DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator

Shang Li, Zhiyuan Yang, Dhiraj Reddy, Ankur Srivastava, and Bruce Jacob

Abstract—DRAM technology has developed rapidly in recent years. Several industrial solutions offer 3D packaging of DRAM and some are envisioning the integration of CPU and DRAM on the same die. These solutions allow higher density and better performance and also lower power consumption in DRAM designs. However, accurate simulation tools have not kept up with DRAM technology, especially for the modeling of 3D DRAMs. In this letter we present a cycle-accurate, validated DRAM simulator, and DRAMsim3, which offers the best simulation performance and feature sets among existing cycle-accurate DRAM simulators. DRAMsim3 is also the first DRAM simulator to offer runtime thermal modeling alongside with performance modeling.

Index Terms—DRAM, cycle-accurate, simulation, 3D-modeling, thermal modeling

1 Introduction

DRAM technology has emerged through out the years led by industry efforts. Other than the already widely used DDR4 introduced in 2012, GDDR5 and GDDR5X was developed to serve graphic applications with tremendous memory bandwidth increase. On the embedded market, LPDDR3 and LPDDR4 were introduced to meet the low power consumption demands. The most interesting of all, in the high-end market, is the stacked DRAM technology. By stacking DRAM dies and connecting them with TSVs, supported and controlled by a bottom logic layer, these DRAMs can achieve very high density and better performance per package than planar DRAMs. The most representative stacked DRAM technologies nowadays are Hybrid Memory Cube (HMC) and High Bandwidth Memory (HBM). The former utilizes high speed SERDES interface to hide the internal details of DRAM while the latter doesn't impose the logic die standard but only defines how the DRAM dies should operate. All these new DRAM technologies comes with new features that potentially boost the performance of a DRAM and lower the power consumption. For example, DDR4 introduced bankgroup architecture; GDDR5/GDDR5x has doubled/quadrupled the data transfer rate with the introduction of a seperate clock domain; LPDDRx devices are tuned to consume much less power with features such as bank-level refresh; High Bandwidth Memory has dual-issue command interface on 8 128-bit buses; Hybrid Memory Cube radically changes the interface and adds more channels to a single package (up to 32).

Additionally, stacked DRAMs propose new challenges in thermal design and modeling. Traditional planar DRAM dies have less concerns in thermal issues, and DRAM protocols enforce constraints on DRAM timings that may prevent thermal issues. However, in stacked DRAM, thermal modeling becomes a more serious issue because DRAM dies in the middle are more difficult to cool down, and as a result, overheated DRAM die may cause data loss or security issues. Current public DRAM simulators are not capable of modeling 3D DRAM thermal characteristics as well as timings.

• *The authors are with the Department of Electrical & Computer Engineering at the University of Maryland, College Park, MD 20742. E-mail: {shangli, zyyang, dhiraj, ankurs, blj}@umd.edu.*

Manuscript received 9 Sept. 2019; accepted 7 Oct. 2019. Date of publication 14 Feb. 2020; date of current version 31 July 2020.
(Corresponding author: Shang Li.)
Digital Object Identifier no. 10.1109/LCA.2020.2973991

To address these issues, we developed DRAMsim3, a successor to DRAMSim2 [14]. DRAMsim3 is fully capable of simulating and modeling almost all modern DRAM protocols along with many of their unique features (seen in Table 1). It also has a thermal modeling component that can model thermal status of DRAM dies on the fly.

2 Simulator Design & Capability

In this section we introduce the design and features of DRAMsim3 as well as how we bridge the architecture simulation with thermal modeling.

2.1 Simulator Design and Features

We build the simulator in a modular way that it not only supports almost every major DRAM technologies existing today, but it also supports a variety of features that come along with these technologies. The idea is to first build a generic parameterized DRAM bank model which takes DRAM timing and organization inputs, such as number of rows and columns, the values of *tCK, CL, tRCD*, etc. Then we build DRAM controllers that initialize banks and bankgroups according to which DRAM protocol it is simulating, and enable controller features that are only available on such DRAM protocol. For example, dual-command issue is only enabled when simulating an HBM system while *t32AW* enforcement is only enabled when simulating a GDDR5 system. On top of the controller models, we build the system-level interfaces to interact with a CPU simulator or a trace frontend. This interface can also be extended to add additional functionality, and we add a cycle-accurate crossbar and arbitration logic specifically for HMC to faithfully simulate its internals.

This parameterized simulator design allows us to add basic support for new protocols as simple as adding a text configuration file without compiling the code. It also enables us to customize protocol-specific features modularly without affecting other protocols. In our code repository, we ship more than 80 configuration files for various DRAM protocols.

DRAMsim3 uses Micron's DRAM power model [12] to calculate the power consumption on the fly, or it can generate a command trace that can be used as inputs for DRAMPower [3]. While there are no public power profiles for some of the DRAM protocols, we try our best to create power profiles for these protocols based on published literature. The power data can be fed into an plugable thermal model running side-by-side or standalone, we will further demonstrate it in Section 2.2.

The software architecture of the simulator is shown in Fig. 1 and the new features are listed in Table 1.

DRAMsim3 can be integrated into popular CPU simulators or simulation frameworks such as SST [13], ZSim [15] and Gem5 [1] as their memory backend simulator. We will open-source the code repository as soon as this paper publishes, along with the glue code to work with above stated simulators.

2.2 Bridging Architecture and Thermal Modeling

Traditional HotSpot based DRAM thermal modeling tools such as [8] usually require one or several power traces generated beforehand. This causes a dilemma: we need to generate a lot of power traces which may be redundant; otherwise when a management is triggered, the existing power trace may not accurately reflect the tuned behavior. Our simulator solves this dilemma by embedding thermal simulation into performance simulation.

Fine-grained thermal simulation can be time consuming due to the amount of calculations need to be done, therefore we offer the freedom to adjust the granularity in both spatial and temporal domains so that the user can choose accordingly and balance the

TABLE 1
Improved and Unique Features of DRAMsim3

Improved Features (vs previous DRAMsim)	Unique Features (vs all other DRAM simulators)
Bankgroup timings	
Self-refresh timings	Performance & Thermal Co-simulation
(GDDR5) t32AW	DDR4 Verilog Validation
(GDDR5X) QDR mode	Cycle-accurate HMC logic simulation
Bank-level refresh	HBM Dual command issue
Flexible address mapping	

simulation speed versus accuracy. For spatial granularity, each DRAM die is divided into smaller grids that in reality would correspond to DRAM subarrays (shown as Fig. 2c). By default it's 512×512 cells but users can also use larger grid to speed up simulation with less accuracy. For temporal granularity, the transient thermal calculation is done once per epoch, and the epoch length can be configured as an arbitrary number of DRAM cycles.

During each thermal epoch, the thermal module needs to know A) how much energy is consumed on that die, and B) what is the energy distribution (in physical location). We use Micron's DDR power model to calculate power and given the time in cycles we can calculate energy. The energy can be broken down into per-command energy (e.g., activation, precharge, read and write) and background energy. We assign those per-command energy only to those locations that the command concerns, for instance, we only distribute the activation energy to wherever the activated row is on the die. Then we distribute the background energy across the whole die evenly.

To know exactly the location to map the per-command energy, the physical layout of the DRAM circuit needs be known. Unfortunately, most of the DRAM circuit designs and layouts are proprietary information that is not publicly available. According to the reverse-engineered results shown in a recent research [7], DRAM manufacturers obfuscate DRAM cell locations by remapping the address bits. *i.e.* the DRAM address sent by the controller is remapped internally in the DRAM circuitry and as a result, the row and column in the controller's view may end up in a different physical row and column on the DRAM die. For example, if, like [7] discovered, the column address sent by controller is internally decoded as:

$$C_{10}...C_3C_2C_1C_0 \rightarrow C_{10}...C_4C_2C_1C_0C_3$$

where C_i is the ith bit of column address, the controller's view of columns 8, 9 and 10 would actually be physical columns 1, 3, and 5. Note that this rearranging is transparent to DRAM controller and works independently from the address mapping that controller has to perform.

To accurately model this, we implement a location mapping function which allows users to input any arbitrary address bits

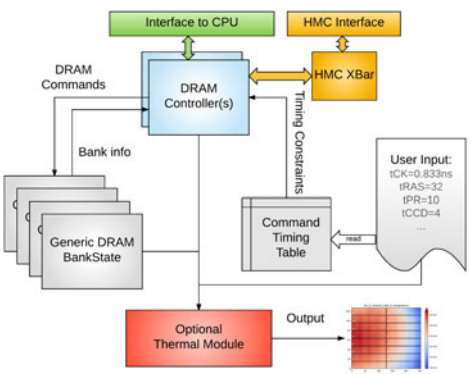

Fig. 1. Software architecture of DRAMsim3.

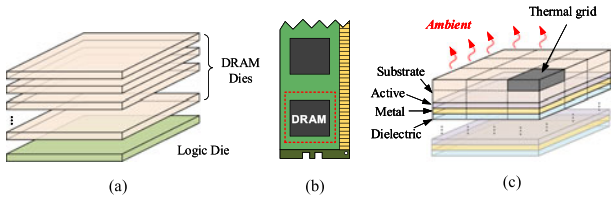

Fig. 2. Illustration of (a) the 3D DRAM, (b) memory module with 2D DRAM devices and (c) layers constituting one DRAM die.

location remapping schemes. e.g., If an DRAM part has 4 bank address bits, 15 row address bits, and 10 column address bits, t he total number of allowed location mapping schemes is $(4 + 15 + 10)! \approx 8.84^{30}$. Therefore while we provide a default mapping scheme, the users can always change the mapping scheme to meet a specific circuit design.

2.3 Thermal Models

Given that our functional model can simulate a variety of DRAM protocols including both stacked or planar designs, the thermal models differentiate for each case in order to achieve more accuracy. For 3D DRAMs (*e.g.*, HMCs, HBMs) as illustrated in Fig. 2a, the temperature of each stacked die is estimated. For 2D DRAMs, however, a memory module comprises several DRAM devices which are separated from each other in distance (Fig. 2b), devices in a rank operate in sync with each other and consumes same amount of power, hence we assume devices in a rank share the same thermal condition and they are independent when calculating the temperature. Therefore, DRAMsim3 only estimates the temperature for a single DRAM device per rank. We assume each DRAM die (or DRAM device) comprises three layers: active layer, metal layer and dielectric layer. The power is generated from the active layer and is dissipated to the ambient through a silicon substrate (as illustrated in Fig. 2c). We assume other surfaces of the device is adiabatic. In the following, we will introduce the thermal modeling method in detail.

2.3.1 Transient Model

We follow the *energy balance method* [2] to model the temperature. In this technique, the dies are divided into small volume elements (called thermal grids) as illustrated in Fig. 2c. Then each thermal grid is modeled as a nodal point and the heat conduction in the DRAM circuit is modeled as shown in Fig. 3. Each pair of adjacent nodal points is connected with a *thermal resistor* (R_{vert}, R_{lat}) which indicates a heat conduction path between the two nodes. The thermal resistance is calculated according to the material's thermal conductivity (k) and the geometrical dimension of the related thermal grids. As shown in Fig. 3, $R^{1,2}_{lat} = \frac{\Delta X/2}{k_1 \Delta Y \Delta Z} + \frac{\Delta X/2}{k_2 \Delta Y \Delta Z}$. R_{vert} is calculated similarly. For the node that connects to the ambient, the corresponding resistance is calculated as $R_{amb} = \frac{\Delta Z/2}{k_3 \Delta X \Delta Z}$. Besides the thermal resistor, each nodal point is connected with a *thermal capacitor* (C) which represents the ability of the thermal grid to store the thermal energy. Given the specific heat capacity (C_h) of the material of a thermal grid, the related capacitance is calculated as $C = \rho C_h \times \Delta X \Delta Y \Delta Z$ (where ρ is the density of the material within the thermal grid). For each thermal grid on the active layer, there is a *heat source* (q_s) connected to the nodal point. q_s represents the heat generation rate within the thermal grid and is calculated based on the power dissipated in that grid. Given the above information, we can estimate the temperature of a node, which represents the average temperature within the corresponding thermal grid.

Suppose there are totally N thermal grids. Let $\boldsymbol{P} \in \mathbb{R}^N$ and $\boldsymbol{T} \in \mathbb{R}^N$ represent the power and temperature for all grids, respectively; $\boldsymbol{G} \in \mathbb{R}^{N \times N}$ represents the matrix of thermal conductance

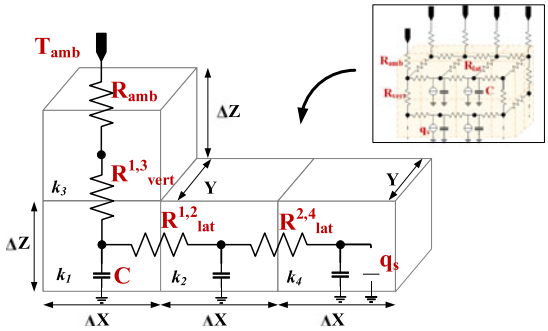

Fig. 3. Illustration of the thermal model.

which is calculated using the thermal resistance; $\boldsymbol{C} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with each element in the diagonal representing the thermal capacitance of the grid. Then the temperature at time t can be calculated by solving the following equation:

$$\boldsymbol{G}\boldsymbol{T} + \boldsymbol{P} = \boldsymbol{C}\frac{d\boldsymbol{T}}{dt}. \tag{1}$$

In practice, the transient temperature profile is calculated every power sampling epoch which is defined by the user. At the end of each epoch, we estimate the average power profile (*i.e.* $\boldsymbol{P}$) during this epoch. This $\boldsymbol{P}$, together with the temperature at the end of previous epoch ($\boldsymbol{T}_{t-1}$), is used to calculate the current temperature ($\boldsymbol{T}_t$). In DRAMsim3, we use *explicit method* [2] to get the solution. This method subdivides the epoch into small time steps (Δt) and calculates the temperature for each Δt iteratively. $\boldsymbol{T}_t$ is calculated at the last time step. In order to guarantee the convergence, this method requires the time step to be small enough:

$$\Delta t \leq \frac{C_{i,i}}{G_{i,i}} \qquad \forall i = 0, 1, 2, \ldots, N-1. \tag{2}$$

In our simulator, users can specify the thermal parameters (including the thermal conductivity, thermal capacitance *etc.*), the dimension of each layer in the DRAM, the size of a thermal grid and the length of a power sampling epoch. Given the above information, $\boldsymbol{G}$ and $\boldsymbol{C}$ will be fixed. Therefore, we only need to calculate $\boldsymbol{G}$, $\boldsymbol{C}$ and Δt (*i.e.* the proper time step) for one time at the beginning of the simulation.

2.3.2 Steady State Model

At the end of simulation, DRAMsim3 also estimates the steady-state temperature profile using the average power profile during the period of simulation. The steady-state thermal model only contains the resistors, hence Equation (1) is reduced to:

$$\boldsymbol{G}\boldsymbol{T} + \boldsymbol{P} = 0. \tag{3}$$

Note that Equation (3) is a linear equation set and $\boldsymbol{G}$ is a sparse matrix [2]. This equation is solved using SuperLU [5], which provides a library to solve large sparse linear equations.

2.3.3 Thermal Model Validation

The proposed thermal model is validated against the Finite Element Method (FEM) results. We use ANSYS to perform the FEM simulation. Our thermal model targets generic 3D ICs, where the die model can be either a processor die or a DRAM die, and the model works the same way for both. So it is reasonable to use either a processor die or a DRAM die to validate the model. The processor layer has much higher power density and variation, which is better for validating the model. Therefore, we use the thermal model to estimate the temperature for a multi-core processor die for our validation.

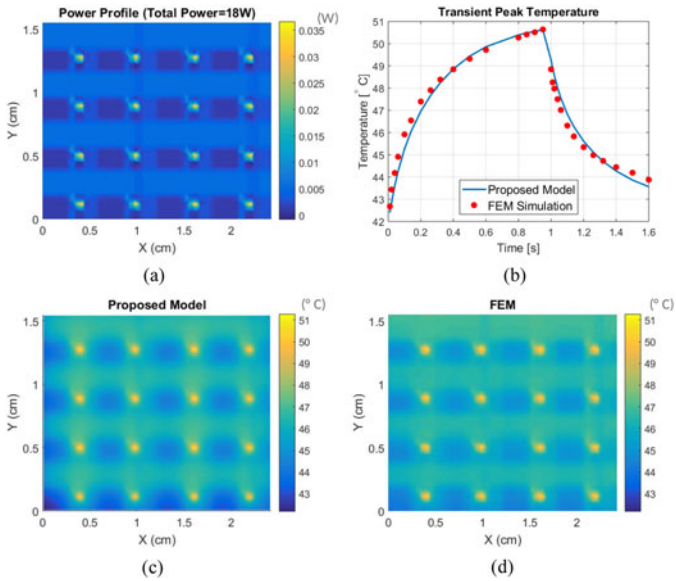

Fig. 4. (a) The original power profile, (b) the transient result for for the peak temperature, (c) the temperature profile at $1s$ calculated using our thermal model and (d) the temperature profile at $1s$ calculated using the FEM method.

The power profile of the multi-core processor is generated based on [11] and is illustrated in Fig. 4a (Total power equals to 18W). This processor die contains three layers as illustrated in Fig. 2c. The simulation is taken for $1.6s$. Before $1s$, the processor power stays constant as shown in Fig. 4a. After 1s, the processor power is reduced by 75 percent. Fig. 4b shows the transient peak temperature using our model and the FEM simulation. Fig. 4c and 4d represent the temperature profile at $1s$ acquired using our model and the FEM method, respectively. According to the figure, the result of our model accurately matches the FEM result.

3 Evaluation

3.1 Simulator Validation

Other than the thermal model validation described in Section 2.3.3, we also validated our DRAM timings against Micron Verilog models. We take a similar approach as [14], that is, feeding request traces into DRAMsim3, output DRAM command traces and convert them into the format that fits into Micron's Verilog workbench. We ran the Verilog workbench through ModelSim Verilog Simulator and no DRAM timing errors were produced. We not only validated DDR3 model as previous works did, but also validated DDR4 model as well. DRAMsim3 is the first DRAM simulator to be validated by both models to our knowledge. While we only have DDR3 and DDR4 Verilog validation, the DRAM timing enforcement implementation is paramerized and universal to all DRAM protocols, assuring that DRAM timing constraints are enforced correctly for other DRAM protocols as well. We also use DRAMsim3 to conduct a thorough memory characterization study of various memory protocols, the results can be found in [10].

3.2 Comparison With Existing DRAM Simulators

We compare DRAMsim3 with existing DRAM simulators including DRAMSim2 [14], ramulator [9], USIMM [4] and DrSim [6]. These are open sourced DRAM simulators that can run as standalone packages with trace inputs, making it viable for us to conduct a fair and reproducible comparison.

Each simulator is compiled by clang-6.0 with O3 optimizations on their latest publicly released source code (except for USIMM where we use officially distributed binary). We use randomly generated memory request traces for all these simulators, the requests

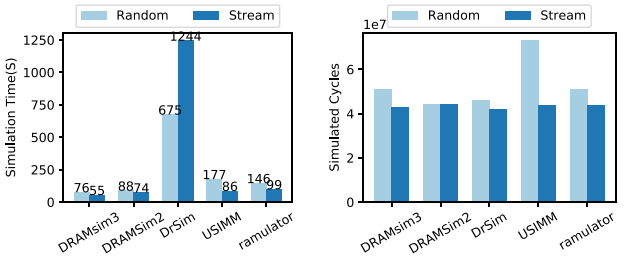

Fig. 5. Simulation time (left) and simulated cycles (right) comparisons for 10 million random & stream requests on various DRAM simulators.

are exactly the same for each simulator while only the trace format is adjusted to work with each specific simulator. The read to write request ratio is 2:1. Since DDR3 is the only protocol all tested simulators support, we run each simulator with a single channel, dual rank DDR3-1600 configuration that have exact same DRAM structures and timing parameters. We also made sure each simulator has comparable system parameters such as queue depth. Note that the thermal model of DRAMsim3 is disabled in this comparison.

We time the host simulation time of each simulator to finish processing 10 million requests from the trace to demonstrate simulation performance. We also examine how many simulated cycles it takes for each simulator to finish these requests, as an indicator for simulator scheduling efficiency.

The results are shown in Fig. 5. In terms of simulation speed, DRAMsim3 offers the best simulation performance among the contestants: it is on average 20 percent percent faster than DRAMSim2, the next fast DRAM simulator, and more than twice faster than the other simulators in both random and stream request patterns. When it comes to simulation throughput, also shown in Fig. 5, DRAMsim3 is on par with other simulators as well, indicating that the scheduler and controller design is just as efficient as the other simulators. Like our Verilog validation procedures, we also provide detailed guideline, source code and scripts needed to reproduce these results.

4 Conclusion

In this paper we present DRAMsim3, a fast, validated, thermal-capable DRAM simulator. We introduced the architectural and thermal modeling capabilities of DRAMsim3. Through the evaluations we demonstrated the validation of the simulator, and showcased the simulation performance of DRAMsim3 with uncompromising simulator design.

Acknowledgments

This work was supported in part by NSF grant 1642424 and DoD contract FA8075-14-D-0002-0007, and TAT 15-1158.

References

[1] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.
[2] Y. Cengel, *Heat and Mass Transfer: Fundamentals and Applications*. New York, NY, USA: McGraw-Hill Higher Education, 2014.
[3] K. Chandrasekar, C. Weis, Y. Li, B. Akesson, N. Wehn, and K. Goossens, "Drampower: Open-source dram power & energy estimation tool," 2012. [Online]. Available: http://www.drampower.info
[4] N. Chatterjee *et al.*, "Usimm: The utah simulated memory module," *University of Utah*, Tech. Rep. UUCS-12-002, 2012.
[5] J. W. Demmel, J. R. Gilbert, and X. S. Li, "An asynchronous parallel supernodal algorithm for sparse gaussian elimination," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 4, pp. 915–952, 1999.
[6] M. K. Jeong, D. H. Yoon, and M. Erez, "DrSim: A platform for flexible dram system research," 2012, Accessed on: [Online]. Available: http://lph.ece.utexas.edu/public/DrSim
[7] M. Jung, C. C. Rheinländer, C. Weis, and N. Wehn, "Reverse engineering of drams: Row hammer with crosshair," in *Proc. 2nd Int. Symp. Memory Syst.*, 2016, pp. 471–476.
[8] M. J. Khurshid and M. Lipasti, "Data compression for thermal mitigation in the hybrid memory cube," in *Proc. IEEE 31st Int. Conf. Comput. Des.*, 2013, pp. 185–192.
[9] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible dram simulator," *IEEE Comput. Archit. Lett.*, vol. 15, no. 1, pp. 45–49, Jan.-Jun. 2016.
[10] S. Li, D. Reddy, and B. Jacob, "A performance & power comparison of modern high-speed dram architectures," in *Proc. ACM Int. Symp. Memory Syst.*, 2018, pp. 341–353.
[11] T. Lu, C. Serafy, Z. Yang, S. K. Samal, S. K. Lim, and A. Srivastava, "TSV-Based 3-D ICs: Design methods and tools," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1593–1619, Oct. 2017.
[12] J. Janzen, "Calculating memory system power for DDR3," *Micron Designline*, vol. 13, no. 1, 2008.
[13] A. F. Rodrigues *et al.*, "The structural simulation toolkit," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 4, pp. 37–42, 2011.
[14] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Comput. Archit. Lett.*, vol. 10, no. 1, pp. 16–19, Jan.-Jun. 2011.
[15] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," in *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, 2013, pp. 475–486.