Architectural Simulation Using gem5

Dipika Deb, Ph.D Scholar
Department of Computer Science and Engineering
Indian Institute of Technology - Guwahati

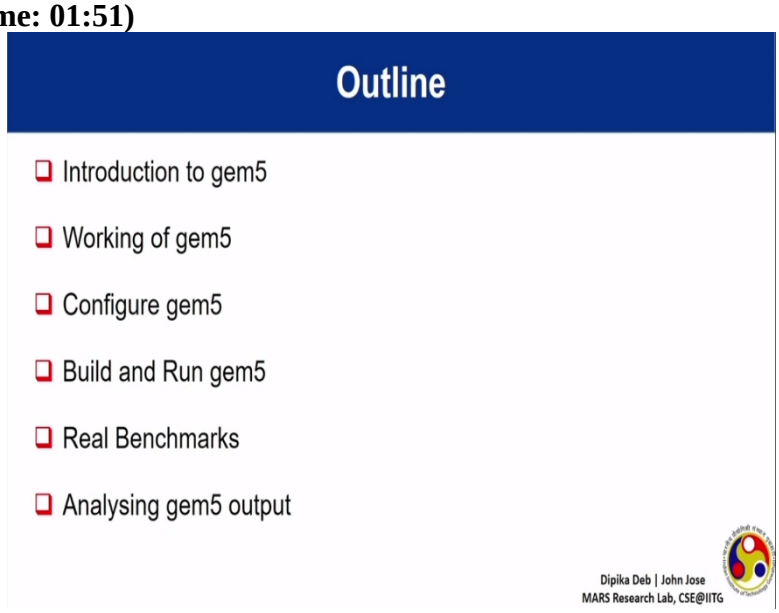
Tutorial 4
MARS Research Lab

Hello, everyone. Welcome to the advanced computer architecture course. I am Dipika Deb, a Ph.D. scholar in the Department of Computer Science and Engineering, IIT, Guwahati. I work under the supervision of Dr. John Jose and my research area of interest is multi core architecture with special focus on prefetching and compression. This is the first video lecture on gem5 session. And we have planned 2 other such videos.

The first session is an introductory session on the gem5 simulator, while the second and the third deals with code optimization and memory optimization. So, with this, I start today's session. gem5 is a full system simulator that is extensively used by the computer architecture community. This is because gem5 simulates a real hardware, where one can adopt any modifications for experimental purpose. The need for such a modification also arises from the fact that fabricating all modifications on a real hardware is not actually possible.

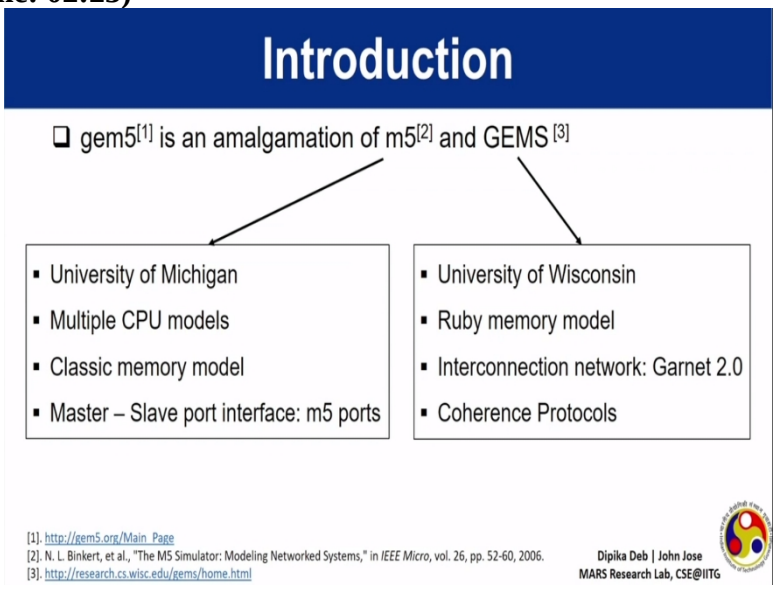
This is because the fabrication cost is too high and also quite time consuming. Due to this reason, an architectural simulator, such as gem5 is used for the experimental purpose.

(Refer Slide Time: 01:51)



This is the outline of today's presentation, I will introduce to the readers the full system simulator that is gem5, followed by how it works. Then I will show you how to configure and build gem5 in your system. I will also show the real benchmarks that can also be used for evaluating the system performance in gem5. Finally, I will explain the output statistics that are generated from the simulated machine.

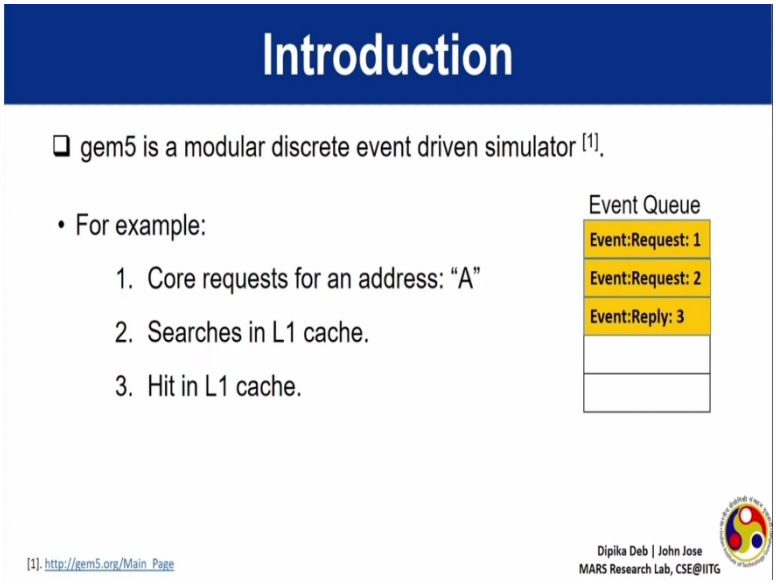
(Refer Slide Time: 02:23)



Gem5 is an amalgamation of m5 and GEMS. m5 is built by University of Michigan with multiple CPU models and instruction set architectures. It also has a classic memory model. The CPU and memory in m5 interacts with each other as master and slave using the available m5 ports. On the other hand, GEMS is built by University of Wisconsin with Ruby memory model and it also has a flexible interconnection backbone that is Garnet 2.0.

It also models different types of coherence protocols. In the previous slide, as I have shown, the CPU models available in gem5 will be described in details in the second video of gem5, while the memory models will be described in the third lecture of the gem5. gem5 is a modular discrete event driven simulator, that is all the actions in the system are represented as events that are time stamped. The events are stored in an event queue and a time is attached with an event. As per the time an event get scheduled.

(Refer Slide Time: 03:24)

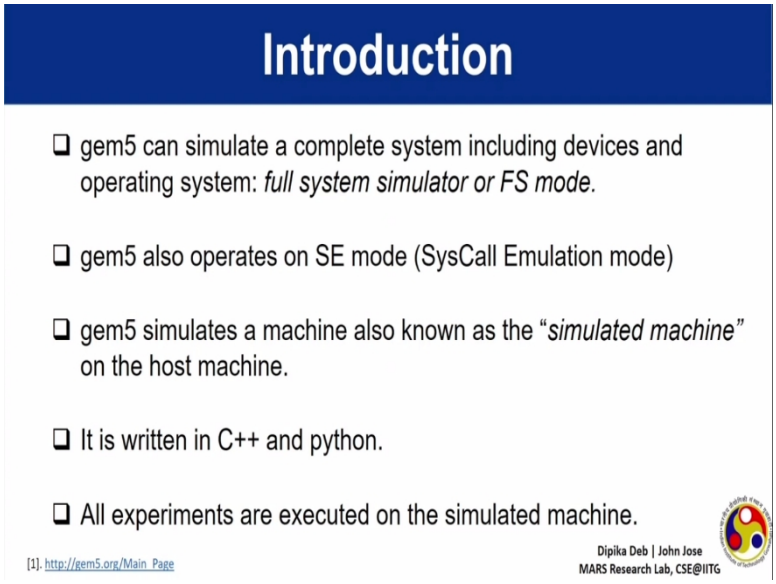


For example, let us suppose that this is an event queue and a core request for an address A. The request is processed as an event that is entered into the event queue. Assuming that the current clock cycle is 1, the event is time stamped with 1. As the event gets dequeued the normal procedure is whenever a core generates an address it is initially searched in the L1 cache.

Hence searching in the L1 cache is another event and that is time stamped with clock cycle 2 and that is entered in the into the event queue if the L1 cache success, turns out to be a hit in the cache, the L1 cache controller sends the block back to the core. This is then generates another event which is known as a reply event with timestamp 2, this event is also inserted into the event queue.

Kindly note that this is a representative example to describe an event driven simulation and this is how the simulation occurs in gem5. Gem5 is also known as a full system simulator.

(Refer Slide Time: 04:59)

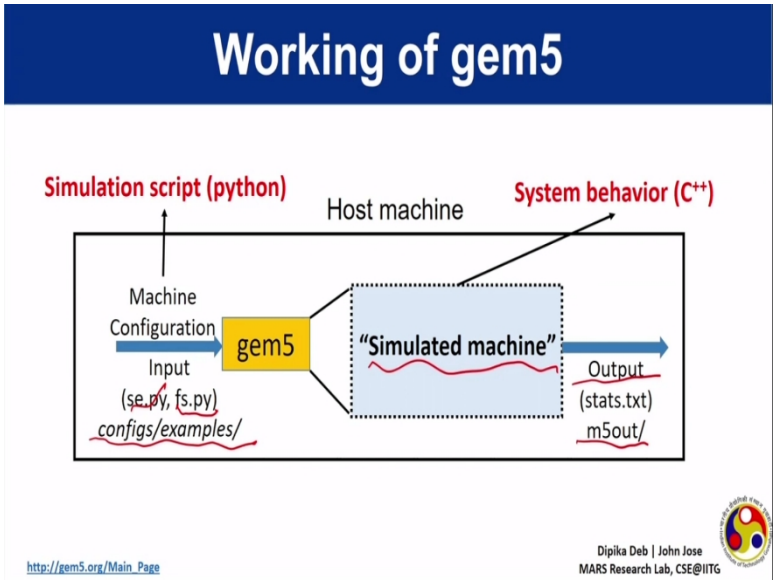


By full systems simulation I mean it executes both user level codes and kernel codes and hence models a complete system including devices and operating system. Gem5 also operates on SE mode or this is also known as Syscall emulation mode. In SE mode, it emulates most of the system level services that is kernal codes and just executes the user code. Hence this mode does not model any operating system or any devices.

Gem5 simulates a machine which we call a simulated machine and the machine on which gem5 is running is known as the host machine. Remember the term host machine and simulated machine as I will use this 2 terms throughout the video lecture. Gem5 is written in C++ and Python and all the operations and experiments are performed on the simulated machine.

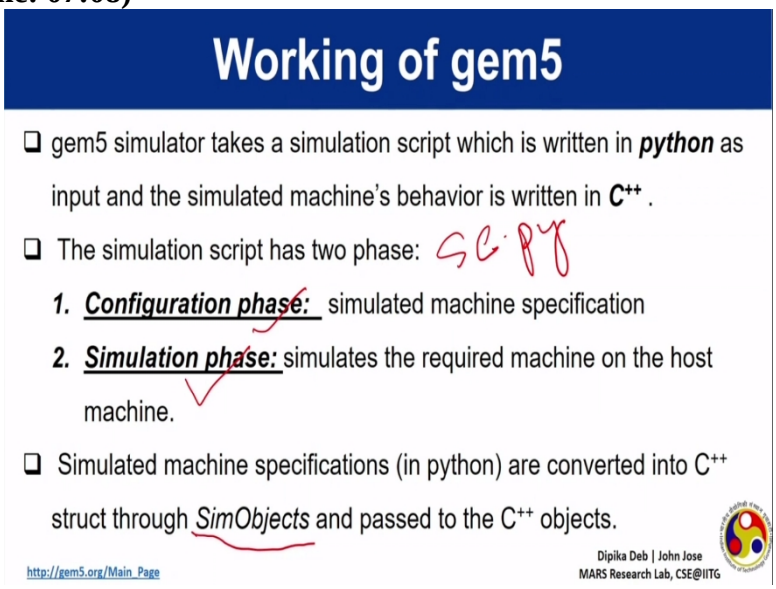
The figure shows a diagram of how gem5 builds a simulated machine on installing gem5 on a host machine it takes an input parameter. This input parameter is actually a simulation script which is written in python. This script contains configuration of the simulated machine that is to be by gem5. Few such input simulation scripts are present in this path. One such script is se.py and the next one is fs.py. I will explain more about the simulation scripts in the next slide.

(Refer Slide Time: 06:02)



Gem5 on taking the input generates and builds a simulated machine the behaviour of the simulated machine can be observed as an output file in the m5 out directory inside gem5 the behaviour of this machine is implemented in C++. As I have already mentioned, that gem5 simulator takes a simulation script which is written in Python as input that is se.py and the simulated machines behaviour is written in C++. Now coming to the simulation script, the Python script se.py contains 2 parts.

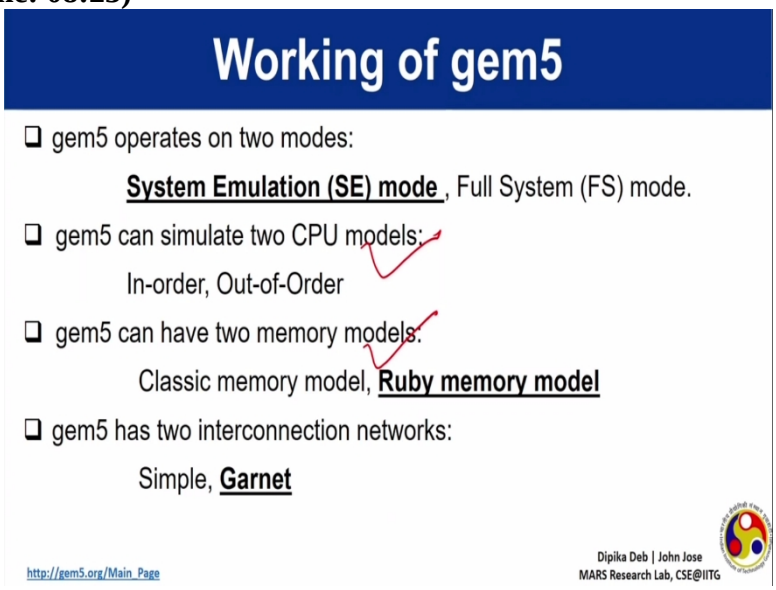
(Refer Slide Time: 07:08)



The first part is configuration phase. This says what the configuration of the simulated machine. And the next phase is simulation phase. This phase mentions how to build the machine. The configuration from the python script are then converted into C++ structures through Sim Objects and these are then passed to the C++ objects which models the system behaviour. This Sim Objects generally represent some physical components such as cache, main memory, or DRAM.

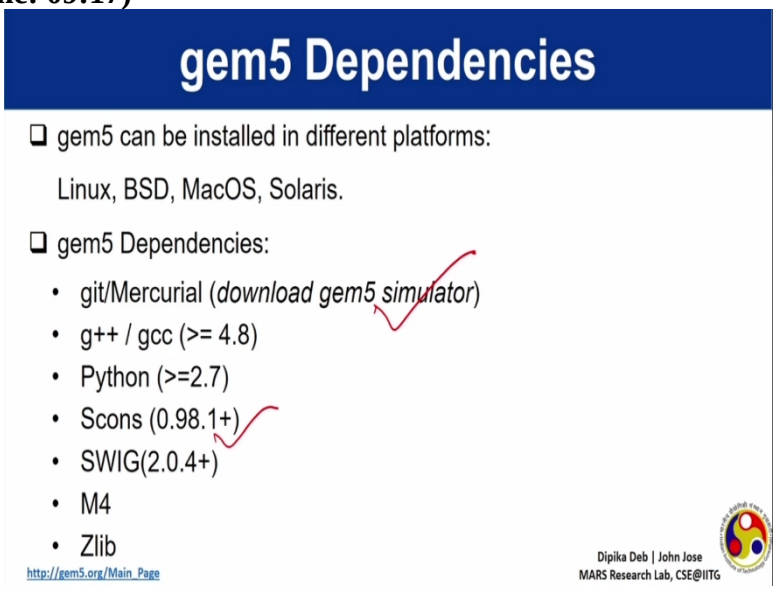
All these components are represented using different Sim Objects. I will explain the whole procedure in greater detail in the subsequent slides. Now moving forward, gem5 operates on 2 modes. SE mode and full system mode. Gem5 broadly simulates 2 CPU models in order CPU and out of order CPU. And it has 2 memory models classic memory and Ruby memory. Gem5 has 2 interconnection networks simple and Garnet. In this course, we will mostly handle with CPU models and memory models.

(Refer Slide Time: 08:23)



Throughout our assignment, we will use the system emulation mode or SE mode. We will use both the CPU models in order as well as out of order, which I will explain in greater detail in the second video lecture of gem5. And we will use Ruby memory model along with the garnet interconnection networks.

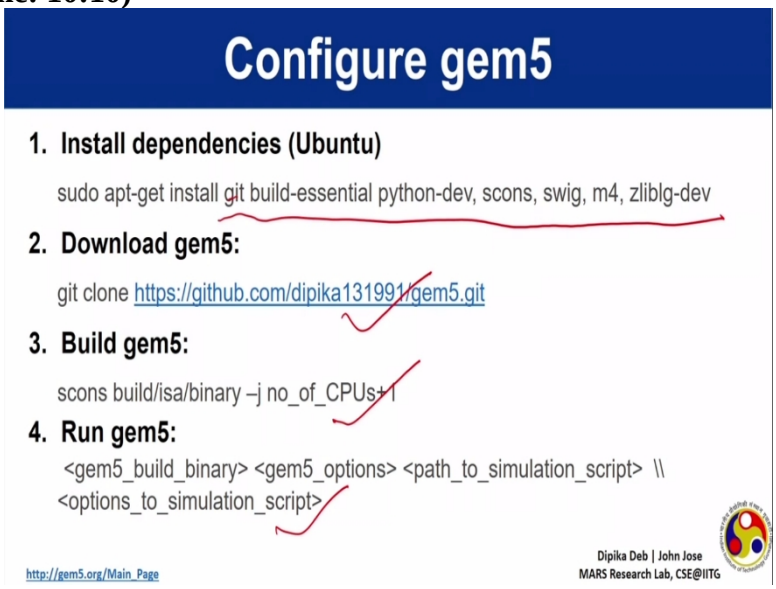
(Refer Slide Time: 09:17)



Gem5 is dependent on few packages. First gem5 can be installed in different plat forms. It can be installed in Linux, BSD Mac OS, Solaris, and it is dependent on the following packages: git on Mercurial, This is used to clone or download gem5. The G ++ or GCC compiler is used to build the C++ system files. Python is required to interpret the simulation scripts and scons is required to build the complete gem5 simulator.

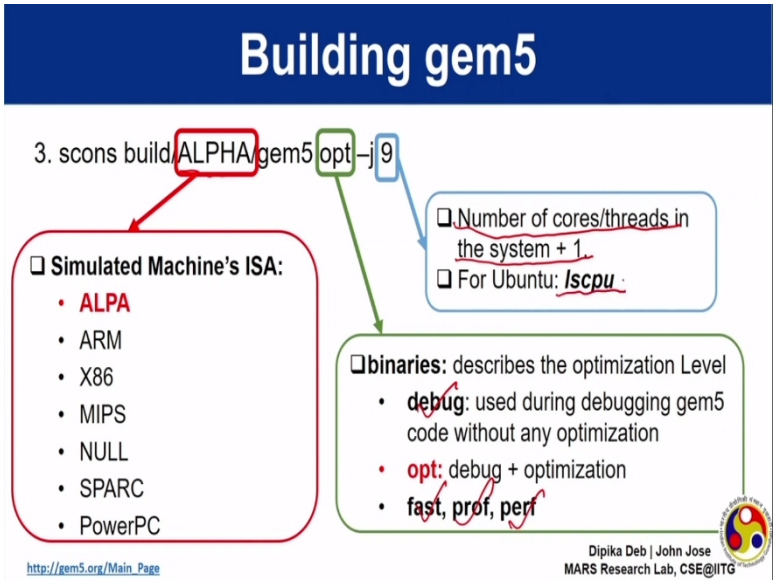
And finally swig M4 Zlib at the extra packages that are required by gem5 for installation. Now let us see how do we configure gem5 in our system.

(Refer Slide Time: 10:10)



These are the commands to install the dependencies type the following command in the terminal. I will use an Ubuntu system to show a demo. You can download gem5 using this command. We can then build gem5 using this command and to run gem5 we will use this command. I will explain the build and run command in greater details.

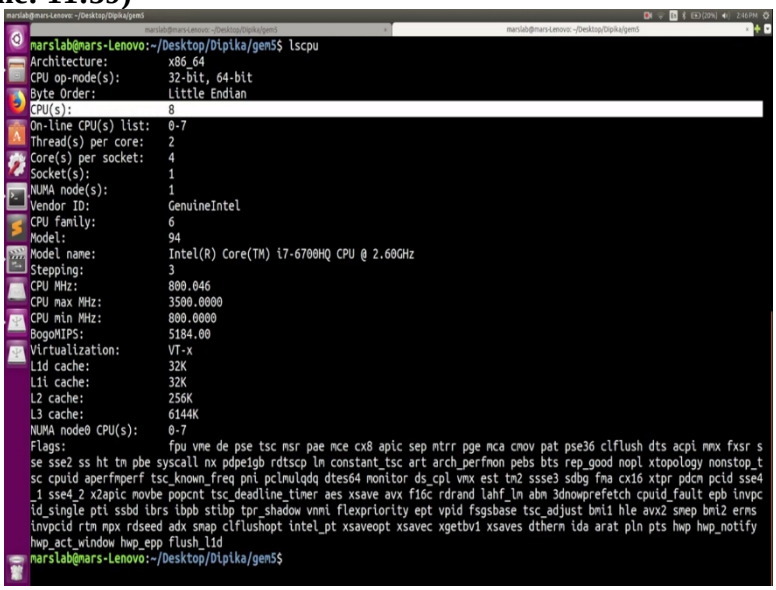
(Refer Slide Time: 10:41)



This is the build command where the instruction set architecture used is alpha. Throughout our simulations and assignment, we will use the alpha instruction at architecture. So this is the simulated machines instruction set architecture. And in gem5 there are several other ISA available that is alpha ARM, x86, MIPS, null, spark and power PC. Gem5.opt is the binary that is commonly used during working with gem5, it is fast and it has almost all of the debug features.

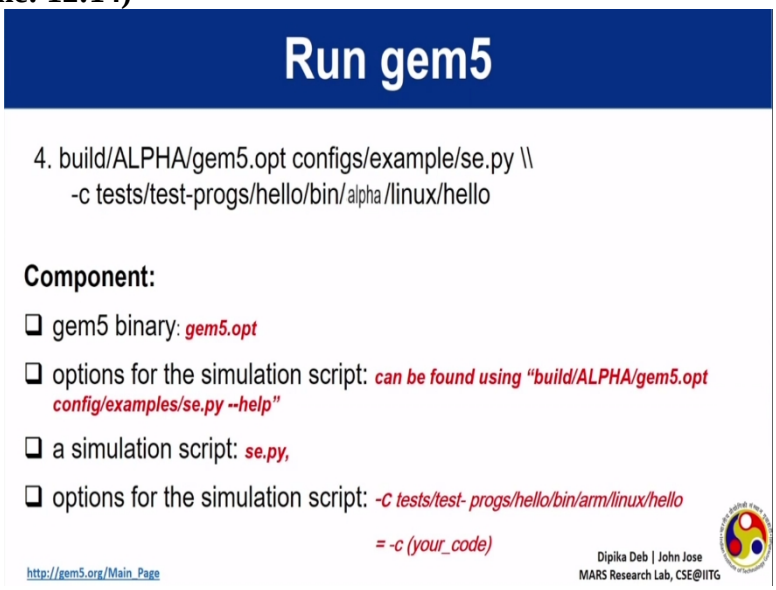
There are other types of binaries available such as perf, prof, fast and gem5.debug. Next, - j 9 this 9 indicates the number of cores or threads in the system + 1. For Ubuntu to users. You can type this command to check the number of course that you have in your system.

(Refer Slide Time: 11:59)



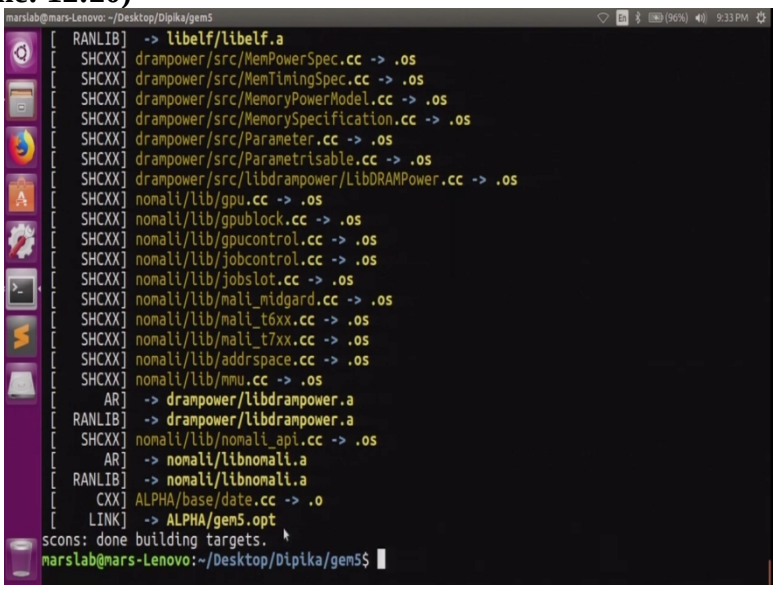
After building gem5, you will notice that inside the gem5 directory a new directory named build is automatically generated. Now, we will run gem5 using the following command.

(Refer Slide Time: 12:14)



Now I will show a demo of the complete process that is to be carried out in your system to build a gem5 system, we will type scons build we will use the alpha architecture gem5.opt binary and the number of cores that I have in my system is 8.

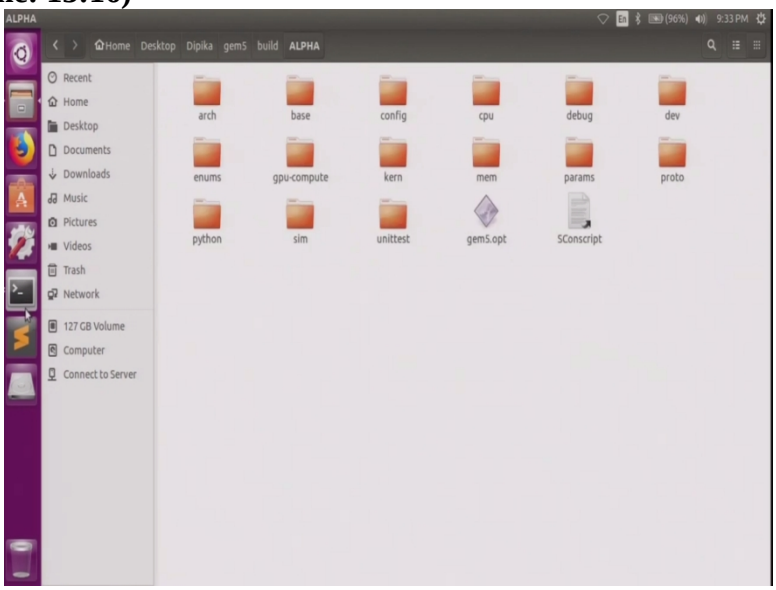
(Refer Slide Time: 12:20)



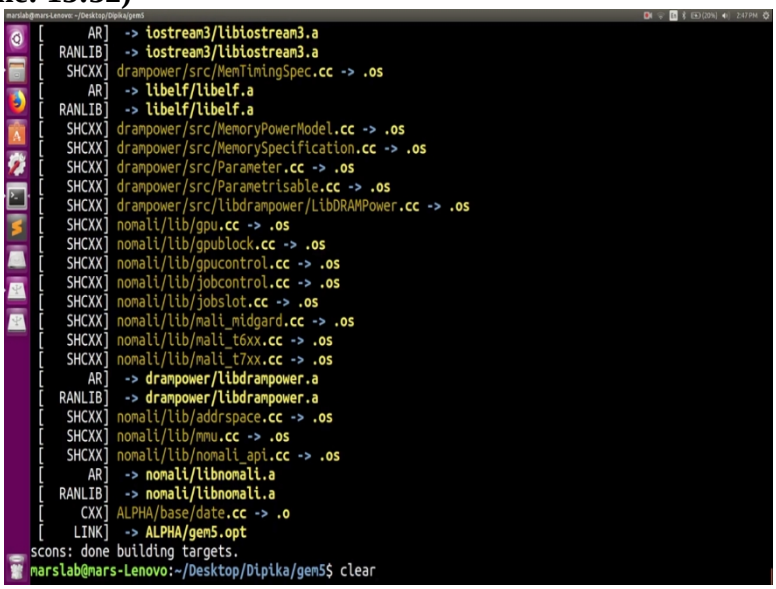
So I will use 8 + 1 that is 9 as the number of threads in my system. This process requires quite a time around 10 minutes to perform the complete building of gem5. So, at this stage, you can see that the gem5.opt binary has been created. If you look inside the folder you will find that inside gem5 a build directories build. Inside the build directory you will find the

alpha architecture that has been build using this scons command. Now, we will run a Hello program in the gem5 build directory.

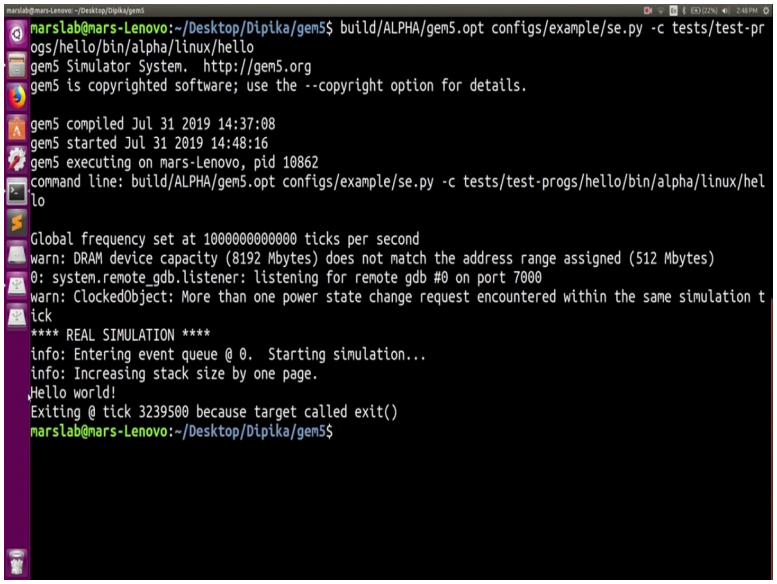
(Refer Slide Time: 13:16)



(Refer Slide Time: 13:32)

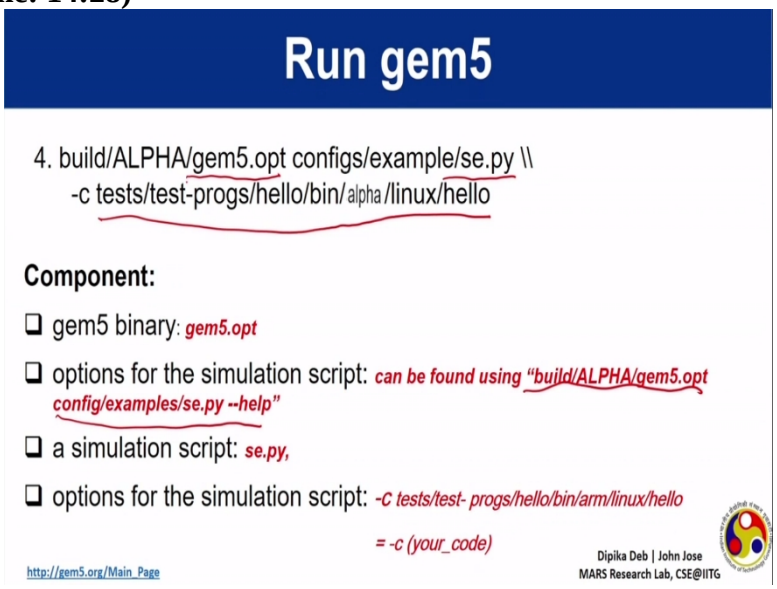


(Refer Slide Time: 13:34)

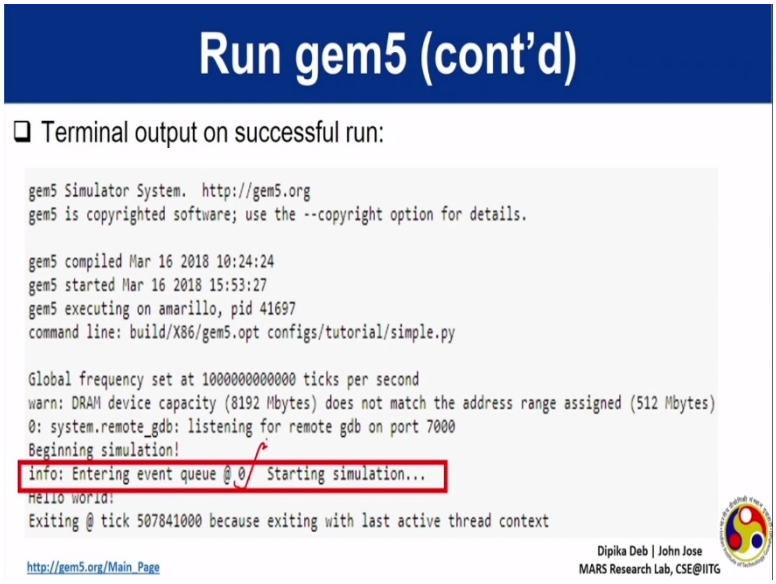


So for this we will type this command. Here the gem5.opt is the gem5 binary. The simulation script that we use is se.py and the options for the simulation script can be found by typing this command in the terminal and - C provides the option for this simulation script. This can be any C code or this can be also any real benchmark. So after the successful run, you will find this as the output. This is a Hello World program and if this hello world prints it means the run was successful.

(Refer Slide Time: 14:28)

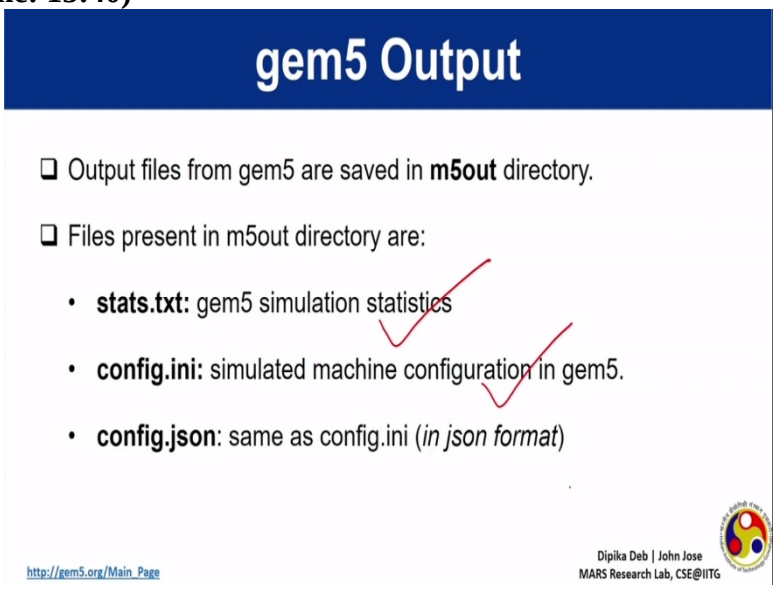


(Refer Slide Time: 14:55)



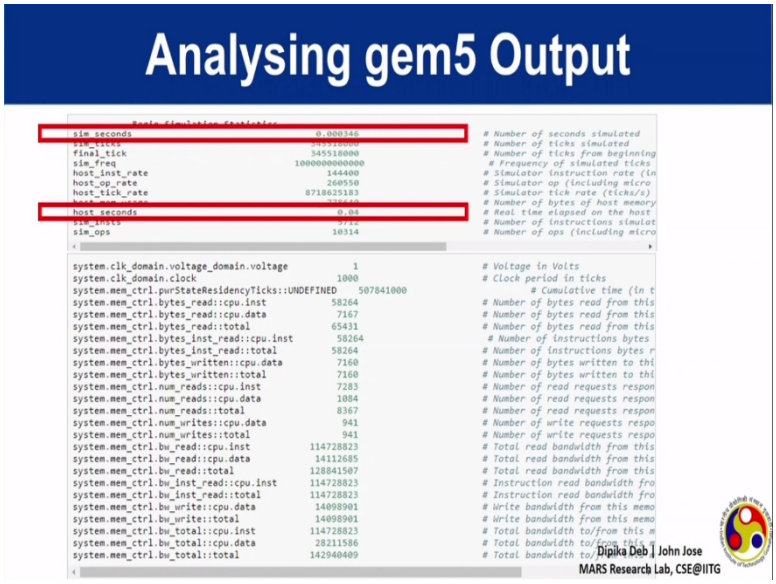
You can also notice here is that it shows an entering event queue. This I have already explained that since gem5 is an event driven simulator it has an event queue where all the events are queued and de-queued as the simulation process. This line indicates that the first event was queued in the event queue at timestamp zero. So now coming to the output of the simulated machine, it can be found in the gem5/m5out directory. Next I will discuss about the output of the target machine.

(Refer Slide Time: 15:40)



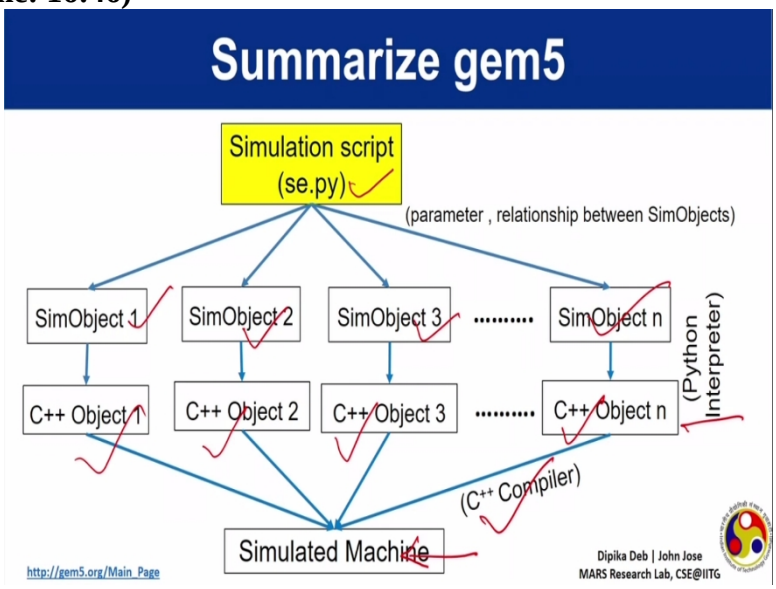
Text file contains the statistics of the output files. Inside gem5 out directory the config.ini contains the machine configuration of the simulated machine. The config.Json is same as the config.ini, but it is in JSON format. So in step stop txt file looks somewhat like this. Here you can see that there is a sim seconds and something known as the host seconds. Sim seconds means that for how much time did the program hello world run in the simulated machine.

(Refer Slide Time: 16:17)



While hosts second means that for how much time gem5 took to run in the host machine that is your machine. This figure summarizes the complete procedure of how gem5 works. First, the input parameters that are given to the gem5 simulator, that is se.py in the previous slide are pursed by Python interpreters. These parameters are then passed on different SimObjects. As mentioned, each of these SimObjects represents 1 physical quantity, this may represent cache, the second SimObjects may represent DRAM.

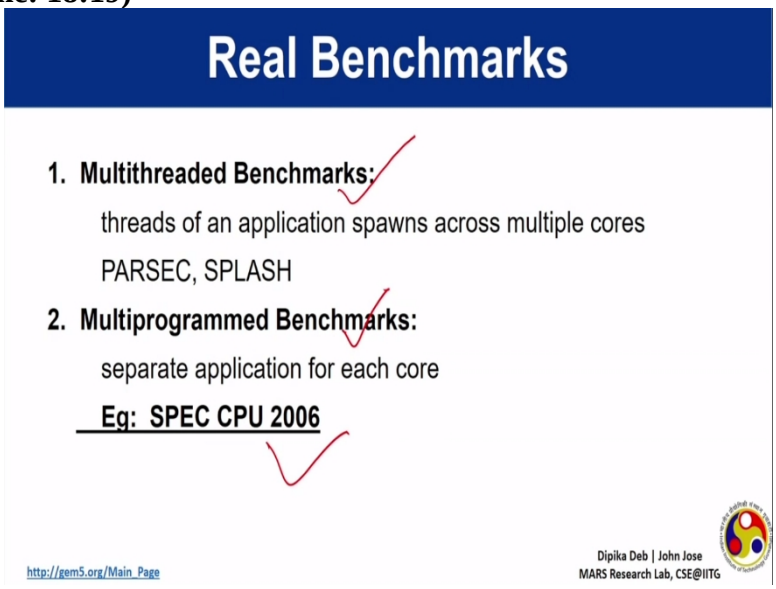
(Refer Slide Time: 16:46)



The next one, SimObjects 3 may represent network and so on. This SimObjects are also written in python fighting. So we required a python interpreter. For example, in cache implementation in Ruby, it has a SimObject known as Ruby cache.PY. Next, the C++ objects which define this system behaviour. That is cache organization implementation of the cache inherits the values from the same objects. And finally, the system is built using GCC or G++ compiler.

And the output of the simulated machine can be found in stats.text. And the configurations that we used for the simulated machine can be seen in config.ini file, and config.JSON file. Now along with running user programs, one can also run real benchmarks to verify the behaviour of a simulated machine with respect to the real world programs. There are 2 types of benchmarks, multi programmed and multi-threaded benchmarks.

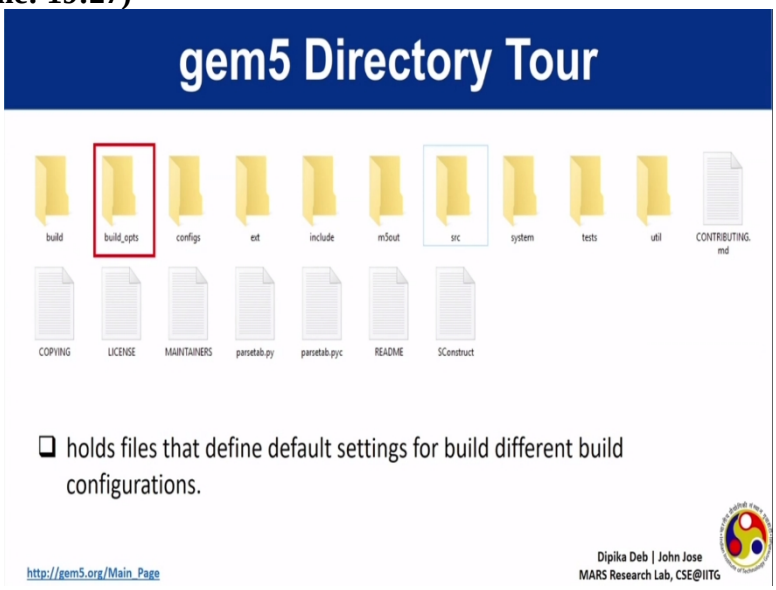
(Refer Slide Time: 18:19)



In multithreaded benchmarks, thread of an applications spawns across multiple cores par second splash are 2 such benchmarks. On the other hand, in multi program benchmarks, separate applications runs on each core like in spec cpu 2006 benchmark suit. We will use this benchmark throughout the gem5 assignments provided in the course. So next we go through a quick directory tour of gem5. Inside the gem5 directory you will find this directories. Build contains the build directory which is generated after typing Scons build command.

(Refer Slide Time: 19:08)

(Refer Slide Time: 19:27)



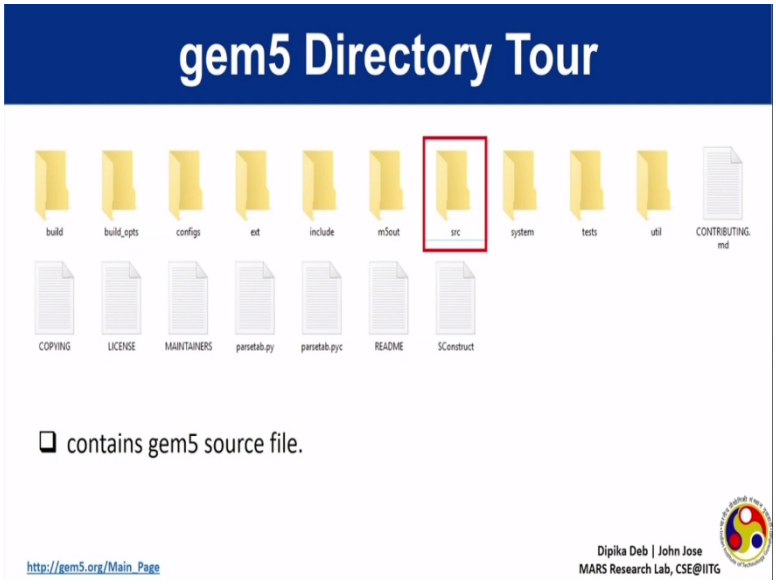
The build opt hold files that define default settings for built different building configurations. This folder contains simulation configurations. This contains gem5 dependency, but this is not a part of actual gem5. This directory contains include files that are used to simulate and compile gem5. The output files are present in this folder. And source contains all the gem5 implementations like caches course, CPU models, interconnection models and everything.

(Refer Slide Time: 19:34)

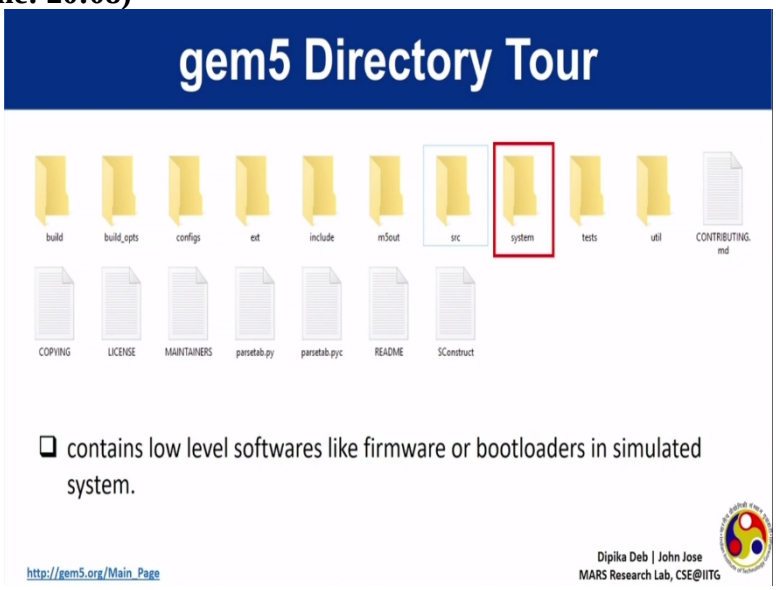
(Refer Slide Time: 19:44)

(Refer Slide Time: 19:53)

(Refer Slide Time: 19:57)

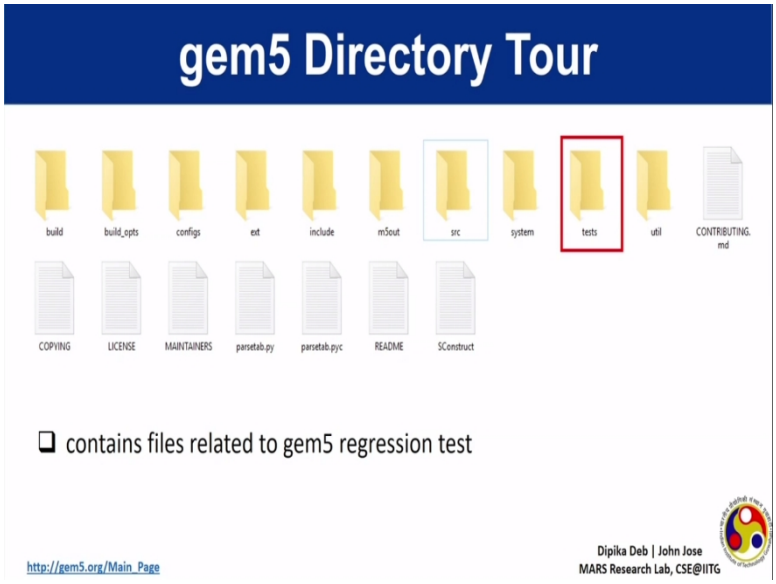


(Refer Slide Time: 20:08)

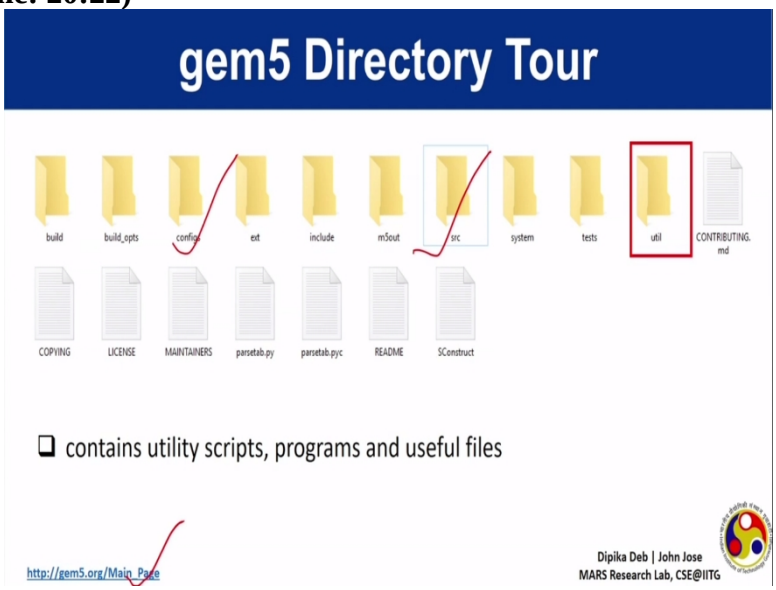


This directory contains low level softwares like firmware or boot loaders in simulated system. This directory contains files related to gem5 regression test. While this directory contains utilities scripts, our programs and useful files we will mostly deal with this folder and the config folders. All the references are mentioned in the slide. With this we come to the end of today's video for any further queries, writer us in the discussion forum of our course web page.

(Refer Slide Time: 20:16)



(Refer Slide Time: 20:22)



Thank you.