DRAMPower 5: An Open-Source Power Simulator for Current Generation DRAM Standards

Lukas Steiner
University of Kaiserslautern-Landau
Kaiserslautern, Germany
lukas.steiner@rptu.de

Thomas Psota
Fraunhofer IESE
Kaiserslautern, Germany
Thomas.Psota@iese.fraunhofer.de

Marco Mörz
Fraunhofer IESE
Kaiserslautern, Germany
marco.moerz@iese.fraunhofer.de

Derek Christ
Julius-Maximilians-Universität Würzburg
Würzburg, Germany
derek.christ@uni-wuerzburg.de

Matthias Jung
Julius-Maximilians-Universität Würzburg
Würzburg, Germany
Fraunhofer IESE
Kaiserslautern, Germany
m.jung@uni-wuerzburg.de

Norbert Wehn
University of Kaiserslautern-Landau
Kaiserslautern, Germany
norbert.wehn@rptu.de

Abstract

As off-chip memory accesses nowadays dominate the overall power consumption of many compute platforms, accurate DRAM power simulation models are an important tool for system designers. Unfortunately, existing open-source models only support older generations of DRAM standards, while current system designs mainly rely on the newest generation including DDR5, LPDDR5 or HBM3. In addition, the existing models are not directly applicable to the new standards because of the much higher data rates and newly introduced features. This paper presents DRAMPower 5, a completely revised version of the popular DRAMPower simulator, which uses newly developed core and interface power models to support the current generation of DRAM standards. In addition, DRAMPower 5 features a redesigned software architecture that enables both fast and accurate simulation. The tool is open source and available on GitHub.

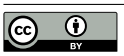
CCS Concepts

• **Hardware** → **Power estimation and optimization**; **Dynamic memory**; • **Computing methodologies** → *Model development and analysis.*

Keywords

DRAMPower, DRAM, power, energy, simulation, interface

ACM Reference Format:
Lukas Steiner, Thomas Psota, Marco Mörz, Derek Christ, Matthias Jung, and Norbert Wehn. 2025. DRAMPower 5: An Open-Source Power Simulator for Current Generation DRAM Standards. In *Rapid Simulation and Performance Evaluation for Design (RAPIDO '25), January 21, 2025, Barcelona, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3721848.3721850



RAPIDO '25, Barcelona, Spain

ACM ISBN 979-8-4007-1471-9/25/01
https://doi.org/10.1145/3721848.3721850

1 Introduction

The recent expansion of memory-intensive applications has led to increased demand for DRAM bandwidth and capacity in current computing systems. This demand is particularly pronounced in AI applications, where specialized accelerator chips with immense DRAM bandwidths beyond 1 TBps are used. On these platforms, memory dominates the total power consumption. When training large AI models, it was found that up to 90 % of the system power is consumed by memory accesses [3]. Therefore, an accurate estimation of DRAM power consumption is critical in the early stages of design to properly dimension the power supply circuits and cooling. In consumer devices, on the other hand, it was also found that in average more than 60 % of the total system power is spent on memory accesses [2]. While the overall power budget for these devices is limited to only a few watts, it is equally important to accurately estimate the DRAM power consumption, for example, to explore the power saving potential of software improvements during system design.

In the current state of the art, there are two widely used open-source simulation tools for estimating DRAM power consumption, namely *DRAMPower* [4] and *CACTI-IO* [8]. DRAMPower focuses on the DRAM core, while CACTI-IO models the DRAM interface. Unfortunately, both tools only provide support for older standards. At the same time, current generation standards like DDR5, LPDDR5 and HBM3 enable much higher interface speeds and offer new features, which requires special consideration for power modeling. In addition, the standards are inconsistent in specifying operating currents, making it difficult to create a universal power model. To the best of our knowledge, there is no open-source DRAM power simulator that provides accurate models of both the DRAM core and interface and supports the current generation of DRAM standards, including DDR5, LPDDR5 and HBM3. To fill this gap, we introduce DRAMPower 5, a completely revised version of the DRAMPower simulator with more accurate power models, an improved software architecture, and support for the latest DRAM standards.

In this paper, we make the following new contributions:

- We present a universal core power model that can handle the different operating current specifications of each standard.

- We introduce a newly developed interface power model with improved accuracy at high operating frequencies.
- We present the updated software architecture of the simulator and evaluate its simulation speed and accuracy.

The remainder of the paper is structured as follows. Section 2 discusses related work on DRAM power modeling. Section 3 provides the reader with the necessary background on DRAM. In Section 4, the core power modeling is explained, while Section 5 addresses the interface power modeling. Section 6 provides a short overview of the new simulator. Finally, Section 7 concludes the paper and gives an outlook on future work.

2 Related Work

In this section, we provide an overview of related work. An often used DRAM power model is the System Power Calculator by Micron [11]. It is provided in the form of spreadsheets for various JEDEC standards. The power estimation is based on DRAM timing and current datasheet values and workload specifications like the read/write ratio. However, this modeling approach can only achieve a limited accuracy because the actual command trace that is issued to the DRAM by the memory controller is not considered. In addition, there exist no spreadsheets for current generation standards. A more accurate simulation tool is DRAMPower [4], which also relies on datasheet values, but in addition uses a real DRAM command trace as input to model the internal state transitions with cycle accuracy. Since initially the power dependence on the number of active DRAM banks was not taken into account, DRAMPower was later extended with a bank-sensitive model [9, 10] to improve its accuracy. Still, the tool has two drawbacks: it only models core power, but no interface power, and it has not been updated to the latest standards yet. Another simulator similar to DRAMPower is VAMPIRE [6]. This tool puts its focus on the power variations between different DRAM modules, within one DRAM module depending on the access location, and the data value dependency. VAMPIRE is calibrated with measurements of real DRAM modules and provides very accurate results. However, this presupposes that real measurements are available for the devices to be used, which is not usually the case in the early stages of design. Additionally, VAMPIRE supports DDR3 only. In [13], an analytical DRAM core power model is presented. It approximates the power consumption based on the internal device architecture and technology and can also be extrapolated to future technologies. Since the model was already developed 15 years ago, it is not clear whether an extrapolation to current technologies will still provide accurate results. The most popular tool for DRAM interface power modeling is CACTI-IO [8]. CACTI-IO does not rely on datasheet currents, but it uses an equivalent circuit diagram of the interface between DRAM controller and devices. The power consumption is then calculated with a simplified network analysis. While the results are accurate for older generation standards, the simplifications introduce a large error for current generation standards as they support higher data rates. In summary, there is no publicly available DRAM power simulation tool capable of modeling both core and interface power of current generation DRAM standards with high accuracy.

3 DRAM Background

This section provides the necessary background on the DRAM core and interface relevant to power modeling. It also briefly introduces the different families of DRAM standards and explains their key differences.

3.1 Core

DRAM is a type of memory primarily optimized for low cost per bit. To achieve high memory density, the chips are internally organized in a hierarchical fashion consisting of *columns*, *rows*, *banks* and, for newer standards, *bank groups*. When data should be read or written from or to a column, the corresponding row must be *activated* first. Within each bank, only one row can be active at a time and the bank must be *precharged* before a new row can be activated. Data is transferred over the interface in a burst fashion, i.e., for a read operation, a large amount of data is first fetched internally in parallel from the array to the interface, before it is transferred to the memory controller in multiple beats. Information is stored as an electrical charge held in a tiny capacitor. As the capacitor leaks this charge over time, each DRAM cell must be *refreshed* regularly (usually every 32 to 64 ms). The refresh operation is triggered externally by the memory controller with a refresh command. During refresh, no data can be accessed within the target bank(s). Thus, only a few rows are refreshed each time to avoid long access delays and a refresh command is sent every few microseconds. To save energy, DRAM devices can be put into a *power-down* mode when no data accesses are performed. This disables parts of the core and interface. However, the power-down mode must be interrupted periodically to perform refreshes. To avoid this, the *self refresh* mode can be entered where data retention is managed by the device itself and no refresh commands need to be provided by the memory controller.

3.2 Interface

All modern DRAM subsystems use a bidirectional single-ended *data bus* (DQ) to transfer data from the memory controller to the DRAM devices or the other way round. To sample the data at the correct time, a differential *data strobe* pair (DQS_t/DQS_c) is provided by the driving side. Since data is sampled both at the rising and the falling edge of the data strobe (intersection of DQS_t and DQS_c), the bus operates at *double data rate* (DDR). Commands and addresses are transferred from the memory controller to the DRAM devices over a unidirectional *command/address bus* (CA). They are sampled on the edges of a differential clock signal pair (CK_t/CK_c) that is also driven by the memory controller. Since all modern DRAM standards operate at frequencies in the gigahertz range with data rates reaching more than 8 Gbps/pin, the signals are terminated at the receiver side to ensure their integrity. To increase the memory capacity of a DRAM channel, multiple devices can be connected to the same memory controller, sharing the command/address and data bus (so-called ranks). The target device is selected by the controller via a *chip select* signal (CS). The physical interconnect between memory controller and DRAM can be realized in different ways, e.g., through a classical *printed circuit board* (PCB), a *package on package* (PoP) arrangement or a silicon interposer. All these channels have different characteristics in terms of load capacitances, reflections and loss, so they need to be modeled individually for

an accurate power estimation. One special interconnection type widely used in PCs and servers is the *dual inline memory module* (DIMM). Multiple DRAM chips are soldered onto a small PCB with pins on the bottom edge, which is then plugged into a socket on the main PCB. DIMMs require extra considerations for power modeling as there are different wiring topologies, off-die termination, and in some cases additional buffer chips for the command/address bus and data bus. Due to space limitations, this is not discussed in detail in this paper.

3.3 DRAM Standards

Over the last quarter century, JEDEC has published more than 20 different DRAM standards. As DRAM application fields become more heterogeneous, so do the standards. Currently, there are four major families:

- *DDR* is used as general-purpose memory for PCs and servers as it provides high capacities at a low cost. It can be organized as single devices or DIMMs.
- *Low-Power DDR (LPDDR)* is optimized for a low power consumption and mainly used in battery-powered devices like smartphones or embedded systems.
- *Graphics DDR (GDDR)* offers higher bandwidths than DDR and is mainly used in GPUs.
- *High Bandwidth Memory (HBM)* provides even higher bandwidths than GDDR by utilizing a much wider data bus and a silicon interposer for connection. It is mainly used in high-performance GPUs and ASICs.

The DRAM core architecture is similar across all standards and has not changed much over the years. Newer standards usually come with higher memory capacities, a slightly reduced core supply voltage and some new commands to improve the performance or power efficiency. An example of this is the different refresh modes, which will be explained in more detail in Section 4.3. However, the interface between memory controller and devices differs greatly from standard to standard. These differences include pin data rate, pin count, termination scheme, channel loss characteristics, signaling voltage and clocking architecture. In order to achieve an accurate interface power modeling, all these differences need to be considered in the calculations. More details are provided in Section 5.

4 Core Power Modeling

This section explains the modeling of core power, while the modeling of interface power is covered in the next section. Core and interface can be considered completely independent of each other because they use different supply voltages. Core power refers to the power consumed by the internal circuitry of the DRAM device, i.e., the memory arrays, sense amplifiers, row and column decoders, I/O gating and control logic. The receiver circuits at the interface are also operated with the core supply voltage and are therefore included in the core power. As the internal architecture of modern DRAM devices is very complex and highly proprietary, core power calculation cannot be based on classical network analysis. Thus, each DRAM standard defines a set of currents for fixed operating scenarios, which are listed in vendor datasheets. Based on these currents, the core power can be estimated. The following section provides an overview of these currents.

4.1 Current Measurement Conditions

The minimum set specified in all DRAM standards includes the following nine currents:

- I_{DD0} (Operating one bank active-precharge current): Activate and precharge commands are sent alternately with minimum spacing.
- I_{DD2N} (Precharge standby current): All banks are precharged and no commands are issued.
- I_{DD2P} (Precharge power-down current): All banks are precharged, no commands are issued and the device is in power-down mode.
- I_{DD3N} (Active standby current): All banks are active and no commands are issued.
- I_{DD3P} (Active power-down current): All banks are active, no commands are issued and the device is in power-down mode.
- I_{DD4R} (Operating burst read current): All banks are active and read commands are issued with minimum spacing.
- I_{DD4W} (Operating burst write current): All banks are active and write commands are issued with minimum spacing.
- I_{DD5B} (Burst refresh current): Refresh commands are issued with minimum spacing.
- I_{DD6} (Self refresh current): The device is in self refresh mode and the external clock is turned off.

Unfortunately, the different JEDEC subcommittees, which are responsible for formulating DRAM standards, are very inconsistent in specifying the currents. Apart from different naming schemes, the measurement conditions mentioned above only apply for standards of the DDR family, while they differ for LPDDR, GDDR and HBM. For example, LPDDR measures I_{DD3N}, I_{DD3P}, I_{DD4R} and I_{DD4W} with only one bank active. GDDR measures I_{DD3N} and I_{DD3P} with one bank active, while I_{DD4R} and I_{DD4W} are measured with one bank in each bank group active. HBM, in turn, measures I_{DD3N} and I_{DD3P} with one bank active and I_{DD4R} as well as I_{DD4W} with all banks active. Section 4.2 explains how these different measurement conditions are treated to achieve a universal bank-sensitive power model. Similarly, the refresh currents are also measured under varying conditions. While DDR standards specify a burst refresh current I_{DD5B} for all available refresh modes, LPDDR standards specify a burst refresh current only for all-bank refresh, while for per-bank refresh, an average current I_{DD5A} is provided. The difference between I_{DD5B} and I_{DD5A} is the spacing between two consecutive refresh commands. It is the refresh cycle time t_{RFC} (i.e., the duration of a single refresh operation) for I_{DD5B} and the much longer average refresh interval t_{REFI} (i.e., the interval at which refresh commands need to be issued in normal operation) for I_{DD5A} as shown in Figure 2. Section 4.3 explains how refresh power can be modeled using the provided currents of each standard. However, even if all missing currents can be calculated, the used approach for core power calculation still faces two problems, which have also been highlighted in [6]. First, there are large device-to-device variations, which forces the vendors to be very pessimistic when specifying operating currents. As a consequence, power is overestimated in most cases. Second, the currents are measured for fixed data and address patterns, i.e., no data dependencies and structural

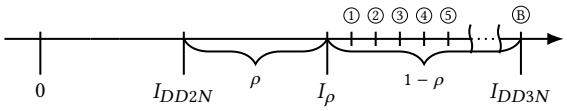


Figure 1: Bank-Sensitive Currents [9]

variations within the device are considered. If a more accurate modeling is required, the calculations have to be refined with additional device measurements. This will be demonstrated in Section 6.3.

4.2 Universal Bank-Sensitive Model

The DRAM core power is composed of background power and command power. A bank-sensitive model is used for the background power, i.e. the more banks are active, the higher the power consumption. This model was already introduced in previous versions of the tool [9, 10] and provides higher accuracy compared to a model that only distinguishes between two states (either active or precharged) like the one from Micron [11]. As shown in Figure 1 for a DRAM of the DDR family with B banks, I_{DD2N} is drawn when all banks are precharged and I_{DD3N} is drawn when all B banks are active. The span in between is not divided linearly depending on the number of active banks, but there is an offset when activating the first bank. This is due to the fact that additional logic must be switched on when the first bank is activated. ρ is a vendor- and device-specific factor between 0 and 1, which can be determined by measurement [9]. Alternatively, the pessimistic assumption of $\rho = 1$ can be made, which leads to the simplified model with only two distinct states. For standards of the DDR family, it is $I_{DD3N} = I_{Ⓑ}$, while for LPDDR, GDDR and HBM, it is $I_{DD3N} = I_{①}$. This difference must be taken into account when calculating the background power. If the current I_{DD2N}, the factor ρ, a current I_{DD3N} measured with M banks active, and the total number of banks B is given, all other currents can be calculated. It is

$$I_{Ⓜ} = I_{DD2N} + (I_{Ⓑ} - I_{DD2N}) \cdot \left(\rho + (1-\rho) \cdot \frac{M}{B}\right). \tag{1}$$

When the DRAM is in power-down mode, the dependence of the current on the number of active banks is much smaller, so we only distinguish between two states characterized by I_{DD2P} and I_{DD3P}.

The average command power is calculated by counting the number of commands of each type, adding up the energy that is consumed for all these commands, and dividing the total energy by the simulated time. As for the background power, the differences among the standards must be taken into consideration for the command power as well. In [9], the energy for a read command E_{RD} is calculated as

$$E_{RD} = V_{DD} \cdot (I_{DD4R} - I_{DD3N}) \cdot \frac{BL}{DR} \cdot t_{CK} \tag{2}$$

where V_{DD} is the core supply voltage, BL is the burst length, DR is the data rate and t_{CK} is the clock period. For a write command, I_{DD4R} is replaced with I_{DD4W}. However, this equation only works if I_{DD4R} and I_{DD3N} are measured with the same number of banks active, which is not the case for GDDR and HBM. Thus, the equations need to be adapted accordingly, i.e., for GDDR, I_{DD3N} must be replaced with $I_{(BG)}$ with BG being the number of bank groups, while for HBM, I_{DD3N} must be replaced with $I_{Ⓑ}$.

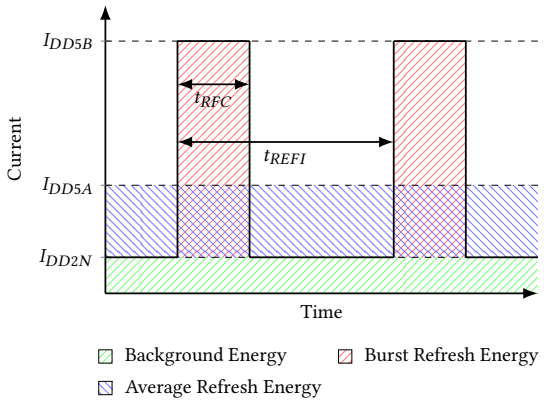


Figure 2: Relation between Burst Refresh Current and Average Refresh Current

4.3 Refresh Power

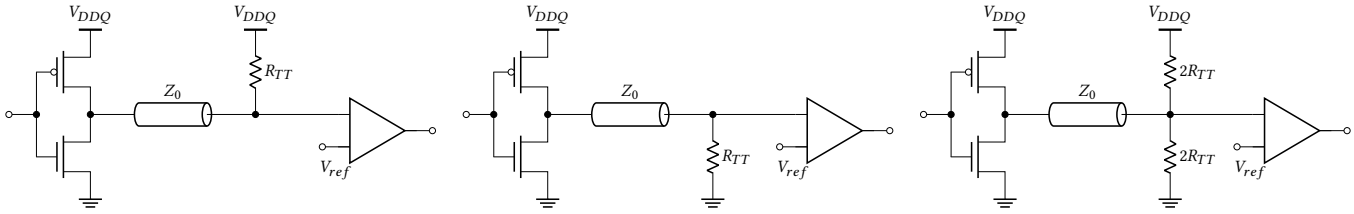
Depending on the DRAM standard, various refresh modes are supported. They differ in the number of banks that are refreshed with a single command. All-bank refresh commands target all banks of the device at once. As no data can be accessed in banks where a refresh is in progress, this mode can cause a large drop in bandwidth. Thus, newer DRAM standards offer improved refresh modes where only a single bank (per-bank refresh), two banks (per-2-bank refresh) or one bank in each bank group (same-bank refresh) of the device are targeted with a single command, while the remaining banks can still be accessed in the meantime. The duration of a single refresh command is the refresh cycle time t_{RFC}, which is also the spacing of refresh commands when measuring the burst refresh current I_{DD5B}. Thus, when a burst refresh current is provided, the energy for a single refresh command E_{REF} can be calculated as

$$E_{REF} = V_{DD} \cdot \left(I_{DD5B} - I_{Ⓜ}\right) \cdot t_{RFC} \tag{3}$$

where M is the number of refreshed banks. As the equation shows, banks with a refresh in progress are considered active, which is the most accurate way of modeling because internally the refresh is performed by successively activating multiple rows within each target bank.

In the cases where only an average refresh current I_{DD5A} is provided, an approximated value for I_{DD5B} can be determined. Figure 2 demonstrates the relation between both refresh currents graphically, where the dashed boxes represent the energy that is consumed. The voltage is constant and can be neglected. From the definitions of the two currents, we know that within a refresh interval t_{REFI}, the burst refresh energy and the average refresh energy are identical. This relationship can be translated into the following equation to calculate I_{DD5B} from I_{DD5A}:

$$I_{DD5B} = I_{DD2N} + (I_{DD5A} - I_{DD2N}) \cdot \frac{t_{REFI}}{t_{RFC}} \tag{4}$$



(a) Pseudo Open Drain Logic (PODL) **(b) Low Voltage Swing Term. Logic (LVSTL)** **(c) Stub Series Terminated Logic (SSTL)**

Figure 3: DRAM Interface Termination Schemes

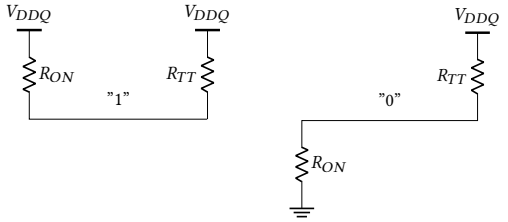
5 Interface Power Modeling

Interface power refers to the power consumed by the drivers for the communication between memory controller and DRAM devices. In contrast to the core power, which is fixed for a specific device, the interface power depends on the complete DRAM subsystem architecture, i.e., the *physical layer* (PHY) of the memory controller, the channel architecture (number of ranks, possible usage of DIMMs, etc.), the channel characteristics (e.g., channel loss and parasitic capacitances) and the DRAM PHYs. Thus, a modeling based on the operating currents specified in vendor datasheets is not possible as they are only measured for one specific subsystem architecture. Instead, we calculate the interface power based on an equivalent circuit diagram of the real interface architecture as is also done by CACTI-IO. Interface power can be divided into *termination power*, which is dissipated across the termination resistances required for signal integrity, and *dynamic power*, which is dissipated through the lossy charging and discharging of parasitic capacitances and the signaling over a lossy transmission line. In the following two sections, the calculation of termination power and dynamic power is explained.

5.1 Termination Power

The termination power depends on the termination scheme and the number of logic zeros and ones transmitted, but it is independent of the operating frequency. There are three commonly used termination schemes for DRAM, shown in Figure 3 for a simple point-to-point connection. *Pseudo open drain logic* (PODL) and *low voltage swing terminated logic* (LVSTL) only use a pull-up or a pull-down resistor for termination, respectively. *Stub series terminated logic* (SSTL) uses both a pull-up and a pull-down resistor. In all three cases, the termination resistance is matched the characteristic impedance of the transmission line, i.e., $R_{TT} \approx Z_0$ (remember that in AC analysis a DC voltage source is treated as a short). To calculate the power, both logic levels are considered separately. The transistor of the driver that is switched on is replaced with an equivalent resistor with resistance R_{ON}, while the transistor that is switched off is replaced with an open circuit. As an example, Figure 4 shows the two equivalent circuit diagrams for a PODL interface. When driving a logic one, both ends of the circuit are connected to V_{DDQ}, which means that no current is flowing and no power is dissipated, i.e.,

$$P^{PODL}_{term,1} = 0. \tag{5}$$



(a) Driving Logic "1" **(b) Driving Logic "0"**

Figure 4: Equivalent Circuit Diagrams for PODL Termination Power

In contrast, when driving a logic zero, one side is connected to ground, while the other side is connected to V_{DDQ}. The dissipated power is calculated as

$$P^{PODL}_{term,0} = \frac{V^2_{DDQ}}{R_{ON} + R_{TT}}. \tag{6}$$

In the case of an LVSTL interface, the equations for both logic levels are reversed. The SSTL interface uses both a pull-up and a pull-down resistor, therefore, power is dissipated at both logic levels. It can be calculated as

$$P^{SSTL}_{term,0} = P^{SSTL}_{term,1} = \frac{V^2_{DDQ}}{(R_{ON}||2R_{TT}) + 2R_{TT}}. \tag{7}$$

The average termination power when transmitting n_0 logic zeros and n_1 logic ones is

$$P_{term} = \frac{P_{term,0} \cdot n_0 + P_{term,1} \cdot n_1}{n_0 + n_1} \tag{8}$$

Because with PODL and LVSTL only one logic level consumes power, data bus inversion can be used to reduce the termination power consumption. With SSTL, the termination power is independent of the transmitted data.

For channel configurations with multiple ranks or DIMMs, the interconnect network can change from a simple point-to-point topology to a more complex topology, e.g., because the non-target dies also terminate the bus. In these cases, termination power can be calculated in the same way by determining the equivalent circuit diagrams for both logic levels.

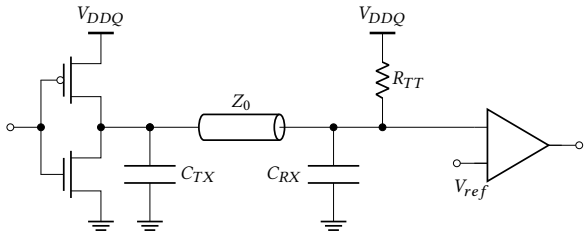


Figure 5: Point-to-Point Connection with Parasitic Caps

5.2 Dynamic Power

As shown in the previous section, termination power is frequency independent because it is dissipated across a purely resistive network. Termination power represents a lower bound for the total power consumption and also dominates at low operating frequencies. However, since current generation DRAM standards support data rates of 8 Gbps/pin and more, the impact of parasitic capacitances is much more significant. Figure 5 shows the simple point-to-point connection with PODL termination scheme as already presented in Figure 3a, but with two added parasitic capacitances, one at the driver side and one at the receiver side. We analyze the power dissipation of this circuit for different operating frequencies as input using SPICE. The components are dimensioned as $R_{ON} = 48\,\Omega$, $R_{TT} = 60\,\Omega$, $C_{TX} = C_{RX} = 1\,\text{pF}$ and $V_{DDQ} = 1.1\,\text{V}$, which is in the order of a real DDR5 interface. For now, the transmission line losses are also modeled with a parasitic capacitance with $C_{TL} = 2\,\text{pF}$. At a frequency of 100 MHz, the dissipated power is 5.7 mW, which is close to the termination power of the circuit of 5.6 mW. However with increasing frequencies, the power also increases because the capacitors start to conduct. At 1600 MHz (i.e., DDR5-3200), the dissipated power is already 8.6 mW, i.e., over 50 % higher than the pure termination power. To calculate the power dissipation analytically, the clock signal with frequency f and voltage swing V_{DDQ} can be expressed as a Fourier series

$$v(t) = \frac{V_{DDQ}}{2} + \Re\left\{\frac{-2j \cdot V_{DDQ}}{\pi} \sum_{k=1,3,5,\dots}^{\infty} \frac{1}{k} \exp(j2\pi f k t)\right\}. \quad (9)$$

with DC component $\frac{V_{DDQ}}{2}$. The complex amplitudes $\underline{\hat{V}}_k$ of the frequency components can be directly determined from this equation as

$$\underline{\hat{V}}_k = \frac{-2j \cdot V_{DDQ}}{\pi} \cdot \frac{1}{k}. \quad (10)$$

With the frequency-dependent complex impedances $\underline{Z}_k$ calculated as

$$\underline{Z}_k = R_{ON} + \frac{1}{j2\pi f k (C_{TX} + C_{RX} + C_{TL}) + \frac{1}{R_{TT}}}, \quad (11)$$

the DC resistance R_{DC} calculated as

$$R_{DC} = R_{ON} + R_{TT}, \quad (12)$$

and the voltage across R_{DC} calculated as

$$V_{DC} = V_{DDQ} - \frac{V_{DDQ}}{2} = \frac{V_{DDQ}}{2}, \quad (13)$$

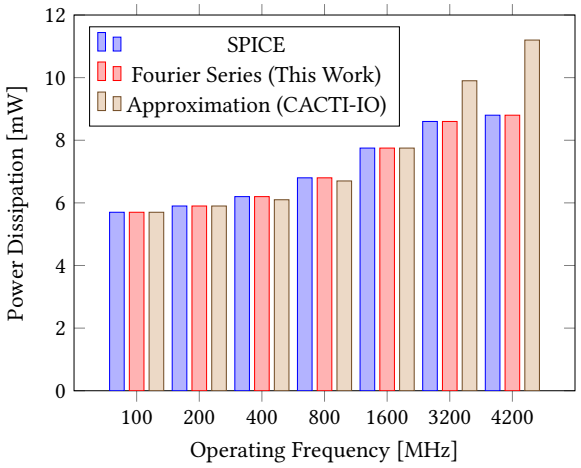


Figure 6: Comparison of Different Calculation Methods for Power Dissipation

the total power dissipation P_{total} can be calculated as

$$P_{total} = \frac{V_{DC}^2}{R_{DC}} + \sum_{k=1,3,5,\dots}^{\infty} \frac{|\underline{\hat{V}}_k|^2}{2} \cdot \Re\left\{\frac{1}{\underline{Z}_k}\right\}. \quad (14)$$

In reality, the series needs to be terminated at a certain k, which can be chosen to match the finite slew rate of the signal. For LVSTL, the same equations can be applied, while for SSTL, the calculation of the DC component needs to be adapted. The dynamic power P_{dyn}, which adds to the termination power due to the toggling between both logic levels, is finally calculated as

$$P_{dyn} = P_{total} - P_{term}. \quad (15)$$

One alternative formula, which is often used to approximate the dynamic power P_{dyn}, is given by

$$P_{dyn} = \left(\sum_i C_i \cdot V_{sw,i}\right) \frac{V_{DDQ} \cdot f}{2} \quad (16)$$

where C_i are the capacitances along the channel and $V_{sw,i}$ are the respective voltage swings at each capacitance [1, 8]. The voltage swings are usually determined using a DC analysis for both logic levels. This is also done by CACTI-IO. While this approximation provides accurate results at low operating frequencies, current generation DRAM interfaces do not reach full voltage swing anymore due to the large parasitic capacitances in combination with high operating frequencies. Figure 6 shows the total power dissipation of the previous circuit (Figure 5) at different operating frequencies calculated with SPICE, Equation 14 and Equation 8 plus Equation 16. While the Fourier series based formula consistently provides the same results as SPICE, the approximate formula is accurate at low frequencies, but overestimates the power dissipation at higher frequencies, e.g., by 15 % at 3200 MHz (DDR5-6400) and even 27 % at 4200 MHz (DDR5-8400, the currently highest specified data rate of the standard).

The loss characteristic of the transmission line can be handled in different ways. In [7], the authors have analyzed various physical DRAM interfaces, i.e., multi DIMM, package on package, PCB

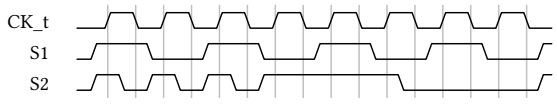


Figure 7: Two Different Signals with Identical α, n_0 and n_1

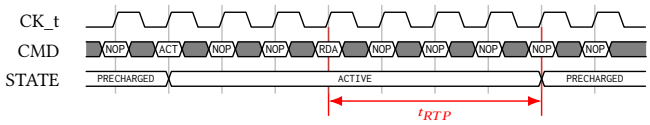


Figure 8: Example for Implicit Command

trace and silicon interposer. They show that the channels have very distinct insertion loss characteristics, which need to be taken into consideration for an accurate power estimation. A linear loss characteristic can be approximated with an additional capacitance, while more complex loss characteristics can be modeled with frequency-dependent impedance values in the Fourier series based calculation.

Up until now, the formulas for dynamic power consumption assume a switching activity of $\alpha = 1$, i.e., the signals transition from logic zero to logic one once every period. While this is true for clock and data strobe signals, the command/address bus and data bus usually experience lower switching activities. Especially when a signal is only operated at SDR, the switching activity is limited to $\alpha_{max} = 0.5$. The problem is that the switching activity α and number of transmitted zeros n_0 and ones n_1 alone do not determine the complete signal behavior, which is demonstrated in Figure 7. Both S1 and S2 have a switching activity of $\alpha = 0.5$ and the number of transmitted zeros and ones is $n_0 = n_1 = 8$. However, S1 operates at half the clock frequency for the whole time, while S2 operates at the full clock frequency in the beginning and only one fifth of the clock frequency in the end. When the dissipated power is calculated section by section using Equation 14, the results for S1 and S2 differ because different voltage swings are reached in each section. In the corner cases, a signal with switching activity α can be either modeled with a constant switching activity for the whole time or with a maximum switching activity α_{max} for one part of the time and a switching activity of 0 for the other part of the time. The actual dynamic power consumption lies between these two corner cases and can be approximated by the mean value

$$\overline{P}_{dyn}(\alpha) = \frac{P_{dyn}(f = \alpha \cdot f_{max}) + \frac{\alpha}{\alpha_{max}} \cdot P_{dyn}(f = \alpha_{max} \cdot f_{max})}{2}. \tag{17}$$

Finally, the switching activity α can be determined by counting the number of zero to one transitions $n_{0\to1}$ in a given time interval τ as

$$\alpha = \frac{n_{0\to1}}{\tau \cdot f_{max}}. \tag{18}$$

6 Simulator Overview

This section provides a short introduction to the internal software architecture of DRAMPower 5. Afterwards, the simulation speed and simulation accuracy are evaluated.

6.1 Software Architecture

The new version of DRAMPower is not designed as a standalone simulator, but as a library that is coupled to a DRAM subsystem simulator which models the memory controller and translates incoming read and write requests into DRAM commands. Alternatively, a DRAM command trace can be provided as an input file. For the interface power calculation, the provided commands, addresses and data are translated into equivalent bit patterns using the command truth table of the simulated standard. Based on this data, the number of transmitted zeros n_0, transmitted ones n_1 and zero to one transitions $n_{0\to1}$ can be calculated. To achieve high simulation speeds, bit manipulation instructions including the population count (`POPCNT`) instruction are used. Instead of real data, it is also possible to provide a switching activity α and a duty cycle D (ratio between logic one and logic zero). In addition to the command/address and data bus, the remaining signals like the clock signal pair, data strobe pairs or chip select need to be considered (see Section 3.2).

The core power calculation is more complex because in addition to counting the number of issued commands of each type, DRAMPower needs to keep track of the clock cycles that the DRAM is in a specific state (i.e., 0 - B banks active, active/precharge power-down, self refresh). If a configuration with multiple ranks is simulated, the counting has to be done separately for each rank. A further difficulty arises from the fact that the internal state is not always changed immediately by an external command, but it can also change after a certain delay. An example for this behavior is shown in Figure 8. When a *read with auto precharge* command (`RDA`) is issued, the target bank is automatically precharged after the read to precharge delay t_{RTP} has expired. This means that the DRAM will internally issue what we call an *implicit command* in the future. Unfortunately, DRAMPower is not based on an event-driven simulation kernel like SystemC where an event can be directly notified in the future. Instead, it is only triggered from the outside when new commands are issued, so the implicit commands need to be handled differently. The actions that are performed by an implicit command are formulated as a lambda expression, which is stored in an internal list ordered by the time stamp of execution. Whenever DRAMPower is triggered from the outside, first, this list is searched from the beginning for implicit commands with time stamps less than or equal to the current simulation time. The lambda expressions of these list entries are then evaluated before the external command is handled. The total power consumption can be queried at any time even when the simulation is still running, which allows to analyze the change of power consumption over time.

6.2 Simulation Speed

Since DRAMPower is not operated as a standalone tool in the normal use case, but rather coupled to a behavioral DRAM subsystem simulator, we evaluate its simulation speed in terms of the overhead of adding power simulation. For this analysis, DRAMPower is coupled with DRAMSys [12], a well-known DRAM subsystem simulator, and executed on a server with two Intel Xeon Silver 4210R processors. Within DRAMSys, one million read and write requests with random addresses and data are generated. This simulation is carried out both with and without power simulation enabled. Moreover, the simulations are also performed without actual data.

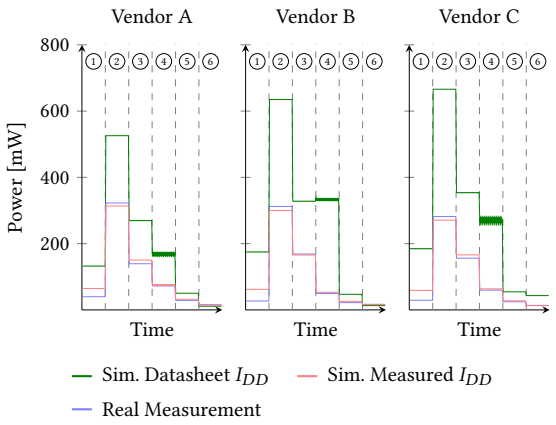


Figure 9: Average Power Consumption of Simulations and Measurements for Different Vendors

In this case, DRAMPower is provided with a switching activity α and a duty cycle D.

For the simulations with data, DRAMSys alone requires on average 9.10 s to finish, while with added power simulation, the average simulation time increases to 11.37 s. This corresponds to an overhead of 25 %. When no data is simulated, DRAMSys alone requires on average only 6.95 s to finish, while with DRAMPower enabled, the simulation time increases to 8.44 s. In this case, the overhead is 21 %. While this overhead may seem relatively large at first glance, there are two things to consider. Firstly, DRAMSys is highly optimized for simulation speed as was already shown in [12]. Secondly, if a full system simulation is performed where DRAMSys is additionally coupled to a much slower processor simulator such as gem5, the overhead of adding DRAMPower becomes negligible.

6.3 Simulation Accuracy

To verify the power estimates of the new DRAMPower implementation, we use core and interface power measurements[1] of LPDDR4 devices from three different vendors, as reported in a study of a memory measurement platform [5]. Unfortunately, no measurement results for a newer standard are publicly available. Each DRAM is operated with six different access patterns, which are analogous to the following I_{DD} currents: ① I_{DD0*} , ② I_{DD4R}, ③ I_{DD4W}, ④ I_{DD5B}, ⑤ I_{DD2N} and ⑥ I_{DD6}. As it was not possible to reproduce the usual I_{DD0} pattern of activate-precharge for the measurement platform, I_{DD0*} is a variation using an activate-read-precharge pattern, which is also resembled in the DRAMPower simulation. In addition, the platform could not accurately measure the operating burst write current I_{DD4W} because only one write request could be issued at a time. Thus, the simulation was also configured to limit the number of outstanding write requests to one. The initial simulations are based on the current values specified in the vendor datasheets. Then, based on the actual measurements, the current values are reapplied to a second simulation. The results are shown in Figure 9. As it can be seen, the currents specified in the datasheets are overly pessimistic for all three vendors: The simulations based on the datasheets show on average a 2.9 × higher power consumption than the actual measurements. However, when the measured currents are applied to the simulation, the deviation drops to only around 18.8 %. The largest share is caused by I_{DD0*}. For this pattern, it is unclear whether the measurement platform was actually able to fully saturate the memory controller's buffer and therefore reports a lower average power consumption than the simulations. Without I_{DD0*}, the deviation is only 2.8 %. This again highlights that truly accurate core power simulations are only possible with measured currents, while datasheet values provide a worst-case estimate.

[1]The measurements do not include the interface power of the memory controller PHY.

7 Conclusion and Future Work

In this paper, we have presented DRAMPower 5, a power simulator for current generation DRAM standards. It uses newly developed core and interface power models to flexibly support different standards and accurately capture the effects of high operating frequencies. In the future, we will continue to update the simulator to emerging standards and new features.

Acknowledgments

This work was funded in part by the German Federal Ministry of Education and Research (BMBF) under grants 16ME0935, 16ME0936 and 16ME0934K (DI-DERAMSys) as well as grants 16ME0717 and 16ME0716K (MANNHEIM-MEMTONOMY).

References

[1] H.B. Bakoglu. 1990. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley Publishing Company.

[2] Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu. 2018. Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (Williamsburg, VA, USA) *(ASPLOS '18)*. Association for Computing Machinery, New York, NY, USA, 316–331. https://doi.org/10.1145/3173162.3173177

[3] Katherine Bourzac. 2024. Fixing AI's Energy Crisis. *Nature* (Oct. 2024). https://doi.org/10.1038/d41586-024-03408-z

[4] Karthik Chandrasekar, Christian Weis, Yonghui Li, Benny Akesson, Omar Naji, Matthias Jung, Norbert Wehn, and Kees Goossens. Last Access 15.08.2019. DRAMPower: Open-source DRAM Power & Energy Estimation Tool.

[5] Johannes Feldmann, Lukas Steiner, Derek Christ, Thomas Psota, Matthias Jung, and Norbert Wehn. [n. d.]. A Precise Measurement Platform for LPDDR4 Memories. In *Proceedings of the International Symposium on Memory Systems* (Alexandria VA USA, 2023-10-02). ACM, 1–8. https://doi.org/10.1145/3631882.3631899

[6] Saugata Ghose, Abdullah Giray Yaglikçi, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X. Liu, Hasan Hassan, Kevin K. Chang, Niladrish Chatterjee, Aditya Agrawal, Mike O'Connor, and Onur Mutlu. 2018. What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study. *Proc. ACM Meas. Anal. Comput. Syst.* 2, 3 (Dec. 2018), 38:1–38:41. https://doi.org/10.1145/3224419

[7] Timothy M. Hollis, Eric Stave, Dave Ovard, Roy Greeff, Worfgang Spirkl, Martin Brox, Jennifer Taylor, and Justin Butterfield. 2019. Recent Evolution in the DRAM Interface: Mile-Markers Along Memory Lane. *IEEE Solid-State Circuits Magazine* 11, 2 (2019), 14–30. https://doi.org/10.1109/MSSC.2019.2910617

[8] Norman P. Jouppi, Andrew B. Kahng, Naveen Muralimanohar, and Vaishnav Srinivas. 2015. CACTI-IO: CACTI With OFF-Chip Power-Area-Timing Models. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 7 (2015), 1254–1267. https://doi.org/10.1109/TVLSI.2014.2334635

[9] Matthias Jung, Deepak M. Mathew, Éder F. Zulian, Christian Weis, and Norbert Wehn. 2016. A New Bank Sensitive DRAMPower Model for Efficient Design Space Exploration. In *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS 2016).*

[10] Deepak M. Mathew, Éder F. Zulian, Subash Kannoth, Matthias Jung, Christian Weis, and Norbert Wehn. 2017. A Bank-Wise DRAM Power Model for System Simulations. In *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools* (Stockholm, Sweden) *(RAPIDO '17)*. Association for Computing Machinery, New York, NY, USA, Article 5, 7 pages. https://doi.org/10.1145/3023973.3023978

[11] Micron. 2014. Micron System Power Calculator. last access 2024-11-12. https://www.micron.com/sales-support/design-tools/dram-power-calculator

[12] Lukas Steiner, Matthias Jung, Felipe S. Prado, Kirill Bykov, and Norbert Wehn. 2020. DRAMSys4.0: A Fast and Cycle-Accurate SystemC/TLM-Based DRAM Simulator. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer International Publishing, Cham, 110–126.

[13] Thomas Vogelsang. 2010. Understanding the Energy Consumption of Dynamic Random Access Memories. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. 363–374. https://doi.org/10.1109/MICRO.2010.42