CSE P548: Computer Architecture
Assignment 4
Due: Wednesday, June 1

This assignment gives you the opportunity to design and evaluate your own cache coherency protocol.

For this assignment you can work in teams of two. I realize this is getting harder, but try to choose someone you haven't worked with yet. If it's too hard, so be it.

Part I: Cache coherency protocol design

In class we will study a simple 3-state snooping cache coherency protocol. Although two bits are needed to encode the coherency state in this protocol, only three of the four possible values are used: invalid, shared, and exclusive. This begs for a coherency protocol which utilizes the fourth value to implement a fourth state that improves multiprocessor performance in some way.

In this project you are to come up with a fourth state and present hypotheses that support its value, i.e., in what situations should it improve performance; why do you think those situations will occur often enough to make a performance difference (Amdahl's Law applies here again); does it have any downsides; why do you think its advantages will outweigh the downsides, etc. Design a finite state machine that carries out the functions of your new protocol with its fourth coherency state, both from the CPU and the snoop points of view. In your report, represent this FSM as nodes and arcs like we will do in class (and is done in the text as well).

Part II: Cache coherency protocol implementation

Implement your new protocol with its fourth state. For this work you will use sim3mp, a multiprocessor simulator from UW CSE that is based on SimpleScalar. In addition to familiar single-processor cache parameters, the simulator has multiprocessor-specific parameters, such as the number of processors and bus speeds.

Evaluate your new protocol relative to the three-state protocol that is already implemented in sim3mp. Don't just say the performance difference is X. Design and implement metrics that show *why* one protocol is better than the other.

You will have to write small test cases to test and debug particular aspects of your protocol, to make sure they do what they are supposed to and don't do what they are not supposed to do. The completeness of these tests will be a component of your project grade.

In addition to the protocol, you should implement a "global coherency state monitor" that tracks the effects on the entire multiprocessor of changes brought about by reads and writes to shared data. This monitor guarantees that mistakes in your protocol are caught immediately,

saving you a lifetime of debugging. I'll explain the monitor in more detail in class.

Part III: The report

Turn in the fruits of your labor in the form of a report. Douglas will get in touch with you if he wants to see your code.