Project Title: Robust Cache Coherence Protocol Verification with Inferno

Team Name: FLY-Bee

Team Member: Zeyu Bu, Xiangfei Kong, Chenxi Lou, Yao Jiang

- **Problem to be addressed:**

Fault-tolerant architectures are emerging to guarantee reliable functionality on vulnerable silicon devices. For a fault-tolerant architecture, the RTL design is more complex than a normal one and is more likely to introduce design bugs.

- **Why does this problem matter?**

Fault-tolerant architectures are popular in current circuits design industry. However, the verification of such a complex RTL design is a time consuming and costly task that is likely to become a bottleneck in the release of new architectures. Thus, finding an efficient way for debugging would speed up the verification process.

- **Idea/Solution to be investigated by the project:**

This project will focus on the debugging process of a fault-tolerant directory-based 'MSI' protocol. We will also apply Inferno++ in our debugging process and evaluate this novel tool. To start with, the robust protocol proposed in 'A Systematic Methodology to Develop Resilient Cache Coherence Protocols' can be used to implement the resilient architecture. In the process of implementing the resilient architecture, we expect to run into several bugs. Both traditional debugging methods and Inferno++ will be used to locate bugs and their efficiency will be evaluated and compared in a systematic way. In our case, the efficiency of inferno++ will be evaluated based on its ability of locating bugs. Our analysis should provide a reference for future verification process of complex designs.

- **Progress so far:**

✓ **Router - done**

A 2x2 mesh router network has been built based on BookSim router RTL model. Three cores and one directory are connected to the router network interfaces. The network is with 4 virtual channels and different passing messages are designed to pass along different virtual channels. The message passing scheme is as following:

VC0 - GetS, GetM, PutS, PutM, PutE
VC1 - Fwd_GetS, Fwd_GetM, Inv, Put_Ack
VC2 - Data, Inv_Ack
VC3 – Not Used

The router network has been tested individually. Xiangfei is mainly responsible for the part.

✓ **Data Cache Controller - done**

Since the previous design is snoopy based, to make it directory based, the input and output connected to bus was removed. And the input and output connected to Router interface was added. Specifically,

the Data Cache Controller receives the command, data, address, requester and acknowledgement from Router, and will send command, data, address to the Router.

State machine has been added to the cache controller. To realize directory based design, many intermediate states are added to the design, such as IM_AD, IM_A etc. The control signal and state transfer is determined by both core memory requests and the command from router. Zeyu manages the component.

✓ **Directory - done**

The directory controller module has one interface communicating with the global memory and one interface communicating with the router. For each line in global memory, there is one corresponding directory entry. It maintains the owner, sharer list and also the state of that line. Fundamentally, the directory controller will stall other requests until it completes the current transaction. To prevent deadlock, we have two separate input FIFOs in the directory controller module to accommodate different kinds of requests, which will mimic the operation of virtual channels. Yao designed the directory and works with any issues related.

✓ **Integration - done**

We have integrated all components, including core, directory, IMEM, DMEM and router network together. A design change to the initial design is that we separated the IDMEM and now each core has its own Instruction memory and one single shared data memory is connected to the directory. We tested the integrated design with low traffic load & store test programs and has verified its correctness with these basic operations.

✓ **Robust MESI Protocol Development - working on**

We are working on the implementation of the resilient feature of directory-based coherence protocol which is illustrated by (Aisopos & Peh, 2011). We have implemented resilient directory-based coherence protocol in mainly two situations. One situation is that when the line in directory is in state I/S, one core initiates a GetS message. In this situation, we can ensure that when any of the intermediate messages is lost, the transaction can be successfully completed. The other situation is that when the line in directory is in state M, one core initiates a GetM message. In this situation, we can ensure that when any of the intermediate messages is lost, the transaction can be successfully completed.

✓ **Debug with Inferno - working on**

Chenxi is responsible for Inferno and has tried the tool with one EECS 470 mini project. The tool is ready to be used for our project. As scheduled, we are putting inferno into use these days and continuously documenting the debugging effort we make with and without Inferno.

✓ **Debug Log - working on**

As mentioned in last checkpoint, we have a google sheet to document the debug efforts we made. Until this week, we have over 30 debug records documented in our 'Debug Log'. Each log entry has fields of:

Date - Time - Problem - Debug Person - Time Taken - Bug Source - Use Inferno or not

- **Issues/Showstoppers:**

We are at an initial stage of using Inferno and in most time it is hard to understand the output diagram generated by Inferno. It may take some time for us to get familiar with Inferno and see if it really can accelerate the debugging process.

To test the robust protocol, Xiangfei is considering about adding a control input signal to drop a valid packet in the router network time to time. For example, the 'drop' control signal can be & to the valid bit in the packet to drop it.

We believe that we succeeded in catching up the design process and are putting more and more efforts on the verification and Inferno usage side.

From now on to the next checkpoint, we will be focusing on Inferno and resilient protocol verification and see if Inferno tool give great debugging help.

Reference

1. Aisopos, K., & Peh, L.-S. (2011). A Systematic Methodology to Develop Resilient Cache Coherance Protcols. *MICRO'11*.