# An Overview of Cycle-Accurate, Event-Driven and Full Systems Simulators for Chip-Multiprocessors

Ahmed Hasan Kadhim Janabi and Michael Opoku Agyeman

***Abstract*—Due to the growing number of Chip-Multiprocessors, the researchers have proposed new designs and architectures. So, there is a constant need to know the most accurate simulators used in this scope, which should be used to identify their outcomes. Computer System Architecture (CSA) simulators are usually used to validate the designs, architectures that new discovered and developments. This paper is to provide an overview and insight into the most critical simulations used in Computer System Architecture and possible standards that distinguish simulators from the other. The essential aspects and parameters that determine these simulators are accuracy, speed, performance, flexibility, and functionality as well. Cycle-Accurate, Event-Driven and Full Systems Simulators of CSA, this taxonomy of simulators that w discusses in this paper.**

***Index Terms*—Computer system architecture (CSA), simulators, sniper, a cycle-accurate, full system, event-driver, Gem5, DRAMSim21.**

## I. INTRODUCTION

Computer System Architecture (CSA) Simulators are software tools which are essential in designing several computer models and components to predict performance level in specific inputs [1]. Computer simulations are usually a crucial point in computer architecture research and design. Such referred to like the intricate design used by researchers in architecture, which often designed in a complicated and non-analytical way, which makes the production of such models at all stages of design costly [1]. A comprehensive simulation is expensive, as well as it requires realistic workloads, many machine cycles, and well-designed simulators [2].

Usually, the required level of detail getting by the low speed, but in this issue, the multi-core processor design needs a high-speed simulation [3]. That may take several weeks and months to complete the task. The increasing of generations grows with the aggregate throughput of the host chip increases. However, there is no enhancement due to energy constraints in the single-thread performance, that is a critical component in the simulation [4]. It is interesting to note that the number of the core is directly proportional to the architecture model, as the number of cores increases, the architecture becomes more complicated than the first. The increasing target complexity does not adapt to the increased performance of the host cores. Researchers recognize the problem by finding out different strategies to reduce simulation time [5]. The method of sampling is the way that applied in most previous works that use to get an analytical description for a particular part in the application. That helps to get comparable visions in a full simulation with the reduce time taken as well [6].

In another hand, other researchers presented the parallel simulation by running it on several threads to reduce a single-simulation period. As well, this approach is appropriateness for design stages, which requires to be evaluated faster for more design points. Although, when passing several parameters, driving Indicators simulations, concurrently provides better performance for simulation [1]. Necessarily to use the reducing simulation period strategies in the cycle-accurate simulators [7]. The simulators in computer architecture implement the essential elements of research to improve understanding of the importance associating to the workload of the microarchitectural structure. Generally, cycle-accurate simulators are not proper for complex and large-scale structures [4].

The rest of the paper formed as follows: Section II explained the importance of studying computer system architecture. The type of system simulators described in Section III. Section IV discusses the selected simulators for detailed study. Section V concludes the paper.

## II. COMPUTER SYSTEM ARCHITECTURE SIMULATORS IMPORTANCE

Simulation is necessary because it facilitates the work of researchers, simulates their work, and implements the complicated structures. As such, the old systems required. A system migration simulation, significant as well, which are costly, can be simulated to maintain their operation. Therefore, the simulators can help the company in moving to the new software and hardware easier. That provides economics and time in the business aspect [6], [7]. Besides, simulation is also essential for hardware development.

According to Magnusson et., simulators are significant in modelling computer systems despite their availability [8]. That contributes to the development of devices that still exist or are under construction. Also, it decreases the costs because it reduces the need for hardware models [9]. Thus, the use of simulation is useful in terms of cost savings. For example, the simulator provides the standard x86-based PC, to save costs and organization [9]. It could help to build software for the developers of device drivers and operating systems as well. It's challenging to debug software that is running in kernel

Manuscript received August 2, 2019; revised October 10, 2019.

Ahmed Hasan Kadhim Janabi is with Faculty of Art, Science & Technology, University of Northampton, Northampton, UK and IT Department, Al-Mustaqbal University College, Hilla, Babil, Iraq (e-mail: ahmedaljanabi95@outlook.com).

Michael Opoku Agyeman is with Faculty of Art, Science & Technology, University of Northampton, Northampton, UK (e-mail: Michael.OpokuAgyeman@northampton.ac.uk).

DOI: 10.7763/IJCTE.2019.V11.1251

mode; but, with using simulators provide debugging options that make the development easier [1]. Also, the simulators operate in a controlled environment that helps the researchers to check security and enhance their work. That enables the companies to examine the new proposed systems before going into the process production. The simulation technology has importance in recovering the past states of simulated computers.

## III. Classification of Computer System Architecture Simulators

The taxonomies in computer architecture simulators based on their contexts. In the beginning, the researchers classified the simulators depend on their field, which involves Cycle Accurate, Full System, Instruction Set, Micro-architecture, and Application-based simulators.

In another hand, there is another classification according to the inputs or workload, which involves event-driven (also called trace-driven simulators) and execution-driven simulators [10].

### A. Full System Simulators

Full system simulators can simulate a fully operating system. Also, it can simulate the I/O devices which needed to boot an operating system. That enables it to be flexible to operate complex multi-threaded workloads. In general, the full system simulators involve memories, peripheral devices, interconnection buses, network connections, and processor cores. There are many examples of full system simulators like Simics [11], Gem5 [10], ML-Rsim [12], SimOS [10], and PTLSim [13].

### B. Event-Driven Simulators

In this case, Event-driven simulators (also called Trace-driven simulators) use trace files as inputs. These files include recorded streams of instructions of a program operating on some real hardware, and this hardware considers it's the target [12].

One of the most problems that faced the developers in this type of simulators generated files could be a considerable size and retrieve such files from disk it may take longer time. However, this problem can solve by utilised trace sampling and reduction techniques [10]. According to the event-driven simulation cannot be used as a simulator to systems based on the parallel system or timing. These Simulators perform simulation corresponding to events, rather than depending on a cycle-by-cycle in simulating the target. Sniper is CSA simulator depends on this technique [9].

### C. Cycle-Accurate Simulators

A cycle-accurate simulator is a CSA simulator that based on a cycle-by-cycle basis. Usually, a cycle-accurate simulator is applied when designing a new microprocessor so they can be experimented and benchmarked accurately without building a hardware device, and it merely changes design many times to reach the required plan [13].

The Cycle-accurate simulators require to check if all orders executed in the proper virtual time – branch prediction, thread context switching, fetch, cache misses, and many other subtle aspects of microprocessors [8]. Cycle-accurate simulators usually add a fixed latency to memory accesses, that lead to significantly under-reporting the real impact of the memory system. Often, it uses Cycle-accurate simulators to help fill the void of accurate memory system simulators. DRAMSim2 is CSA simulator depends on this technique [7].

## IV. Selected Simulators for Detailed Study

In this paper, has selected three types of simulators for detailed study, which are gem5 simulator as an example of full system simulators [8], Sniper simulator as an example of Event-driven simulators [6], and DRAMSim21 simulator as an example of Cycle-Accurate simulators, because they have several design strategies with respect to detail and concept [7], [14]. Next, is a brief discussion of all these simulators along with previous validation works.

### A. Gem5 Simulator

Gem5 is a full system simulator that used instruction set architectures (ISAs) with many CPU models [4]. Gem5 obtains the full memory system modelling from GEMS [10] and the detailed CPU modelling from M5 [4]. Also, it supports various CPU models, including 'TimingSimple', 'AtomicSimple', 'O3' and 'InOrder'. AtomicSimple and TimingSimple are nonpipelined single-cycle micro-architectures [10].

The 'InOrder' and 'O3' used as a pipelined out-of-order (OOO) and in-order (IO) core models. Both are 'execute-in-execute' meaning that instructions executed in the execute step after the fixed all the dependencies. These can be configured to simulate a different number of pipeline stages, issue widths and number of hardware threads. Gutierrez and his team experimented the accuracy of gem5 by modelling real systems based on ARM. They made some modifications in the simulator, Gutierrez et al. [15] were obtained 5% as a low percentage in a runtime error, and they gained 13% as a low absolute percentage in a runtime error for SPEC CPU2006 benchmarks [5], [15].

Usually, the researchers also require a simulation structure that enables them to cooperate with others in both academia and industry [15]. However, the licensing terms of the simulators and quality of code can prevent this collaboration. Moreover, some limitations like the lack of modularity and the poor of quality code may cause some difficulties in understanding and modify the code for new users.

The gem5 simulator exceeds those restrictions by providing a flexible and modular simulation system that can evaluate a broad range of systems, and it is reachable and available for researchers. This infrastructure affords flexibility by providing several sets of memory system models, CPU models, and system execution modes [16]. The license of BSD-based makes the code reachable and available for others (researchers and developers) without awkward legal restrictions [4].

### B. Sniper Simulator

Sniper is a parallel simulator, a high-speed and accurate x86. This simulator depending on the Graphite simulation infrastructure and the interval core model, providing a realistic and fast simulation to able to use a range of flexible

simulation possibilities while searching various heterogeneous and homogeneous multi-core architectures [9]. Besides, it achieves timing simulations at high speed, and this considered faster than the current-used simulators. The timing simulations performed by the sniper simulator for both multi-threaded, multi-program workloads and shared-memory applications, are among 10s to 100+ cores. The critical feature of the sniper simulator is has a core model that depends on the period simulation [10]. The main reason which makes the simulator be faster development and evaluation times by using intervals which mean 'jumping' between miss events [17].

Sniper simulator has been examined with Nehalem systems and multi-socket Intel Core2 and get c performance errors about 25% at a speed of simulation up to several MIPS [9].

This simulator has a friendly environment and easy for the researchers; that the reason to be balanced between accuracy and performance. However, Sniper simulator doesn't support all instructions of a different systems model (i.e. x86-bit 64 and the SSE4 ) [10].

Sniper simulator is considered very important in vital studies that require in-depth detail. It also has used in conventional one-IPC models when the cycle-accurate simulators are not fast to allow logical simulation assignments [1].

After the simulation process ends; the sniper simulator generates different files which are (sim.cfg, sim.out, and sim.stats) [1]. The sim.cfg file contains configuration alternatives that used in the simulation process. While the sim.out files include the primary statistics, also the sim.stats file comprises several sets of tools to capture critical points of the simulation and most importantly is CPI (Cycles Per Instruction) Stacks. The CPI stack is sets of instructions includes different components which contribute to the overall performance, as presented in Fig. 1. The core CPI exists at the bottom and displays the vital work done. The CPI stack is essential while obtaining insight toward the performance of the simulation [1].

An additional essential feature is that the interval core model supports the making Cycles Per Instruction (CPI) stacks. These stacks represent the cycles number that gets lost due to the variable features of the system. This feature allows Sniper use in the categorisation of applications and hardware design/software [7]. The topology which uses in Sniper as shown in Fig. 2.

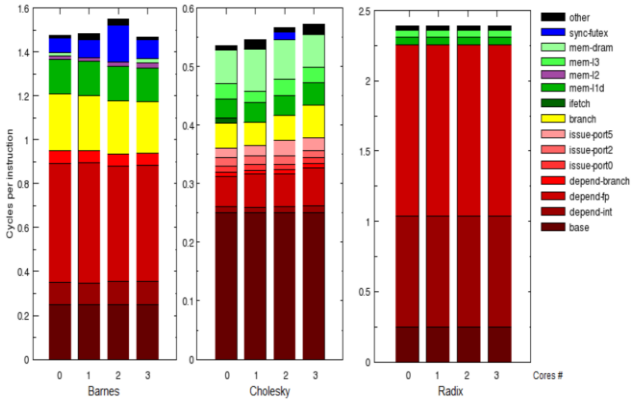


Fig. 1. Cycles per instruction stack [1].

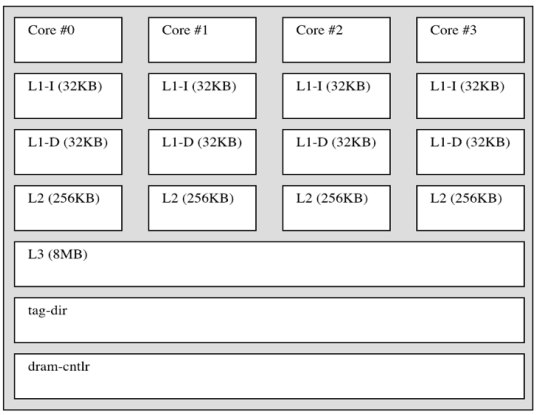


Fig. 2. The topology which uses in Sniper.

### C. DRAMSim2 Simulator

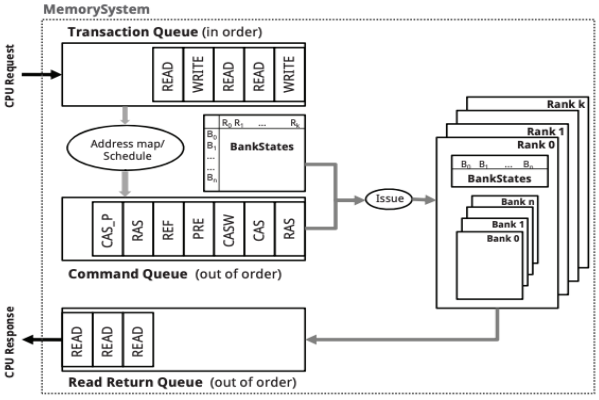


Fig. 3. Overview of DRAMSim2 elements [7].

Fig. 3 Demonstrated the main structure of the DRAMSim2 elements. DRAMSim2 is considered one of the most popular DDR2/3 memory system simulators which implemented by using C++. The primary purpose of design this simulator is to improve the accuracy of the simulation results and make the simulator freely accessible DDR2/3 memory system model as well can utilise for trace-based and full system simulations [7].

As shown in Fig. 3 the core of the DRAMSim2 represented by the object named memory system which needs two types of files: the first file is a device ini which includes parameters that define a particular DRAM device like power consumption and the timing constraints of the device. The second file is a system ini contains independent parameters of the real DRAM device [7]. As an example of these parameters are: ranks, memory controller queue structures, row buffer, the address mapping scheme, debug options. Also, the DRAMSim2 can be run in two different modes, which are a shared library or as a standalone binary [13].

Moreover, this type of simulator has a flexible object-oriented design and programming interface. More critical, the DRAMSim2 supports a verification tool which can help to verify the results of the DRAMSim2 simulation without caring about the front end driver2 [13].

Besides shown that the DRAMSim2 offers a robust visualisation tool that allowed users to view and analyse the impacts of memory system parameters on performance metrics like power, latency and bandwidth [7]. All of these characteristics make the DRAMSim2 an essential tool for

researchers, especially those who want to comprise memory models in their computer architecture research [7], [10].

## V. Conclusion

More generally, these essential findings are consistent with research showing that the best simulators tool that used model multi-core micro-architecture. It is significant to use simulators features at a sooner speed than the full-system simulators identified through hardware and software expenses. The main challenge of the simulators is how to make a simulation been for thousand-core chips; however, having such simulators is impossible to reach that level. Furthermore, it may face challenges to achieve the primary goal because of the limitation of some cores that causes by the Instruction Set Architectures (ISA). In general, it possible x86's Advanced Programmable Interrupt Controllers (xAPICs) able to support 256 cores. In this level of operation, selecting a simulator must depend on the user-level simulator model.

The continuous increase in the numbers of processes in cores and systems leads to a rise in the system sizes that may cause some difficulties in the simulating. The constant development of multi-core technology with much larger cache sizes requires accurate and high-level simulations to validate the initiation of system architectures and designs.

## References


[1] M. Al-manasia and Z. Chaczko, "An overview of chip multi-processors simulators technology," *Progress in Systems Engineering, Advances in Intelligent Systems and Computing*, vol 366, pp. 877-884, 2015.

[2] A. V. Laer, T. Jones, and P. M. Watts, "Full system simulation of optically interconnected chip multiprocessors using gem5," in *Proc. Opt. Fiber Commun. Conf. Fiber Opt. Eng. Conf. 2013*, 2013.

[3] M. O.Agyeman, Q.-T. Vien, A. Ahmadnia, A. Yakovlev, K.-F. Tong, and T. Mak, "A resilient 2-D waveguide communication fabric for hybrid wired-wireless NoC design," *IEEE Trans. Parallel Distrib. Syst.*, p. 1, 2016.

[4] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1, 2011.

[5] A. Al-Mahmood and M. O. Agyeman, "On wearable devices for motivating patients with upper limb disability via gaming and home rehabilitation," in *Proc. 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, 2018, pp. 155–162.

[6] A. Akram and L. Sawalha, "A comparison of x86 computer architecture simulators," in *Proc. the CEUR Workshop*, vol. 1691, 2016, pp. 21–27.

[7] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Comput. Archit. Lett.*, vol. 10, no. 1, pp. 16–19, 2011.

[8] M. T. Yourst, "PTLsim: A cycle accurate full system x86-64 microarchitectural simulator," in *Proc. the ISPASS 2007 IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2007, pp. 23–34.

[9] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proc. the 2011 Int. Conf. High Perform. Comput. Networking, Storage Anal. - SC '11*, 2011, p. 1.

[10] A. Akram and L. Sawalha, "×86 computer architecture simulators: A comparative study," in *Proc. the 34th IEEE Int. Conf. Comput. Des. ICCD 2016*, 2016, pp. 638–645.

[11] P. S. Magnusson *et al.*, "Simics: A full system simulation platform," *Computer (Long. Beach. Calif).*, vol. 35, no. 2, 2002.

[12] J. E. Miller *et al.*, "Graphite: A distributed parallel simulator for multicores," in *Proc. HPCA - 16 2010 Sixt. Int. Symp. High-Performance Comput. Archit.*, 2010, pp. 1–12.

[13] Y. Paik, M. Han, K. H. Choi, M. Kim, and S. W. Kim, "Cycle-accurate full system simulation for CPU+GPU+HBM computing platform," in *Proc. Int. Conf. Electron. Inf. Commun. ICEIC 2018*, vol. 2018, 2018, pp. 1–2.

[14] M. O. Agyeman, "Optimizing heterogeneous 3D networks-on-chip architectures for low power and high performance applications," Doctoral thesis, Glasgow Caledonian University, 2014.

[15] A. Gutierrez *et al.*, "Sources of error in full-system simulation," in *Proc. ISPASS 2014 - IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2014, pp. 13–22.

[16] M. O. Agyeman, A. Ahmadinia, and N. Bagherzadeh, "Performance and energy aware inhomogeneous 3D networks-on-chip architecture generation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1756–1769, Jun. 2016.

[17] R. Chis and L. Vintan, "Multi-objective hardware-software co-optimization for the SNIPER multi-core simulator," in *Proc. 2014 IEEE 10th Int. Conf. Intell. Comput. Commun. Process. ICCP 2014*, 2014, pp. 3–9.


**Ahmed H. Janabi** is currently a PhD researcher with the Faculty of Arts, Science, and Technology at the University of Northampton in the United Kingdom. He finished his bachelor of engineering in computer networking in 2017 with high honours, and previously studied at the University of Babylon in the Republic of Iraq.

He currently works as a researcher with the University of Northampton in the United Kingdom. His previous positions include teaching assistant within the University of Northampton, University of Babylon, and Al-Mustaqbal University College. Currently he is in progress of a journal write-up in the field of software-defined network and security. His research interests include (but not limited to) software-defined networks, internet of things, security on the network level, computer architecture and engineering, and microprocessors.

**Michael Opoku Agyeman** is a charted engineer (CEng) of the IET and a fellow of Higher Education Academy (UK) who is a senior lecturer and the programme leader of BEng, HND, MEng computer systems engineering at the Department of Computing at the University of Northampton, UK. He is an associate member of the Chartered Management Institute (ACMI). Previously, he was with Intel Embedded System Research Group of the Chinese University of Hong Kong (CUHK) as a research fellow, where he worked on reliable wireless network-on-chip solutions. He received the PhD in embedded and distributed systems from Glasgow Caledonian University, UK, in 2014 and the MSc. degree in embedded and distributed systems from London South Bank University, UK, in 2009. Michael received the BSc. (Hons.) in electrical and electronics engineering from Kwame Nkrumah University of Science and Technology (KNUST), Ghana, in 2008. He is the author of one book and over 50 publications in significant conference proceedings and journals. His research interests include VLSI SoC design, computer architecture, reconfigurable computing, wired and wireless NoCs, smart rehabilitation solutions, embedded systems and Internet-of-Things (IoT). He has been a guest editor of the EAI Endorsed Transactions on Industrial Networks and Intelligent Systems. Michael is currently a reviewer of several conferences and journals. He serves as a technical committee member (TPC) of over 10 conferences including IEEE ICCSN, IEEE ICBDA, ICIIP and ICBDSC.