Lab Assignment 5: Frequently Asked Questions

1. **General Simulator Issues and Debugging Questions**

   (a) Why do I get this error from executing the following command:

   ```
   scons PROTOCOL=MSI CPU_MODELS=TimingSimpleCPU build/ALPHA/gem5.opt
   scons: Reading SConscript files ... Checking for leading underscore in global variables...no Checking for C header file Python.h... no Error: can't find Python.h header in ['/usr/include/python2.7']
   ```

   Answer: Make sure to exactly follow the instructions on what to add to your environment variables in section 3.1 on page 5 and 6 of the handout. Instructions are given assuming you are using bash. If not, you must adapt the commands accordingly.

   Specifically, `echo $PATH` should start with:

   `/cad2/ece552f/gem5_dependencies/bin:`

   and `echo $LD_LIBRARY_PATH` should print:

   `/cad2/ece552f/gem5_dependencies/lib:`

   (b) What does this line mean: Ruby Tester "hack: be nice to actually delete the event here"?

   Answer: You don't need to be concerned about this message.

   It has to do with the internals of the simulator. When the simulator is started, it schedules a `limit_event`, which is the upper limit of simulation cycles. Running the tests from the lab handout is usually shorter than this upper limit and that's why the simulation script is complaining that there is still an event scheduled when we reach the end of the test.

   (c) Is it possible to isolate debugging events? For example, I have a deadlock that appears 113 million cycles into the program. I'm looking for a way to isolate the problem, since dumping the entire ProtocolTrace to a file with 113M cycles is simply not feasible.

   Answer: Have you looked into these options yet?

   ```
   Trace Options
   -------------
   --trace-start=TIME      Start tracing at TIME (must be in ticks)
   --trace-file=FILE       Sets the output file for tracing [Default: cout]
   --trace-ignore=EXPR     Ignore EXPR sim objects
   ```

   (d) "`Action/check failure`"; what does this message mean?

   Answer: This error refers to a data inconsistency for a specific address. There's a brief mention of this error in Section 5.4 of your handout.

   One way to think about it is that at some point a byte(s) of a cache block got modified with a store request, but then this modification was not properly propagated. It could be that you didn't copy the correct information to the cache, or you didn't properly write it back to memory, or you didn't propagate it to another requestor etc.

   (e) I have added a transition and I am now getting a seg fault. How should we go about debugging this (as gdb doesn't work)?

   Answer: The first approach to debugging should be `DPRINTF` statements, which you can display using the RubySlicc debug flag.

If that doesn't help, you can actually use gdb with gem5. It is a little more involved, as you have to compile a separate debug version of the simulator, but it is possible.

There's a description how to use gdb with gem5 here: `http://www.m5sim.org/Debugger_Based_Debugging`

You can compile the debug version using the following command:

```
scons PROTOCOL=MSI CPU_MODELS=TimingSimpleCPU build/ALPHA/gem5.debug
```

(f) How is the auto marker going to know that our program is functional? We can complete all the example test cases and get message "`Ruby Tester Completed.`" Does this mean that the program is correct?

Answer: We will run a couple of additional tests in addition to the ones mentioned in the handout. So, you should try to expand your test coverage a little bit as well.

Successfully completing all the tests mentioned in the handout is definitely a good thing though and should get you most of the marks.

(g) What should the `--debug-flag` value be to enable the printing of the `DPRINTF` messages?

Answer: Assuming you categorized everything as RubySlicc, use: `--debug-flag=RubySlicc`. If you want to turn on multiple debug flags, delimit them with commas. Example: `--debug-flag=ProtocolTrace,RubySlicc,RubyCache`

(h) Is it OK if we add a couple APPEND statements in some of our actions? This changes what the trace looks like as it adds some extra prints to the trace output. Will this impact the marking of our lab?

Answer: Yes, its alright to add append statements; it will not impact the auto-marker.

(i) Do I have to re-build the simulator with the following command every time I make changes to the MSI-cache.sm and MSI-dir.sm?

```
scons PROTOCOL=MSI CPU_MODELS=TimingSimpleCPU build/ALPHA/gem5.opt
```

Answer: Yes, you need to recompile the simulator every time you change those files.

SLICC is not an executable format. SLICC files are being parsed into C++ files during the compilation process, which are then compiled into the simulator.

(j) Will the test keep running after deadlock has occurred? For example, when I run the test with `-n 16 -l 1000000` with the `-debug-flag=ProtocolTrace`, the test keeps running and I can see the number of cycles going over 2000000. Is this test very long or does my code have a deadlock condition?

Answer: The flag `-l 1000000` indicates that there will be 1000000 loads issued. There's no indication of how many cycles this will take. If your simulation doesn't tell you that there is a deadlock, you can assume that there is none.

(k) I get an error when I try to run the test with the following config parameters:

```
./build/ALPHA/gem5.opt ./configs/example/ruby_random_test.py -n 2 --l1d_size
8192 --l1d_assoc 4
```

Answer: Please try either 8192B or 8kB instead of simply 8192.

(l) I get Deadlock detected for processor 0 in the test run with the following configuration: `-n 2 --l1d_size 8192B --l1d_assoc 4`

I manually went through all the state transitions in L1Cache0 and all the blocks in the cache are in a stable state (i.e. either M or I). Can a deadlock exist for processor even if all the blocks are in stable state?

Answer: The deadlock detection in this case is not a real deadlock detection. As mentioned in section 5.3 of the handout, the "deadlock detected" error is triggered whenever a processor has not made any progress in a fixed number of cycles. This could be the case for a number of reasons. For example, if you forget to pop messages off a queue, then no more messages will be processed, which will lead to starvation and this is detected as a "deadlock". So, in short, yes, given this particular deadlock detection algorithm, you can see a "deadlock" even if all blocks are in a stable state.

2. **State and Event Questions**

(a) Is `Event:PUTM` ever triggered in MSI-dir.sm?

Answer: No, `PUTM` and `PUT_Ack` are never triggered - you can safely ignore those. `PUTM_Owner` and `PUTM_NotOwner` cover everything.

(b) Consider the following situation: Assume `addressXYZ` is in a shared state among CPUs and it currently contains some data labeled old_data. On the same processor the following two instruction are issued:

```
store new_data, addressXYZ
load register, addressXYZ
```

According to the table 8.1, the first store instruction will cause `addressXYZ` to transition to state `SM_AD` where we are waiting for write permissions on the block. If while we are in this state we begin processing the load above, the state table indicates this is a cache hit and upon this logic we will return the data named old_data, not new_data. New data will not be written until we get write permissions (at the point of entering the M state). Shouldn't we preserve the order of instructions (ie. not allow the load to race ahead the store)?

Answer: The `store new_data, addressXYZ` instruction will trigger a transition to state `SM_AD`, where it will wait for data and acks to return. Load requests during that time will result in a "cache" hit. If this "cache" hit would indeed be served by the cache, it would lead to a race condition, as you mentioned. However, the data is not being served by the cache, but by the TBE, which holds 'new_data'.

Upon the store instruction, its data is written into the TBE and subsequent loads will be served by the TBE, as long as the data resides in there. In MSI-cache.sm it's not obvious where the data is served from, so your question was more than reasonable. There is a small hint how this is handled when you look at `in_port(mandatoryQueue_in, RubyRequest, mandatoryQueue, desc="...")`. In the else branch you can see

```
trigger(mandatory_request_type_to_event(in_msg.Type), in_msg.LineAddress,
cache_entry, TBEs[in_msg.LineAddress]);
```

This calls the function that ultimately does the cache lookup and it checks the TBE, as well as the cache.

(c) Are we allowed to change some of the state transitions that are described in Tables 8.1/8.2 in the required reading? Are alternatives/optimizations permitted?

Answer: Yes, you can optimize; however it is not recommended. Even small changes can have major implications on the coherence protocol and can introduce substantial new complexity. Think very carefully before you decide to deviate from the given protocol.

(d) Why do we need to read data before going from the Invalid state to the Modified state (e.g., transient state `IM_AD`)?

Answer: When a cache block is in Modified state, that particular cache is the owner of the entire block, which means it has the only valid copy in the system. Subsequent reads and writes to any address in that block will be cache hits and that's why we need to bring in the rest of the data of that block, aside from the specific address that was written.

(e) Is it acceptable to do nothing on a state transition between two transient states?

Answer: You should be doing some action in every transition.

(f) How can the Invalid state in Directory be receiving `PUT` requests? If the directory has a block in the invalid state how would another cache issue a `PUT` request to writeback the block?

Answer: Here's an example to help answer this question. If there are 2 cores and one core (C2) is the owner of blk A in state M and the directory has blk A as M and the owner as C2. Say C1 wants to write to blk A and at the same time at C2 a replacement was invoked for blk A as well. C1 says to the directory controller that it wants to write to blk A, the directory sets C1 as the owner and forwards the `GetM` to C2 since it has the most up to date value for blk A. At this stage in C2, the state switched from M to `MI_A` because of the concurrent replacement. Once it receives the forwarded `GETM`, originally from C1, it sends its data to C1 and C1 state for blk A switches to `M`. Then lets say the `PUTM` invoked by the replacement in C2 is delayed for some reason and C1 blk A was replaced, now the directory has Inv as the state for blk A and now the delayed `PUTM` arrives!!

(g) How do I use the TBE in the directory controller? Specifically, what does `vv_allocateTBEFromRequestNet` do?

Answer: If you're unsure about how a particular action works, you can look at its definition. For example, the `vv_allocateTBEFromRequestNet` has the following definition:

```
action(vv_allocateTBEFromRequestNet, "\v", desc="Allocate TBE") {
    peek(requestQueue_in, RequestMsg) {
      TBEs.allocate(address);
      set_tbe(TBEs[address]);
      tbe.DataBlk := in_msg.DataBlk;
    }
  }
```

Here you can see that when a message is received from the requestQueue, a TBE entry is allocated with the address from that incoming message. Afterwards, the data from the incoming message is copied to the TBE entry at that address.

In general, the TBE is used to temporarily store data and state for outstanding requests during transient states.

(h) Do messages between the directory and a core arrive in order? Say the directory sends a `INV` and a `WB_ACK` to Core1. Is it possible for Core1 to receive the `WB_ACK` and then `INV`?

Answer: It's possible for messages to arrive out-of-order if they're handled by different queues. When messages are transmitted using the same queue, they must arrive in order. We basically assume a single hop bus as the communication substrate, so there cannot be any reordering during transmission.

(i) Do we have to use all the actions available? We've completed our implementation and see that all the cases for tables 8.1 and 8.2 including the states we have to add are

covered and we tested it and no faults have been found, but we didn't use all the actions available for the directory. Do we have to use all these actions in order to get full marks?

Answer: You probably won't need all the actions.

(j) What is `Data_from_Dir_Ack_Cnt_Last` and in what scenario will this message be sent?

Answer: That event happens whenever the cache transitions to modified state and is waiting for data and acks to arrive. If data (along with the ack count) arrives before all other acks arrive ("Data from Dir (ack>0)"), it transitions to a state where it only waits for further acks. If all acks arrive before the data arrives it counts as "Data from Dir (ack=0)" and transitions into modified state right away.

3. **Report Requirements**

(a) The lab handout says to include in our report a "small table documenting all your protocol modifications with respect to Tables 8.1 and 8.2." Do we need to include every action that occurs at every state transition in our code? For example, must we mention every time we allocate/deallocate a TBE entry or an L1 cache entry?

Answer: You don't need to include every action in detail. It's important that all additional states and transitions show in the table and other than that a high-level description similar to what tables 8.1 and 8.2 provide is enough.