following types of tests are tentatively planned to be done:

- Tests that exhaustively check every state transition and its side effects corresponding to each protocol.
- Tests for checking correctness of the simulator's caches' and bus's address and line management.
- Tests for checking correctness of the statistics provided by the simulator.