x86 Computer Architecture Simulators: A Comparative Study

Ayaz Akram and Lina Sawalha
Department of Electrical and Computer Engineering, Western Michigan University, Michigan, USA
{ayaz.akram, lina.sawalha}@wmich.edu

***Abstract*—The significance of computer architecture simulators in advancing computer architecture research is widely acknowledged. Computer architects have developed numerous simulators in the past few decades and their number continues to rise. This paper explores different simulation techniques and surveys many x86 simulators. Comparing simulators with each other and validating their correctness has been a challenging task. In this paper, we compare and contrast x86 simulators in terms of flexibility, level of details, user friendliness and simulation models. In addition, we measure the experimental error and compare the speed of four contemporary x86 simulators: gem5, Multi2sim, PTLsim and Sniper. We also discuss the strengths and limitations of these simulators. We believe that this paper provides insights into different simulation strategies and aims to help computer architects understand the differences among existing simulation tools.**

I. Introduction

Due to huge costs associated with building real systems for testing and verification purposes, computer architects rely on simulation and modeling techniques to evaluate different design options. Major percentage (approximately 90%) of papers published in top conferences use simulation as performance evaluation methodology [1]. Previous surveys on computer architecture simulators are old and do not include recent simulators [2]. Some of them focus only on teaching or memory related simulators [3], [4]. This paper compares and contrasts x86 simulators based on different characteristics and features, and show several examples of contemporary simulators. x86 is one of the oldest and widely used instruction set architectures (ISAs) in desktops and servers. There is not much literature dealing with the evaluation of x86 simulators by comparing them to each other and to the state-of-the-art processors. The absence of performance validation of simulators may cause experimental errors that can result in incorrect conclusions, if the errors are large. Finding and correcting errors in simulators is difficult because of [5]: implicit assumption of simulator features, absence of detailed documentation, inefficient and mislabeled micro-operations that causes unnecessary register dependencies, etc. Academic researchers often use experimental error of a simulator's ability to model an existing hardware [6], [7]. In this paper, we configure four state-of-the-art x86 computer architecture simulators to model Intel's Haswell microarchitecture. Then we quantify the experimental errors and evaluate the accuracy of such an approach. In addition, we compare the performance/speed results of the simulators among each other, and pinpoint some causes of inaccuracies. This paper also discusses the strengths and limitations of different x86 simulators. The purpose of this paper is not to criticize a particular simulator for showing high experimental error, but to emphasize the importance of validating simulators and to help the community understand some sources of inaccuracies. Our main contributions are:

- An up-to-date survey of various types of simulators that support x86 ISA with a comparison of their features.
- A detailed comparison of four modern x86 simulators: gem5 [8], Multi2sim [9], PTLsim [10] and Sniper [7].
- A comparison of simulation speed of these four simulators, in addition to quantifying the experimental errors of each simulator modeling real hardware.

The rest of this paper is organized as follows: section II discusses x86 simulators' categories with examples. Section III details four contemporary x86 simulators. Section IV discusses our experiments methodology to measure the experimental error, and section V shows the experimental results with analysis. Finally, section VI concludes the paper.

II. Classification and Survey of Simulators

We discuss the classes of x86 simulators below based on: level of simulation details, target's scope, and input type.

A. Functional vs. Timing Simulators

Functional simulators simulate the functionality of the target only and do not model microarchitectural details. Examples of x86 functional simulators are 'sim-safe' and 'sim-fast' models of SimpleScalar [11] and Simics [12].

Timing simulators provide detailed information about the real timing/performance of a simulated computer system or parts of a computer system. Timing simulators that keep track of all cycles on a simulated processor while a benchmark is executing on it, are known as *cycle-level* simulators. Keeping track of such details makes them slow and resource intensive in comparison to functional simulators and other subtypes of timing simulators [13]. For example, SimpleScalar's detailed (cycle-level) simulation model is 25 times slower than its functional model. Other examples of cycle-level simulators are GEMS [14], gem5 [8], PTLsim [10] and Flexus [15].

Event-driven simulators perform simulation corresponding to events, instead of simulating the target on a cycle-by-cycle basis, which makes them faster. It is also possible to combine cycle-level to event-driven simulators by modeling some parts of the simulated structures on a cycle-by-cycle basis and others on an event-driven basis. McSimA+ [16] is a timing simulator that supports both in-order (IO) and out-of-order (OOO) x86 pipelines. SimFlex [17], which is based on sampling microarchitecture simulation (SMARTS) [18]

978-1-5090-5142-7/16/$31.00

framework, and Flexus (Simics) [15] are cycle-level timing simulators while some components are event-driven internally. Many alternatives to cycle-level and event-driven simulators have emerged recently, which provide a tradeoff between simulation speed and accuracy, for example, interval simulation [19]. The fundamental idea of interval simulation is that the miss events like branch mispredictions and cache misses divide the normal flow of instructions through the pipeline into intervals. These miss events can be determined by respective simulator blocks (e.g. branch predictor or memory simulator) and an analytical model can be used to find the timing for each interval. Using both the miss event simulators and mechanistic analytical model, performance information can be acquired. Sniper [7], an x86 multicore simulator, is based on this technique.

B. Application-Level vs. Full-System Simulators:

Application-level/user-mode simulators are able to run only target applications instead of a full fledge target operating system (OS). Thus, they have to simulate microprocessor and only limited peripherals. In user-level simulations, any system service requests by the simulated application/benchmark bypass the user-level simulation and are serviced by the underlying host OS. For compute intensive benchmarks for example SPEC CPU, little time is spent in the execution of system-level code [20], thus relying only on the simulation of user-level code can be a good choice as it is usually faster than full-system simulation. However, for many other workloads (server workloads, transaction processing and database benchmarks), this simulation will not be enough because a considerable amount of time is spent in the execution of system-level code. Application-level simulators are usually less complex but may show some inaccuracies due to lack of system level code support. SimpleScalar, Sniper [7] and McSimA+ [16], Zesto [21] (built on top of SimpleScalar) and ZSim [22] are application-level timing simulators that support x86 ISA.

Full system simulators can simulate an entire OS for the target. I/O devices needed to boot an OS are also simulated. gem5 [8], ML-RSim [23], MARSSx86 [24] and PTLSim [10] are some examples of full system timing simulators.

C. Trace-Driven vs. Execution-Driven Simulators:

Trace-driven simulators use trace files as their inputs. These files contain recorded streams of instructions from a program's run on real hardware. These simulators do not need to emulate the ISA of the target, which makes them relatively simple. One problem with the trace-driven simulators is that the generated trace files can be very large in size and reading such large files from disk can be slow. This can be solved by using trace sampling and reduction techniques [25].

In case of execution-driven simulation, instructions of any particular benchmark are executed on the simulated machine directly. Different performance aspects are calculated at the same time as the program is executing. As opposed to trace-driven simulators, these simulators can simulate misspeculated code paths. But, they are more complex compared to trace-driven simulators and can take longer to run. Examples include SimpleScalar [11], Augmint [26] and gem5 [8].

There are four main factors to consider when comparing different computer architecture simulators: performance, flexibility, detail of simulation model and accuracy. Table I shows a comparison of some detail and flexibility features of several x86 computer architecture simulators. In the next sections, we compare four of these simulators with more details.

III. Selected Simulators for Detailed Study

We selected four simulators for detailed study, $gem5$ [8], $Multi2Sim$ [27], $PTLsim$ [10] and $Sniper$ [7], because they have diverse design strategies with respect to detail and abstraction. All of them are contemporary simulators with active development, except PTLsim that is currently not in active development but still being heavily used. Next, is a brief discussion of all these simulators along with validation efforts that were previously performed for them.

A. gem5

gem5 [8] is a full system simulator that supports many ISAs (x86, ARM, MIPS, Alpha, SPARC) with various CPU models (Table I). gem5 borrows the detailed CPU modeling from M5 [31] and the detailed memory system modeling from GEMS [14]. gem5 primarily supports four CPU models: 'AtomicSimple', 'TimingSimple', 'InOrder' and 'O3'. AtomicSimple and TimingSimple are non-pipelined single-cycle microarchitectures. The 'InOrder' and 'O3' are pipelined IO and OOO core models. Both are 'execute-in-execute' meaning that instructions are executed exactly at the execute stage. These can be configured to simulate different number of pipeline stages, issue widths and number of hardware threads. The O3 model can simulate CPUs with simultaneous multithreading (SMT).

Gutierrez et al. [32] and Butko et al. [33] 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 based), Gutierrez et al [32] were able to achieve a mean percentage runtime error of 5% and a mean absolute percentage runtime error of 13% for SPEC CPU2006 benchmarks [32]. Butko et al. [33] have analyzed the accuracy of gem5 for simulation of a multicore embedded system (ARM Cortex A9 based). Various benchmarks related to scientific workloads (SPLASH-2), media applications (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.

Because gem5 is the most configurable simulator among the four studied ones, it can be a good choice for microarchitecture/architecture optimization research. In addition it is a good fit for heterogeneous multicore research as it supports different architectures and it has been recently integrated to $GPGPUsim$ and $Aladdin$ accelerator simulator.

TABLE I
COMPARISON OF X86 SIMULATORS

Simulator	Supported hosts (ISA/OS)	Category	Supported Processor Models	MultiCore Support
Gem5 [8]	x86, ARM, SPARC, Alpha, PPC/Linux,MacOSx,Solaris,OpenBSD	FS/MOD/TIM (cycle-accuate)	IO, OOO	yes (HMP)
Multi-2sim [27]	x86/Linux	UM/MOD/TIM	OOO, multithreaded	yes (HMP)
Sniper [7]	x86/Linux	parallel UM/TIM (interval simulation)	IO, OOO, SMT cores	yes (HMP)
PTLsim [10]	x86/Linux	FS/TIM simulator (cycle-accurate)	OOO	yes
ZSim [22]	x86/Linux	parallel UM/TIM	IO, OOO	yes (HMP)
Simple-Scalar [11]	x86/Linux, Win2000, SPARC/Solaris	UM/ED/TIM	OOO	no
SIMFLEX [17]	x86	FS/MOD/TIM (decoupled functional and timing)	OOO	yes
SIMICS [12]	Alpha, PPC, UltraSparc, x86/Linux, Windows	FS/functional	functional	yes
Augmint [26]	x86/Unix and Windows NT	ED/TD	functional	yes
Mc-SimA+ [16]	x86/Linux	UM/TIM (decoupled functional and timing)	IO, OOO	yes (HMP)
GEMS [14]	x86/Linux, AMD64-linux, and SparcV9 (Solaris 8)	FS/TIM (decoupled functional and timing)	OOO	yes
MARSSx86 [24]	x86-64/Linux	FS/TIM	IO, OOO	yes
Graphite [28]	x86/Linux	parallel UM/TIM (decoupled)	IO	yes
OVPsim [29]	x86/Windows and Linux	functional	IO	yes
Flexus [15]	x86/Linux	FS/TIM/ED (cycle-accurate)	IO, OOO	yes
Zesto [21]	x86	UM/TIM cycle-level	OOO	yes
McPAT [30]	x86/Linux	power, area and TIM	IO, OOO, NOCs, shared caches	yes (HMP)

Note: UM=user mode, **FS**=full-system, **ED**=execution-driven, **EvD**=event-driven, **TD**=trace-driven, **TIM**=timing, **MOD**=modular, **IO**=in-order, **OOO**=out-of-order, **HMP**=hetrogeneous multiprocessor

B. Multi2Sim

Multi2Sim is another recently developed simulator for CPU-GPU computing, which supports various ISAs (x86, ARM, MIPS, AMD Evergreen, NVIDIA Fermi). Multi2Sim [9] uses three different simulation models: functional, detailed and event-driven. Multi2sim supports only OOO execution mode, and can support SMT. Memory hierarchy and interconnect networks are highly configurable. It does not simulate an OS but can execute parallel workloads with dynamic threads creation. Ubal et al [27] validated Multi2Sim for GPU simulation using commercial AMD Radeon 5870 GPU as a reference model. They verified functional correctness of the GPU simulator. The average percentage error between the execution time and simulated execution time vary from 5% to 30%. We have not found any validation efforts for x86 CPUs on Multi2Sim. Choosing Multi2Sim simulator would work the best for heterogeneous GPU-CPU architectures' related research, as it is validated for GPUs. On the other hand, the CPU side is not very configurable compared to some other simulators.

C. PTLsim

PTLsim is a cycle-level full-system x86 simulator. It can model an OOO core with SMT and a simple non-pipelined sequential core designed for functional simulation only. The default OOO pipeline is based on a combination of features from the Intel Pentium 4, AMD K8, Intel Core 2, IBM Power4/Power5 and Alpha EV8 [10].

Yourst [10] configured PTLsim processor pipeline and memory hierarchy as close as possible to real 2.2 GHz AMD Athlon 64 (K8 micro-architecture) based machine, for experimental evaluation. He used $rsync$, client server benchmark to include kernel-level effects in simulation. By comparing PTLsim statistics and the real K8-based core's performance counters, a 5% error was measured.

D. 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 one-IPC simulation model (one-IPC model is defined as an IO single issue pipeline model). Sniper is based on Graphite [28] 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 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 [34]. Validation of this model against Nehalem microarchitecture showed a single-core error of 11.1% using a subset of SPLASH-2.

TABLE II
FEATURE COMPARISON

Feature	gem5	Sniper	PTLsim	Multi2Sim	ZSim
Platform support	P++	P	P	P+	P
Target support	T++	T	T	T+	T
Full system	✓	X	✓	X	X
Fast forwarding & cache wARMup	✓	✓	X	✓	✓
Checkpointing	✓	X	X	✓	X
Trace generation	✓	✓	✓	✓	✓
Details of generated performance stats.	D++	D	D+	D+	D+
Pipeline depth configuration	✓	X	✓	X	✓
Energy and power modeling	E++	E	E	E-	E
In-order pipeline support	✓	✓	X	X	✓
HMP support	M,G,S	S	X	M,G	S
GPU-Modelling	✓	X	X	✓	X
Multi-threaded app. support	✓	✓	✓	✓	✓
Community support	C++	C++	C-	C	C+

Note: [feature's 1st letter]++ is better than [Feature's 1st letter]+ which is better than [feature's 1st letter] which is better than [Feature's 1st letter]- , S=Single-ISA, M=Multi-ISA, G=GPU

Combining interval and detailed simulation models allows Sniper to run fast as compared to the other three simulators with good accuracy. In addition, as Sniper is the only simulator out of these four simulators that was validated for an x86 Nehalem microarchitecture, it is expected that it is the most accurate simulator. For these reasons, Sniper is the best choice for multi- and many-core x86 architectures for CPU workloads. It can also work well for single-core workloads, depending on the study being done, as Sniper is not very configurable compared to gem5 for example. However, if the target is full-system workloads, such as workloads with large OS interaction, Sniper would not be a good choice as it is a user-mode simulator and does not simulate all OS-hardware interaction.

Simulators' Feature Comparison: Table II shows a comparison among different features and configuration parameters of the aforementioned four simulators in addition to a recently developed simulator, ZSim [22]. gem5 supports the highest number of OSes and architectures as shown in Table I. Fast forwarding and cache warmup are supported by gem5 and Sniper. Multi2Sim and ZSim support only fast-forwarding. PTLsim does not have a working fast forwarding or warmup

functionality. Sniper cannot create checkpoints itself, but it can use checkpoints created by Pin [35] and Simpoint tools. Only gem5 and Multi2Sim support the creation and later use of checkpoints during simulation. All of these simulators support the creation of executed instruction traces during simulation. gem5 produces the most detailed simulated performance statistics (including pipeline stages throughput, instruction mix and performance of various structures).

In gem5, PTLsim and ZSim, branch misprediction penalty can be configured implicitly by varying the pipeline depth. Sniper and Multi2Sim do not have explicit options to configure pipeline depth, but they have options to set misprediction penalty.To support power and energy modeling, gem5 and Sniper provide support for dynamic voltage and frequency scaling (DVFS) to run experiments to simulate energy efficiency. Multi2Sim, gem5, Sniper and ZSim can be easily integrated to McPAT [30] to evaluate power and energy consumption. PTLsim does not provide any support for energy consumption modeling. In terms of the support for heterogeneous multicore (HMP) simulation, Sniper and ZSim support only single-ISA (also known as asymmetric) HMP simulation. gem5 can be modified to simulate single- and multi-ISA HMP system, as it supports multiple ISAs. Multi2Sim has extensive support for GPU-CPU computing simulation. gem5 and Sniper have large development communities and support forums. Multi2Sim and ZSim also maintain support groups but there is currently no maintained support forum for PTLsim.

IV. Experimentation Methodology

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

A. 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 Intel documentation [36], [37] and other resources [38], [39], [40], [41]. The basic features of this target system are in Table III. The simulators are configured to model a 19-stage pipeline with 14-cycles misprediction penalty. All simulators in this study do not model fused μ-ops, thus we use pipeline stages' widths in equivalent mico-operations (μ-ops). As we do not have the exact specifications of branch predictors in Haswell μ-architecture, and not all simulators implement the same branch predictors, we have tried to configure similar tournament branch predictors in all simulators with the available branch predictor implementations. We use a 4K-entries branch target buffer (BTB) and 16-entries return address stack (RAS) with a tournament predictor where at least one of them is a 2-level predictor. Details of these predictors are available in next sub-sections.

TABLE III
Target Configurations.

Parameter	Core i7 Like
Pipeline	Out of Order
Fetch width	6 instructions per cycle
Decode width	4-7 fused μ-ops
Decode queue	56 μ-ops
Rename and issue widths	4 fused μ-ops
Dispatch width	8 μ-ops
Commit width	4 fused μ-ops per cycle
Reservation station	60 entries
Reorder buffer	192 entries
Number of stages	19
L1 data cache	32KB, 8 way
L1 instruction cache	32KB, 8 way
L2 cache size	256KB, 8 way
L3 cache size	8 MB, 16 way
Cache line size	64 Bytes
L1 cache latency	4 cycles
L2 cache latency	12 cycles
L3 cache latency	36 cycles
Operation latency	Based on [37], [38]
Branch predictor	Hybrid
Branch misprediction penalty	14 cycles
Branch target buffer	4096 entries, 4 way
Return Address Stack	16 entries
Physical Int/FP registers	168 each

B. Configurations of Simulators for Core i7 Target

Our main objective while configuring simulators is to be as close as possible to the real target system. Another, important aim is to configure these simulators as close as possible to each other, so that we could perform an impartial comparison. Although these simulators have diverse support for configuration parameters, we tried our best to keep the simulation configurations the same. Next, is a brief description of the configuration settings on these simulators.

gem5's pipeline configuration is very flexible, in fact it is the most flexible out of the four studied simulators. A 19-stage pipeline is set by configuring different delay values 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 of fused 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]. This scheme maintains local and global history to predict the direction of a given branch, where a choice predictor selects one out of the two predictions. Both tables of local predictor are 2k entries each while global, choice predictor and direct-mapped BTB are 4K entries each. Each entry contains a 2-bit counter.

In Multi2Sim, the configured branch predictor is a tournament predictor consisting of a bimodal predictor and two-level adaptive predictor in addition to a set associative BTB with similar sizes as other simulators. PTLsim is also quite flexible in terms of feature configurability. PTLsim directly fetches pre-decoded μ-ops from a basic block cache (a simulator structure, used to speed up simulation). Branch prediction is configurable in PTLsim. It includes a hybrid G-share and

bimodal predictors and set-associative BTB, each set to 4k entries. PTLsim supports distributed and shared issue queues. It is not possible to model a shared instruction queue as well as clusters in PTLsim. Thus, distributed instruction queues (with extra entries for compensation) are configured as done by Hayes et al [43]. In case of Sniper, we have overwritten the basic configurations with those for Haswell. Available pipeline configurations are also changed according to values in Table III. The branch predictor modeled in Sniper is based on Intel Pentium M's branch predictor [44]. We have changed the table sizes of this branch predictor in the code itself so that global predictor, bimodal predictor and BTB each are 4K entries in size. 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. More details on each simulator's configurations can be found in [45].

C. Experimental Workloads

We ran our experiments using SPEC CPU 2006 and a subset of the MiBench benchmark suites. We simulated complete MiBench benchmarks, however; running complete SPEC benchmarks in simulated environment can take impractically long time. Thus, each application was executed for 500 million x86 instructions chosen from a statistically relevant portion of programs [46]. Simulation is fast-forwarded for all benchmarks and simulators are warmed up for 100 million instructions. We changed Multi2Sim and PTLsim's code to support the warmup functionality.

D. Performance Measurement on Real Hardware

We used PAPI [47] to measure IPC (instructions per cycle) values for complete MiBench runs. In case of SPEC CPU, we measured the IPC values for the same 500 million instructions interval that we simulated. Benchmarks were run five non-consecutive times and results were averaged and compared with simulation results to measure the experimental error. We also used PAPI to measure the number of cache misses and branch mispredictions on the real hardware. The compiler that we used is *gcc 4.4.7*. We started with 32-bit binaries for all simulators because Multi2Sim supports only 32-bit binaries, but many benchmarks failed to run on gem5, so we have used 64-bit binaries for gem5 only. The 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 machine. Reference hardware performance metrics are measured on both systems as well. We have used gem5's version of Sep. 2015, Multi2Sim 5.0, Sniper 6.0 and PTLsim version available at [48] for all experiments.

V. Results and Analysis

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 includes all benchmarks, *avgERORR*, and the other excludes outliers, *avgERROR-NO*; where an outlier corresponds to more than 50% inaccuracy in a metric.

Fig. 1 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%, 44.6% (37.6% excluding outliers), 38.2% and 47.06% (41.8% no outliers) for Sniper, gem5, PTLsim and Multi2Sim respectively. The MAPE for integer benchmarks for Sniper, gem5, PTLsim and Multi2Sim are 12.5%, 41.1% (24.0% excluding outliers), 36.5% and 57.4% (40.4% excluding outliers) respectively. For floating point (FP) benchmarks, the MAPE excluding outliers are 8.2%, 28.8% and 41.4% for Sniper, gem5 and Multi2Sim respectively. PTLsim shows much greater inaccuracy in FP benchmarks, with most of the benchmarks showing more than 50% error (PTLsim shows very high μ-ops to instructions ratio for most of them). One of the reasons for lower inaccuracy for FP benchmarks on all simulators is the limited configurability of FP instructions' latencies and limited support for x87 FP on all simulators.

We have been able to observe various reasons which can contribute to overall inaccuracy shown by the simulators. For example, one of the primary reasons for the high underestimation of IPC for some benchmarks in PTLsim is the way x86 instructions are decoded into μ-ops. We observed that for all benchmarks including outliers, which have high inaccuracies in IPC values for PTLsim, the ratio of committed μ-ops to committed x86 instructions is very high compared to other simulators. For example, using PTLSim the ratio is 9 for $gemsFDTD$ (1.36 for gem5 and 2.27 for Multi2Sim), 6.07 for $gamess$ (1.5 for gem5 and 2.16 for Multi2Sim), 5.43 for $povray$ and 4.2 for $soplex$. As all pipeline widths are configured in the number of μ-ops, thus when this ratio is very high, the overall IPC can diminish significantly.

Figures 2 and 3 show normalized L1 data cache misses and L3 cache misses per instruction for SPEC CPU2006 benchmarks. Fig. 4 shows normalized values of branch mispredictions to hardware performance counters results. There are many cases where the average error goes above 100%. These figures help to understand very high underestimation of IPC values by simulators for some benchmarks. For example, most of the outliers in gem5 in Fig. 1 ($h264ref$, gcc_200, $gobmk$, $perlbench$, $libquantum$, $namd$, $povray$) have much higher branch mispredictions and cache misses compared to real hardware runs. The reason for these inaccuracies in simulated branch predictors is that the modeled branch predictor fails to mimic the behavior of branch predictor of the real core for these benchmarks. The benchmarks that have high overestimated branch mispredictions have significant number of committed branch instructions (20% or more) compared to other benchmarks (4-5%). This exposes the inefficiency of modeled

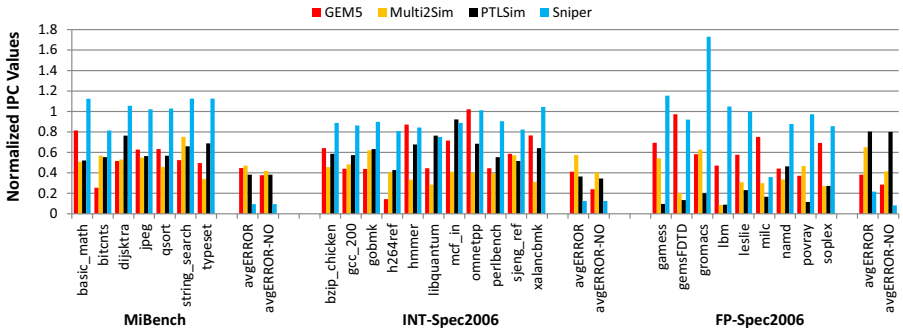

Fig. 1. Normalized IPC values for all benchmarks

branch predictor compared to Haswell branch predictor. Note that in this work, we only configure the simulator to measure experimental error and we do not change the simulators' code to model real hardware. Other sources of IPC inaccuracies can be due to the way some of the x86 instructions are decoded and implemented in gem5. For example, integer division is implemented using loops of division μ-ops, where each μ-op computes a single bit of the quotient (each μ-op will have minimum one cycle latency). This implementation is different than the division algorithm on the real hardware. In addition, some inaccuracies in gem5 were seen due to the mislabeled and inefficient microoperations [5].

Similarly for Multi2Sim most outliers (gcc_200, $h264ref$, $hmmer$, $perlbench$, $gemsFDTD$, $namd$) have high overestimated branch mispredictions, that obviously highly affects overall IPC values. As was observed for gem5, most of these benchmarks have high number of overall committed branch instructions. It should be noted that the high overestimated branch mispredictions might not always explain underestimated IPC values. For example, branch mispredictions for $gromacs$ on Sniper are highly overestimated, however; the total number of branch mispredictions is too low to affect overall IPC value (Sniper highly overestimates the IPC value for $gromacs$). It is worth noting that most of the benchmarks that show highly overestimated L3 cache (L3$) misses ($h264ref$, $xalancbmk$, $perlbench$, $leslie3d$, $namd$) by the simulators have low number of overall L3$ misses on the real hardware. As a result this high inaccuracy might not affect the overall IPC accuracy significantly. Another reason for high L3$ misses inaccuracy can be the accumulation of inaccuracies from L1$ and L2$ into L3$. We also noted that the inaccuracy in L2$ misses to L3$ misses ratio is lower than the inaccuracy of L3$ misses alone. Other sources of inaccuracies are resulted because of the lack of support of fused μops, and μop cache of Haswell by the simulators, which can significantly reduce the effective pipeline depth in case of μop cache hits. The abstract level of some simulators and the lack of other microarchitectural details can also induce errors in simulation models.

As many studies rely on the relative performance of simulators to model a new architecture, we ran two relative performance experiments: one after reducing the pipeline stages' width, and the other for using half size of all caches. Figures 5 and 6 show the percent change in IPC values for pipeline width and cache size change respectively, from the normal configurations in table III. For many benchmarks, the simulators show similar relative change in performance. PTLsim is the least sensitive on average among all other simulators. Multi2Sim seems to be more sensitive to change in cache sizes than other simulators as its memory mode is pretty detailed. Sniper and gem5 show similar sensitivity to cache size change. They also show, on average, very close IPC change in pipeline width, being more sensitive than PTLSim and Multi2Sim.

In addition to comparing the simulators accuracy, we also compare the time it takes for each simulator to simulate 500 million instructions, including fast forwarding time, for SPEC bencmarks, and complete embedded benchmarks. Fig. 7 depicts the average simulation time for each simulator for the different types of benchmarks. As shown from the figure, Sniper is the fastest simulator. Simulators speed in the figure is based on a subset of overall simulation experiments, which were run in an isolated environment on host systems. Fast forwarding is the fastest on Sniper. Multi2Sim and PTLsim take relatively similar time to fast forward while gem5 is the slowest. For example, to fast forward 50 million instructions for mcf, Sniper took 2sec. Mult2sim, PTLsim took $\approx$ 5X, while gem5 took $\approx$ 11X that of Sniper. Our preliminary experiments with ZSim [22] shows that it is faster than Sniper with a higher average experimental error than Sniper but lower than the other simulators. Our main observations based on our experiments are following:

- The *main* sources of inaccuracies in simulated statistics are: highly overestimated branch mispredictions, imprecise decoding of instructions into microoperations, high cache misses, and the lack of modeling of all optimization structures and instruction latencies of real hardware.
- Although it is widely accepted that simulators cannot provide speed and accuracy simultaneously, for a given target, Sniper shows the most accurate and fastest results out of these four 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 Nehalem μ-architecture [34]. On the other hand, we did not find any validation efforts for x86 processors for other simulators.
- Sniper though shows greater accuracy, is not very flexible

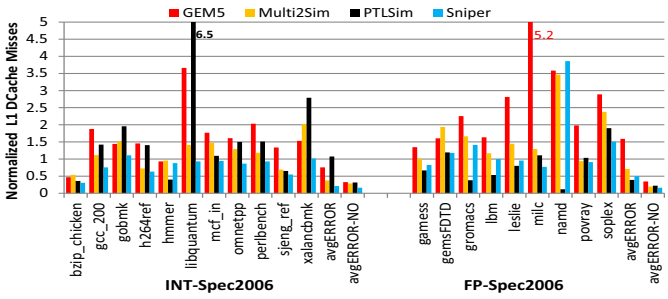

Fig. 2. Normalized L1 DCache Misses/Instruction for SPEC benchmarks

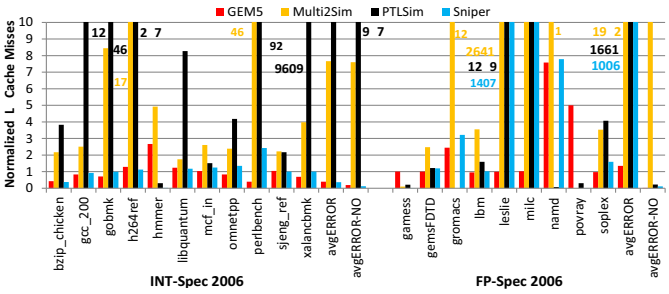

Fig. 3. Normalized L3 Cache Misses/Instruction for SPEC benchmarks

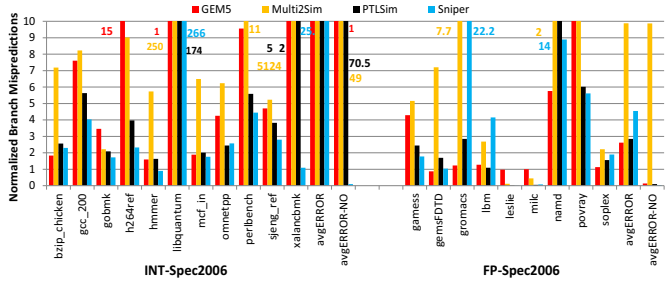

Fig. 4. Normalized branch mispredictions per instruction for all benchmarks

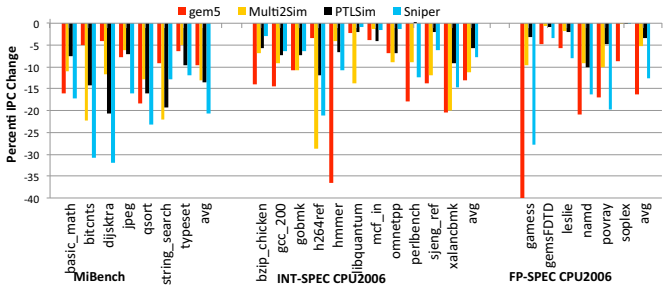

Fig. 5. Normalized IPC values for reduced pipeline width

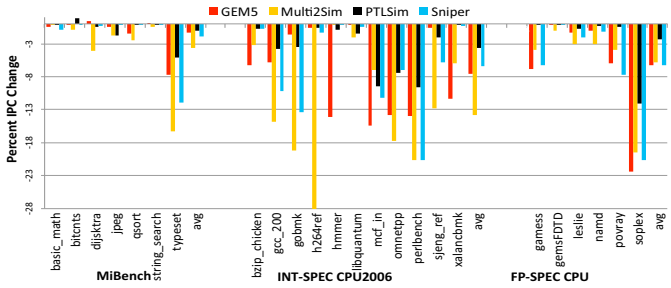

Fig. 6. Normalized IPC values for reduced cache size

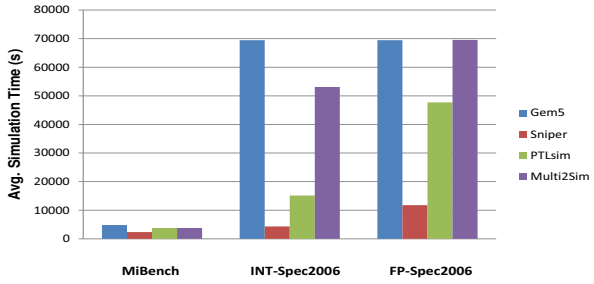

Fig. 7. Average simulation time for all benchmarks

to allow one to model new micro-architectural features compared to gem5. Sniper does not support full-system simulation and the effect of OS for applications. It is best use is for x86 many-core user-mode workloads. On the other hand, gem5 and PTLsim are more flexible and can be used for studies of particular microarchitectural blocks, and full-system workloads, with gem5 being more configurable and showing higher accuracy.

- An unvalidated/uncalibrated simulator for a particular architecture can show significant simulated performance differences when compared to real hardware, which confirms the conclusion of recent work on calibrating MARSSx86 [49].
- 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 community makes it a good choice to perform detailed experiments on a particular processor sub-system, multi-ISA heterogeneous multicore studies, OS interaction with hardware related studies, etc. Multi2Sim can be a good option for CPU-GPU architecture simulation. PTLsim can serve as base x86 simulator to develop more diverse simulators, such as in MARSSx86 [24] simulator.

VI. Conclusion

In this paper, 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 measured the experimental error of four modern simulators compared to real hardware runs. The experimental error rate shown by the simulators does not necessarily mean that main causes are bugs in the simulator, but that simulators need to be calibrated to model an actual hardware with high accuracies. Some reasons of inaccuracies seen in the simulators 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 micro-operations; and different labeling of micro-operations of simulators. The results show that Sniper has the least absolute experimental 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 performed studies or research, which also include requirements for: targeted workloads, microarchitecture optimizations used/required, configurability of simulator, estimated simulator's code modifications, OS interaction

requirements, energy studies, etc. In future work, we will consider making changes to these simulators to further validate them, discover more sources of inaccuracies and perform more relative performance runs. We will also use μbenchmarks to study the effect of different structures on inaccuracies. In addition, we plan to study other new x86 simulators such as ZSim and MARSSx86.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable feedback.

REFERENCES

[1] K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, and V. S. Pai, "Challenges in Computer Architecture Evaluation," *Computer*, vol. 36, pp. 30–36, Aug. 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, pp. 449–458, Sep. 2009.

[4] R. A. Uhlig and T. N. Mudge, "Trace-driven memory simulation: A survey," *ACM Computing Surveys*, vol. 29, pp. 128–170, Jun. 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. Desikan, D. Burger, and S. Keckler, "Measuring experimental error in microprocessor simulation," in *ISCA*, pp. 266–277, Feb. 2001.

[7] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation," in *ACM Int. Conf. for High Performance Comp., Net., Storage and Analysis*, pp. 1 – 12, Nov. 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, Apr. 2007.

[11] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An Infrastructure for Computer System Modeling," *Computer*, vol. 35, p. 59, Aug. 2002.

[12] P. S. Magnusson, F. Larsson, A. Moestedt, B. Werner, J. Nilsson, P. Stenström, 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. Sherwood and J. Y. Joshua, "Computer Architecture Simulation and Modeling," *IEEE Micro*, vol. 26, pp. 5–7, Jul./Aug. 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, and D. A. Wood, "Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) Toolset," *SIGARCH Comp. Arch. News*, vol. 33, pp. 92–99, Nov. 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, Apr. 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, *Computer Architecture Performance Evaluation Methods*. Morgan & Claypool Publishers, Dec. 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, Apr. 2009.

[22] D. Sanchez and C. Kozyrakis, "ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems," *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 Microarchitectural 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] 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.

[29] "Technology ovpsim." http://www.ovpworld.org/technology_ovpsim.

[30] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. 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, Dec. 2009.

[31] 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, Jul./Aug. 2006.

[32] 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, Mar. 2014.

[33] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy Evaluation of GEM5 Simulator System," in *Int. Workshop on Reconfigurable Communication-centric Systems-on-Chip*, pp. 1–7, 2012.

[34] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM TACO*, vol. 11, p. 28, Oct. 2014.

[35] 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, Jun. 2004.

[36] http://www.intel.com/content/www/us/en/processors/architectures-software-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] 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, Dec. 2012.

[44] V. Uzelac and A. Milenković, "Experiment Flows and Microbenchmarks for Reverse Engineering of Branch Predictor Structures," in *IEEE ISPASS*, pp. 207–217, May 2009.

[45] A. Akram and L. Sawalha, "A comparison of x86 computer architecture simulators," Tech. Rep. TR-CASRL-1-2016, Western Michigan University, Kalamazoo, MI, 2016.

[46] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *ASPLOS*, pp. 45–57, Oct. 2002.

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

[48] https://github.com/stephand/ptlsim.

[49] M. Asri, A. Pedram, L. K. John, and A. Gerstlauer, "Simulator calibration for accelerator-rich architecture studies," in *SAMOS*, Jul. 2016.