2011 14th Euromicro Conference on Digital System Design

Improved Power Modeling of DDR SDRAMs

Karthik Chandrasekar
Computer Engineering
TU Delft, The Netherlands
Email: k.chandrasekar@tudelft.nl

Benny Akesson, Kees Goossens
Electronic Systems
TU Eindhoven, The Netherlands
Email: {k.b.akesson, k.g.w.goossens}@tue.nl

***Abstract*— Power modeling and estimation has become one of the most defining aspects in designing modern embedded systems. In this context, DDR SDRAM memories contribute significantly to system power consumption, but lack accurate and generic power models. The most popular SDRAM power model provided by Micron, is found to be inaccurate or insufficient for several reasons. First, it does not consider the power consumed when *transitioning to power-down and self-refresh modes*. Second, it employs the *minimal timing constraints* between commands from the SDRAM datasheets and *not the actual duration between the commands* as issued by an SDRAM memory controller. Finally, without adaptations, it can only be applied to a memory controller that employs a *close-page policy* and *accesses a single SDRAM bank* at a time. These critical issues with Micron's power model impact the accuracy and the validity of the power values reported by it and resolving them, forms the focus of our work.**

In this paper, we propose an improved SDRAM power model that estimates power consumption during the state transitions to power-saving states, employs an SDRAM command trace to get the actual timings between the commands issued and is generic and applicable to all DDR*x* SDRAMs and all memory controller policies and all degrees of bank interleaving. We quantitatively compare the proposed model against the unmodified Micron model on power and energy for DDR3-800. We show differences of up to 60% in energy-savings for the precharge power-down mode for a power-down duration of 14 cycles and up to 80% for the self-refresh mode for a self-refresh duration of 560 cycles.

***Index Terms*—DDR SDRAMs; Power Modeling; Power Estimation; State Transitions; Power-Down; Self-Refresh; Bank-Interleaving; Open-page; Close-page; SDRAM Command Trace;**

I. Introduction and Motivation

Design-time and run-time power estimation is often used for obtaining power/performance trade-offs, as per the design requirements of modern embedded Systems-On-Chip (SoCs). DDR SDRAM memories contribute considerably to SoC power consumption [4] and accurate power analysis of SDRAMs is critical for defining their run-time power management policies and for overall SoC design space exploration. For this purpose, Micron's SDRAM power model [1] is a widely accepted and employed tool. However, it is found to be inaccurate or insufficient for several reasons including:

1) It *does not consider* the power consumed during the *state transitions* from any arbitrary SDRAM state to the power-down and self-refresh states, reporting optimistic power saving numbers for these modes. Schmidt et al., also empirically verified this shortcoming of Micron's power model in [2].

2) It employs the *minimal timing constraints* between successive commands from SDRAM datasheets [13], [14] and not the *actual duration* between them as issued by an SDRAM controller, which may well be greater than the minimum constraints. Direct scaling of the power estimates obtained from Micron's power model gives pessimistic power consumption values for basic SDRAM operations, such as reads and writes.

3) It cannot directly provide power consumption values when an *open-page policy* or a *multi-bank-interleaved memory access policy* [19] is employed. This is because, *it assumes a close-page policy* by default and is directly applicable only when a single SDRAM bank is accessed. When multiple banks are accessed in parallel, Micron's power model requires adaptations for proper power and energy estimation.

4) It does not take into account the *power consumed during the pre-refresh clock cycles used to precharge all banks before executing a Refresh*, as a part of Refresh power.

This paper addresses all of the aforementioned issues by proposing an improved SDRAM power model for all DDR*x* SDRAMs. The proposed power model takes into account all possible state transitions from any arbitrary SDRAM state to the power-down and self-refresh states based on JEDEC specifications [15] [16]. Our generic power model accepts a cycle-accurate SDRAM command trace of any length (from a single transaction to an entire application trace) from any memory controller, supporting both open and close-page policies and any degree of bank-interleaving memory access scheme. Our proposed power model employs the actual timings between commands obtained from any such SDRAM command trace, in combination with the measured current and voltage values reported by memory vendors in SDRAM datasheets. Current users of Micron's model, such as DRAMSim [17] and MemScale [7], can benefit from our proposed power model, as it can report improved SDRAM energy estimates for any window of analysis (from a single transaction to an entire application).

The remainder of this paper is organized as follows: Section II discusses the related work in power modeling of SDRAMs. Section III gives the background information on SDRAM organization, operation and timing constraints. Section IV describes our approach to deriving the proposed power model. In Section V, we propose power equations for the basic power components to address issues (2) to (4) mentioned above. We then address the issue of state transitions to power-down modes and the self-refresh mode in Sections VI, VII and VIII. Specifically, Section VI provides power equations for the transitions from the stand-by (idle) mode to the power-down modes. Section VII addresses transitions from any arbitrary active mode of the memory to the power-down modes and Section VIII discusses the transitions to the self-refresh mode. In Section IX, we compare our power model against Micron's for different memory states and transitions and for an H.263 video decoder application. Section X concludes the paper, highlighting the significance of our contributions.

II. Related Work

Micron's SDRAM power model [1] is most widely accepted and has several current users including the likes of DRAMSim [17], MemScale [7] and [12]. However, it is found to be inaccurate or insufficient mainly due to the four issues discussed in Section I. Schmidt et al., in [2] and [3] empirically measured the power values from a DDR SDRAM and showed that Micron's power model provided approximate and worst-case power consumption numbers and over-estimated the actual savings of the Self-Refresh mode for SDRAMs. They also attributed these discrepancies to the fact that Micron's power model does not cover the state transitions to the Self-Refresh or the other power saving modes and verified this using different benchmark applications.

Other existing SDRAM power models suggested by Rawson [5], Joshi et al. [6] and Ji et al. [9], propose similar SDRAM power modeling like Micron, but none of them identified or addressed the state transitions issue and hence, do not provide any improved power estimation numbers. On the other hand, Joo et al. in [8] employed an energy state machine for SDRAMs and derived energy coefficients for the different memory states and state transitions to obtain more accurate power estimates. However, their power model cannot directly employ an SDRAM command trace and obtain the actual timings between the commands, and therefore, cannot be used to obtain accurate power estimates.

Memory power estimation tools like CACTI [11] can provide more accurate power consumption values than the other analytical power models and can be used for evaluating different memory features during the design space exploration of memory architectures. However, CACTI requires detailed understanding of memory architectures and cannot be employed for obtaining run-time memory power estimates for a given application, which is what is required by SoC designers.

Another promising tool by Thomas Vogelsang at Rambus Inc [10], also aims to provide accurate power consumption numbers for every component in the DRAM architecture. It employs device-level details and technology specifications to calculate the power numbers based on the switching activities and associated frequency of operation. However, since memory vendors do not provide easy access to such detailed specifications for their SDRAM memory architectures, the use of this model is very limited. Hence, the most viable method of estimating SDRAM power consumption in an SoC, is still to use the current and voltage values from the SDRAM datasheets [13], [14] that are based on real measurements, as used by Micron and us. However, it should be kept in mind that the correctness of the power model using these current measures, defines the accuracy of the reported power values.

In this paper, we propose an accurate and generic SDRAM power model that employs these measured current and voltage values from SDRAM datasheets, to provide transaction-accurate design-time and run-time power and energy estimates. Our proposed power model employs the actual timings between commands from an SDRAM commands trace, caters to all DDR*x* SDRAM memories and all memory controllers with any arbitrary scheduling policy and provides accurate power and energy estimates for any window of analysis.

III. SDRAM Organization and Operation

This section introduces the generic SDRAM architecture, its operation and the associated timing constraints [18], for better understanding of the proposed power model.

SDRAMs are organized in banks, rows and columns, as shown in Figure 1. A bank includes memory elements (cells) arranged in a matrix structure and a row buffer (with sense amplifiers) to store contents of an active memory row. The banks in an SDRAM operate in a parallel and pipelined fashion, though only one bank can perform an I/O operation at a particular instance in time and only one SDRAM command may be issued to the memory per clock cycle. The memory may operate in active, idle or power-down state and can have one or more banks active in parallel, based on the degree of bank-interleaving employed by the memory controller.

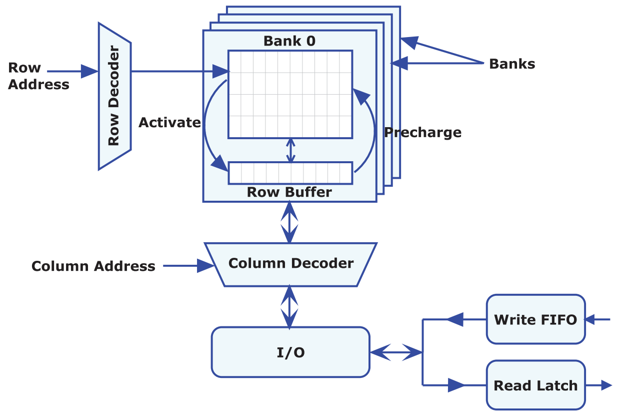


Fig. 1. SDRAM Architecture

The basic SDRAM commands issued to the memory (shown in Figure 1) include the following:

(a) A *Precharge* (PRE) command: Precharges the bit lines (columns) across all the memory rows to the reference voltage level and restores the contents of the row buffer (if any) back into the memory array.

(b) An *Activate* (ACT) command: Activates the word line of the indicated memory row and transfers the contents from the memory cells in that row to the row buffer for further access.

(c) A *Read* (RD) command: Reads out a burst of data (Burst Length of 4 or 8 words) from the row buffer.

(d) A *Write* (WR) command: Writes the accompanying burst of data (4 or 8 words) to the specified columns in row buffer.

(e) A *Refresh* (REF) command: Refreshes the rows in the memory at regular intervals to recharge the memory cells to retain the data in the memory.

In addition to these commands, it is also possible to transition to power-down state by disabling the clock at run-time to reduce power consumption, if the memory is not in use. It is also possible to retain the memory contents in the power-down state by employing the Self-Refresh feature, to refresh the memory at significantly lower power consumption.

For proper SDRAM operation, the commands discussed above must be issued by the memory controller in a specific order, while satisfying the associated timing constraints (for DDR2 [15] and for DDR3 [16]). For instance, between issuing an Activate and a Read command, the minimum timing constraint of *tRCD* should be respected. Some of these constraints that need to be satisfied when issuing commands to a DDR3-800 memory [14] are specified in Table I:

TABLE I
MICRON DDR3-800 TIMING CONSTRAINTS

Constraint	*Description (Minimum Time between)*	*Time (cycles)*
tRC	Two ACTs to the same bank	20
tRAS	An ACT and a PRE to the same bank	15
tRCD	An ACT and a RD/WR to the same bank	5
tRP	A PRE and next ACT to the same bank	5
tRFC	A REF and the next ACT	44

These timing constraints obtained from the datasheets are the minimal timings between two commands. However, most SDRAM controllers do not always issue commands as soon as these minimal constraints are satisfied. Instead, they schedule commands based on different command scheduling and row-buffer management policies, where the actual duration between any two issued commands may be greater than the minimum. For instance, the memory controller may employ an open-page policy [18] and delay issuing a precharge to a bank until there is a row-miss on the subsequent access to that bank.

In general, memory controllers decide to employ the open-page policy or the close-page policy [18] based on the presence or absence of data locality in the target application. The former policy keeps the row buffer active to reduce the access time for subsequent accesses to the same memory row in the same bank, by not issuing a *Precharge* command at the end of a transaction. The latter policy strictly closes the active row buffer at the end of every transaction to a bank with a *Precharge* command, for faster accesses to any other random location in the memory in the subsequent transaction. Additionally, read and write transactions can also be issued with an auto-precharge flag to automatically precharge as soon as the transaction completes. Our generic power model is devised to support both these row-buffer management policies.

IV. OUR APPROACH

In this paper, we present equations to accurately model power consumption of different SDRAM operations and estimate power savings for the different power-down modes and the self-refresh mode. For this, we employ the actual timing durations between successive commands issued by an SDRAM controller (obtained using an SDRAM command trace), instead of the minimal timing constraints from the datasheets, as employed by Micron [1]. We take into account the power consumed during the *state transitions* from any arbitrary active/idle mode to the power-down mode and from an idle state to the self-refresh mode. In short, we propose a generic power model that is applicable to all DDR*x* SDRAM memories and can be used with any memory controller using any row-buffer management policy (open-page or close-page), any command scheduling policy, and any degree of bank parallelism or interleaving. We achieve this by employing a five-step approach, as described below:

1. We execute a given application on a cycle-accurate instruction set simulator and *filter the accesses to the SDRAM memory*. These are forwarded to the SDRAM memory controller, where *the memory commands with all the relevant signals are logged*, to get the SDRAM command trace.

2. We the employ this *SDRAM command trace* to get *the actual timings between commands*, as opposed to the minimum timings from the datasheets.

3. We observe *the changes in the signals to the memory* in the logged trace, to identify the state transitions and the usage of the power-saving modes.

4. We identify *the current values for the states and state transitions* using JEDEC specs and the signals to the memory.

5. We *derive the power consumption values* for the different SDRAM states and state transitions, using all the details and specifications collected in steps (1) to (4).

To identify appropriate current consumption values for the different state transitions to the power-down or self-refresh mode, we observe: (a) the state the memory is in (active/precharged) before entering the power-down/self-refresh mode, (b) the state it is expected to be in (active/precharged) after powering back up or after exiting the self-refresh mode, and (c) the changes to the CKE (Clock Enable) signal. We obtain the timing requirements for those state transitions from the JEDEC specified requirements (shown for DDR3 in Figure 2), identify the duration of the transitions from the trace and accurately calculate their energy consumption. Using this approach, we obtain the power consumption values for any such transition and compare our estimates against Micron's.

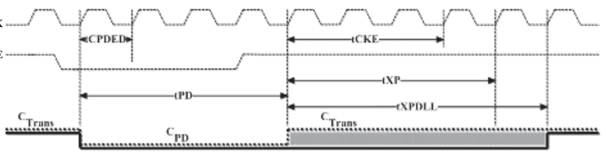
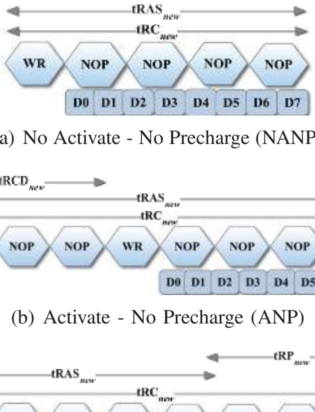


Fig. 2. DDR3 Power-Down Transitions and Power Consumption

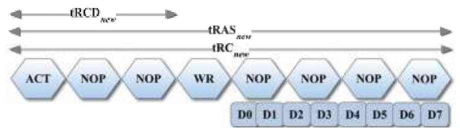
As shown in the figure, *tCPDED*, *tCKE*, *tXP* and *tXPDLL* contribute to the transition periods when switching to the power-down mode. Micron's model assumes the power-down current consumption (C_{PD}) for both the transition period and the actual power-down period *(*tPD) (as indicated by the solid line at the bottom in Figure 2). We corrected this flaw, and identified the correct current consumption (C_{Trans}) (shown by the dotted line at the bottom in Figure 2) for the transition periods. The shaded area in Figure 2, refers to the difference in the current estimates reported by Micron and our model during the clock cycles covering this transition period.

When it comes to regular transactions, Micron's model employs the minimal timing constraints (Table I) like *tRC* (minimum duration between two Activates to the same bank), as the transaction length to calculate power consumption for the transaction. We instead propose to employ the actual *transaction length* denoted by (tRC_{new}) for every individual transaction, as observed in the command trace of an SDRAM memory controller, to calculate the exact power consumption for that particular transaction. Note: tRC_{new} is used to represent the transaction length for all transactions and should not be misread as being the actual timing between two ACTs to the same bank, instead of *tRC*. All the actual timing parameters are hereafter referred with a suffix 'new' and the minimal timing parameters, without this suffix. *Note:* In the context of our power model, *a read or write transaction ends when the corresponding data transfer or the associated auto-precharge* *(if any) finishes*. Similarly, *an idle transaction is defined by the duration of the continuous period of idle clock cycles*. We further clarify on the transaction lengths associated with every operation, as and when we discuss them.

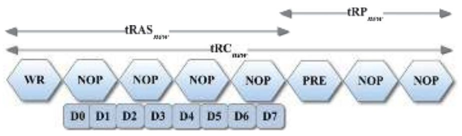
Micron's model [1] assumes that an *Activate* is always followed by a *Precharge* (close-page policy) in every transaction, at the end of the minimum active period *tRAS*. This assumption rules out other policies like the open-page policy, where a *Precharge* is not always used in every transaction and the active period and the transaction length may be longer than the minimum. Our generic model addresses this issue by estimating power consumption of a transaction on a case-by-case basis, where a transaction may or may not have an Activate and/or a Precharge command (open-page policy). Our model also addresses scenarios when two or more transactions are executed in parallel in different banks, as it independently monitors the commands and signals to every bank and each bank state on every clock cycle. Our generic power equations can hence be employed individually for every transaction, thus arriving at transaction-accurate power estimates. It should also be noted that Micron's power model is directly applicable only when a single SDRAM bank is accessed, whereas our generic model can be applied with any degree of bank-parallelism. Figure 3 depicts the actual timing values for the parameters in Table I and the command and data transfer cycles for different transactions based on different memory access policies.



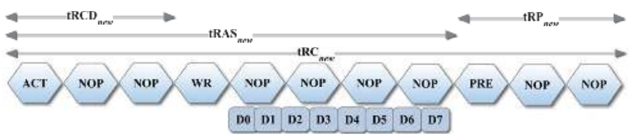
(a) No Activate - No Precharge (NANP)



(b) Activate - No Precharge (ANP)



(c) No Activate - Precharge (NAP)



(d) Activate - Precharge (AP)

Fig. 3. Actual Timing Parameters for different transaction types

As indicated in Figure 3, these transactions differ in their usage of activates and precharges as per the different policies:

(1) NANP: No Activate and no Precharge (Figure 3(a)), because the last and the next transactions are to the same row in the same bank as the current transaction.

(2) ANP: An Activate but no Precharge (Figure 3(b)), because the last transaction precharged the bank and the next transaction is to the same row in the same bank as the current.

(3) NAP: No Activate but a Precharge (Figure 3(c)), because last transaction was to the same row in the same bank as the current and the next is to a different row in the same bank.

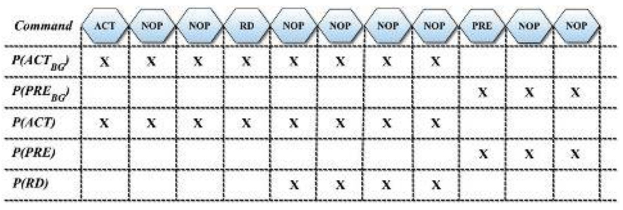
(4) AP: Both an Activate and a Precharge (Figure 3(d)), because the last and the next transactions are to a different row in the same bank.

In a nutshell, our approach addresses state transitions, employs actual timings between commands and is applicable to all memory controller policies. Our approach adheres to JEDEC specifications for current and timings and we derive our power model on the logical basis of this approach.

V. Basic SDRAM Power Model

Micron has identified the basic power components that add up and contribute to overall memory power consumption [1]. These basic components include background power components (contributing mainly to static power consumption), such as Active Background (Act_{BG}) and Precharged Background (Pre_{BG}) power, and active power components (contributing mainly to dynamic power consumption), such as Activate (*ACT*), Precharge (*PRE*), Read (*RD*), Write (*WR*) and Refresh (*REF*) power. Unfortunately, Micron employs the minimal constraints (issue (2) raised in Section I) to calculate power consumption of these basic components. In this section, we present alternatives to Micron's equations for these basic power components, considering the actual timings between commands. In V-A, we cover the background power components, in V-B, we detail power consumption of ACT and PRE commands and in V-C and V-D, we derive equations to provide power consumption of read, write and refresh commands. In V-E, we discuss the auxiliary power components that correspond to every read and write command, including the 'I/O' power and the 'Termination' power. In V-F, we provide a generic equation that combines all the basic power components to compute power estimates for a trace of any length from a single transaction to an entire application.

A general rule of thumb is that the background power components (static power elements) scale up with increase in the timing parameters, since they are always consumed whenever the memory is 'ON' (leakage). On the other hand, the active power components (dynamic power elements) scale down with increase in timing parameters, since they contribute to power consumption only for the period when they are used (based on the switching activity), and get averaged over the actual transaction length (tRC_{new}). The basic power components that add up for a sample read transaction with burst length 8 (using a close-page policy) are shown in Figure 4. The clock cycles in which they are consumed are indicated by 'X', for instance, *P(RD)* is consumed over 4 cycles of data transfer.



Command	ACT	NOP	NOP	RD	NOP	NOP	NOP	NOP	PRE	NOP	NOP
$P(ACT_{BG})$	X	X	X	X	X	X	X	X			
$P(PRE_{BG})$									X	X	X
$P(ACT)$	X	X	X	X	X	X	X	X			
$P(PRE)$									X	X	X
$P(RD)$					X	X	X	X			

Fig. 4. Basic Power Components in a Read Transaction

A. Background Power

If all memory banks are in the precharged stand-by state, the memory consumes a precharge background current (static power component) of I_{DD2N} [16]. However, even if a single bank is in the active state, the memory consumes an active background current (also static power component) of I_{DD3N}. Using these current specifications with the actual timings, if a bank stays in the active state for a period of $tRAS_{new}$ cycles out of the total transaction length of tRC_{new} cycles, it consumes an average $P(Act_{BG})$ static power per cycle for the entire transaction length, as shown in Equation (1). If on the other hand, all the banks remain in the precharged state for tRP_{new} cycles, the memory consumes an average $P(Pre_{BG})$ static power per cycle, given by Equation (2) for the entire transaction length.

$$P(Act_{BG}) = \sum_{n=1}^{tRAS_{new}} I_{DD3N} \times V_{DD}/tRC_{new} \tag{1}$$

$$P(Pre_{BG}) = \sum_{n=1}^{tRP_{new}} I_{DD2N} \times V_{DD}/tRC_{new} \tag{2}$$

As shown in Equations (1) and (2), to estimate these power values for any given transaction length tRC_{new}, the power consumption due to these background power components is scaled over the transaction length. These actual timings can be derived from a command trace by calculating the duration for which any of the banks is in the active state, and for which all the banks are in the precharged standby state.

B. Activate and Precharge Command Power

I_{DD0} is specified as the average current consumed by the memory when it executes an *ACT* command (to transfer the data from the memory array to the row buffer) and a *PRE* command (to charge the bit lines and restore the row buffer contents back to the memory array), within the minimum timing constraints. The I_{DD0} current value also includes the active background current I_{DD3N} for the minimum period for which the row is active (*tRAS*) and the precharge current I_{DD2N} for the minimum period for which the row is precharged (*tRC - tRAS*). Hence, these should be subtracted from I_{DD0} for the appropriate durations and averaged over the transaction length tRC_{new} to identify the average power consumed only due to the *ACT* and *PRE* commands. The unmodified Micron power model specifies these two power components as one, assuming by default, a close-page policy. However, we split them as *P(ACT)* and *P(PRE)* and provide estimates by using the same total average current of I_{DD0} and apply it separately to the two components, based on the ratio of the number of active cycles to precharge cycles in the transaction, as shown in Equations (3) and (4), respectively. This partitioning enables us to provide power estimates when using the open-page policy.

$$P(ACT) = \sum_{n=1}^{tRAS} (I_{DD0} - I_{DD3N}) \times V_{DD}/tRC_{new} \tag{3}$$

$$P(PRE) = \sum_{n=tRAS+1}^{tRC} (I_{DD0} - I_{DD2N}) \times V_{DD}/tRC_{new} \tag{4}$$

C. Read and Write Command Power

A *Read* command consumes I_{DD4R} average current during the cycles of the data transfer, while a *Write* command consumes I_{DD4W}. Since these also include the active background current values consumed during the read or the write, I_{DD3N} must be subtracted from the I_{DD4R} and I_{DD4W} currents, to identify the power associated only with the *Read* and the *Write* commands, respectively. To calculate the power associated with the *Read* and *Write* commands, we first sum the current values over the number of cycles the data is on the data bus when reading from or writing to the SDRAM, identified here using *tR* and *tW*, respectively. These cycles of data transfer for a single burst of data can are be derived using the ratio of burst length (BL) to data rate (DR). For DDR memories this equates to BL/2. The power values are scaled over the transaction length tRC_{new} to get the average power consumed by a *Read* and a *Write*, given by Equations (5) and (6), respectively.

$$P(RD) = \sum_{n=1}^{tR} (I_{DD4R} - I_{DD3N}) \times V_{DD}/tRC_{new} \tag{5}$$

$$P(WR) = \sum_{n=1}^{tW} (I_{DD4W} - I_{DD3N}) \times V_{DD}/tRC_{new} \tag{6}$$

D. Refresh Power

A refresh operation is used to retain the data in the SDRAM by recharging the capacitors in the memory cells. A refresh can be executed only when all the banks of the memory are in the *precharged state*. A refresh thus consists of a single *Refresh* command along with a set of pre-refresh NOPs that gives enough time (at least *tRP* cycles) to precharge all the banks each before executing the refresh. If all the banks all already in the precharged idle state or the last command of the last transaction was issued with an auto-precharge, since the refresh would start only at the end of the auto-precharge of the last transaction, no explicit precharges will be required. Accordingly, *P(PRE)* (Equation (4)) is consumed (with a transaction length of *tRP*) for the number of precharges (*N(PRE)*) issued and I_{DD2N} current is consumed for the *tRP* cycles associated with those *Precharges*. Micron's model fails to consider the power consumed during pre-refresh clock cycles, as a part of refresh power (issue (4) raised in Section I). The refresh command by itself, consumes I_{DD5} current over the refresh cycles (*tRFC*). The refresh and pre-refresh power components add up over *tREF* (=*tRP*+*tRFC*) cycles to give the total refresh power, as shown in Equation (7).

$$P(REF) = \sum_{n=1}^{tRP} \Big(\big(I_{DD2N} \times V_{DD}\big) + \big(N(PRE) \times P(PRE)\big) \Big)/tREF + \sum_{n=1}^{tRFC} I_{DD5} \times V_{DD}/tREF \tag{7}$$

E. Auxiliary Power Components

Besides these basic power components, other auxiliary power components are associated with every read and write operation. When a write is issued, the external signal used to drive the data to the memory needs to be terminated on the memory module to avoid distortions of other signals on the memory, using a termination resistor. This termination power is consumed whenever a write is issued. Similarly, when a read is issued the power required to drive the data out through the device I/O, must also be accounted for and is referred to as the I/O power. These power components are not described in this paper, since they can be employed directly from Micron's power model [1]. In order to calculate the total power for termination during a write operation, the termination power per data bit, $P(W_{DQ})$, and the number of data bits written, $N(W_{DQ})$, must be multiplied. Similarly, to calculate the total power for data I/O during a read operation, the I/O power per data bit, $P(R_{DQ})$, and the number of data bits read, $N(R_{DQ})$, must be multiplied. In addition to these power equations, power is consumed when switching from a read-to-write or a write-to-read or when the memory is idle. Their power equations are not explicitly provided here, since they are relatively simple and can be derived from the $P(Act_{BG})$ and $P(Pre_{BG})$ power equations previously shown in Equations (1) and (2) respectively, using the duration of these switching or idle cycles as the transaction length.

F. Transaction and Trace Power Computation

To estimate power consumption of an entire trace or a transaction, we provide a generic power equation which applies to the whole of a trace. This equation can be employed for any window of analysis from a single transaction (including idle transactions) to the entire application trace and is valid for any degree of bank-parallelism and any memory access (open/close page) policy (issue (3) raised in Section I). This equation (shown in Equation (8)) is highly parameterized and together with the individual components described earlier fits a transaction of any length. All the power values are obtained from Equations (1) to (6) described before, while $P(R_{DQ})$ (I/O power) and $P(W_{DQ})$ (Termination power) are obtained from Micron's power model [1] for the total numbers of data bits read or written. These parameters can be obtained from any memory controller for every memory transaction.

$$\begin{aligned} P(\mathit{Trace}) = {} & P(\mathit{Act_{BG}}) \times N(\mathit{Act_{BG}}) + P(\mathit{Pre_{BG}}) \times N(\mathit{Pre_{BG}}) \\ & + \sum_{i=0}^{\mathit{nBanks\text{-}1}} \Big(P(\mathit{ACT}) \times N(\mathit{ACT})_{(i)} + P(\mathit{PRE}) \times N(\mathit{PRE})_{(i)} + P(\mathit{RD}) \\ & \times N(\mathit{RD})_{(i)} + P(\mathit{WR}) \times N(\mathit{WR})_{(i)} \Big) + P(\mathit{REF}) \times N(\mathit{REF}) \\ & + P(R_{DQ}) \times N(R_{DQ}) + P(W_{DQ}) \times N(W_{DQ}) \qquad (8) \end{aligned}$$

In Equation (8), $nBanks$ refers to the number of banks accessed in parallel in that transaction/trace and $N(\mathit{Act_{BG}})$ and $N(\mathit{Pre_{BG}})$ refer to the number of active and precharge cycles, respectively. $N(\mathit{ACT})_{(i)}$ and $N(\mathit{PRE})_{(i)}$, refer to the number of *ACT* and *PRE* commands, and $N(\mathit{RD})_{(i)}$ and $N(\mathit{WR})_{(i)}$ refer to the number of reads and writes per bank (i). $N(\mathit{REF})$ refers to the number of Refreshes in the trace. It should be noted that each *Read* and *Write* corresponds to a burst count *(BC)* of one and hence, transactions with burst counts greater than one are defined by as many read and write commands. Since this power equation is highly parameterized, it can be employed for any transaction from any given memory controller. For instance, if a memory controller employs two bank access with close-page policy, for a read transaction with a burst count (BC) of 4, i = 2, $N(\mathit{ACT})_{(i)}$ = 2, $N(\mathit{PRE})_{(i)}$ = 2, $N(\mathit{RD})$ = 8, $N(\mathit{WR})$ = 0, $N(\mathit{REF})$ = 0, $(N(R_{DQ}))$ = 1024 and $(N(W_{DQ}))$ = 0. If an open-page policy is employed, the $N(\mathit{ACT})_{(i)}$ and $N(\mathit{PRE})_{(i)}$ values are determined by the need for activating and precharging the particular banks. Equations (1) through (8) can thus be employed to resolve issues (2) to (4) raised in Section I. In Sections VI, VII and VIII, we provide equations to resolve the issue (1) regarding modeling of state transitions from different memory states to the power-down and the self-refresh modes.

VI. Stand-by to Power-Down Mode Transitions

Micron's power model does not provide power values for the transition period to power-down modes, resulting in optimistic estimates of power savings. This section corrects this optimism with power equations related to the transition periods from stand-by modes to different power-down modes. As specified before, certain timing constraints are to be respected when the memory controller decides to employ one of the power-down modes, as detailed in the following sections.

A. Active Power-down

When an active power-down is issued in the active stand-by mode, a time period of *tCPDED* is required to enter the power-down mode and block all the input signals. Including this time period, the DRAM must be in power-down mode for a time period of *tPD*, which may vary from a minimum of *tCKE* to a maximum of 9× *tREFI* for any DDR*x* SDRAM. When employing this mode, either a fast-exit or a slow-exit policy can be selected for DDR2. The fast-exit power-down mode has an exit transition period of *tXARD*, which is shorter than that of the slow-exit power-down mode given by *tXARDS*. The difference between the two modes is that the former saves less power than the latter, owing to the shorter transition period and thus, with a smaller performance penalty. For DDR3, only the slow-exit active power-down mode is supported with an exit timing constraint of *tXP*. The memory consumes active standby I_{DD3N} current during the transition periods when switching to the power-down mode. For DDR2, during *tPD* (actual power-down time), the memory consumes I_{DD3P0} current for the fast-exit mode and I_{DD3P1} current for the slow-exit mode. For DDR3, it is given by I_{DD3P}. As before, the power values are scaled over the total active power-down duration (including transition period) taken as the transaction length ($\mathit{tRC_{new}}$). Using these current values and actual timing parameters, we derive the power equations for fast-exit and slow-exit active power-down modes, in Equations (9) and (10):

$$P(\mathit{APD_F}) = \left(\sum_{n=1}^{\mathit{tPD}} I_{DD3P0} + \sum_{n=1}^{\mathit{tXARD}} I_{DD3N} \right) \times V_{DD} / \mathit{tRC_{new}} \qquad (9)$$

$$P(\mathit{APD_S}) = \left(\sum_{n=1}^{\mathit{tPD}} I_{DD3P1} + \sum_{n=1}^{\mathit{tXARDS}} I_{DD3N} \right) \times V_{DD} / \mathit{tRC_{new}} \qquad (10)$$

B. Precharge Power-down

When a power-down is issued in the precharge stand-by mode, a time period of *tCPDED* is required to enter the power-down mode. Including this time period, the DRAM must be in power-down mode for *tPD* cycles, as defined in the previous sub-section. When employing this mode, either a fast-exit or a slow-exit policy can be selected for DDR3. The fast-exit power-down mode has an exit transition period of *tXP* and the slow-exit power-down mode has an exit transition period of *tXPDLL*. For DDR2, only the slow-exit precharge power-down mode is supported with similar timing constraints as DDR3.

In the precharge power-down mode, the memory consumes precharge standby I_{DD2N} current during the transition periods when switching to the power-down mode. During *tPD* (power-down time), the memory consumes I_{DD2P1} and I_{DD2P0} currents in the fast-exit mode and slow-exit mode, respectively. For DDR2, this is given by I_{DD2P}. Using these current values, we derive the power equations for the fast-exit and slow-exit precharge power-down modes, as shown in Equations (11) and (12), respectively. The total precharge power-down duration (including transitions) is the transaction length (tRC_{new}).

$$P(PPD_F) = \left(\sum_{n=1}^{tPD} I_{DD2P1} + \sum_{n=1}^{tXP} I_{DD2N}\right) \times V_{DD} / tRC_{new} \quad (11)$$

$$P(PPD_S) = \left(\sum_{n=1}^{tPD} I_{DD2P0} + \sum_{n=1}^{tXPDLL} I_{DD2N}\right) \times V_{DD} / tRC_{new} \quad (12)$$

VII. Active to Power-Down Mode Transitions

This section presents power equations that address transitions from active memory states to the power-down states. These include transitions after issuing commands like *RD*, *WR*, *REF*, *ACT*, *PRE* and *MRS* (Mode Register Set), thus covering almost all transitions to the power-down modes. When such SDRAM commands are involved in state transitions to the power-down modes, for accurate trace energy analysis, the equations presented in this section must be substituted appropriately for the command power equations in Section V.

A. Read or Write to Power-down

If a power-down is scheduled after a Read or a Write (without an auto-precharge), the memory controller must wait at least *tRDPDEN* or *tWRPDEN* cycles, respectively, before issuing the (active) power-down. During these cycles, active stand-by current of I_{DD3N} is consumed. In addition, *P(RD)* and *P(WR)* is consumed during the *BL/2* cycles of data transfer for read and write, respectively. These are calculated over *tRDPDEN* and *tWRPDEN*, which must be taken as the transaction lengths of the read and write transactions used in Equations (5) and (6), respectively. Also, $P(R_{DQ})$ (I/O power) is consumed for each data bit read out and $P(W_{DQ})$ (Termination power) is consumed for every data bit written. Equations (13) and (14) give the power for transition from a Read and a Write (without an auto-precharge) to an active power-down mode. The power numbers for the active power-down modes can be obtained using Equations (9) and (10).

$$P(R_{PD}) = \Bigg(\sum_{n=1}^{tRDPDEN} I_{DD3N} \times V_{DD} + \sum_{n=1}^{tRDPDEN} P(RD) + (P(R_{DQ})) \times (N(R_{DQ})) \Bigg) / tRDPDEN \quad (13)$$

$$P(W_{PD}) = \Bigg(\sum_{n=1}^{tWRPDEN} I_{DD3N} \times V_{DD} + \sum_{n=1}^{tWRPDEN} P(WR) + (P(W_{DQ})) \times (N(W_{DQ})) \Bigg) / tWRPDEN \quad (14)$$

If a power-down is scheduled after Read is issued with an auto-precharge, the waiting time before entering a (precharge) power-down mode is also defined by *tRDPDEN* and hence, Equation (13) holds for this transition as well. However, if a Write is issued with an auto-precharge, the memory controller must wait *tWRAPDEN* cycles before issuing the (precharge) power-down. The active stand-by current of I_{DD3N} is consumed during the *tWRAPDEN-1* cycles before the auto-precharge is issued and I_{DD2N} is consumed for the precharge cycle. In addition, *P(WR)* is consumed during the *BL/2* cycles of data transfer (calculated using Equation (6) with transaction length *tWRAPDEN*), besides the $P(W_{DQ})$ (Termination power) for every data bit written to the memory. Also, *P(PRE)* (Equation (4)) is consumed for the auto-precharge with transaction length *tWRAPDEN*. Equation (15) gives the power for transition from a write (with an auto-precharge) to a power-down mode. The power estimates for the precharge power-down modes can be obtained using Equations (11) and (12).

$$P(WA_{PD}) = \Bigg(\Big(\sum_{n=1}^{tWRAPDEN-1} (I_{DD3N}) + I_{DD2N} \Big) \times V_{DD} + \sum_{n=1}^{tWRAPDEN} (P(WR) + P(PRE)) + (P(W_{DQ})) \times (N(W_{DQ})) \Bigg) / tWRAPDEN \quad (15)$$

B. Other Active Modes to Power-down

Here, we look at transitions to power-down modes after Refresh (REF), Precharge (PRE), MRS (Mode Register Set) and Activate (ACT) commands are issued. As in the previous cases, a particular timing constraint needs to be respected after each of these commands is issued, before the memory is asked to power-down by the memory controller. For instance, if a Refresh command has been issued, this timing constraint is given by *tREFPDEN*, for a Precharge command by *tPRPDEN*, for an MRS command by *tMRSPDEN* and for an Activate command by *tACTPDEN*. In the case of Refresh, Precharge and MRS commands, the precharge stand-by current (I_{DD2N}) and the corresponding command current are consumed during this transition period and the memory transitions to the precharge power-down mode. In the case of an Activate command, the active stand-by current (I_{DD3N}) and the activate command current *(P(ACT))* are consumed during this transition period and the memory transitions to the active power-down mode. The power values for the transitions from ACT, PRE and REF commands to power-down modes, can be obtained using Equations (16), (17), and (18), respectively.

$$P(ACT_{PD}) = \sum_{n=1}^{tACTPDEN} \Big(I_{DD3N} \times V_{DD} + P(ACT) \Big) / t_{ACTPDEN} \quad (16)$$

$$P(PRE_{PD}) = \sum_{n=1}^{tPRPDEN} \Big(I_{DD2N} \times V_{DD} + P(PRE) \Big) / t_{PRPDEN} \quad (17)$$

$$P(REF_{PD}) = \sum_{n=1}^{tREFPDEN} \Big(I_{DD2N} \times V_{DD} + P(REF) \Big) / t_{REFPDEN} \quad (18)$$

In the equations presented above, *P(ACT)*, *P(PRE)* and *P(REF)* are obtained from Equations (3), (4) and (7), respectively, with the transition periods used as their transaction lengths. For the transition from MRS command to the precharge power-down mode, the power value can be obtained by multiplying I_{DD2N} (precharge stand-by current) by V_{DD} (supply voltage) and the energy value by multiplying this power number by *tMRSPDEN* constraint. Again, the power values for the active and precharge power-down modes can be obtained using Equations (9), (10), (11) and (12), respectively.

VIII. Self-Refresh Mode Transition

Sections VI and VII covered all possible transitions to the different power-down states. In this section, we discuss the transitions to the Self-Refresh mode. The Self-Refresh command is used in DDR SDRAMs to retain data even when the clock is stopped. In this condition, the rest of the memory system is powered down, but the memory internally performs refreshes to maintain its contents without an external clock.

In order to switch to the Self-Refresh mode, it must be ensured that the SDRAM is idle and all its banks are in the precharge state with *tRP* satisfied. After issuing the Self-Refresh command, the Clock Enable Signal (CKE) must be kept 'Low' to maintain the memory in the Self-Refresh mode. The minimum time that the SDRAM must remain in Self-Refresh mode is given by *tCKESR*. This includes *tCPDED* to block all the input signals, *tCKSRE* (Self-Refresh entry time), *tCKSRX* (Self-Refresh exit time) and a minimum *tCKE* period within which the SDRAM memory must initiate at least one Refresh command. When exiting the Self-Refresh mode, it must be ensured that the clock is stable and then, the CKE can be changed to 'High'. A timing constraint of at least *tXSDLL* must be satisfied before any other valid command is issued to the memory. The total time required for the Self-Refresh to finish is indicated by *tSREF*.

During the *tXSDLL* cycles, I_{DD2N} (precharge stand-by) current is consumed and during the *tCKSRE* and *tCKSRX* cycles, slow-exit precharge power down current (I_{DD2P0} for DDR3 and I_{DD2P} for DDR2) is consumed. *P(PRE)* (Equation (4)) is consumed (with a transaction length of *tRP*) for the number of precharges (*N(PRE)*) issued before self-refresh and I_{DD2N} current is consumed for the *tRP* cycles associated with those *Precharges*. During the Self-Refresh cycles, the self-refresh current (I_{DD6}) is consumed. Equation (19) gives the power consumption for the entry and exit transition periods of the Self-Refresh mode and Equation (20) gives the total power consumption for the Self-Refresh mode.

$$P(SR) = \left(\sum_{n=1}^{\substack{tRP \\ +tXSDLL}} I_{DD2N} \times V_{DD} + \sum_{n=1}^{tRP} N(PRE) \times P(PRE) \right) / tSREF \quad (19)$$

$$P(SREF) = \left(\sum_{n=1}^{\substack{tCKSRE \\ +tCKSRX}} I_{DD2P0} + \sum_{n=1}^{\substack{tCKESR- \\ (tCKSRE+tCKSRX)}} I_{DD6} \right) \times V_{DD} / tSREF + P(SR) \quad (20)$$

IX. Results and Analysis

This section compares the unmodified Micron model against our proposed model in terms of power and energy consumption and power savings estimation.

A. Basic Experimental Setup

To evaluate our generic power model, we employed a cycle-accurate model of our SDRAM memory controller [20] that interleaves over a given number of banks determined at design time. Most importantly, the memory controller is flexible and supports different configurations and policies. For our analysis, we used power numbers for DDR3-800 from Micron datasheets [14]. We perform five experiments (presented in Sections IX-B through IX-F) to highlight the contributions of our proposed power model and present significant improvements in power and energy estimation compared to Micron's SDRAM power model. Modifications to the memory controller configuration and policies are presented for every experiment individually and the platform setup and configuration for the H.263 decoder experiment is discussed in Section IX-F.

B. Actual Transaction Lengths and their Impact on Energy

In our first experiment, we derive the actual transaction lengths (*tRC*$_{new}$) obtained when varying the number of banks interleaved and the burst count (BC) of the data read out in a *Read* transaction when employing a close-page policy with a burst length (BL) of 8 words using our memory controller [20].

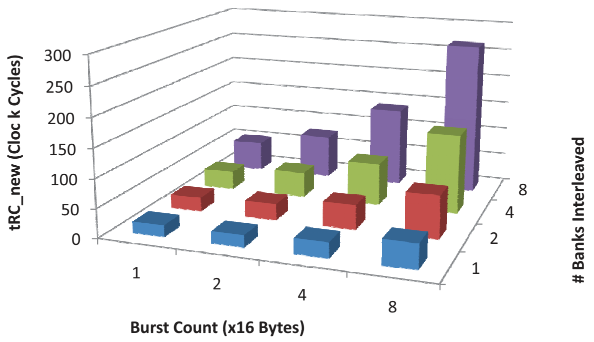


Fig. 5. Transaction Lengths (*tRC*$_{new}$) for different Burst Counts

As can be noticed in Figure 5, the actual transaction lengths (*tRC*$_{new}$) for the different accesses are very different from the minimal *tRC* of 20 clock cycles reported by the datasheets.

To analyze the impact of the actual transaction lengths on the energy estimates, we scale the energy consumption values obtained from Micron's power model for the basic power components to the actual timing parameters and compare them against those from our power model. As an example, we select 4 bank-interleaving read and write transactions with burst count of 1 and burst length of 8 words (2 bytes per word) to access 64 bytes of data using a close-page policy.

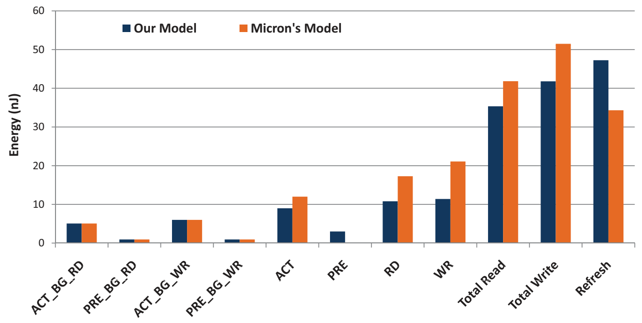


Fig. 6. Read, Write and Refresh Transaction Energy Comparison

In the graph depicted in Figure 6, the read transaction has a tRC_{new} of 32 cycles and $tRAS_{new}$ of 27 cycles, whereas the write transaction has a tRC_{new} of 37 cycles and $tRAS_{new}$ of 32 cycles. The refresh request has a transaction length tRC_{new} = 49 cycles. The minimum values from the DDR3-800 datasheet are tRC = 20 cycles, $tRAS$ = 15 cycles and $tRFC$ of 44 cycles.

As can be seen in Figure 6, Micron's power model over-estimates the Read (RD) and Write (WR) power components, since it employs the minimum timing constraints and combines the energy consumption for the *ACT* and *PRE* commands. The ACT(BG) and PRE(BG) estimates (background power components) are found to be similar in both the models. The I/O and Termination power components are employed directly from Micron's model and added appropriately to the total read and write transaction energy estimates. Note that Micron reports higher energy consumption numbers for *RD* and *WR* commands by around 37% and 45%, respectively, and in terms of the total *Read* and *Write* transaction power by around 15% and 18%, respectively. We also observe that Micron's numbers for Refresh are lower by around 27%, since it does not consider the power consumed during the required pre-refresh precharging of all the banks.

C. Energy Comparison for Open and Close Page Policies

In our next experiment, we configure the memory controller to support open-page and close-page policies while still inter-leaving over 4 banks with burst count of 1. We observe the energy consumption values for the transactions corresponding to different memory access policies discussed in Section IV.

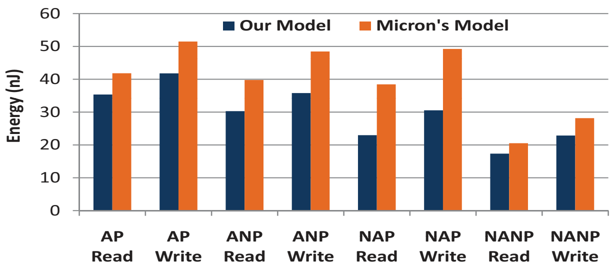


Fig. 7. Read/Write Energy Comparison for different Memory access policies

Figure 7 shows that the difference in the values reported by Micron's model against ours ranges from 18% in the case of AP Read to 67% in the case of NAP Read. In the latter case, the difference in higher since Micron's model assumes a close-page policy by default, but the NAP Read does not have an Activate command in the transaction (Figure 3(c)). The splitting up of the *ACT-PRE* command power in the Equations (3) and (4), to support the open-page policy in our power model, enables us to visualize this difference.

D. Impact of Transitions from Idle to Power-Saving states

In the next experiment, we compare the percentage energy-savings reported by Micron's model against those obtained from our model, for the self-refresh, the active and the slow-exit precharge power-down modes. Figure 8 depicts this difference across different granularities (power-down/self-refresh duration) of the power-down and self-refresh modes. The fast-exit precharge power-down mode has similar savings as the active power-down mode and is hence not depicted in graph.

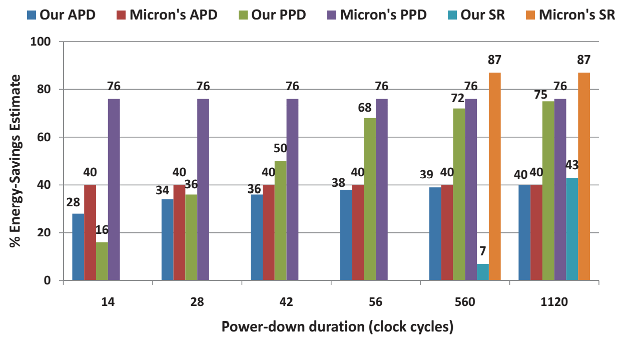


Fig. 8. Energy Comparison for Stand-by to Power-Down and Self-Refresh

The graph in Figure 8 shows that Micron's model over-estimates the energy-savings in the slow-exit precharge power-down mode (PPD) by up to 60% compared to our estimates, when the power-down duration is at its minimum ($tPPD_{min}$) of 14 clock cycles (in the case of DDR3-800). The corresponding difference in the savings reported for the active power-down mode (APD) is 12%. As the power-down duration increases (in this case, by a multiple of $tPPD_{min}$), the impact of power consumption during the transition period on the total energy savings, reduces. For the self-refresh (SR) mode (minimum transaction length of 529 cycles for DDR3-800), the difference in power savings swings from 80% for a transaction length of 560 cycles (40 times $tPPD_{min}$) to 44% for a transaction length of 1120 cycles (80 times $tPPD_{min}$). At the granularity of 560 cycles, the self-refresh period is very close to the minimum self-refresh transaction length of 529 cycles, which gives minimum power savings. As the granularity of the power-down or self-refresh increases, the savings increases and the difference between our model and Micron's reduces.

E. Impact of Transitions from Active to Power-Down states

In our next experiment, we compare the energy consumption estimates of both the models when transitioning from active states to power-down states. These include transitions after issuing Read (RD), Write (WR) and Refresh (REF) commands to active power-down (APD) or slow-exit precharge power-down (PPD) states. In Table II, we present the percentage difference in the energy estimates for these transitions, between the unmodified Micron model and our power model.

TABLE II
PERCENTAGE DIFFERENCE IN ENERGY (MICRON VS. OUR MODEL)
ACTIVE MODES TO POWER-DOWN TRANSITIONS

Transition Mode	*Operation Energy %*	*Background Energy %*	*Transition Energy %*
RD to APD	37.3	-93.7	-16.25
RD to PPD	37.3	-93.7	-16.25
WR to APD	42.3	-95	-12.04
WR to PPD	42.3	-95.2	-13.75
REF to PPD	-27.64	0	-27.64

As can be observed from Table II, Micron's power model, in general, underestimates the transition period energy consumption for transitions from active to power-down states. For instance, when transitioning from RD to APD, the transition period is 16 cycles, before entering power-down mode. The unmodified Micron model does not include the transition period and under-estimates background energy by 93.7% and over-estimates energy for the RD operation by 37.3%. These erroneous estimates however, cancel each other out to some extent, resulting in an overall energy under-estimation error by 16.25%. This does not take away the fact that these differences are significant and can lead to incorrect run-time decisions. For instance, the memory controller may decide to power-down the memory for shorter durations than may be actually required.

F. Energy Analysis of H.263 Decoder Application

In our final experiment, we execute an H.263 video decoder application to decode one video frame and compare the energy results obtained when using the unmodified Micron power model against our proposed power model.

System Setup: The H.263 video decoder application [21] is executed on the Simplescalar [22] tool, with a 2KB L1-Data cache, 2KB L1-Instruction cache, a 32KB shared L2 cache and 64 bytes cache lines configuration. We filtered out the L2 cache misses (meant for the SDRAM memory) and forwarded them to a trace-based traffic generator (a processing element) in our SystemC simulation model of a predictable MPSoC [23]. The processing elements in our system setup communicate with the SDRAM memory controller [20] (described before) through the AEthereal network-on-chip [24]. At the memory controller, the transactions are executed and the commands to the memory with all the relevant signals are logged to get the SDRAM command trace, which we employ to obtain the actual timings between commands. Since the burst length (BL) employed by DDR3-800 is 8 words each 16 bits (2 bytes) long, we employ a burst count (BC) of 1 and interleave transactions over 4 banks, to obtain 64-byte accesses to the SDRAM memory.

Energy Comparison: The overall composition of the H.263 decoder application, in terms of SDRAM transactions included 166788 reads and 7296 writes. When using the close-page policy, the H.263 decoder application took 1578414 cycles for completion. On the other hand, when using the open-page policy, it took 1017166 cycles, indicating that the open-page policy was able to exploit the spatial locality of data in the memory. This also had a significant impact on energy numbers.

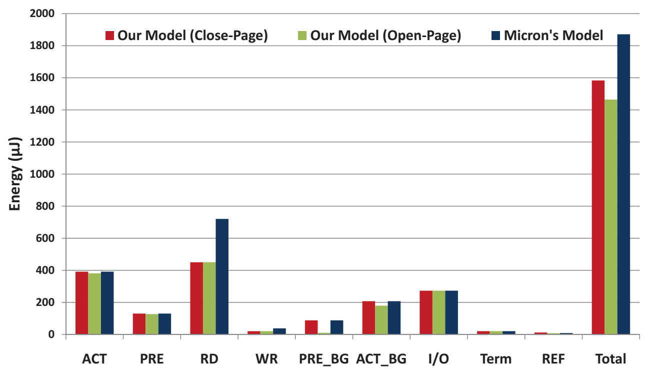


Fig. 9. H.263 Energy Comparison

Figure 9 shows the difference in the energy numbers for reads, writes and refreshes, besides the total application's SDRAM energy consumption. As can be noticed, Micron (assuming a close-page policy) over-estimates total energy consumption by 18% compared to our close-page policy implementation and by 28% compared to our open-page policy.

X. Conclusion

This paper presents an improved SDRAM power model that takes into account all the state transitions to power-down and self-refresh states and the actual timing parameters, from any given memory controller. Our generic power model also supports both open and close page policies and any degree of bank-interleaving. We evaluated our power model and showed differences of up to 67% in a read transaction energy when employing the open-page policy and up to 60% in energy savings for the precharge power-down mode and up to 80% for the self-refresh mode between Micron's model and our model. These significant differences highlight the importance of modeling transition period power and employing actual timings in estimating energy consumption of DDR SDRAMs.

Acknowledgments

This work is carried out under the COMCAS project (CA501), a project labelled within the framework of CATRENE, the EUREKA cluster for Application and Technology Research in Europe on NanoElectronics. The authors would also like to thank Tim Kouters and Sven Goossens at TU Eindhoven for their valuable suggestions.

References

[1] Micron Technology Inc., *TN-41-01: Calculating Memory System Power for DDR3*, Technical Report, 2007.
[2] D. Schmidt et al., *DRAM Power Management and Energy Consumption: A Critical Assessment*, Proc. of SBCCI, 2009.
[3] D. Schmidt et al., *A Review of Common Belief on Power Management and Power Consumption*, White Paper, 2009.
[4] O. Vargas, Infineon Technologies AG, *Achieve minimum power consumption in mobile memory subsystemsin*, Technical Report, 2006.
[5] F. Rawson, IBM Austin, *MEMPOWER: A Simple Memory Power Analysis Tool Set*, Technical Report, 2004.
[6] A. Joshi et al., University of Texas Austin, *Power Modeling of SDRAMs*, Technical Report TR-040126-02, 2004.
[7] Q. Deng et al., *MemScale: Active Low-Power Modes for Main Memory*, Proc. of ASPLOS, 2011.
[8] Y. Joo et al., *Energy Exploration and Reduction of SDRAM Memory Systems*, Proc. of DAC, 2002.
[9] J. Ji et al., *System-Level Early Power Estimation for Memory Subsystem in Embedded Systems*, Proc. of SEC, 2008.
[10] T. Vogelsang, Rambus Inc., *Understanding the Energy Consumption of Dynamic Random Access Memories*, Proc. of MICRO, 2010.
[11] S. Thoziyoor et al., Hewlett-Packard Advanced Architecture Laboratory, *CACTI 5.0*, Technical Report HPL-2007-167, 2007.
[12] H. Zheng et al., *Power and Performance Trade-Offs in Contemporary DRAM System Designs for Multicore Processors*, IEEE Tran. on Comp., vol. 59, no. 8, 2010.
[13] Micron Technology Inc., *DDR2 SDRAM 1Gb Data Sheet*, 2004.
[14] Micron Technology Inc., *DDR3 SDRAM 1Gb Data Sheet*, 2006.
[15] JEDEC SST Association, *DDR2 SDRAM Standard*, JESD79-2F, 2009.
[16] JEDEC SST Association, *DDR3 SDRAM Standard*, JESD79-3E, 2010.
[17] D. Wang et al., *DRAMsim: A memory system simulator*, ACM SIGARCH Comp. Arch. News, vol. 33, no. 4, 2005.
[18] B. Jacob et al., *Memory Systems: Cache, DRAM, Disk*, Morgan Kaufmann Publishers, 2007.
[19] S. Rixner et al., *Memory access scheduling*, ACM SIGARCH Comp. Arch. News, vol. 28, no. 2, 2000.
[20] B. Akesson et al., *Architectures and Modeling of Predictable Memory Controllers for Improved System Integration*, Proc. of DATE, 2011.
[21] G. Cote et al., *H.263+: Video Coding at Low Bit Rates*, IEEE Tran. on Circuits and Systems, vol. 8, no. 7, 1998.
[22] D. Burger et al., *The SimpleScalar tool set, version 2.0*, ACM SIGARCH Comp. Arch. News, vol. 25, no. 3, 1997.
[23] A. Hansson et al., *CoMPSoC: A template for composable and predictable multi-processor system on chips*, ACM Trans. Des. Autom. Electron. Syst., vol. 14, no. 1, 2009.
[24] K. Goossens, *AEthereal network on chip: concepts, architectures, and implementations*, IEEE Design & Test of Computers, vol. 22, no. 5, 2005.