System-Level Power Estimation of SSDs under Real Workloads using Emulation

Sangmin Kim, Kwanghyo Ahn, Changhoon Han, Hyunsik Kim, Jaewoo Im
Samsung Electronics Co., Ltd., Hwaseong-si, Korea
sang.min.kim@samsung.com, kwanghyo.ahn@samsung.com, ch4.han@samsung.com,
hs1107.kim@samsung.com, jaewoo.im@samsung.com

***Abstract*-As the performance of Solid-State Drives (SSDs) increases, the importance of system-level power estimation is also increasing. In this paper, we propose a system-level power estimation methodology for SSD designs that uses silicon measured power of each command. We use emulation to verify the hardware along with the software under real workloads. Experiment results show average and peak power was estimated with a maximum error of 8%.**

I. INTRODUCTION

Due to faster data transfer speed requirements, the performance of Solid-State Drives (SSDs) is constantly increasing. SSDs now support the PCIe 5.0 interface with a sequential read speed of up to 13,000 megabytes per second (MB/s) [1]. Higher performance often results in larger power consumption, which is why accurate power estimation is important. Fig. 1 shows the components of an SSD. An SSD consists of the controller (SoC), the memory storage (NAND), and cache (DRAM) components. For the controller we are able to use a mature SoC power estimation methodology, but the power consumption of other components are hard to estimate.

There have been previous attempts to estimate the power of memory subsystems. J. Ji et al. used an equation-based method to estimate NAND and DRAM power [2]. Using power models, they were able to estimate average power values. However, this method does not consider how power fluctuates during a command and is not able to estimate the power at a certain time. Therefore, they cannot estimate the peak power. Peak power consumption estimation of a device is also important, since the instantaneous current consumption should not pass the PMIC's max input current.

In this paper, we introduce our SSD system-power estimation methodology. Estimating power of realistic I/O operation usage scenarios requires a verification methodology that can handle the large design size of SSD controllers, which is usually over tens of millions of instances, and support the firmware. Simulation time needs to be relatively long (100ms), as peak power usually occurs only on corner cases. We use an emulation based hardware/software co-verification method to satisfy the hardware and software requirements. Fig. 2 shows our UVM based verification environment, which is accelerated using hardware emulator, similar to the method presented by A. Jain et al. [3]. For memory subsystems, we use internal status registers of emulator behavior models to analyze which command each component is running. We use real silicon measurements of each command to convert the command information to power values. This allows concurrent estimation of each component's power and thus the average and peak power consumption of SSDs. We compare the SSD power estimation results with silicon measurements.

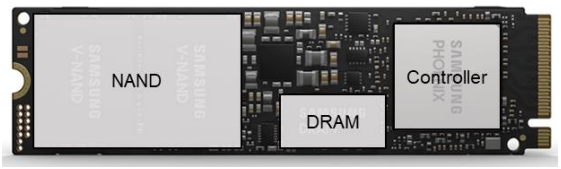

Figure 1. Components of an SSD

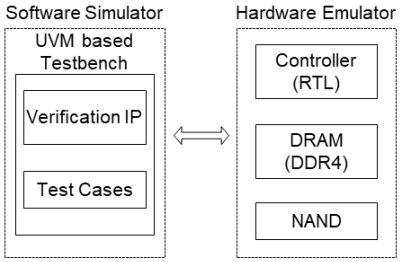

Figure 2. Verification environment

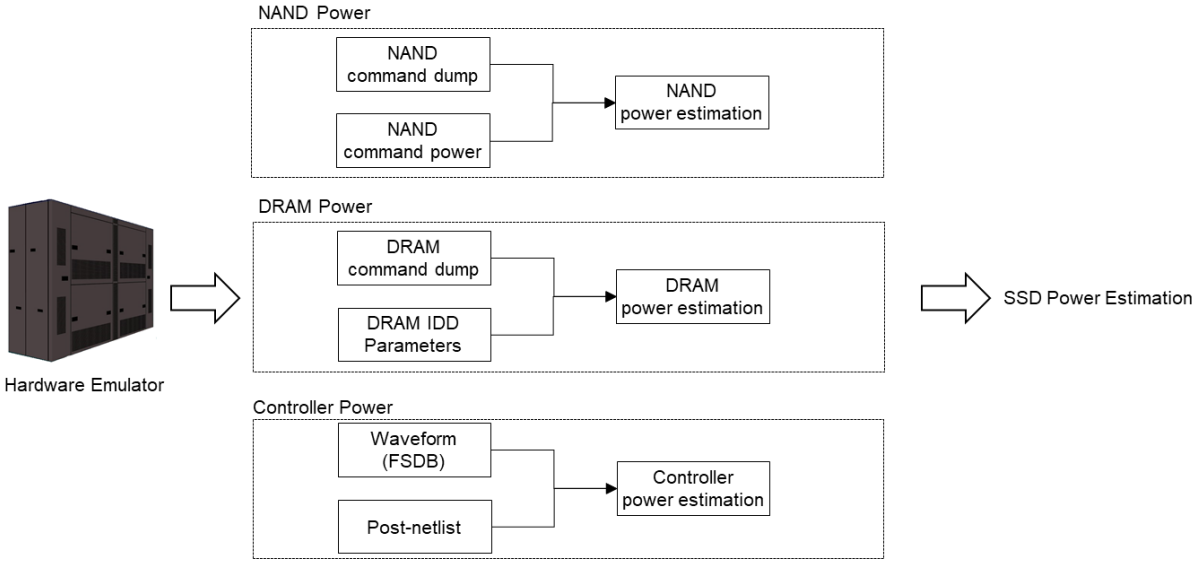

Figure 3. Overview of SSD system-level power estimation methodology

II. SSD System-Level Power Estimation

In this section, we describe in detail how we perform power estimation and how we obtain the required inputs for power estimation, especially the DRAM and NAND command logs. Fig. 3 is an overview of our SSD system-level power estimation methodology. We estimate SSD system power by estimating the power of each component, NAND, DRAM and Controller. We run real usage scenario simulations by utilizing a hardware emulator.

A. NAND Power

For NAND power estimation, we modify the behavior model and create an internal status register that represents the command status. The active commands we consider are Read, Program, Erase, Read DMA, and Write DMA. We also output if the NAND is in single-level cell (SLC) mode or triple-level cell (TLC) mode. We create status bits that represent multi-plane operations also. We create a dump file for each NAND channel's status registers. Fig. 4(a) is an example with a 1-channel 2-way NAND configuration. The command dump log has the timestamp and command status of each way (Fig. 4(b)).

We measure the input power values for each NAND command, while considering SLC/TLC mode and multi-plane operations. To increase accuracy of our simulations, we set the AC timing parameters of the behavior model to the parameters measured alongside the power, so that the simulated SSD read/write performance will match the measured performance. We apply the measured NAND command power when each command occurs to estimate the power of each NAND channel, and accumulate the values to obtain the NAND power.

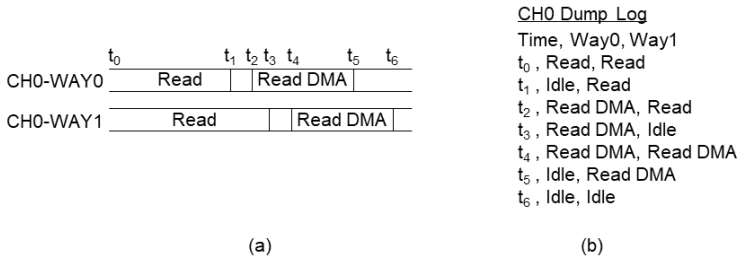

Figure 4. (a) NAND operation of a 1-channel 2-way configuration (b) NAND command dump log

B. DRAM Power

For estimating DRAM power, we modify the emulator DDR4 memory model [4]. Although the model is mostly encrypted, monitoring signals are provided in a non-encrypted module to help analyze the DDR4 traffic. We use the command condition flags, which activate when the corresponding command is detected, to create the DRAM command dump log by detecting precharge, activate, write, and read commands. We disable the condition flag for the refresh command, as the rapid occurrence of the command increases the size of the dump file, which makes it problematic when running multi-millisecond simulations.

Power calculation is similar to the method presented in DRAMPower [5]. We use the command dump log and the measured IDD current values to calculate the power of each command. Refresh power calculation uses the average refresh interval, tREFI, and the refresh cycle time, tRFC, from the JEDEC DDR4 standard [6], and burst refresh current IDD5B.

C. Controller Power

We estimate the controller power using a standard SoC power estimation flow. We obtain the FSDB dump files and input it to Synopsys PrimePower [7], a gate-level netlist power estimation tool. To improve the correlation with the power measurements of a silicon chip, we set the process, temperature, and voltage conditions to be similar to nominal operating conditions. To analyze peak power situations,

We also would like to note that by probing the state signals of the controller we are able to track the controller operation at the peak power situation. We could also perform power-aware simulations by adding UPF files and tracking the isolation/retention/power gating signals. This would allow us to check the state of each power domain during the simulation. However, for our usage cases we only estimated the SSD power for maximum performance situations, which did not require considering power saving modes.

D. SSD System Power

Using the above-described NAND, DRAM, and controller estimated power values, we estimate the SSD system power. We estimate system power differently for systems that have an internal PMIC or use an external PMIC. For systems with an external PMIC, we only need to estimate the sum of the currents of each component. For systems with an internal PMIC, we measure the PMIC efficiency for varying output currents to create an efficiency graph similar to the one used by L. G. Salem and P. P. Mercier [8]. We then convert the PMIC output current values to obtain the PMIC input current.

III. EXPERIMENTAL RESULTS

We applied our system-level power estimation methodology to a commercial SSD and compared average (Table I) and peak power (Table II) to silicon chip measurements. The controller design size is on the scale of tens of millions of instances. We used the Veloce hardware platform to perform our power estimation [9]. However, since we do not use a vendor specific feature, we can use other emulation platforms in a similar way. Also for smaller designs, we can use Verilog simulations to obtain power estimation results since concept-wise our methodology does not require emulation. We normalize the estimated values in the tables to the measured power. Identical product firmware was used during silicon measurement and simulation so firmware operation difference does not cause estimation errors. We cannot perform estimation on a similar time range to silicon chip measurements, as the silicon chip measurements were performed on a seconds range and estimations were performed for 10~100 milliseconds.

TABLE I
AVERAGE POWER ESTIMATION RESULTS

Component	Sequential Read		Sequential Write		Random Read		Random Write	
	Measured	Estimated	Measured	Estimated	Measured	Estimated	Measured	Estimated
NAND	1.00	0.97	1.00	1.05	1.00	0.91	1.00	1.05
DRAM	1.00	0.94	1.00	1.05	1.00	0.90	1.00	0.90
CTRL	1.00	1.01	1.00	1.00	1.00	1.00	1.00	0.97
Total	1.00	0.98	1.00	1.03	1.00	0.95	1.00	1.02

TABLE II
PEAK POWER ESTIMATION RESULTS

Component	Sequential Read		Sequential Write		Random Read		Random Write	
	Measured	Estimated	Measured	Estimated	Measured	Estimated	Measured	Estimated
NAND	1.00	0.88	1.00	0.95	1.00	0.93	1.00	0.97
DRAM	1.00	0.78	1.00	0.91	1.00	0.84	1.00	0.87
CTRL	1.00	1.00	1.00	0.98	1.00	0.96	1.00	0.95
Total	1.00	0.92	1.00	0.96	1.00	0.95	1.00	0.96

We estimated power of 4 I/O performance scenarios, Sequential Read/Write and Random Read/Write. For average power, the average error is 3%, and the largest error is 5%. If we look at each component, the maximum error of NAND was 9%, DRAM 10%, and controller 3%. SSD power estimations and measurements have a periodic waveform, which occurs multiple times inside our estimation time range, which is why we were able to estimate the average power estimation accurately.

Peak power estimations have an average error of 5%, and the largest error is 8%. One of the reasons we think the accuracy drops compared to average power is due to cases where peak power occurs in corner cases. If our simulation does not include the corner case where the silicon chip measurement peak occurs, we would miss these peak power events. These would require specific test cases to recreate the peak power scenario, which the verification engineers would be able to collect over time when validating similar designs.

IV. SUMMARY

We propose a system-level power estimation methodology that uses silicon measured command power to estimate memory subsystems. We obtain the command info by modifying the memory behavior models. Emulation is used to accelerate the verification time. Results show that the average and peak power was estimated with a maximum error of 8%.

REFERENCES

[1] Samsung Press Release, “Samsung Develops High-Performance PCIe 5.0 SSD for Enterprise Servers”, 2021.

[2] J. Ji, C. Wang and X. Zhou, "System-Level Early Power Estimation for Memory Subsystem in Embedded Systems,” *2008 Fifth IEEE International Symposium on Embedded Computing*, 2008, pp. 370-375.

[3] A. Jain, P. Gupta, H. Gupta and S. Dhar, “Accelerating System Verilog UVM Based VIP to Improve Methodology for Verification of Image Signal Processing Designs Using HW Emulator,” *International Journal of VLSI design & Communication Systems*, vol. 4, no. 6, Dec. 2013.

[4] Siemens EDA, “Veloce DDR4 Softmodel User Guide”, Software Version 4.0.0, 2021.

[5] K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn, and K. Goossens, “DRAMPower: Open-Source DRAM Power & Energy Estimation Tool,” http://www.drampower.info.

[6] JEDEC, "DDR4 SDRAM Standard," http://www.jedec.org/standards-documents/docs/jesd79-4a.

[7] Synopsys, “PrimePower User Guide,” 2021.

[8] L. G. Salem and P. P. Mercier, "A Battery-Connected 24-Ratio Switched Capacitor PMIC Achieving 95.5%-Efficiency," *2015 Symposium on VLSI Circuits*, 2015, pp. C340-C341.

[9] C. Selvidge and V. Chobisa, “The Veloce Strato Platform: Unique Core Components Create High-Value Advantages,” Whitepaper, Siemens EDA.