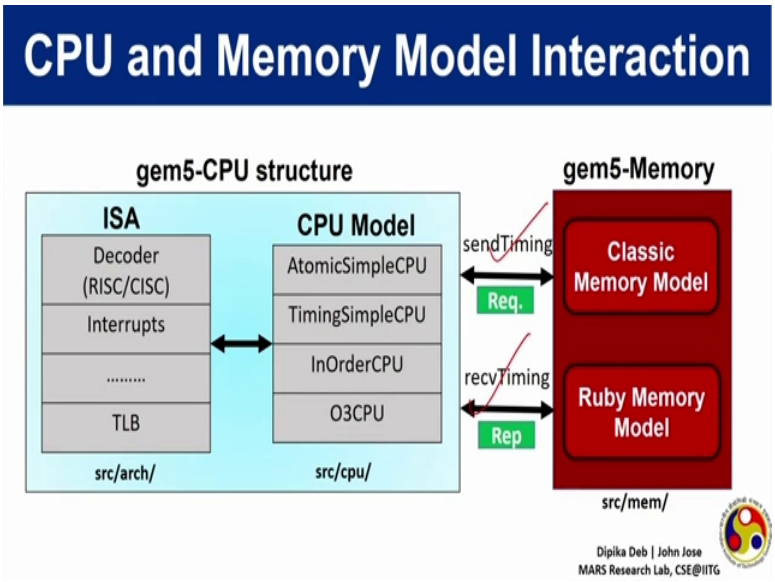
Advanced Computer Architecture
Dipika Deb
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati

Tutorial No. – 8
gem5 – Cache Optimization

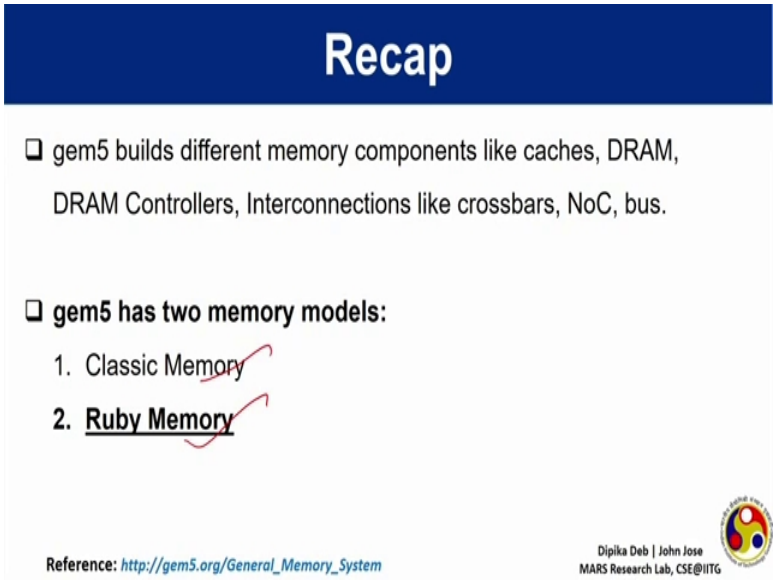
Welcome to the advanced computer architecture course. I am Dipika Deb and in this video I will explain how memory models are implemented in gem5. I will also provide some insights on how to solve the second assignment on gem5.

(Refer Slide Time: 00:31)



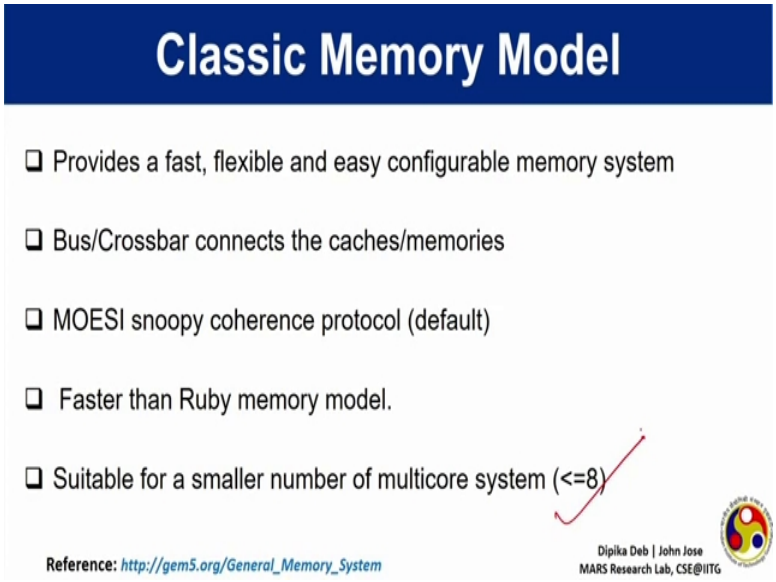
As a quick recap of the previous video on gem5, the figure shows how a CPU and memory interacts in gem5. In gem5 the CPU-memory interaction works in a master slave fashion, where the CPU is the master and it requests for something from the memory. The memory executes the request and replies back to the CPU. This is done using the send timing and receive timing signals. These signals are implemented in gem5 as packets. In this video, we will deal with the memory models only.

(Refer Slide Time: 01:19)



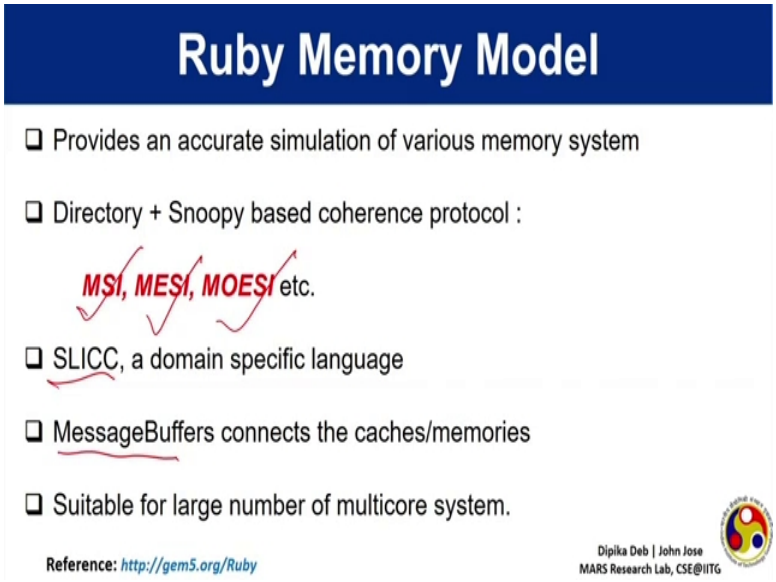
Gem5 builds different memory components such as caches, DRAM, interconnects, etc. There are two main memory models available in gem5, classic memory and ruby memory. Throughout our assignment we will use ruby memory.

(Refer Slide Time: 01:38)



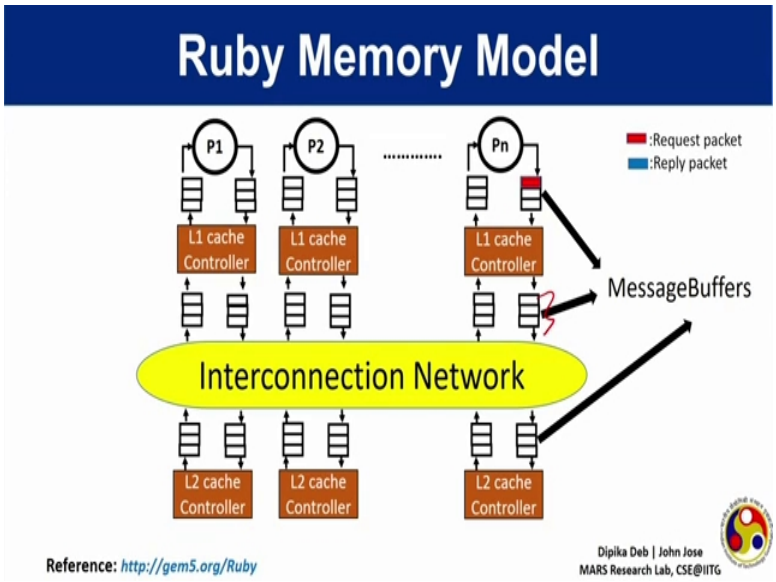
The classic memory model provides a fast, flexible, and easy configurable memory system. It uses a bus or crossbars to connect the caches and memories in the system. For multiple processors implemented in classic memory system, MOESI snoopy coherence protocol is implemented. This memory model is faster than the ruby memory model, but it is suitable only for a small number of multicore processors, generally less than 8 cores in a system.

(Refer Slide Time: 02:11)



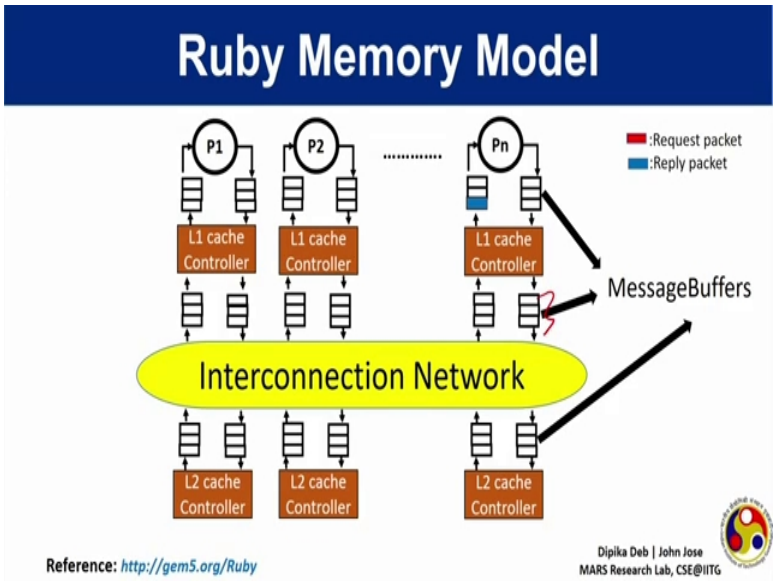
On the other hand, ruby memory model provides an accurate and detailed simulation of various memory systems. It also has both snoopy and directory based coherence protocols, that is, MSI, MESI, MOESI and so on. The cache controllers in ruby model are implemented using a domain specific language known as SLICC, and the caches and memories are connected together using a spatial buffer known as message buffers. Ruby is suitable for a larger multicore system unlike classic memory. Hence, we will use ruby memory model in our gem5 cores.

(Refer Slide Time: 02:56)



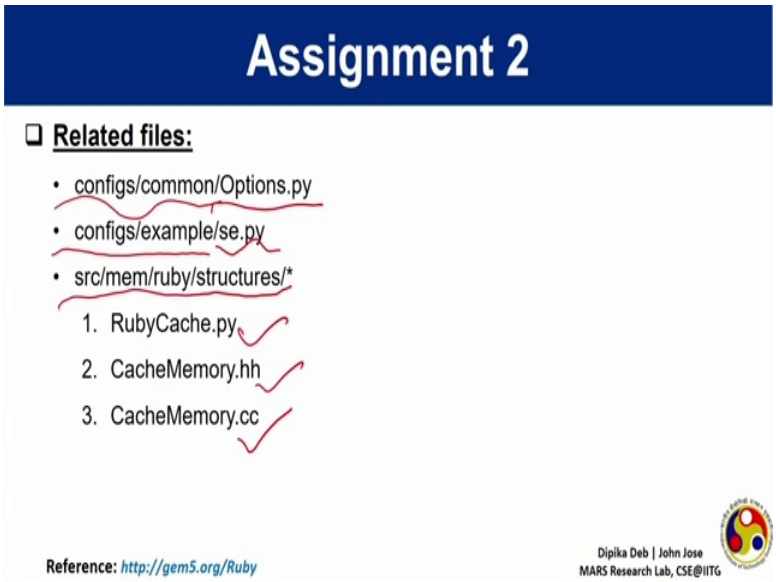
The figure shows an overview of the ruby memory model. Here the processors are connected with L1 cache controllers with message buffers. Similarly, the L1 cache controllers are connected with the interconnection backbone using message buffers. The memory components communicate with each other by generating packets such as request and reply.

(Refer Slide Time: 03:30)



For example, the processor generates a request to the L1 cache, from the L1 cache if it is a hit, it replies back to the core. In the previous video, I have already shown how to configure a unicore processor with different L1 cache and L2 cache configurations. The second assignment on gem5 will contain such modifications which you can either enter through the command line or Options.py file.

(Refer Slide Time: 03:52)



So these are the related files for assignment 2. You can find this Options.py in this directory. The simulation script can be found in this directory. You may also require these three files which can be found in src/mem/ruby/structures directory. This actually implements the caches in ruby module. These three files in association with Options.py file are of prime importance in solving assignment 2. So, by this we have come to the end of today's video.

For any further queries, you can write us in the discussion forum of our course web page. Thank you.