

Comparative Analysis of Simulators



**Comparative Analysis of Simulators**

**Abstract**

The importance of computer architecture simulators in advancing computer architecture research is widely acknowledged. In the past few decades, computer architectures have developed numerous simulators, and their number continues to grow. Comparing simulators with each other and validating their correctness has been a challenging task. In terms of flexibility, level of details, user-friendliness, and simulation models, we use x86 simulator. We measure the experimental error and compare the speed of two x86 simulators: gem5 and Sniper. We also discuss the strengths and limitations of the different simulators.

**Table of Contents**

[Introduction 4](#_Toc28900118)

[Motivation 4](#_Toc28900119)

[Simulator 4](#_Toc28900120)

[Computer Simulation 5](#_Toc28900121)

[Simulator for Details Study 5](#_Toc28900122)

[Gem5 5](#_Toc28900123)

[Sniper 6](#_Toc28900124)

[Feature Comparison 7](#_Toc28900125)

[Simulators Verification Methodology 7](#_Toc28900126)

[Target System 7](#_Toc28900127)

[Experimental Workloads 11](#_Toc28900128)

[Performance Measurement on Real Hardware 11](#_Toc28900129)

[Result and Analysis 11](#_Toc28900130)

[Conclusion 13](#_Toc28900131)

[References 14](#_Toc28900132)

# Introduction

To build a real system for testing and verification purposes, it needs a lot of money. Computer architects rely on simulation and modeling techniques to evaluate different designs. Approximately 90% of papers published in top conferences use simulation as a performance evaluation methodology. Previous surveys on computer architectures simulations are old and do not include new simulators. Some of them only focus only on teaching or memory-related simulators. This paper compares and contrasts x86 simulators. X86 is one of the oldest and widely used instruction set architectures in desktops and servers. We compare the performance results of the simulators with each other and also try to find out the strengths and limitations.

# Motivation

The motivation for doing this project was primarily an interest in undertaking a challenging project in an interesting area of research. The opportunity to learn about a new area of computing not covered in lectures was appealing. This will help us greatly on how to do research in our futures if we pursue post-graduate degrees in this topic. Also, this is a great opportunity for us to determine how we function in a group setting. I hope learn teamwork and courage through our group project.

# Simulator

A machine that simulates environmental and other conditions for purpose of training and experimentation is called simulator. And simulation is a technique of studying and analyzing the behavior of a real world or an imaginary system by mimicking it on a computer application. A simulation is works on a mathematical model that describes the system. In a simulation, one or more variable of mathematical model is changed and resulted changes in other variables are observed. Simulation enable user to predict the behavior of the real-world system. Simulation help designers to optimize their systems by doing necessary changes and obtain good results. User can run simulations slower and faster than the real world and that may help to figure out more details.

# Computer Simulation

A computer simulation or also known as “sim” is an attempt to model a real life or hypothetical situation on a computer so that it can be studied to see how the system works. By changing variables in the simulation, predictions may be made about the behavior of the system. A computer architecture simulator is a program that simulates the execution of computer architecture. Computer architecture simulators are used for the following purposes:

* Lowing cost by evaluating hardware designs without building physical hardware systems
* Enabling access to unobtainable hardware.
* Increasing the precision and volume of computer performance data.
* Introducing abilities that are not normally possible on real hardware such as running code backwards when an error is detected or running in faster than real time.

A cycle accurate simulator is a computer program that simulates a microarchitecture on a cycle by cycle basis. An Instruction set simulator simulates an instruction set architecture usually faster but not cycle accurate to a specific implementation of architecture. A cycle – accurate simulator is used when designing new microprocessor that can be tested and benchmarked accurately without actually building a physical chip and easily change design many times to meet expected plan.

# Simulator for Details Study

We selected two simulators for detailed study gem5 and Sniper because they have diverse design strategies with respect to details and abstraction. All the contemporary simulator with active development.

# Gem5

Gem5[8] is a full system simulator that supports many ISAs with various CPU models. Gem5 Borrows the detailed CPU modeling from M5[28] and the detailed memory system modeling from GEMS[14]. Gem5 primarily supports four CPU models: ‘AtomicSimple’, ‘TimingSimple’, ‘InOrder’ and ‘O3’. AtomicSimple and TimingSimple are nonpipelined single-cycle microarchitectures. The ‘InOrder’ and ‘O3’ are pipelined IO and OOO core models. Both are ‘execute-in-execute’ meaning that instructions are only executed in the execute stage after all dependencies have been resolved. These can be conﬁgured to simulate different number of pipeline stages, issue widths and number of hardware threads.

Gutierrez et al. [29] and Butko et al. [30] validated the accuracy of gem5 by modeling real systems based on ARM. After making some modiﬁcations to the simulator, apart from conﬁguring it to match the experimental board (ARM Cortex A15), Gutierrez et al [29] were able to achieve a mean percentage runtime error of 5% and a mean absolute percentage runtime error of 13% for SPEC CPU2006 benchmarks [29]. Butko et al. [30] have analyzed the accuracy of gem5 for simulation of a multicore embedded system (ARM Cortex A9). Various benchmarksrelatedtoscientiﬁcworkloads(SPLASH-2),mediaapplications(ALPBench)and memory bandwidth (STREAM) were used for validation. The results show that the accuracy varies form 1.39% to 17.94%. We have not been able to ﬁnd any validation effort for x86 based targets for gem5 simulator.

# Sniper

Sniper [7] is a parallel simulator for simulating large scale multicore systems using interval simulation [19], which provides a balance between detailed cycle-level simulation and oneIPCsimulationmodel(one-IPCmodelisdeﬁnedasanIOsingleissuepipelinemodel). Sniper is based on Graphite [31] that supports various one-IPC models. Carlson et al [7] validated Sniper against real hardware (4-socket 6-core Intel Xeon X7460 Dunnigton shared-memory with simultaneous multithreading (SMT) support) using SPLASH-2 benchmark suite. They concluded that Sniper’s interval simulation is within 25% accuracy on average compared to real hardware. Carlson et al. introduced the instruction-window centric core model that improved the accuracy of base interval simulation model [32]. Validation of this model against Nehalem microarchitecture based system showed a single-core error of 11.1% using a subset of SPLASH-2 benchmarks.

# Feature Comparison

|  |  |  |
| --- | --- | --- |
| Feature | Gem5 | Sniper |
| Platform Support | P++ | P |
| Target support | T++ | T |
| Full System | Yes | No |
| Fast forwarding & Cache warmup | Yes | Yes |
| Checkpointing | Yes | Yes |
| Trace generation | Yes | Yes |
| Pipeline depth configuration | Yes | No |
| Energy and Power modeling | E++ | E |
| In-order Pipeline Support | Yes | Yes |
| GPU – Modelling | Yes | No |
| Multi-threaded app support | Yes | Yes |
| Community Support | C++ | C++ |

Table 1: Simulators Feature Comparison

To support power and energy modeling, gem5 and Sniper provide support for dynamic voltage and frequency scaling (DVFS), making it possible to run experiments to simulate energy efﬁciency. The support for heterogeneous multicore (HMP) simulation, Sniper support only single-ISA (also known as asymmetric) HMP simulation. Gem5 has currently integrated a GPUsim model to run cpu-gpu heterogeneous simulations. Moreover, it can be modiﬁed to simulate single- and multi-ISA HMP system, as it supports multiple ISAs.

# Simulators Verification Methodology

To experimentally evaluate the error in the selected simulators to model real hardware, we have conﬁgured them to model an existing processor, Corei7. The target system is similar to Haswell µ-architecture (Intel core i7 machine i7-4770 cpu, 3.40 GHz). We then compare the performance of simulated benchmarks with their runs on the real hardware. This section discusses the conﬁgurations used for the simulators, experimental workloads and the method followed for measuring real hardware metrics.

# Target System

As all the exact conﬁgurations for this processor are not published by Intel, we tried our best to model Haswell microarchitecture based on some Intel documentation [36, 37] and other resources [38, 39, 40, 41]. The basic features of this target system are in Fig 1.

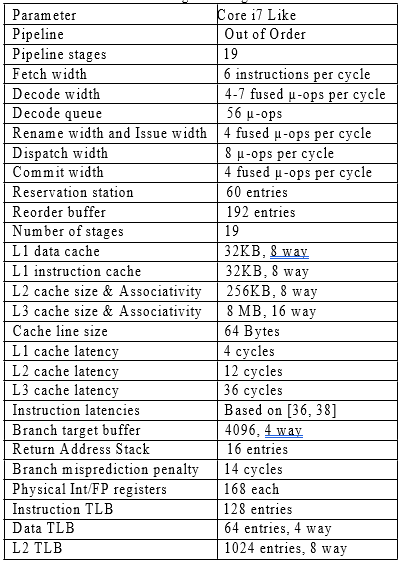


Fig: Target Configuration

Our main objective while configuring simulators was to be as close as possible to the real target system. Another, important aim was to conﬁgure these simulators as close as possible to each other, so that we could perform a fair and impartial comparison. Although these simulators have diverse support for conﬁguration parameters, we tried our best to keep the simulation conﬁgurations the same. First column describes the conﬁguration parameter, whilerestofthecolumnsdescribethecorrespondingsimulator’sparameternameandsetvalue for this parameter. Deﬁnition of that parameter for a particular simulator is also mentioned in brackets where required.

|  |  |  |
| --- | --- | --- |
| Parameter | Gem5 | Sniper |
| Branch Predictor | Tournament | Pentium M |
| BTB associativity | 1 way | 4 way |
| TLB associativity | 1 way | 4,8 way |
| Instruction Queue | Unified(60 entries) | Unified(60 entries) |
| Memory Address disambiguation | Store set | Not Configurable |
| Delay Between pipeline stage | Configured to achieve overall 19 stage pipeline | Not configurable |
| TLB levels | Single | Double |

Table 2: Differences in Simulator Configurations

Gem5’s pipeline conﬁguration is very ﬂexible, in fact it is the most ﬂexible out of the four simulators that we have studied. A19-stagepipelineissetbyconﬁguringdifferentdelayvalues between the existing pipeline stages of gem5’s OOO pipeline, such that branch misprediction penalty stays 14 cycles. Since µ-ops are known at the fetch stage in gem5, all pipeline width parameters are set in µ-ops. Moreover, to incorporate the effect off used µ-ops(not supported in gem5), we have translated the width of 4 fused µ-ops to 6 normal µ-ops. The branch predictor simulated is a tournament predictor similar to the one present in Alpha 21264 machine [42]. Thisschememaintainslocalandglobalhistorytopredictthedirectionofagivenbranch, where a choice predictor selects one out of the two predictions. The size of local, global and choice predictors and direct-mapped BTB are 4K entries each. Each entry contains a 2-bit counter.

In case of Sniper, to conﬁgure any target system, a primary conﬁguration script base.cfg is already provided. Various scripts used to conﬁgure different core models and structures are available that can override the default conﬁguration parameters. Moreover, conﬁgurations for some modern Intel processors are also provided. We have overwritten the basic conﬁgurations with those for Haswell.

|  |  |  |
| --- | --- | --- |
| Parameter | Gem5 Parameter | Sniper Parameter |
| Clock Frequency of core | 3.4GHz | 3.4GHz |
| Core model | cpu-type detailed | Nehalem |
| Fetch width | 8 | No parameter |
| Fetch buffer | 16 | No parameter |
| Fetch queue | 56 | No parameter |
| No. of stage between fetch and decode | 3 | No Parameter |
| No. of stages between decode and rename | 3 | No parameter |
| No. of stage between rename and issue | 4 | No parameter |
| No. of stages between issue and execute | 2 | No parameter |
| No of stage between writeback and commit | 4 | No parameter |
| Decode width | 6 | No parameter |
| Rename width | 6 | No parameter |
| Dispatch width | 6 | No parameter |
| Issue width | 8 | No parameter |
| Squash width | 8 | No parameter |
| Writeback width | 8 | No parameter |
| Commit width | 8 | 4 |
| Load queue entries | 72 | 72 |
| Store queue entries | 42 | 42 |
| Instruction queue entries | 60 | 60 |
| Count of int ALU’s | 4 | No parameter |
| Cache read ports | 2 | No parameter |
| Cache write ports | 1 | No parameter |
| Branch target buffer entries | 4096 | 4096 |
| Return address stack size | 16 | No parameter |
| Branch misprediction penalty | 14 | 14 cycle |
| Cache line size | 64 | 64 |
| Main Memory Latency | 57ns | 57ns |

Available pipeline conﬁgurations are also changed according to values in Table 3. The branch predictor modeled in Sniper is based on Intel Pentium M’s branch predictor [43]. We have changed the table sizes of this branch predictor so that global predictor, bimodal predictor andBTBeachare4Kentriesinsize. Pentium M branch predictor also contains a loop predictor (128 entries) and a direct-mapped iBTB (indirect branch target buffer) with 256 entries for indirect branches.

As details of cache prefetchers on Haswell are not available, we did not conﬁgure any prefetcher on any simulator to eliminate one source of inaccuracy. We also deactivated cache prefetchers on real hardware before collecting reference performance statistics.

# Experimental Workloads

We ran our experiments using SPEC CPU 2006 [45] and subset of the MiBench embedded benchmarks [46] benchmark suites. We simulated complete MiBench benchmarks, however; running complete SPEC benchmarks in simulated environment will take impractically long time. Thus, each application was executed for 500 million x86 instructions chosen from a statistically relevant portion of the program [47]. Simulation is fast-forwarded for all benchmarks and are warmed up for 100 million instructions before start of detailed simulation.

# Performance Measurement on Real Hardware

We used PAPI [48] to measure IPC (instructions per cycle) values for entire execution of embedded benchmarks. In case of SPEC benchmarks we measured the IPC values for the same 500 million instructions interval that we simulated. Benchmarks were run ﬁve non-consecutive times and results are averaged to compare with simulation results and measure experimental error. We also used PAPI to measure number of cache misses and branch mispredictions on the real hardware. The compiler that we used is gcc 4.4.7. We have used 64-bit binaries for gem5 only. The host OS used for compilation and the execution of benchmarks is Scientiﬁc Linux 2.6.32. For gem5, experiments are performed on a Ubuntu 14.04 host OS. Reference hardware performance metrics are measured on both systems as well. We have used gem5’s stable version of September 2015 and Sniper version 6.0.

# Result and Analysis

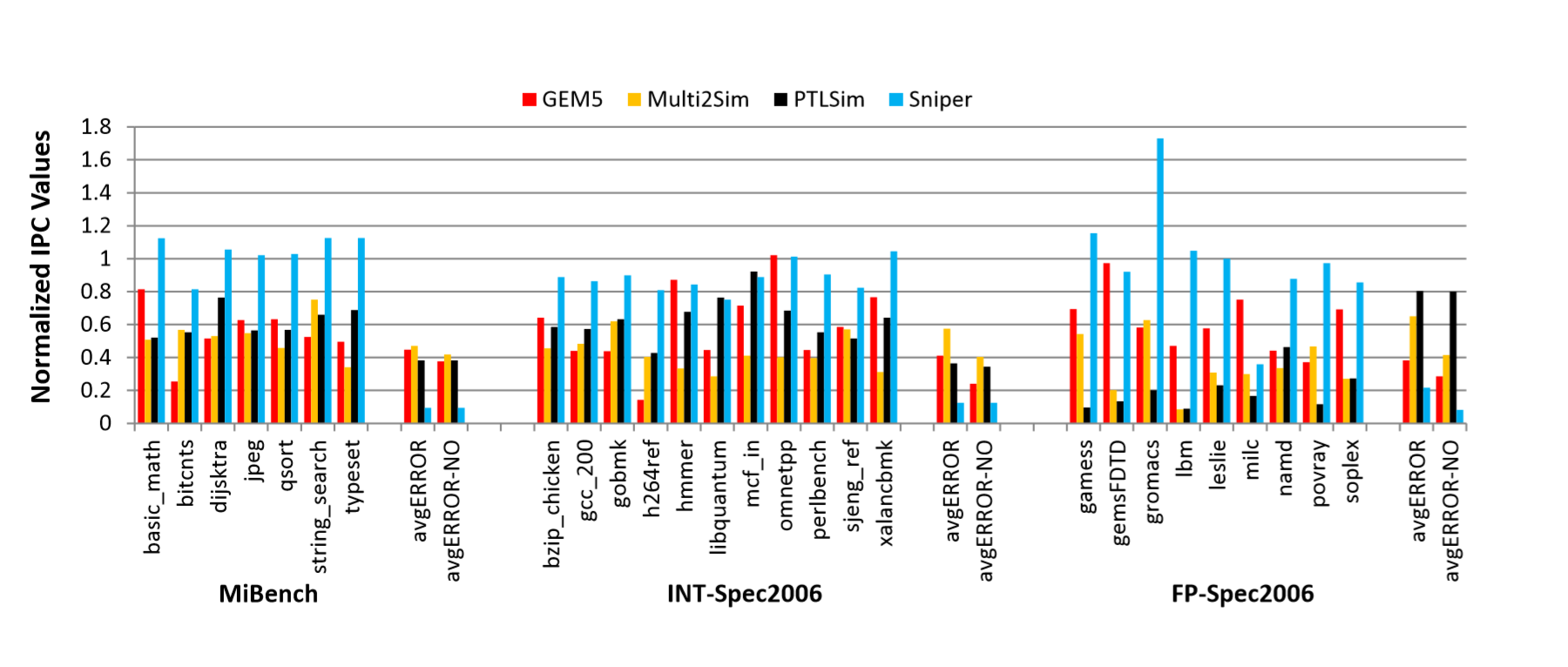


Fig: Normalized IPC values for all benchmarks

In this section, we compare the IPC, cache hit rate and branch misprediction results of the four simulators to that of real hardware runs on a Haswell processor. We also quantify the errors and evaluate the accuracy of this approach. All ﬁgures in this section show two types of average errors for all evaluated metrics: one including all benchmarks, avgE, and the other without outliers, avgE−NO; where an outlier corresponds to more than 50% inaccuracy in a metric. Figure 2 shows the normalized IPC values for all benchmarks to the real hardware results. From the ﬁgure we can see that Sniper shows the least error and only include one outlier. For embedded benchmarks, the mean absolute percentage error (MAPE) compared to real hardware runs are: 9.5% and 44.77%for Sniper and Gem5 respectively.

The MAPE for integer benchmarks for Sniper and Gem5 are 12.5% and 41.1%(24.0%excludingoutliers) respectively. For ﬂoating point benchmarks, the MAP Excluding outlier are 8.2% and 28.8% for Sniper and Gem5 respectively.

In addition to comparing the simulated results of the simulators, we compare the time it takes for each simulator to simulate 500 million instructions including the fast forwarding time. Figure7depictstheaveragesimulationtimeforeachsimulatorfor the different types of benchmarks. As shown from the ﬁgure, Sniper is the fastest simulator, followed by PTLsim. gem5 and Multi2Sim show close simulation time on average. It should be noted that the average simulation time is based on only a subset of overall simulation experiments, which were run in an isolated environment on host systems. Our main observations based on our experiments are following:

* The main sources of inaccuracies in simulated statistics are highly over estimated branch mispredictions, imprecise decoding of instructions into micro operations, high cache misses and lack of modelling of all optimization structures of real hardware.
* Although it is widely accepted that simulators cannot provide speed and accuracy simultaneously, for the given target, Sniper shows the most accurate and fastest results out of those studied simulators. Sniper is fast because its simulation model combines both interval and cycle-level simulation. Moreover, Sniper shows the least experimental error as it is the only one of the four simulators that has been modiﬁed and validated for x86 architectures, including Nehalemµ-architecture[32].On the other hand, we did not ﬁnd any validation efforts for x86 processors for other simulators.
* An uncalibrated/unvalidated simulator for a particular architecture can show signiﬁcant simulated performance differences when compared to real hardware. This is also demonstrated in an attempt to calibrate MARSSx86 for a particular target machine.
* A more accurate simulator may still not be able to ﬁt your needs. For instance, Sniper though shows greater accuracy, is not very ﬂexible to allow one to model new microarchitectural features compared to Gem5. On the other hand, Gem5 is more ﬂexible and can help in studying performance of particular micro-architectural blocks.
* For given workloads Sniper appears to be the most accurate, for other type of workloads specially full system workloads it might not show similar accuracy as it does not have extensive device modeling support. On the other hand Gem5 might be better suited for such workloads as it supports full system simulation and supports vast device models.
* Better accuracy and high speed makes Sniper a convincing choice for many-core x86 architectures (specially modern Intel processors like). Gem5’s detailed processor modeling, multi-ISA support, full system simulation support and active development communitymakesitagoodchoicetoperformdetailedexperimentsonaparticularprocessor sub-system, to study OS interaction with hardware or to study interaction of an x86 core with a different ISA core.

# Conclusion

we presented a comprehensive study of x86 architectural simulators. We have surveyed and compared many x86 computer architecture simulators and grouped them in respective categories. We performed veriﬁcation tests of four modern computer architecture simulators and measured their experimental error compared to real hardware runs. The experimental error rate shown by the simulators does not necessarily mean that main cause are bugs in the simulator. Some reasons of inaccuracies include: not modeling all microarchitecture optimization details as real hardware, which makes it harder to validate; varying degree of ﬂexibility and conﬁgurability; inaccurate decoding of instructions into microoperations; and different labeling of microoperations of simulators. The results show that Sniper has the least absolute error. In terms of single-core simulations speed, Sniper comes out at the top with shortest simulation time. However, choosing a simulator can differ depending on the main focus of the research. For example, Sniper is targeted for multicore simulations and is the most accurate among the studied simulators, however; it does not produce detailed performance statistics for simulated system and not very ﬂexible compared to the others. It can be a best choice for some symmetric and asymmetric x86 multicore research project but not for multi-ISA heterogeneous multicore because it only supports x86. On the other hand, gem5 produces very detailed results and can be helpful in studying only particular blocks of processor and supports multiple ISAs. In future work, we will consider digging deeper to ﬁgure more sources of inaccuracies by performing further experimentation using certain µ-benchmarks to ﬁnd more sources of inaccuracies. We will also study some other new x86 simulators.

# References

[1] K.Skadron,M.Martonosi,D.I.August,M.D.Hill,D.J.Lilja,andV.S.Pai,“Challenges in Computer Architecture Evaluation,” Computer, vol. 36, pp. 30–36, August 2003.

[2] <http://www.diva-portal.se/smash/get/diva2:829764/FULLTEXT01.pdf>.

[3] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, “A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization,” IEEE Transactions on Education, vol. 52, no. 4, pp. 449–458, 2009.

[4] R. A. Uhlig and T. N. Mudge, “Trace-driven memory simulation: A survey,” ACM Computing Surveys, vol. 29, no. 2, pp. 128–170, 1997.

[5] T. Nowatzki, J. Menon, C.-H. Ho, and K. Sankaralingam, “Architectural simulators considered harmful,” IEEE Micro, vol. 35, pp. 4–12, Dec. 2015. [6] R. D. D. B. S. Keckler, “Measuring experimental error in microprocessor simulation,” in ISCA, 2001.

[7] T.E.Carlson,W.Heirman,andL.Eeckhout,“Sniper: ExploringtheLevelofAbstraction for Scalable and Accurate Parallel Multi-Core Simulation,” in ACM Int. Conf. for High Performance Comp., Net., Storage and Analysis, pp. 1 – 12, 2011.

[8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, et al., “The gem5 Simulator,” SIGARCH Comp. Arch. News, vol. 39, pp. 1–7, May 2011.

[9] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez, “Multi2sim: A simulation framework to evaluate multicore-multithreaded processors,” in Int. Symp. on Comp. Arch. and High Perf. Comp., pp. 62–68, Oct. 2007.

[10] M. T. Yourst, “PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator,” in IEEE Int. Symp. on Perf. Analysis of Systems & Software, pp. 23–34, 25-27 April, 2007.

[11] T. Austin, E. Larson, and D. Ernst, “SimpleScalar: An Infrastructure for Computer System Modeling,” Computer, vol. 35, p. 59.

[12] P. S. Magnusson, F. Larsson, A. Moestedt, B. Werner, J. Nilsson, P. Stenstr¨om, F. Lundholm, M. Karlsson, F. Dahlgren, and H. Grahn, “SimICS/Sun4m: A VIRTUAL WORKSTATION,” in Usenix Annual Technical Conference, pp. 119–130, Jun. 1998.

[13] T.SherwoodandJ.Y.Joshua,“ComputerArchitectureSimulationandModeling,” IEEE Micro, vol. 26, pp. 5–7, July/August 2006.

[14] M. M. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K.E.Moore,M.D.Hill,andD.A.Wood,“Multifacet’sGeneralExecution-DrivenMultiprocessorSimulator(GEMS)Toolset,”SIGARCHComp.Arch.News,vol.33,pp.92–99, November 2005.

[15] <http://parsa.epfl.ch/simflex/>.

[16] J. H. Ahn, S. Li, O. Seongil, and N. P. Jouppi, “McSimA+: A Manycore Simulator with Application-level+ Simulation and Detailed Microarchitecture Modeling,” in IEEE ISPASS, pp. 74–85, April 2013.

[17] N. Hardavellas, S. Somogyi, T. F. Wenisch, R. E. Wunderlich, S. Chen, J. Kim, B. Falsaﬁ, J. C. Hoe, and A. G. Nowatzyk, “Simﬂex: A Fast, Accurate, Flexible Full-System Simulation Framework for Performance Evaluation of Server Architecture,” ACM SIGMETRICS Perf. Eval. Review, vol. 31, pp. 31–34, Mar. 2004.

[18] R. E. Wunderlich, T. F. Wenisch, B. Falsaﬁ, and J. C. Hoe, “SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling,” in ISCA, pp. 84–95, 9-11 June 2003.

[19] D. Genbrugge, S. Eyerman, and L. Eeckhout, “Interval Simulation: Raising the Level of Abstraction in Architectural Simulation,” in IEEE Int. Symp. on High Perf. Comp. Arch., pp. 1–12, 9-14 Jan. 2010.

[20] L.Eeckhout,ComputerArchitecturePerformanceEvaluationMethods. Morgan&Claypool Publishers, 2010.

[21] G. H. Loh, S. Subramaniam, and Y. Xie, “Zesto: A cycle-level simulator for highly detailed microarchitecture exploration,” in IEEE ISPASS, pp. 53–64, April 2009.

[22] D. Sanchez and C. Kozyrakis, “ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems,” in Int. Symp. on Comp. Arch., vol. 41, pp. 475–486, June 2013

[23] L. Schaelicke and M. Parker, “Ml-rsim reference manual,” Dept. of CSE, Univ. of Notre Dame, Tech. Rep. TR, pp. 02–10, 2002.

[24] A. Patel, F. Afram, and K. Ghose, “MARSS-x86: A Qemu-Based Micro-Architectural and Systems Simulator for x86 Multicore Processors,” in Int. QEMU Users Forum, held in conjunction with Design Automation and Test in Europe, pp. 29–30, 18 Mar. 2011.

[25] P. Crowley and J.-L. Baer, “On the Use of Trace Sampling for Architectural Studies of Desktop Applications,” in Workload Characterization: Methodology and Case Studies, pp. 15–24, Dallas, TX, 1999.

[26] A. Sharma, A.-T. Nguyen, J. Torellas, M. Michael, and J. Carbajal, “Augmint: A Multiprocessor Simulation Environment for Intel x86 Architectures,” Tech. Rep. 1463, Center for Supercomputing Research and Development, UIUC, 28 Mar. 1996.

[27] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, “Multi2Sim: A Simulation Framework for CPU-GPU Computing,” in Int. Conf. on Parallel Arch. and Compilation Techniques, pp. 335–344, 2012.

[28] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, “The M5 Simulator: Modeling Networked Systems,” IEEE Micro, vol. 26, pp. 52–60, July/August 2006. [29] A. Gutierrez, J. Pusdesris, R. G. Dreslinski, T. Mudge, C. Sudanthi, C. D. Emmons, M. Hayenga, and N. Paver, “Sources of Error in Full-System Simulation,” in ISPASS, pp. 13–22, 2014. [30] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, “Accuracy Evaluation of GEM5 Simulator System,” in Int. Workshop on Reconﬁgurable Communication-centric Systems-onChip, pp. 1–7, 2012.

[31] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald III, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, “Graphite: A Distributed Parallel Simulator for Multicores,” in IEEE Int. Symp. on High Perf. Comp. Arch., pp. 1–12, Jan. 2010.

[32] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, “An evaluation of high-level mechanistic core models,” ACM TACO, vol. 11, no. 3, p. 28, 2014.

[33] <http://developer.amd.com/tools-and-sdks/opencl-zone/amd-acceleratedparallel-processing-app-sdk/>.

[34] V. J. Reddi, A. Settle, D. A. Connors, and R. S. Cohn, “PIN: A Binary Instrumentation Tool for Computer Architecture Research and Education,” in Proceedings of the workshop on Comp.arch. education: held in conjunction with ISCA, p. 22, June 2004.

[35] S.Li,J.H.Ahn,R.D.Strong,J.B.Brockman,D.M.Tullsen,andN.P.Jouppi,“McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures,” in IEEE/ACM Int. Symp. on Microarch., pp. 469–480, 12-16 Dec. 2009.

[36] <http://www.intel.com/content/www/us/en/processors/architecturessoftware-developer-manuals.html>.

[37] http://www.intel.com/content/www/us/en/architecture-and-technology /64-ia-32-architectures-optimization-manual.html.

[38] <http://www.agner.org/optimize/instruction_tables.pdf>.

[39] <http://www.realworldtech.com/haswell-cpu/>.

[40] <http://www.anandtech.com/show/6355/intels-haswell-architecture/6>.

[41] <http://xania.org/201602/haswell-and-ivy-btb>.

[42] R. E. Kessler, E. J. McLellan, and D. A. Webb, “The Alpha 21264 Microprocessor Architecture,” in ICCD: VLSI in Computers and Processors, pp. 90–95, 1998.

[43] V. Uzelac and A. Milenkovi´c, “Experiment Flows and Microbenchmarks for Reverse Engineering of Branch Predictor Structures,” in IEEE ISPASS, pp. 207–217, 2009.

[44] T. Hayes, O. Palomar, O. Unsal, A. Cristal, and M. Valero, “Vector extensions for decision support dbms acceleration,” in IEEE/ACM MICRO, pp. 166–176, 2012.

[45] “SPEC CPU 2006.” <https://www.spec.org/cpu2006/>.

[46] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, “Mibench: A free, commercially representative embedded benchmark suite,” in IEEE WW, pp. 3–14, Dec. 2001.

[47] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, “Automatically characterizing largescaleprogrambehavior,”inthe10thInter.Conf.onArchitecturalSupportforProgr. Languages and Oper. Sys., pp. 45–57, October 2002.

[48] “Performance Application Programming Interface.” http://icl.cs.utk.edu/papi/. [Online; accessed 5-August-2015].