International Journal of Latest Trends in Engineering and Technology
IJLTET Special Issue- ICRACSC-2016 , pp.147-153
e-ISSN: 2278-621X

CHARACTERIZATION AND DYNAMIC MITIGATION OF EMBEDDED MEMORY PREFETCHERS

M Devendra[1] and Dr. K.E. Sreenivasa Murthy[2]

Abstract—In emerging and future high-end processor systems, tolerating increasing latency and properly managing memory bandwidth will be critical to achieving high performance. Prefetching, in both hardware and software, is among one of the most important available techniques for doing this analysis. The analysis of energy consumption and performance is essential to design and optimize mobile systems because of their limited battery capacity. Full-system simulation provides detailed performance metrics for an entire system. Thus it has been widely used for designing and optimizing Chip Multiprocessor (CMP) micro architectures. The gem5 simulator provides full-system simulation based on the ARM architecture. Furthermore, gem5 provides only performance statistics without power consumption data. This paper presents a mobile full-system simulation framework based on an enhanced gem5 that includes power models for the main components of mobile systems: CPU/caches, DRAM. Using SPEC CPU2006 benchmarks, we show that the proposed mobile full-system simulator achieves power modeling accuracy within 12.8% error rate, compared with Nexus 5.

Keywords—component, high-end processor, energy consumption, battery capacity Chip Multiprocessor, mobile full-system simulator

I.INTRODUCTION

With more and more cores packaged in a Chip Multiprocessor (CMP), memory hierarchy is becoming one of the key design problems. But unfortunately, for almost all of computer architectures, quantitative evaluation of memory subsystems is possibly only by using simulators [1]. In other words, simulation is indispensable for a computer architect. Yet even as many simulators and other simulating methods have been presented for CMP researchers, to date there is still some main problems remained unsolved. One such a problem is unbearable simulating speed. It is virtually impossible to simulate all benchmarks of a whole suite to completion, especially by using those full-system simulators [1]. Typically, an architect simulates a subset of benchmarks, using a reduced or truncated input set or a representative set instead, to minimize the simulating time. As a result of trading speed for accuracy, these simulators bring out the second problem: poor accuracy, while an evaluation with poor accuracy is un-trusted. Besides, these simulators are often hard to use or to implement new architecture solutions. To address these problems, this paper proposes a fast and cycle accurate memory subsystem evaluation framework, called gem5 simulator.

II. RELATED WORK

During the past decades, many simulators have been used in architecture performance evaluation and optimization field. Practically, there are three main types of them: execution-driven simulation, instrumentation-driven simulation and trace-driven simulation. Execution-driven simulation relies on existing functional performance models to execute a binary application [2], [3], [4], [5]. The functional performance model provides the memory addresses in real time. A cache simulation model starts after receiving the access information. GEMS [4], as a simulation tool set, is a typical case of execution-driven simulator, created by Wisconsin Multifaceted Project, to characterize and evaluate the performance of multiprocessor hardware systems. Fahringer et al. [5] have used an execution-driven model to analyze performances of distributed and parallel systems. Woo et al. [2] have also used an execution-driven simulating model with the Tango Lite Tracing tool [3].
An instrumentation-driven simulator uses a dynamic binary instrumentation tool to gather the running information of an application at any expected instrumented point [6], [7]. Then, after receiving the running information such as memory addresses, the cache model works. A binary instrumentation approach is relatively faster, suitable for conducting accurate memory performance studies. Jaleel et al. [6] have described an instrumentation-based approach to characterize memory behaviors of workloads. They use CMPsim, an instrumentation driven memory simulator, to evaluate the memory performance. Since binary instrumentation normally occurs at native execution speed, an instrumentation-driven simulator can run at MIPS (Million

[1] *Research Scholar, ECE, Rayalaseema University Kurnool, AP, INDIA*
[2] *ECE Department G. Pullaiah College of Engineering and Technology, Kurnool, AP, INDIA*

Instructions Per Second) speed. However, they do not simulate a detailed timing model, and do not provide a prefacer either. Above all, CMPsim is not available to public.

A trace-driven simulator uses some specific tool(s) to collect the memory references (called an address trace) of a running application, and applies the trace sequence to the simulation model to mimic the way that a real processor might exercise the design [8], [9]. A trace-based simulation method is conceptually simple, and easy to reappear experimental results. Since functional models of modern ISAs are considerable slow and complicated to be built. Trace-driven simulation is more popular for conducting memory subsystem performance studies [8], such as the performance characterization and performance optimization. Uhlig et al. [8] have made a detailed survey for existing trace-driven simulators. They have investigated more than 50 trace-driven simulators and showed that none of them is best when all criteria (including accuracy, speed, memory, flexibility, portability, ease-of-use, etc.) are considered together. Simple Scalar [9] is one of the hugely popular set of simulation tools during the past 15 years. But in this multicore era, on-chip cache access model cannot be applied to it directly. Other researchers have also proposed simulation environments to meet their own research needs. Li et al. [10] use IBM's Turandot Power Timer to generate single-core L2 cache access traces that are annotated with timestamps and power values, then feed these traces to a cache simulator developed by themselves. The simulator uses hits and misses to shift the time and power values in the original traces. They propose joint optimization across multiple design variables. Bononi et al. [11] use OMNeT++ simulation Framework to analysis some architectures for network on chip. Sun et al. [12] construct a prototype using a public domain network simulator ns-2 and evaluated design options for a specific network-on chip (NoC) architecture. In summary, to our knowledge, few researchers have comprehensively considered many cores and mutual effects of timing components under the CMP environment; namely, CMP researchers still lack of a useful timing-detailed memory subsystem performance model to support the x86 CMP platform. Some full system simulators such as SIMICS [13] and M5 [14] do support x86 ISA, but they are bulky and emulate the whole system with peripherals and the operating system: apt to be accurate but much slower. It is expected that this paper fills the gap.

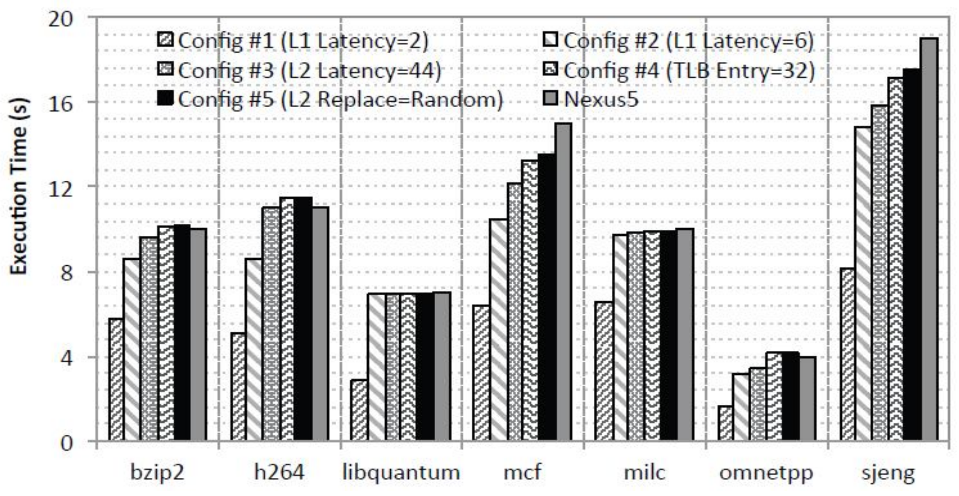

Fig. 1. The execution time comparison of SPEC CPU2006 benchmarks for various configurations. The configurations of L1/L2 cache latency, TLB, and L2 cache replacement policy are same as TABLE II, and the rest of the configurations is same as TABLE I. Note that the Config #5 has the minimum error rate of 4.4% compared to Nexus 5.

TABLE I MOBILE FULL-SYSTEM SIMULATOR CONFIGURATION AND NEXUS 5 SPECIFICATION

Type	**Mobile Full-System Simulator based on gem5**	**Nexus 5**
Execution Core	300Mhz ~ 2.26GHz quad-core CMP, ARM ISAs, out of order, tournament branch predictor, 4,096 BTB entries, 64 reorfer buffer, 32 fetch queue	300Mhz ~ 2.26GHz quad-core Krait 400, 11 stage integer pipeline with 3-way decode, 4-way out-of-order speculative issue superscalar execution, 7 execution ports, Pipe lined VFPv4, 128-bit wide NEON
GPU	-	Adreno 330, 450 MHz

Caches	L1 I/D-cache: 16kB, 4-way, private, 64B block size, LRU, 4 mshrs, 6 hit latency L2 cache: 2MB, 8-way, shared, 64B block size, random, 20 mshrs, 44 hit latency	L0 I/D-cache: 4kB direct mapped, L1 I/D-cache: 16kB, 4-way L2 cache: 2MB, 8-way
DRAM	LPDDR3 800MHz, 2GB, Dual-channel, 32-bit bus width, 8 banks per rank, 4kB row buffer size, 1 rank per channel	LPDDR3 800MHz, 2GB, Dual-channel, 32-bit bus width (12.8GBps)
TLB	I-TLB: ArmTLB type, 32 entry, D-TLB: ArmTLB type, 32 entry	I-TLB: 32 entry, D-TLB: 32 entry
Type	**Mobile Full-System Simulator based on gem5**	**Nexus 5**
Execution Core	300Mhz ~ 2.26GHz quad-core CMP, ARM ISAs, out of order, tournament branch predictor, 4,096 BTB entries, 64 reorfer buffer, 32 fetch queue	300Mhz ~ 2.26GHz quad-core Krait 400, 11 stage integer pipeline with 3-way decode, 4-way out-of-order speculative issue superscalar execution, 7 execution ports, Pipe lined VFPv4, 128-bit wide NEON
GPU	-	Adreno 330, 450 MHz
Caches	L1 I/D-cache: 16kB, 4-way, private, 64B block size, LRU, 4 mshrs, 6 hit latency L2 cache: 2MB, 8-way, shared, 64B block size, random, 20 mshrs, 44 hit latency	L0 I/D-cache: 4kB direct mapped, L1 I/D-cache: 16kB, 4-way L2 cache: 2MB, 8-way
DRAM	LPDDR3 800MHz, 2GB, Dual-channel, 32-bit bus width, 8 banks per rank, 4kB row buffer size, 1 rank per channel	LPDDR3 800MHz, 2GB, Dual-channel, 32-bit bus width (12.8GBps)
TLB	I-TLB: ArmTLB type, 32 entry, D-TLB: ArmTLB type, 32 entry	I-TLB: 32 entry, D-TLB: 32 entry

III. Performance Validation of Full-System Simulation based on ARM Architecture CPU Models

In [15] [16], the authors evaluated the performance accuracy of the gem5 full system simulator based on ARM architecture CPU models by comparing the execution times of benchmarks on hardware development boards. Butko et al. [15] used three types of benchmarks, which are SPLASH-2 for scientific research, ALP Bench for media, and STREAM for engineering, on the Snowball SKY-9500-ULP-C01 development kit. Gutierrez et al. [16] used SPEC CPU2006 and PARSEC benchmarks on the versatile express TC2 board. They could achieve accuracy within 20% error rate on average. However, they did not verify their method on the mobile platform such as Android, and did not use real mobile applications. We evaluate web browsing performance and the execution times of SPEC CPU2006 benchmarks on Android-based full-system simulation in comparison with Nexus 5.

TABLE II Configurations for Validating Simulation Performance

Configuration	**#1**	**#2**	**#3**	**#4**	**#5**
L1 Cache Latency (cycles)	2	6	6	6	6
L2 Cache Latency (cycles)	20	20	44	44	44
TLB Entry (count)	64	64	64	32	32
L2 Cache Replacement	LRU	LRU	LRU	LRU	RANDOM

IV. GEM 5 ENHANCEMENTS

Our simulator is an enhanced version of gem5. First, we made two simulated systems, which are an Android-based mobile system and an embedded Linux based server system, and they are connected by a simulated Ethernet. gem5 supports only Alpha systems for connected two systems simulations [17]. We modified the configuration files of gem5 to allow two simulated systems that support ARM architectures to use a network with IP addresses. Second, we upgraded the Android version to Jelly Bean 4.1, while gem5 supports the ICS 4.0 version. Before the Jelly Bean 4.1 version, Android uses the *display event* thread in surface flinger, which composes all surfaces into a single display buffer. The thread generates many instructions because of no sleeping thread due to software rendering without a GPU. From the Jelly Bean 4.1 version, Android uses the *vsync* thread that sleeps for the designated duration while waiting for a next sync event. Third, we use a reconfigured Linux Kernel with version 3.14, which enables networking and the *netem* tool to be used. Fourth, we modified the DVFS handler in gem5 to enable two-system simulations. The original DVFS handler in gem5 has a problem of handling only one system, when two simulated systems are working. Finally, we modified the output mechanism of the simulation for each application trace. Our simulator can dump out simulation results when a context switching of processes occurs or at every designated periodic time.

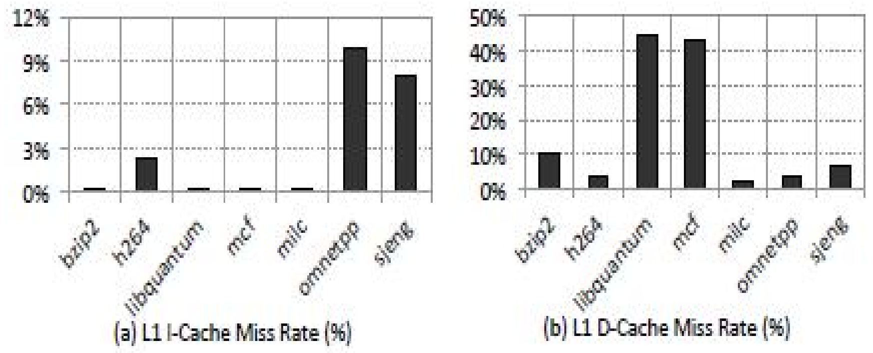

Fig. 2. L1 I/D-cache miss rates for configuraiton #2 in TABLE II. Notethat the benchmarks, which have high error rates such as h264 (21.3%),mcf (30.2%), omnetpp (20.6%), and sjeng (22.0%), have higher miss rates for L1 I/D cache than other benchmarks, which are bzip2 (13.4%), libquantum (0.9%), and omnetpp (2.5%). This means L2 cache hit latency is the one of error causes.

V. PERORMANCE VALIDATION USING NEXUX-5

To minimize performance errors of our simulator, we validated execution times of simulations, which run SPEC CPU2006 and BBench benchmarks, by comparing with Nexus 5. As a result, we discovered the cache hit latency and replacement policy are very important factors.

1) Execution Time Comparison using SPEC CPU2006

benchmarks: To evaluate CPU performance, we chose seven benchmarks from SPEC CPU2006, and we prepared execution binaries for the ARM architecture to run on Nexus 5 and our simulator. We also controlled input parameters of each benchmark for appropriate execution times on CPU core 0 in the simulations. At first, we used default configurations, which are given in gem5, which is configuration #1 in TABLE II.

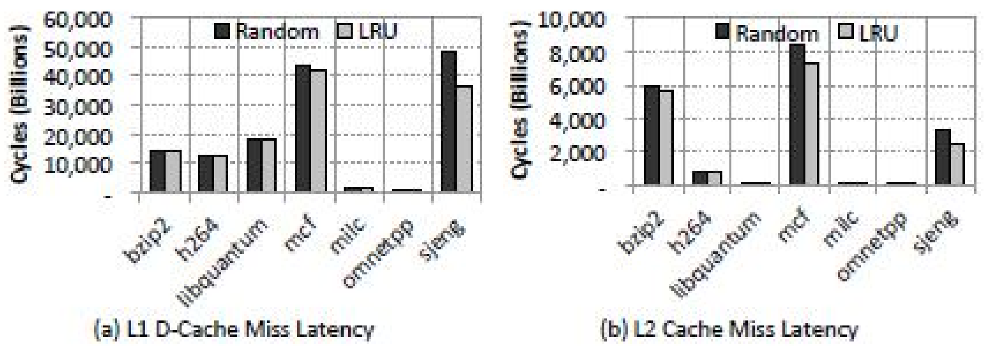

Fig. 3. The miss latencies of L1 D-cache and L2 cache for configurations #4 and #5. We can see that the benchmarks, which have high error rates with configuration #4 such as mcf (11.9%) and sjeng (9.9%), has the larger differences of miss latency between random and LRU replacement policies of L2 cache than other benchmarks. This means the performance with configuration #5 is more accurate than other configurations.

However, execution times of all benchmarks in the simulation are almost about a half of actual execution times, in which the average error rate is 51.9%, in Nexus 5, as shown in Fig. 1. To reduce the error rate, we applied the configurations #2 and #3 based on the experimental results of 7-Zip LZMA benchmark [18] and our observations. We can see that h264, **omnetpp,** and **sjeng** have high L1 I-cache miss rates, and bizp2, lib-quantum, and **mcf** have high L1 D-cache miss rates, as shown in Fig. 2. Those benchmarks also have the highest error rates with configuration #2. High L1 cache miss rates mean that those benchmarks need many L2 cache accesses. In addition to hit latency of caches, we discovered that the cache replacement policy is important to reduce performance errors. We can see that the error rates for execution times of **mcf** and **sjeng** are only 10%, as shown in the Config #5 bar of Fig. 1. So, we adopted the random replacement policy into the L2 cache instead of the LRU policy based on an ARM reference manual [19] and our observations. These replacement policies make a difference to L2 cache latency only for mcf, sjeng, and bzip2, as shown in Fig. 3. Thereby, we can confirm that the hit latencies of L1 and L2 caches and the L2 cache replacement policy are the main reasons for performance errors. We achieved the error rate of only 4.4% with configuration #5.

VI. POWER MODELS

Energy converters, which were introduced in Section IV, are based on our power models of the main components of a mobile system: CPU/caches, DRAM, network interfaces (3G/4G/WiFi), and display. Because our power models are based on a statistics file, which is the results of a full-system simulation, and a system configuration file, we can analyze not only the overall energy consumption of a mobile system, but also the energy consumption of each hardware component. In addition to the analysis of the hardware energy consumption, our energy converters with power models allow us to analyze the energy consumption of each application.

A. Power Modeling of Main Components of a Mobile System

1) CPU/Caches: Our power model for CPU and caches is based on the McPAT framework [26] and its ARM Cortex-A9 CPU power model, which provides multi-core processor configurations with out-of-order processor cores *_NumCores Pcore*, a shared cache *PL2Cache*, a network onchip *PNoC*, a memory controller *PMC*, a flash controller *PFC*, a PCIe controller *PPCIeC*, and a network interface unit *PNIU*, as shown in the following Equation (1):

$$Pcpu = NumCores\_\ Pcore + PL2Cache + PNoC + PMC + \mathrm{p}FC + PPCIeC + PNIU \qquad (1)$$

Each *Pcore* consists of five units: the instruction fetch unit *PIF* , renaming unit *PR*, load store unit *PLS*, memory management unit *PMM*, and execution unit *PE*, as shown in the following Equation (2):

$$Pcore = PIF + PR + PLS + PMM + PE \qquad (2)$$

To calculate the energy consumption based on CMOS circuits for Equation (1) and Equation (2), we need various parameters of dynamic, short-circuit, and leakage power [26], as shown in Equation (3), where *α* is an activity factor, *C* is total load capacitance, *Vdd* is supply voltage, Δ*V* is voltage swing during switching, *fclk* is a clock frequency, *ES* is shortcircuit energy per switch operation, and *Ileakage* is leakage current:

$$PCMOS = \alpha CVdd\Delta V\, fclk + \alpha ESfclk + VddIleackage \qquad (3)$$

Equation (3) is divided into *dynamic power* (*αCVdd*Δ*V fclk* + *αESfclk*) and *static power* (*VddIleakage*) by the influence of activity factor *α*. The static power has two types, which are subthreshold and gate leakages. As a threshold voltage is lower and channel length is shorter, the *subthreshold leakage* tends to increase. Also, thinner gate oxide increases the *gate leakage* [26]. McPAT uses an XML interface with various hardware components as inputs, but it cannot support DVFS (dynamic voltage and frequency scaling). Therefore, we modified gem5 to report when the frequency and voltage change by DVFS governors. Moreover, to automate execution of McPAT, we made an McFAT converter tool, which generates an XML input file from a system simulation configuration file and a statistics file as simulation results, and then reports the final energy consumption results after automatic execution of McFAT. Fig. 4 shows an example of the CPU power breakdown when BBench is executed with DVFS performance governor with a simulation configuration as in TABLE I.

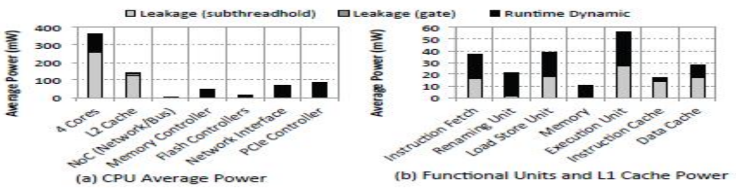

Fig. 4. The CPU power example of BBench execution with the DVFS performance governor.

2) DRAM: We use gem5's DRAM model [30] based on Micron DDR3, as our DRAM power model. The gem5 DRAM model shows in average latency 1%, and in average power 3% inaccuracies [30], compared to DRAMSim2. The DRAM power model consists of precharge PPRE, write PWR, background precharge PPRE BACK, refresh PREF , activation PACT , read PRD, and background activation PACT BACK powers, as shown in Equation (4):

$$PLPDDR3 = PPRE + PWR + PPRE\ BACK + PREF + PACT + PRD + PACT\ BACK \qquad (4)$$

Fig. 5 shows an example of DRAM power breakdown when a web browser loads BBench benchmarks in the simulation with a configuration as in TABLE I.

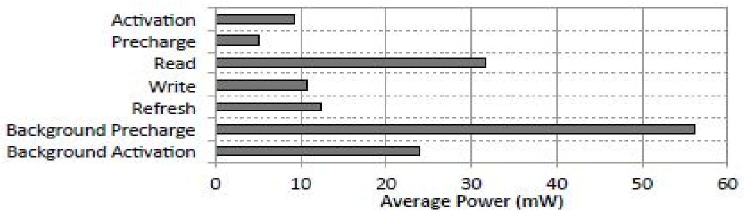

Fig. 5. The DRAM power example of BBench execution with the DVFS performance governor. The total DRAM average power is 149.4mW.

VII. POWER MODEL EVALUATION

To evaluate our power models, we compared the average power of the real measurements on Nexus 5 and a mobile full-system simulator with configuration as in TABLE I.

1) The Comparison for Overall Power Model: For the validation of the overall mobile system, we compared the energy consumption of various types of applications, which are a web browser, a slideshow, a utility, and a game, as shown in Fig. 6. For web browsing, we executed the BBench benchmarks and conducted that the web browser loads only BBC website, which takes about 2 seconds, for a peak power validation. In a slideshow, a QuickPic application loads five images and shows for 10 seconds. For utility and game applications, we launched the calculator and TalkingTom applications and waited for 10 seconds. As a result, error rates with the *ondemand* governor range from 1.5% to 20.6%, and are 11.3% on average. With the *performance* governor, the error rates range from 4.6% to 20.7%. We achieved the total average error rate of 12.8%.

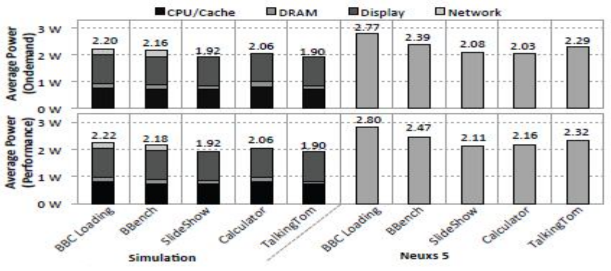

Fig. 6. The average power comparison of mobile applications on a mobile full-system simulator and Nexus 5.

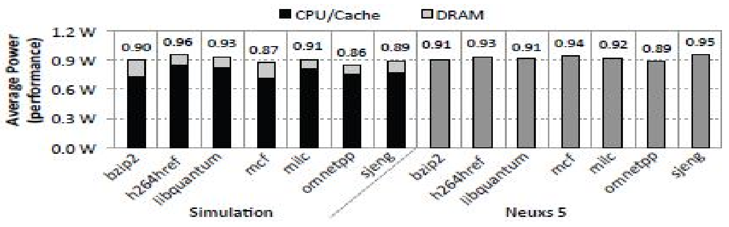

Fig. 7. The average power comparison of SPEC CPU2006 benchmarks on a mobile full-system simulator and Nexus 5

2) SPEC CPU2006 Comparison for CPU, Caches, and DRAM Power Models: For more accurate validation of CPU/caches and DRAM, which are the important components in the aspect of applications, we executed SPEC CPU2006 benchmarks on Nexus 5 with turning off a screen and all network interfaces such as cellular, WiFi, and Bluetooth.

Fig. 7 compares the average power of each benchmark. We can see that the bzip2, mcf, and sjeng benchmarks, which have many memory operations, consume more power than other benchmarks. The error rate of the simulated power consumption is 3.5% on average compared to the measurements on Nexus 5. CPU/caches and DRAM power models are more accurate. Thereby, the absence of GPU and other components in a mobile system incurs some inaccuracy of overall power models, but the difference of total average power compared to Nexus 5 is small.

VIII. CONCLUSION

Because of the limited battery capacity of mobile systems, the analysis of performance and energy consumption is very important in mobile systems. Full-system simulation with OS enables system designers and analysts to explore and optimize many kinds of hard wares and future mobile systems. To apply this full-system simulation to mobile systems based on wireless communication, various networking environments should be reflected in the simulator, because network quality and conditions affect performance and energy consumption. Although gem5 provides the full-system simulation, it does not provide a method to configure various network conditions, and to analyze overall energy consumption. To address these limitations, this paper presents a mobile full-system simulation framework called *MofySim*, which provides the configuration of network conditions and power models for CPU/caches, DRAM, network interfaces, and display. To evaluate the performance of our simulator and power models, we compared the execution times and the average power consumption of mobile applications, BBench, and SPEC CPU2006 benchmarks on our simulation with Nexus 5. We found that the cache hit latency and replacement policies are the sources of errors. The results show that the error rate of the power models is 12.8%, and the inaccuracy of the performance is 4.4% for SPEC CPU2006 and 26.8% for web page loading. Finally, we demonstrated convincingly that mobile systems depending on various network types and network errors such as packet losses have different characteristics. We also found energy inefficiency and unnecessary tasks such as the progress bar and intermediate rendering images, on network delays due to packet losses by analyzing the energy consumption of each application and the behaviours of CPU frequency migrations by the DVFS on demand governor.

REFERENCES

[1] J. J. Yi and D. J. Lilja, "Simulation of computer architectures: Simulators, benchmarks, methodologies, and recommendations," IEEE Transactions on Computers, vol. 55, pp. 268–280, March 2006.

[2] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodology considerations," in Proceedings of the 22nd International Symposium on Computer Architecture (ISCA), Santa Margherita Ligure, Italy, 1995.

[3] S. R. Goldschmidt and J. L. Hennessy, "The accuracy of trace-driven simulations of multiprocessors," Tech References Rep. CSL-TR-92-546, Stanford University, Sept. 1992

[4] M. Martin, D. J. Sorin, B. M. Beckmann, and M. R. Marty, "Multifacet's general execution-driven multiprocessor simulator(gems) toolset," ACM SIGARCH Computer Architecture News, vol. 33, n.4, Nov. 2005.

[5] T. Fahringer, B. Scholz, and X.-H. Sun, "Execution-driven performance analysis for distributed and parallel systems," in In 2nd International ACM Sigmetrics Workshop on Software and Performance (WOSP 2000), Ottawa, Canada, Sep 2000.

[6] A. Jaleel, "Memory characterization of workloads using instrumentationdriven simulation - a pin-based memory characterization of the spec cpu2000 and spec cpu2006 benchmark suites," tech. rep., VSSAD, 2007.

[7] A. Jaleel, S. R. Cohn, C.-K. Luk, and B. Jacob, "Cmp$im: A binary instrumentation approach to modeling memory behavior of workloads on cmps," tech. rep., UMD-SCA, 2006

[8] R. A. Uhlig and T. N. Mudge, "Trace-driven memory simulation: A survey," ACM Computing Surveys, vol. 29, 1997. [11] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," IEEE Computer, vol. 35(2), 2002.

[9] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," IEEE Computer, vol. 35(2), 2002.

[10] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron, "Cmp design space exploration subject to physical constraints," in 12th International Symposium on High Performance Computer Architecture (HPCA-12), 2006.

[11] L. Bononi and N. Concer, "Simulation and analysis of network on chip architecture: Ring, spidergon and 2d mesh," in Proceedings of the conference on Design, automation and test in Europe: Designers' forum, 2006.

[12] Y. Sun, S. Kumar, and A. Jantsch, "Simulation and evaluation for a network on chip architecture using ns-2," in 20th IEEE Norchip Conference, 2002.

[13] P. Magnusson, M. Christensson, J. Eskilson, and D. Forsgren, "Simics: A full system simulation platform," IEEE Computer, vol. 35(2), Feb. 2002.

[14] N. L. Binkert, R. G. Dreslinski, J. Eskilson, and D. Forsgren, "The m5 simulator: Modeling networked systems," in IEEE Micro, vol. 26, no. 4, July/August, 2006.

[15] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy evaluation of gem5 simulator system," in Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on. IEEE, 2012, pp. 1–7.

[16] A. Gutierrez, J. Pusdesris, R. G. Dreslinski, T. Mudge, C. Sudanthi, C. D. Emmons, M. Hayenga, and N. Paver, "Sources of error in fullsystem simulation," in Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on. IEEE, 2014, pp. 13–22.

[17] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modelling networked systems," *IEEE Micro*, no. 4, pp. 52–60, 2006.

[18] 7-CPU, LZMA benchmark results of Qualcomm Krait 300. [Online]. Available: http://www.7-cpu.com/cpu/Krait.html [Accessed: August 10, 2015].

[19] ARM, Cortex-A15 technical reference manual. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0438c/ CHDDDHFD.html [Accessed: August 15, 2015].