Assumptions

As we began the development process, we made the following assumptions for our multi-core cache coherence simulator:

1. At any time, each processor has only one outstanding request
2. The bus only support atomic transaction
3. We don't support read and write to actual data. We only concern the addresses issued by each processor

As part of this project **we plan to remove assumption 1 by enhancing the cache implementation to incorporate non-blocking semantics**.

Updated Schedule for Project Milestone

We've been working diligently to keep to the schedule. So far we've completed the following portions:

1. Study SST Core API

- We went through the SST Core documentation explaining the basic primitives components and APIs
- We also checked out the tutorials online and the simple examples given in the SST-Elements repository
- The next step was building the SST-Core and SST-Elements repository locally and executing a few examples to get hands-on and understand the process of building our own components on top of SST-Core.

2. Completed development for simulated CPU load store generator
3. Completed building a cache component

- We have already completed the basic development process of a multi-core cache.
- The cache has three ports
  - One to receive requests and transmit responses back to the processor
  - Second to submit requests to the bus arbitrator to access the bus
  - Third to transmit the actual request on the bus and receive the response
- We have a working implementation with broadcast based MSI cache coherency protocol

4. Tested the complete implementation on a trace computing the sum of an array in parallel

- Broadcast based interconnect
- Arbiter with round-robin and FIFO policies
- Multi-core cache with MSI coherence protocol

5. Understand how to gather multi-threaded memory traces using PIN tool

Below is the updated schedule for the coming 2 weeks in half-week granularity (between milestone and project deadline).

Week Number	Checkpoint	Assignee	Status

Week Number	Checkpoint	Assignee	Status
0.5	Complete implementation of cache component	Tanay	Done
1.0	Complete implementation of bus and arbitrator	Xuan	Done
1.5	Enhance implementation of cache component for MESI protocol and additional statistics	Tanay	In progress
1.75	Enhance implementation to incorporate non-blocking cache semantics and additional statistics	Both	In progress
2.0	Devise characteristic multi-threaded programs to stress test simulator and study workload patterns	Both	In progress
2.0	Generate characteristic cache traces using Pintool	Both	In progress
2.25	Perform analysis and gather data using our simulator	Both	To Do
2.5	Complete the extended implementation of directory component	Xuan	To Do
2.75	Incoporate changes to cache and bus for directory based protocol	Tanay	To Do
3.0	Work on report and poster session prep	Both	To Do

Updated Goals and Deliverables

We have been sticking farily well to the planned schedule and targeted development goals. We already have a working multi-core cache coherency simulator with the following features:

1. Variable cache block size
2. Variable total cache size
3. Configurable associativity
4. Configurable replacement policy

- Round robin
- LRU
- MRU

5. Configurable cache coherence protocol

- MSI
- MESI (in progress)

6. Configurable arbitration policy:

- Round robin
- FIFO

In the upcoming weeks we plan to enhance the implementation to inocporate non-blocking cache semnantics after discussing it and taking feedback from Professor Skarlatos. Post that we would carry out performance studies using our simulator to gather insight into the behaviour of shared memory parallel programs with different communication patterns and plan to reproduce what we learned in class regarding artifactual communication with actual data and statistics collected using our simulator (XTSim).

Since our poster session is on December 9th, leaving us with less than 10 days, we are skeptical of achieving our 125% goal but will try our best to keep in line with original GOALs and deliverables.

DEMO

We aim to have an interactive demo showing the capabilities of our simulator. We plan to present insights such as reporting the following statistics -

1. Different types of cache statistics -

- Miss rate
- Number of invalidations due to coherecncy protocol

2. Bus Traffic Classficiation -

- Memory traffic: Request served directly from memory
- Coherency traffic: Request served from one of the caches due to sharing

3. Latency of different cache events

- Read
- Write

4. Effect of cache block size on performance producing a plot of miss rate vs cache block size
5. Study the effect of different types of memory access patterns and sharing (such as Ocean simulation, stencil etc) to gather insights from the simulator
6. Scalability of cache coherency implementation styles (125%)

- snoop-based
- directory based

We aim to present our deliverables in terms of graphs

1. Miss rate vs Programs with different access patterns
2. Number of invalidations vs Programs with different access patterns
3. Miss rate vs cache block size
4. Miss rate vs total cache size
5. Coherence Traffic vs Programs with different access patterns
6. Memory Traffic vs Programs with different access patterns

Oustanding Concerns

We primarily have the following tasks remaining

1. Enhance cache implementation to incorporate non-blocking semantics
2. Generate test plan for carrying out performance study using our simulator (XTSim)

We are majorly concerned with the remaining time we have since we're doing an early poster session. We are confident of achieving the 100% goal but are not completely sure of completing the 125% goal of enhancing the implementation to incorporate directory based coherency protocol.