A Data Effect Aware Microcomponent-Based Estimation Approach For Accurate System-Level Memory Device Power Evaluation

[1,2]Chi-Kang Chen rc@itri.org.tw
[1]Hsin-I Wu { hiwu.dery ,
[1]Chi-Ting Hsiao lineagenoya ,
[1]Ren-Song Tsay rstsay }@gmail.com

[1]Department of Computer Science, National Tsing-Hua University, Hsinchu, Taiwan
[2]Industrial Technology Research Institute, Hsinchu, Taiwan

Abstract - As memory is a major power dominant, we propose a highly efficient microcomponent-based approach with data-aware refinement for accurate system-level power estimation. The proposed method pre-calibrates the power consumption pattern of each identified microcomponent for power simulation. To achieve high accuracy, the data variation effect is considered and a simple interpolation technique is proposed to further boost accuracy. The proposed approach produces accurate results of less than 2% error rate in average for system-level power analysis.

I. Introduction

Energy efficiency has become a key concern in a typical computing system. According to recent reports [3, 14], the associated memory system devours 25% of total power and is expected to consume even more in the near future [3]. To perform memory system power optimization at an early design stage, designers strongly desire a fast and accurate memory system power estimation method for evaluation of target architectures. However, many recent research works, such as those of Vogelsang [7] and Chandrasekar [13], have raised issues regarding the fact that existing system-level memory power models are mostly based on the worst-case hardware measurement value, which may mislead designers in making decisions. To close the gap, we adapt the microcomponent-based power analysis idea, which has been proven effective for CPU designs [10], and propose a fast and accurate system-level memory system power estimation approach.

Traditional memory power models take each memory request instruction as a basic unit for power calculations and assume that each instruction consumes a fixed power value [1, 2, 4, 5]. In general, the traditional request-based power models work well for SRAM because the circuit structure of SRAM is simpler and the access operations are predictable. Therefore, SRAM power consumption behavior is relatively manageable. In contrast, DRAM access behavior is more complicated. Before a memory request instruction is sent into DRAM dice, it is first sent to the memory controller and translated into a sequence of internal DRAM commands that drive the actual DRAM circuit. The DRAM controller schedules these commands according to the DRAM's state (such as row hit/miss). For example, a *load word (lw)* instruction issued from CPU is translated by the DRAM controller into a sequence of three internal commands, *ACTIVE* (ACT), *READ* (RD), and *PRECHARGE* (PRE), and each command drives specific memory circuit components. The ACT command moves data out through the row decoder, data array and sense amplifier. The RD command puts data onto the bus from row buffer through I/O gating. The PRE command re-initializes sense amplifier for the next incoming command. Additionally, a DRAM memory hierarchy often consists of complicated channel, rank, bank, row and column structures. Moreover, the internal command execution time is in fact variable rather than fixed. To this issue, we find that independent of architecture difference, the DRAM microcomponents involved are basically the same. Therefore, with pre-characterized microcomponents, we can easily compose various memory architectures and estimate system-level memory power usage quickly and accurately based on precise internal command timing information derived from the given memory system scheduling policy of the corresponding architecture.

For the rest of this paper, we will focus our study on DRAM power analysis and hence, for clarity's sake, will use "command" to indicate an internal command and use "instruction" for an external command. To further understand and analyze the power consumption of DRAM circuits, we first show a sample power waveform in Figure 1. The DRAM circuit power consumption includes two parts, dynamic and static power. When each internal command is issued, input signals begin to propagate through the corresponding activated circuit and then hold at a certain stable state until the next internal command arrives. Accordingly, we further divide the active command interval into two parts: processing time and hold time. The command processing time, or signal propagation time, is a fixed value and can be computed through timing analysis. During the command processing time, circuit switching consumes dynamic power in addition to continuously consuming static leakage power. During the hold time, the circuit is in a stable state after the completion of command processing and consumes only static power. In the past, the most widely adopted DRAM power model is the Micron-like model [4], which assumes that each command type consumes a fixed energy. However, in practice almost all modern memory designs adopt concurrent interleaving memory access schemes with varying hold times and result in varying energy consumption in practice that can be far from the Micron-like model. To improve accuracy, IPMDS [6] assumes a constant power level for each command but computes actual hold time and derives more accurate energy consumption values. However, this model does not differentiate between power consumption values of command processing time

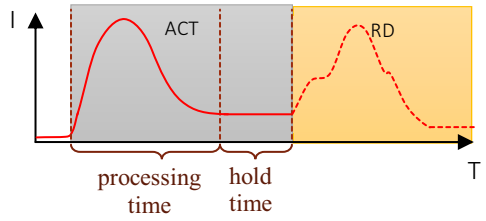


Fig. 1. An example of electric current waveform for a sequence of ACT and RD commands

and hold time. Hence, the estimation can be unsatisfactory for practical use. To improve accuracy, we made an important discovery: since most memory circuit microcomponents are engaged in passing data, the dynamic switching power is roughly proportional to the number of input bits whose values are changed. Thus, we have created a more accurate dynamic power model that considers data content, something never previously done before. To verify the accuracy of the proposed approach, we test on a few real memory designs. The experimental results show that our approach has less than 2% error rate on average, while the traditional approach has 29%~54% error rate with reference to detailed circuit simulation results.

II. Related Work

Most existing system-level memory power models estimate total power consumption by using the average power consumption per command, known as the per command power consumption value. The referred power values are from a standard datasheet based on measurement results. Among these models, the Micron model [4] has been most widely adopted due to its simplicity. Basically, to compute the standard energy consumption of each command, it multiplies the average recorded current through a specified period with the driving voltage. In order to differentiate the energy consumption value under specific scenarios, a few Micron-like models [1, 2, 5] take an input command sequence and multiply this standard energy consumption value with the command count number to calculate energy consumption. Essentially, Micron-like models do not consider hold time variations and hence can be far from accurate, particularly for advanced memory designs [6].

To fix the above-mentioned issues, IPMDS [6] takes into account timing information, which is determined by the memory controller. As opposed to the Micron-like model, IPMDS multiplies each recorded command power consumption value with the actual command issuing time interval to get the total energy value, so take into account the power effect of contention delays. In sum, IPMDS improves accuracy over the basic Micron-like model by considering more general timing policy. Nevertheless, both Micron and IPMDS models use the worst-case value for power calculation, which can be overly pessimistic in practice. Additionally, the measurement-based method is generally known to lack detailed power information regarding when and where power is consumed [7].

In contrast, other approaches mostly focus on power value calculation using a transistor model or established circuit floorplan [7, 8, 9], according to the circuit model provided. These approaches are not restricted to circuit or gate-level models. For instance, Vogelsang [7] studies the effect of circuit parameters, such as gate oxide thickness or wire capacitance, on power value. Chandrasekar et al. [12, 13] consider the power effect of 3D-stack and process variations. DArT [15] also model basic circuit units (such as a memory cell) as components, then construct memory design and estimate circuit area, power and timing values. While this component concept seems similar to our approach, we maximize evaluation efficiency by modeling the component based on function, rather than structure as DArT does. In general, the above approaches do provide precise power evaluations but are only applicable to small scale designs due to their tremendous computational complexity, making them impractical for gigabit-scale calculations. We therefore propose a microcomponent-based approach that precisely identifies the activated circuit components of each command and accurately models the dynamic and static power consumption values of each component in relation to the driving command. In the following, we elaborate our proposed approach.

III. Microcomponent-Based Approach

To construct an accurate system-level power estimation method, we propose a microcomponent-based approach, which partitions the whole memory circuit into microcomponents according to DRAM commands. We then use the power model of each microcomponent to derive detailed power information for the target design. In Figure 2, we show the flowchart of our proposed approach, for which we have a preparation phase and a simulation phase. In the one-time preparation phase, we first run through all possible commands and identify microcomponents. Then, we analyze each microcomponent's power consumption behavior under different active commands and construct a microcomponent-command power table.

In the simulation phase, we produce the power waveform and arrive at the total power consumption value. First, we analyze the timing behaviors of the input memory requests. The timing information identifies which command at specific time points. With the corresponding microcomponents activated by the memory commands, we construct the final power waveform simply by sequentially stacking up the power waveform from the microcomponent power table of each activated microcomponent. In our approach, a memory technology provider can independently complete the preparation phase and supply the microcomponent-command power table for the user's system-level evaluations.

A. Microcomponents

The microcomponent method for accurate power evaluation has been successfully applied to CPU designs [10]. The main observation is that the same instruction always activates the same specific circuit components. Similarly, for DRAM operations, the same command issued from the memory controller also always activates the same specific circuit components. We exploit this property and identify microcomponents in the memory circuit for precise power calculations. A unique benefit of using the microcomponent model for power analysis is that we can focus on analyzing the power effect on a limited active region without the need to consider other un-affected regions. By doing so, we can greatly reduce estimation errors. Next we will first explain how to identify microcomponents systematically.

Identify Microcomponents: A microcomponent is defined as the largest set of circuits used by a group of commands that activates

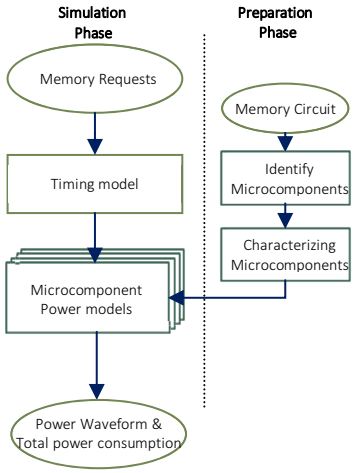


Fig. 2. The flowchart of our proposed microcomponent-based approach

this set of circuits during execution. Note that with this definition, no two microcomponents can be activated by a same set of commands. Therefore, we shall produce a minimum number of microcomponents for the target design investigated. In practice, we apply an iterative refinement procedure to generate the microcomponents. To start, we first set the whole circuit as a region and then propagate the control signals of a command and identify the activated circuit region. Note that this identified region can overlap with previously identified regions. We then separate each overlapped region into a new region, which obviously is activated by multiple commands. The non-overlapped area of each existing region will also be a separate region. Running through each command in sequence will thus produce regions each corresponding to a microcomponent. The above process can be illustrated using Table 1, which lists the activated DRAM units of five basic commands and the final identified microcomponents based on a typical DRAM architecture shown in Figure 3. To generate the table, we run through the five basic commands, *ACTIVE* (ACT), *READ* (RD), *WRITE* (WR), *PRECHARGE* (PRE), and *REFRESH* (REF), and list the activated units of each command in Table I. Following the microcomponent definition, we find that *Row Decoder* and *Data Array* are the largest group of DRAM units commonly activated by *ACT, PRE* and *REF* and hence are grouped as a microcomponent, uc_1. Similarly, *Sense Amp* is marked as the microcomponent uc_2 as it is the only unit used by every command. Likewise, *I/O gating* and *Col Decoder* are the largest group used by both *RD* and *WR* commands and are marked as microcomponent uc_3. The remaining two components *Write Driver* and *Read Latch* are used by *WR* and *RD* respectively and form separated microcomponents uc_4 and uc_5.

Microcomponent Power Model: Note that each microcomponent is activated by at most one command at a time. When a different command occupies the microcomponent, the circuit will execute a different function and hence perform at a different power level. Each microcomponent power model basically follows a same general pattern, which includes a fixed processing time period T_p and a variable hold time period T_h. In Figure 4, we show an example of a modeled power current waveform of a microcomponent, with the blue line indicating the average electric current value, a sum of both dynamic and static current values.

When a microcomponent is activated by a command, the input data start to propagate through the microcomponent during the processing period T_p. During this period, transistors in the microcomponent may switch states and consume dynamic power along with the constant static power consumption. After T_p, all data signals propagate through the microcomponent and the circuit state is stabilized, hence consuming only static power, no dynamic power. Then it turns into the hold time period T_h, during which the microcomponent remains at the same state and consumes only static power until the next activating command arrives. For each

TABLE I
THE ACTIVATED DRAM UNITS OF EACH COMMAND AND THE MICROCOMPONENTS IDENTIFIED.

DRAM unit \ Command	ACT	RD	WR	PRE	REF	uc
Row Decoder	✓			✓	✓	uc_1
Data Array	✓			✓	✓	
Sense Amp	✓	✓	✓	✓	✓	uc_2
I/O Gating		✓	✓			uc_3
Col Decoder		✓	✓			
Write Driver			✓			uc_4
Read Latch		✓				uc_5

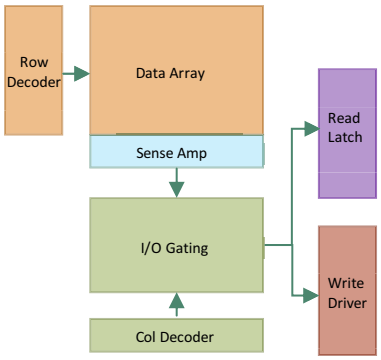


Fig. 3. A typical DRAM architecture.

microcomponent, we can use a detailed circuit analysis tool, such as SPICE, to compute the current waveform and average value of these two periods based on the given command and input data. In practice, the data content can significantly affect the estimated value. Nevertheless, due to the fact that the memory circuit handles mostly data pass-through, the variation of power consumption caused by different input data can be accurately estimated using a simple model, which we shall elaborate below.

Data Effect: In the past, the data effect is usually ignored in power modeling mainly because it is complex and hard to quantify. However, we find that the data content does significantly affect memory circuit power consumption values. For example, with extensive SPICE simulations, we observe that the minimum power consumption is only 14% of the maximum value for the command *WR* taking various data content. Therefore, considering data content change is crucial for accurate power estimation. To this issue, we propose a simple interpolation approach. We observe that the memory circuit is mainly for data storage or passing and hence it is mostly composed of the regular bitline array, with each data bit value affecting its own bitline independently. Therefore, if the bitline value is unchanged, the corresponding bitline circuit simply maintains the same static power level with no dynamic power consumption, whereas if the bitline changes value, the switching causes dynamic power consumption and the difference of the final state results in a different static power level. With this observation, we simply use the number of data bit value changes to estimate accurately both the dynamic and static power consumption values.

In doing so, we first analyze two extreme cases of the hold time for each microcomponent, using the power values P_0^h and P_1^h, which have input data bits of all 0's and all 1's, respectively. Then, we change the bits to all 1's or 0's, respectively, to compute the processing power values $P_{0\to1}$ and $P_{1\to0}$. Then, we have the following linear interpolation formula to compute the processing power

$$P_{proc} = (P_{0\to1}n_{0\to1} + P_{1\to0}n_{1\to0} + P_{0\to0}n_{0\to0} + P_{1\to1}n_{1\to1})/n,$$

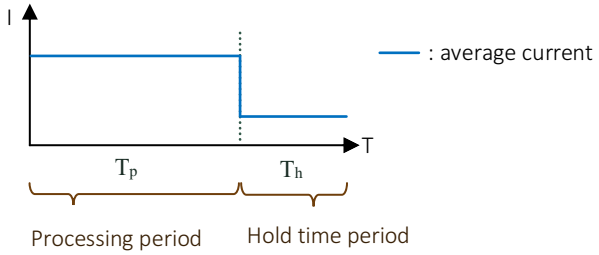


Fig. 4. An illustration of a modeled microcomponent power pattern.

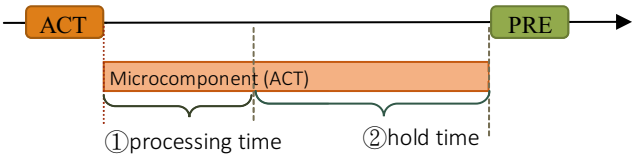


Fig. 5. An illustration of the microcomponent timing model.

and the following formula to compute hold time power

$$P_{hold} = (P_1^h n_1 + P_0^h n_0)/n,$$

where n is the total number of data bits, n_1 is the number of data bits whose final value is 1, n_0 is the number of data bits whose final value is 0, and $n_{x\to y}$ is the number of data bits whose value changes from x to y.

Verify Data-aware Power Correction: Our experiments reveal an interesting fact, that the active data value to be processed has more impact on power than the previous data value. This is because a DRAM circuit charges bitlines following the active data value. In contrast, previous data values affect only initial logic switching but has nearly no effect on later bitline charging, which actually consumes most power. To verify our proposed data-aware interpolation power correction model described previously, we perform tests on a 32-bit wide data array circuit. We first use SPICE to generate the corresponding power consumption golden references by charging various numbers of bitlines from low to high value. Then, we apply our proposed interpolation approach for power consumption estimation. The comparison results are summarized in Table II, which shows that our interpolation method produces results of less than 1% error rate. To verify whether the interpolation method is sensitive to bitline position, we conduct another test by charging the same number of bitlines on a few sets of randomly selected bitlines. The results show that they all produce the same power value. This suggests that our interpolation approach is insensitive to bitline position and requires only knowing the number of bitlines of value 0 or 1. Now, using the microcomponents and their associated power information, we can accurately calculate the total power consumption once we know precisely when the processing period and the hold time period take place.

B. Memory Timing Model

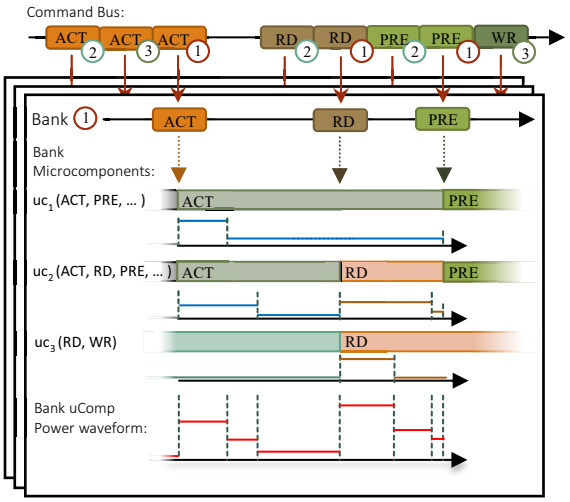
Using the accurate power models of each microcomponent generated based on the methods discussed above, to calculate energy consumption value precisely we simply need to know the processing time and hold time of each command on the microcomponent. In fact, the memory controller controls the scheduling of all commands according to the statuses of the channel, bank and bus. Therefore, to precisely compute the timing information, we simply monitor the command sequence at the output of the memory controller. Once we know the target bank of a command, we can easily derive the command timing interval.

After a microcomponent is triggered by a command, the timing model requires two parameters: processing time T_p and hold time T_h, as shown in Figure 5. The processing time is the minimum time that the microcomponent to complete the data propagation. Note that we have setup time included in the processing time to simplify the model. The hold time is a varying time period during which signals are stabilized while waiting for next activating command. Processing time is a constant and determined by the circuit characteristics of the microcomponent. After identifying the exact processing time and hold time of each command, we simply look up the corresponding processing time power and hold time power from the microcomponent power table and compose the final total power waveform.

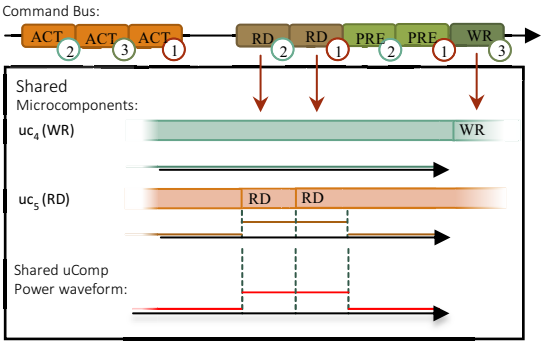
TABLE II
COMPARING THE ACCURACY OF INTERPOLATION APPROACH WITH GOLDEN REFERENCE

# bit changes	0	1	12	22	31	32
Error Rate	0%	0.9%	1%	1%	1%	0%

Note that the same microcomponents are used regardless of the bank-channel architecture; except that write driver and read latch microcomponents are shared by all banks while other microcomponents all have bank-specific copies. We illustrate our approach in Figure 6 with the memory bank configuration and the corresponding bank-specific and shared microcomponents. First, we monitor the command bus and derive a precisely timed command trace, as shown on the top of Figures 6(a) and 6(b). Figure 6(a) demonstrates how to compute individual bank power. For example, suppose that Bank 1 executes the command sequence ACT→RD→PRE. With the command timing information as shown, we then check how each activated microcomponent contributes to the final total power waveform. For instance, uc_1 is first triggered by ACT and then re-initialized by PRE, but not affected by the RD command. For the ACT activating time period, which stops when PRE is activated, we first have the fixed ACT processing time and



(a). The power waveform of individual bank. ① indicates bank i.



(b). The power waveforms of shared microcomponents.

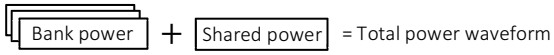


Fig. 6. An illustration of generating the total power waveform using command issuing time and microcomponent power waveform.

leave the rest as the hold time. We put the ACT-uc_1 processing time power and hold time power into the corresponding time periods just computed. Afterward we use a similar process to match the PRE power pattern to the right time point corresponding to uc_1.

In contrast, uc_2 is sequentially activated by ACT, RD and then PRE. Therefore, the hold time period for ACT on uc_2 is clearly shorter than that of ACT on uc_1. In a similar procedure as used above for uc_1, we produce the power waveform of uc_2.

After power waveforms of all three bank-specific microcomponents are generated, we simply sum up all power values and produce the final total bank power waveform of Bank 1, shown on the bottom. Similarly, we can generate the power waveform for other banks and use them to produce the total bank power.

For the two shared microcomponents shown in Figure 6(b), we monitor the command bus and find RD and WR commands that activate shared microcomponents. For example, uc_5 takes RD of bank 1 and RD of bank 2 as its input command and produces the corresponding power waveform. As with bank power, we sum up all the shared microcomponent power to derive the shared power. With bank and shared power, we can eventually generate the total memory system power waveform.

IV. Experimental Results

To demonstrate our proposed approach, we adopt the DRAMsim2 [1], which include both DRAM controller and dice model, as the base system-level DRAM simulator. The cycle-accurate DRAMSim2 takes a stream of memory instructions (read/write) as the input. The DRAM command scheduler within the controller generates timing-annotated DRAM commands (ACT, RD, WR, PRE, and REF) according to given timing constraint. These timing-annotated DRAM commands are then used to calculate the power waveform, as illustrated in Figure 6, and simultaneously to compute the IPMDS waveform.

In fact, the DRAMSim2 can also perform execution-driven simulation. In this mode, it integrates with a full system simulator [16-17] and receives memory instructions dynamically from the full system simulator at run time instead of from a trace file. However, since full system simulations are excessively time consuming; this method is adopted less often in practice. Before we proceed to system-level simulation, we first verify the model accuracy.

A. Verify Accuracy

To validate the accuracy of our power model, we simulate a 2 MB 16 Banks 250 MHz eDRAM design and use the results of SPICE simulation as golden references. We first identify five microcomponents from the eDRAM design and then generate the power pattern of each microcomponent and command based on SPICE simulations. At the same time, we also use the SPICE results of the whole circuit and generate the equivalent Micron and IPMDS power models. The proposed microcomponent approach maintains an average error rate of less than 2%, which is far more accurate than the MICRON and IPMDS models in reference to the SPICE golden results, as shown in Table III. In the case of the ACT_NOP_PRE command sequence, which occurs during the idling phase, both Micron and IPMDS produce results with a 5~7% error rate as there is no hold time variation in accesses of short transaction length.

With WR_8, we have a typical access command sequence, which includes a WR and a burst RD sequence of length 8, to represent cases with command switching, data processing and long transaction intervals. In this scenario, due to the insensitivity of processing time and hold time differentiation, the Micro-like and IPMDS approaches results in significant accuracy loss. In general, for command sequences with longer transaction lengths and for more complex command schedules such as those found in WR_8, both the Micron and IPMDS models show significant error rate increases, with the Micron-like models producing more errors. As discussed previously, our approach outperforms existing methods mainly because we use microcomponents for analysis and ignore uninvolved regions. Additionally, our data interpolation approach, which take into account the power effect of data content differences, also contributes to our more accurate results.

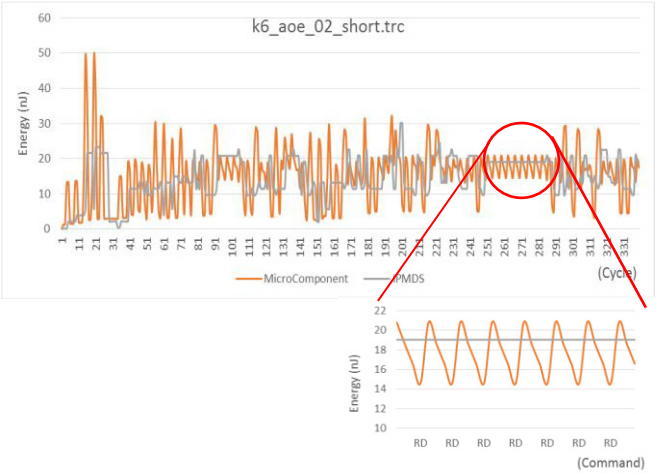


Fig. 7. Comparing the system simulated power waveform of IPMDS and our approach.

B. Trace-Driven Memory Simulation

To test our approach, we use the memory access trace file "k6_aoe_02_short.trc" that comes with the DRAMSim2 project as the input for our experiment. The target 2MB 16 Banks 250 MHz eDRAM design was simulated using the trace file to generate the timing-annotated command trace file. Both the proposed approach and the IPMDS approach were then simulated using the timing-annotated command trace file to generate power waveforms for comparison. Figure 7 shows a partial waveform segment from the system power simulations generated from both our approach (Microcomponent) and IPMDS. The proposed microcomponent-based power waveform is realistic and contains detailed and accurate timing information compared to the traditional approach. For instance, examine the segment from time clock cycle 251 to cycle 281 encircled in red, in which the DRAM circuit repeatedly issues RD commands and reading from different banks. In this scenario, the corresponding power waveform shows a periodical waveform, in orange, which reflects the repetitive executions of RD commands. In contrast, IPMDS cannot show the

TABLE III
Comparing The Accuracy of Our Method with Micron and IPMDS Based on The Total Energy Value.

Command Sequence		ACT_NOP_PRE		WR_8			
Transaction length		short		long			
Process data		No effect		All 0		All 1	
Initial data		All 0	All 1	All 0	All 1	All 0	All 1
Error rate	Micron	7.02%	5.76%	29.50%	31.28%	54.78%	53.49%
	IPMDS	7.02%	5.76%	11.92%	13.45%	33.77%	32.65%
	Microcomponent	1.11%	2.07%	1.55%	2.38%	2.32%	2.35%

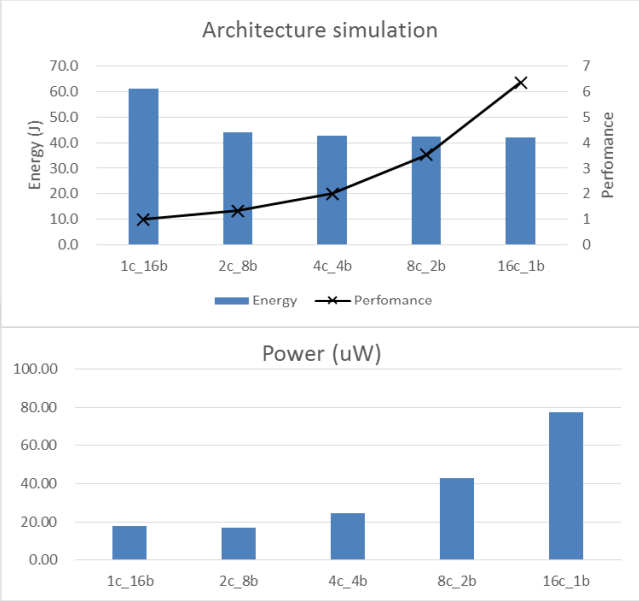


Fig. 8. Comparing power, energy and performance of various architectures.

rise and fall of power variations.

C. Architecture Evaluation

To demonstrate the effectiveness of our approach for architecture evaluation, we take a DRAM design with a total size of 64MB and use our tool to evaluate a few different architectures. Note that in general, the microcomponents are the same even though the architectures differ. However, if needed, we check if there are new microcomponents. For our test, we use different channel/bank numbers (1 channel*16 banks, 2c_8b, 4c_4b, 8c_2b, 16c_1b) and run the trace simulation described in Sec. 4.2, then compare power, energy consumption and performance. In Figure 8, we show the total memory energy consumption and data access performance, normalized to 1c_16b. We can observe that the performance increases when the channel number increases. This is mainly due to the fact that more channels facilitate higher potential for parallelism. Although the power increases sharply as the channel number increases, energy consumption actually decreases due to the shorter execution time. To maximize energy efficiency and performance, 16c_1b is obviously the best choice among these five cases, but if we consider peak power, then 4c_4b or 8c_2b may be a better choice.

V. Conclusion

In this paper, we have proposed an accurate microcomponent-based system-level power estimation method. We leverage the fact that each memory command is active only in certain microcomponent regions in order to achieve highly accurate power simulations. Our approach maintains high accuracy even under arbitrary scheduling policies and can easily produce detailed power waveforms for examination. The reusability of microcomponents is ideal for early stage power evaluation of systems with new design elements or architectures. In particular, with the advent of 3D-IC, our proposed approach is useful for detailed yet fast and accurate power analysis. Additionally, a unique advantage is that vendors can independently provide microcomponent power models for users without fear of revealing technology details.

Acknowledgements

This research was supported by MOST TAIWAN project 102-2221-E-007-141-MY3. We also thank Prof. Meng-Fan Chang and his lab at National Tsing-Hua University, Hsin-Chu, Taiwan, for providing memory test circuits for experiments.

References

[1] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, “Dramsim2: A cycle accurate memory system simulator,“ *Computer Architecture Letters*, 2011.

[2] Q. Deng, L. Ramos, R. Bianchini, D. Meisner and T. Wenisch, "Active low-power modes for main memory with MemScale," in *IEEE Micro*, vol. 32, no. 3, pp. 60-69, May-June 2012.

[3] L. Minas and B. Ellison, “The problem of power consumption in servers”, *Intel Corporation. Dr. Dobb's,* 2009.

[4] Micron Technology Inc., “TN-41-01: Calculating memory system power for DDR3”, *Technical Report*, 2007.

[5] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna and C. Le, "RAPL: Memory power estimation and capping", *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on, Austin,* TX, USA, 2010, pp. 189-194.

[6] K. Chandrasekar, B. Akesson and K. Goossens, “Improved power modeling of DDR SDRAMs”, *Digital System Design (DSD), 2011 14th Euromicro Conference on. IEEE*, 2011. p. 99-108.

[7] T. Vogelsang, “Understanding the energy consumption of dynamic random access memories”, *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society*, 2010. p. 363-374.

[8] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, *CACTI 6.0: A tool to model large caches,* HP Laboratories, 2009.

[9] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman and N. P. Jouppi, "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies", *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, Beijing, 2008, pp. 51-62.

[10] Z.-C. Huang, C.-K. Chen and R.-S. Tsay, "AROMA: A highly accurate microcomponent-based approach for embedded processor power analysis", *The 20th Asia and South Pacific Design Automation Conference*, Chiba, 2015, pp. 761-766.

[11] S. M. Min, H. J. and S. Parameswaran, “Xdra: Exploration and optimization of last-level cache for energy reduction in ddr drams”, *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013. p. 22.

[12] K. Chandrasekar, C. Weis, B. Akesson, N.t Wehn and K. Goossens, “System and circuit level power modeling of energy-efficient 3D-stacked wide I/O DRAMs”, *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013. p. 236-241.

[13] K. Chandrasekar, C. Weis, B. Akesson, N. Wehn and K. Goossens, “Towards variation-aware system-level power estimation of DRAMs: an empirical approach”, *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013. p. 23.

[14] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich and R. S. Schreiber, “Future scaling of processor-memory interfaces”, *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on.*, 2009. p. 1-12

[15] H.-C. Shih, et al., “DArT: A component-based DRAM area, power, and timing modeling tool”, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* (Volume:33, Issue: 9), 2014. p. 1356-136

[16] A. Patel, F. Afram, S. Chen, and K. Ghose, “MARSS: A full system simulator for multicore x86 CPUs”, *Design Automation Conference (DAC)*, 48th ACM/EDAC/IEEE, New York, NY, 2011, pp. 1050-1055.

[17] N. Binkert, et al., “The gem5 simulator”, *SIGARCH Computer Architecture News 39*, 2 (August 2011), 1-7.