Modeling Power Consumption of NAND Flash Memories Using *FlashPower*

Vidyabhushan Mohan, Trevor Bunker, Laura Grupp, Sudhanva Gurumurthi, *Senior Member, IEEE,* Mircea R. Stan, *Senior Member, IEEE,* and Steven Swanson

***Abstract*—Flash is the most popular solid-state memory technology used today. A range of consumer electronics products, such as cell-phones and music players, use flash memory for storage and flash memory is increasingly displacing hard disk drives as the primary storage device in laptops, desktops, and servers. There is a rich microarchitectural design space for flash memory, and there are several architectural options for incorporating flash into the memory hierarchy. Exploring this design space requires detailed insights into the power characteristics of flash memory. In this paper, we present *FlashPower*, a detailed power model for the two most popular variants of NAND flash, namely, the single-level cell (SLC) and 2-bit Multi-Level Cell (MLC) based flash memory chips. *FlashPower* is built on top of CACTI, a widely used tool in the architecture community for studying various memory organizations. *FlashPower* takes several parameters like the device technology, microarchitectural layout, bias voltages and workload parameters as input to estimate the power consumption of a flash chip during its various operating modes. We validate *FlashPower* against chip power measurements from several different manufacturers and show that our results are comparable to the actual chip measurements. We illustrate the versatility of the tool in a design space exploration of power optimal flash memory array configurations.**

***Index Terms*—Analytical models, NAND Flash memory, power.**

I. Introduction

FLASH IS THE most popular solid-state memory technology used today. Consumer electronics products, such as cell-phones and portable music players, widely use flash memory, and flash-based solid-state disks (SSDs) are increasingly displacing hard disk drives as the storage of choice in laptops, desktops, and even servers. Computer architects have been exploring a variety of topics related to flash including the design of SSDs [1], [2], disk-caches [3], new flash-based server architectures [4], even hybrid memory systems [5], and Flash-translation layer (FTL) design [6], [7]. Such studies show that flash-based disks offer several advantages, including lower power and higher performance, as compared to traditional hard disk drives (HDDs). While most studies evaluate the performance impact due to the use of flash-based media, very few studies examine in detail the power benefits of using flash.

Accurately quantifying the energy consumption of flash is necessary for many reasons. Power is an important consideration because the design of flash memories is closely tied to the power budget within which they are allowed to operate. For example, flash used in consumer electronic devices has a significantly lower power budget compared to that of an SSD used in data centers, while the power budget for flash used in disk based caches is between the two. In such scenarios, an accurate knowledge of flash power consumption is beneficial. Insights into flash power consumption are also necessary because hybrid memories use flash in conjunction with other new nonvolatile memories like phase change RAM and spin-transfer torque RAM [8], [5]. Power-aware design of such hybrid memory systems is possible only when an accurate estimate of flash energy consumption is available. So far, most studies that quantify the energy consumption of flash were limited to using datasheets. However, research has shown that such datasheet-derived estimates are too general and do not take the intricacies of flash memory operations into account. Datasheets only provide average energy estimates (which in certain cases can deviate from the actual behavior by nearly 3X) and cannot be used when accurate estimates are required. These inaccuracies chiefly arise because they cannot account for properties like workload behavior that have an significant impact on flash energy consumption. In particular, we lack simulation tools that can accurately estimate the power consumption of various flash memory configurations.

This paper addresses this void by presenting *FlashPower*, a detailed power model for the two most popular variants of NAND flash namely, the single-level cell (SLC) and 2-bit multilevel cell (MLC) based NAND flash memory chips. *FlashPower* models the key components of a flash chip during the read, program, and erase operations, and when idle, and is highly parameterized to facilitate the exploration of a wide spectrum of flash memory organizations. *FlashPower* is built on top of CACTI [9], a widely used tool in the architecture community for studying memory organizations, and is suitable for use in conjunction with an architecture simulator. We

Manuscript received June 7, 2012; revised September 30, 2012; accepted January 22, 2013. Date of current version June 14, 2013. This work was supported in part by NSF CAREER Award (CCF 0643925), the NSF Variability Expedition Award (CCF 1029783) and funding by Google. This paper was recommended by Associate Editor H. S. Lee.

V. Mohan and S. Gurumurthi is with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903 USA (e-mail: vm9u@virginia.edu; gurumurthi@virginia.edu).

T. Bunker, L. Grupp, and S. Swanson are with the Department of Computer Science, University of California, San Diego, CA 92093 USA (e-mail: tbunker@eng.ucsd.edu; lgrupp@eng.ucsd.edu; swanson@eng.ucsd.edu).

M. R. Stan is with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22903 USA (e-mail: mircea@virginia.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2249557

0278–0070/$31.00 © 2013 IEEE

validate *FlashPower* against power measurements from chips from various manufactures and we show that *FlashPower* estimates are comparable to the measured values.

II. The Microarchitecture of *NAND* Flash Memory

In this section, we describe the components in a NAND flash memory array, how they are architecturally laid out and their operation.

A. Components of NAND Flash Memory

Flash is a type of EEPROM (electrically erasable programmable read-only memory) that supports read, program, and erase as its basic operations. A NAND flash memory chip includes command status registers, a control unit, a set of decoders, some analog circuitry for generating high voltages, buffers to store and transmit data, and the memory array. An external memory controller sends read, program or erase commands to the chip along with the relevant physical address. The main component of a NAND flash memory chip is the flash memory array. A flash memory array is organized into banks (referred to as *planes*). Fig. 1 describes a flash plane. Each plane has a page buffer (composed of sense-amplifiers and latches) which senses and stores the data to be read from or programmed into a plane. Each plane is physically organized as *blocks* and the blocks are composed of *pages*. A controller addresses a flash chip in blocks and pages. Thus a plane is a two dimensional grid composed of rows (bitlines) and columns (wordlines). At the intersection of each row and column there is a floating gate transistor (FGT) which stores a logical bit of data. In this paper, the terms “cell" and “FGT" refer to the same physical entity and are used interchangeably. In SLC flash, the FGT stores a single logical bit of information, while in MLC flash, the FGT stores more than one bit of information. Even though the term MLC flash means the number of bits stored in a FGT can be more than 1, we use the term MLC flash mostly in the context of a 2-bit MLC flash.

A group of bits in one row of a plane constitute a *page*. Each NAND flash block consists of a *string* of FGTs connected in series with access transistors to the string select line (SSL), the source line (SL), and the ground select line (GSL), as shown in Fig. 1. The string length is equal to the total number of FGTs connected in series. The number of pages in a block and the size of each page are integer multiples of the string length, and the number of bitlines running through the block, respectively. The multiplication factor depends on the architectural layout of the block. Section II-C explains how the blocks are laid out architecturally. A pass transistor is connected to each wordline to select/unselect the page.

B. Basic Flash Operations

NAND flash uses Fowler-Nordheim (FN) tunneling to move charges to/from the floating gate. A program operation involves tunneling charges to the floating gate while an erase operation involves tunneling charges off the floating gate. Tunneling of charges to and from the FGT varies its threshold voltage, and bits are encoded as varying threshold voltage levels. A read operation involves sensing the threshold voltage of the FGT. An erase operation is performed at the granularity of a block while the read and program operations are performed at a page granularity. Since each FGT in a MLC flash has more than two threshold voltage levels, a read operation for MLC flash involves multiple stages to sense the correct threshold voltage level. An SLC flash has only one stage in a read operation. Most NAND flash memories employ an iterative programming method like incremental step pulse programming (ISPP) to program an FGT. In iterative programming, a FGT attains its target threshold voltage through a series of small incremental steps. By controlling the duration and the magnitude of program voltage in each step, iterative programming ensures that precise threshold voltage levels are reached. After each iteration, a verify operation is performed to ensure the required threshold voltage level is reached. If not, the operation is repeated until an upper bound on the number of steps is reached. This upper bound is fixed during the design of the chip. If the upper bound is reached and the target threshold voltage level is still not reached, then the program operation fails. As the iteration count increases, the energy required for a program operation also keeps increasing. Section III-E5 provides more details about how *FlashPower* estimates program energy consumption using iterative programming. Brewer *et al*. provide more details on circuit and micro-architecture of flash memory [10], [11], [12].

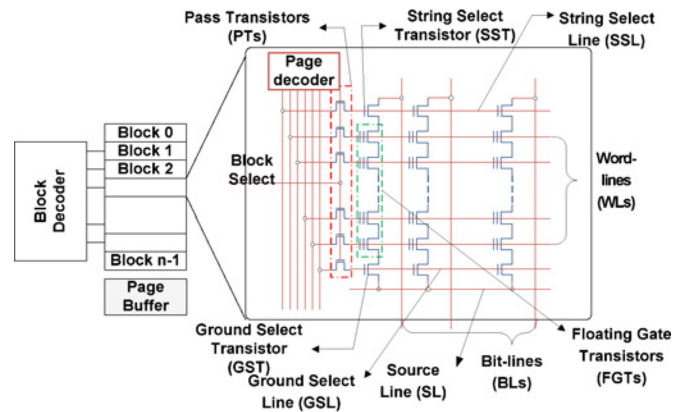

Fig. 1. Circuit layout of a NAND Flash memory plane. Adapted from [10].

C. Architectural Layout of Flash Memory

While Fig.1 shows the circuit level layout of a flash memory plane, this section explains the page layout in a flash memory block. We illustrate the architectural layout using a SLC and a 2-bit MLC flash as examples, but the layout is similar for all N-bit flash. We note that the architectural layout of SLC and MLC flash can vary even though the underlying circuit layout and implementation are identical for the two. Assuming that there are a total of N_{pages} in a block and the size of each page is in $N_{pagesize}$, Tables I and II depict the architectural layout of a flash memory block. In case of SLC flash, each cell stores 1-bit of information and is mapped to a logical page. The time taken to read and program this bit is nearly the same for all pages. In case of 2-bit MLC flash some pages are mapped to

TABLE I
ARCHITECTURAL LAYOUT OF A SLC BLOCK. ADAPTED FROM [13]

Row	$N_{pagesize}$ cells	$N_{pagesize}$ cells
1	Page 0	Page 1
2	Page 2	Page 3
..	..	..
..	..	..
$N_{page}/2$	$N_{page}-2$	$N_{page}-1$

TABLE II
ARCHITECTURAL LAYOUT OF A 2-BIT MLC BLOCK. ADAPTED FROM [13]

Row	MSB of first $N_{pagesize}$ cells	LSB of first $N_{pagesize}$ cells	MSB of last $N_{pagesize}$ cells	LSB of last $N_{pagesize}$ cells
1	Page 0	Page 4	Page 1	Page 5
2	Page 2	Page 8	Page 3	Page 9
..	..	..	..	..
..	..	..	..	..
$N_{page}/4$	$N_{page}-6$	$N_{page}-2$	$N_{page}-5$	$N_{page}-1$

the MSB while some other pages are mapped to the LSB. The pages corresponding to LSB are read and programmed faster compared to pages mapped to the MSB. We will refer to pages mapped to the LSB (pages 0, 1, 2, 3, etc. in Table II) as *fast pages* and pages mapped to the MSB (pages 4, 5, 8, 9, etc. in Table II) as *slow pages*.

III. *FlashPower*: A DETAILED ANALYTICAL POWER MODEL

This section presents the details of the analytical power model that we have developed for NAND flash memory chips. *FlashPower* uses analytical models to estimate NAND flash memory chip energy dissipation during basic flash operations like read, program and erase, and when the chip is idle. Before delving into the details of the model, we list the components that are modeled and the power state machine. We then explain how *FlashPower* was integrated with CACTI [9], followed by the details of the model itself.

A. Circuit Components

With respect to Fig. 1, the components that dissipate energy are as follows:

1) the bitline (BL) and wordline (WL) wires.
2) the SSL, GSL and SL.
3) the drain, source, and the gate of the SST, GST, and PTs.
4) the drain, source, and control gate of the FGTs.
5) the floating gate of the FGTs—energy dissipated during program and erase operation.
6) the sense amplifiers (SAs)—energy dissipated during read, program, verify, and erase verify operations.

In addition to the above components, *FlashPower* models the energy dissipated by the block and page decoders, and the charge pumps (that provide voltages for read, program, and erase operations). The energy per read, program and erase operation are the sum of the energy dissipated by all the

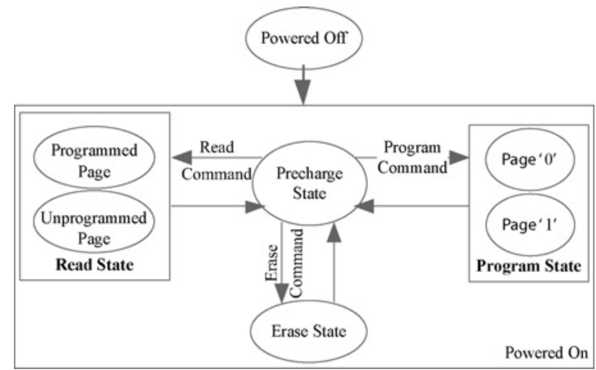

Fig. 2. Power state machine for an SLC NAND Flash chip.

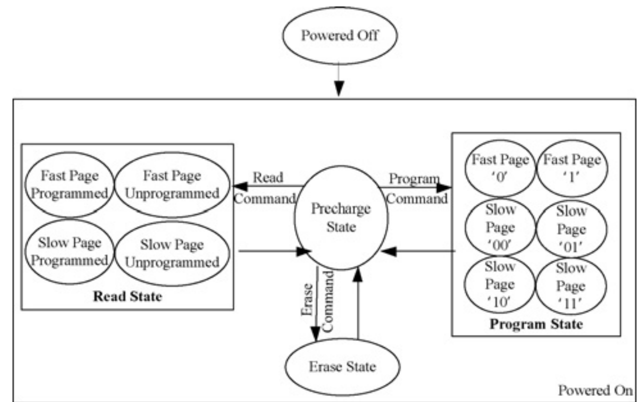

Fig. 3. Power state machine for a 2-bit MLC NAND Flash chip.

aforementioned components. Since the energy dissipation of I/O pins varies significantly with the design of the circuit board using the flash chip, we do not model their energy.

B. Power State Machine

Figs. 2 and 3 describe the power state machine for an SLC and an MLC NAND flash chip, respectively. The circles represent the individual states, while the solid lines denote state transitions. We model the energy dissipated when the chip is powered. When the chip is on, but is not performing any operation, it is in the precharge state. In this state, the bitlines are precharged while the wordlines, and the select lines (SSL, GSL, and SL) are grounded. The select lines isolate the array electrically but the chip is ready to respond to commands from the controller.

Upon receiving a read, program, or erase command from the controller, the state machine switches to the corresponding state. When the command is complete, the state machine switches back to the precharge state. We model the energy dissipation of the actual operation and each state transition. For an SLC read operation, depending on whether a page is programmed or not, the chip is in either the programmed page or in the unprogrammed page state. This can be sensed in only one read cycle. For an SLC program operation, depending on whether the bit to be programmed is a 0 or a 1, the state of the chip is either in page 0 or page 1 state. For an MLC read, the chip transitions to one of the four states, depending on whether it is programmed or not and whether it is a slow or a fast page.

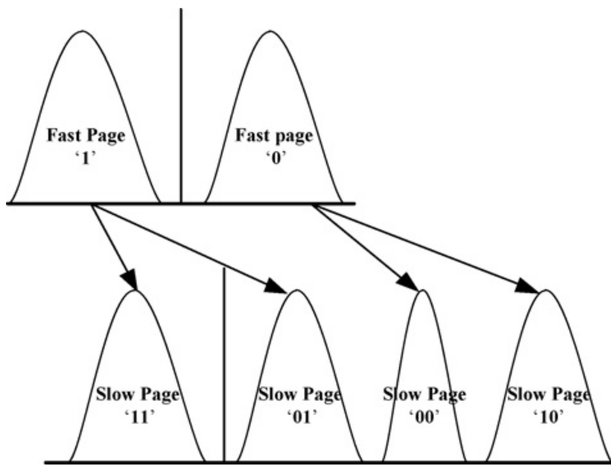

Fig. 4. Transition between MLC program states. The transitions are based on the multi-page programming algorithm proposed in [13].

FlashPower requires at least 2 read cycles to sense one of the four states. For an MLC program operation, the state transition of the chip is a function of the bit pattern to be programmed and whether the page is a slow page or a fast page. *FlashPower* models a conventional multipage architecture programming method as described in [13]. According to this method, each bit cell stores two pages—a fast page and a slow page. Fig. 3 lists all the states for an MLC program while Fig. 4 explains how the chip transitions between these states. When a fast page is programmed, the chip transitions to a fast page 1 state or a fast page 0 state depending on the bit pattern. When a slow page is programmed, the chip transitions to one of the four slow page states depending on the bit pattern to be programmed and the current state of the fast page. The main difference between a transition to a slow page 11 state or to a slow page 01 state from a fast page 1 state is the number of steps in the iterative programming method and the threshold voltage difference between the two states. A transition to a slow page state can occur only if the chip is already in one of the two fast page states.

While *FlashPower* models a canonical MLC program operation, it can also model other programming algorithms as long as they employ iterative programming techniques similar to the multipage architecture model. Modifications to the MLC power state machine can support such algorithms. For example, full sequence programming is one programming algorithm in which both the slow and fast pages are programmed together. To support such an algorithm, the state machine must perform transition directly from the idle state to one of the four slow page states. Since the main difference between transitions is the number of steps in the iterative programming method and the threshold voltage difference between the states, the programming model does not change but the values for individual parameters that the model takes will change.

For an erase operation, we consider only a single state. Note that while we consider the bit pattern for program operation, we do not consider the bit pattern for read or erase operations. This is because the read operation only involves sensing the threshold voltage of the cell, which is deterministic if all cells in the page are already programmed and is non-deterministic otherwise. For an erase operation, all cells in the block are erased irrespective of their threshold voltage state. In case of the program operation, which involves setting the threshold voltage of the cells to one of the 2^n states, the input bit pattern makes a significant difference in the power consumption. As shown in Section V, the energy dissipated to reach a fast page 0 state is significantly different from the energy dissipated to reach a slow page 00 state, which in turn is different from the energy dissipated to reach a slow page 01 state. The important parameters that affect the energy dissipation for the program operation are the threshold voltage difference and the number of verify cycles required for transition to the final threshold voltage state from the start state, both of which are governed by the input bit pattern.

C. Extending FlashPower for n-bit MLC Flash

While the state machines in Figs. 2 and 3 are for SLC and 2-bit MLC flash, *FlashPower* can be extended for other n-bit MLC flash variants like three level cell (TLC) flash. The state machine can be generalized for a n-bit MLC flash as follows: *FlashPower* has a total of 2^n states for the read operation and we can sense one of these states within n cycles. Assuming that we employ iterative programming and multi-page architecture similar to 2-bit MLC, the model requires a total of $2^{n+1} - 2$ states for the program operation. Since an erase operation is for an entire block irrespective of the threshold voltage of the individual cells in the block, *FlashPower* has only one state for the erase operation. For TLC flash, this corresponds to 8, 14, and 1 states for read, program, and erase operations, respectively. The number of steps in programming and erase method can vary from one implementation to another and so would the difference in threshold voltage of states between transitions. Since the program and erase model are based on these parameters, they can be extended to support n-bit MLC flash.

D. Integration with CACTI

FlashPower is designed as a plug-in to CACTI [9], a widely used memory modeling tool. We estimate the power consumption of array peripherals such as decoders and sense amplifiers assuming that they are high performance devices and for the bitlines and wordlines assuming aggressive interconnect projections. We also model an FGT as a CMOS transistor and then use the CMOS transistor models of CACTI to calculate the parasitic capacitance of an FGT. However, unlike CACTI, which models individual components of SRAM and DRAM based memory systems, we have chosen to model the basic operations on the flash memory. This is because, in the memories that CACTI models, individual operations operate at the same supply voltage to drive the bitlines and the wordlines. For example, in an SRAM-based memory, both read and write operations use the same voltage for wordlines and bitlines. The granularity (total bytes) of a read/write is also the same. However for NAND flash, read, program, and erase operate at different bias conditions and the granularity of an erase differs from read/program. Since the circuitry behaves

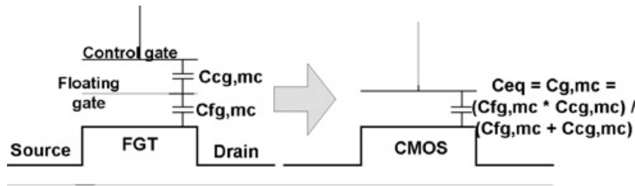

Fig. 5. Modeling a floating gate transistor.

differently for these different operations, we believe that it is more appropriate to model energy for the basic operations on the flash memory. Table III summarizes the inputs to *FlashPower*.

E. Power Modeling Methodology

1) *Modeling the Parasitics of an FGT:* To calculate the energy dissipation of an FGT, it is necessary to estimate the FGT's parasitic capacitance—i.e. an FGT's source, drain and gate capacitance. While the source and the drain of an FGT and a CMOS transistor are similar, an FGT has a two gate structure (floating and control) while a CMOS transistor has a single gate. We model a dual gate FGT structure as an equivalent single gate CMOS structure by calculating the equivalent capacitance of the two capacitors (one across the inter-poly dielectric and other across the tunnel oxide). We then use CACTI to calculate the parasitics of this transistor. Fig. 5 illustrates this. We calculate the control gate capacitance $C_{cg,mc}$ using the information on gate coupling ratio (GCR) available in [14], while we calculate the floating gate capacitance $C_{fg,mc}$ using the overlap, fringe, and area capacitance of a CMOS transistor of the same feature size. The source and drain capacitance of the transistor depend on whether the transistor is folded or not. *FlashPower* assumes that the transistor is not folded as the feature size is in the order of tens of nanometers. We use CACTI to model the drain capacitance of the FGT and the parasitic capacitance of other CMOS transistors like the GST, PT and SST.

2) *Derived Parameters:* The length of each bitline $L_{bitline}$ and each wordline $L_{wordline}$ are derived as

$$L_{wordline} = N_{bitlines-perblock} * N_{bcols} * pitch_{bit-line} \quad (1)$$
$$L_{bitline} = (N_{pages} + 3) * N_{brows} * pitch_{wordline} \quad (2)$$

where $N_{bitlines-perblock} = (N_{pagesize} + N_{sparebytes}) * 8$. "3" is added to N_{pages} because for each block, the bitline crosses three select lines (GSL, SSL and SL). The terms $pitch_{bit-line}$ and $pitch_{wordline}$ refer to the bitline and wordline pitch respectively, and are equal to $2 * featuresize$.

The total capacitance to be driven along a wordline is given by:

$$C_{wl} = C_{d,pt} + C_{g,mc} * N_{bitlines-perblock} + C_{wl,wire} * L_{wordline} \quad (3)$$

where, $C_{d,pt}$ is the drain capacitance of the pass transistor, $C_{g,mc}$ the equivalent gate capacitance of FGT and $C_{wl,wire}$ is the wordline wire capacitance. Similarly, the total capacitance to be driven along the bitline is equal to:

$$C_{bl} = 2 * C_{d,st} + C_{d,mc} * N_{pages} + C_{bl,wire} * L_{bitline} \quad (4)$$

where $C_{d,st}$ is the drain capacitance of the select transistors (SST), $C_{d,mc}$ is the drain and source capacitance of the FGT and $C_{bl,wire}$ is the bitline wire capacitance. series with the GST and SST at either ends, it is assumed that the drain of one FGT is shared with the source of the neighboring FGT. Adjacent FGTs connected in series share the source and the drain. Because of this assumption, the source capacitance of one FGT is considered equal to the drain capacitance of the neighboring FGT. The source of SST is shared with the drain of the first FGT in the string while the source of the last FGT in the string is shared with the drain of GST. The total capacitance of the SSL is

$$C_{ssl} = C_{gsl} = C_{d,pt} + C_{g,st} * N_{bitlines-perblock} + C_{wl,wire} * L_{wordline}. \quad (5)$$

Since the GSL needs to drive the same components as the SSL, we have $C_{ssl} = C_{gsl}$. The capacitive component of the SL is equal to:

$$C_{srcline} = C_{wl,wire} * L_{wordline} + C_{d,st}. \quad (6)$$

We assume that the source capacitance of GST is equal to its drain capacitance ($C_{d,st}$).

3) *Transition from the Powered Off to the Precharged State:* When this transition occurs, the bitlines are precharged and wordlines to $V_{bl,pre}$ and $V_{wl,pre}$, respectively. We bias the SST, GST and the PTs so that the current flowing through the FGTs is disabled. We bias the source line to ground. Hence the energy to transition to the precharge state (E_{pre}) equals

$$E_{pre} = E_{bl,pre} + E_{wl,pre} \quad (7)$$
$$E_{bl,pre} = 0.5 * C_{bl,wire} * (0 - V_{bl,pre})^2 * N_{bitlines-perblock} * L_{bitline} \quad (8)$$
$$E_{wl,pre} = 0.5 * C_{wl,wire} * (0 - V_{wl,pre})^2 * N_{pages} * L_{wordline}. \quad (9)$$

While we precharge the bitlines to reduce the latency of read accesses [15], the wordlines are not precharged. During our validation we set $V_{wl,pre}$ to zero, but the model includes the wordline precharge $V_{wl,pre}$ as a parameter so that devices that perform wordline precharging can use this parameter.

4) *The Read Operation:* The read operation for SLC and MLC flash are quite similar. In case of SLC flash and fast pages in 2-bit MLC flash, the sequence of operations required to sense the threshold voltage of the cells is the same since we only need to distinguish between two states. In case of slow MLC pages, we need two stages (and hence two read operations) to distinguish between four states.

To perform a read operation for both SLC and MLC flash, the block decoder selects one of the blocks to be read while the page decoder selects one of the N_{pages} inside a block. For SLC flash and fast pages in MLC flash, we bias the wordline of the selected page to ground while we change the bias of unselected wordlines to V_{read} from $V_{wl,pre}$. This causes the unselected pages to serve as transfer gates. Hence the energy to bias the wordline of the selected page to ground, and the

TABLE III

INPUTS TO *FlashPower*. *FlashPower* HAS 12 REQUIRED(R) AND 23 OPTIONAL(O) PARAMETERS

Microarchitectural Parameters	
$N_{pagesize}$ (R)	Size (in bytes) for the data area in each page.
$N_{sparebytes}$ (R)	Size (in bytes) for the spare area in each page.
N_{pages} (R)	Number of pages per block.
N_{brows} (R)	Number of rows of blocks in a plane.
N_{bcols} (O)	Number of columns of blocks in a plane. Defaults to 1.
N_{planes} (O)	Number of planes per die. Defaults to 1.
N_{dies} (O)	Number of dies per chip. Defaults to 1.
$tech$ (R)	Feature size of FGTs.
$Bits_per_cell$ (R)	Number of bits per FGT.
Device-level Parameters	
N_A, N_D (O)	Doping level of P-well and N-well. Defaults to $10^{15}cm^{-}3$ and $10^{19}cm^{-}3$.
β (O)	Capacitive coupling between control gate. and P-well. Defaults to 0.8.
GCR (O)	Ratio of control gate to total floating gate capacitance. Obtained from ITRS.
t_{ox} (R)	Thickness of tunnel oxide in FGTs.
$\frac{W}{L}$ (O)	Aspect ratio of FGTs. Defaults to 1.
Workload Parameters	
N_{bits_1} (O)	Number of 1's to be read, or programmed.

Timing Parameters	
$t_{program}$ (R)	Latency to program a page for SLC flash (fast page in MLC flash).
$t_{program,slow}$ (O)	Latency to program a slow page in MLC flash. Defaults to $t_{program}$ * 2.
t_{read} (R)	Latency to read a page in SLC flash (fast page in MLC flash).
$t_{read,slow}$ (O)	Latency to read a slow page in MLC flash. Defaults to t_{read} * 2.
t_{erase} (R)	Latency to erase a flash block.
Bias Parameters	
V_{dd} (R)	Maximum operating voltage of the chip.
V_{read} (O)	Read voltage for SLC flash (fast page in MLC flash). Defaults to 4.5 V.
$V_{read,slow}$ (O)	Read voltage for slow page in MLC flash. Defaults to 2.4V
$V_{bl,drop_[0/1]}$ (O)	Bit-line swing for read operation. Defaults to 0.7V.
V_{pgm} (O)	Program voltage for selected page. Obtained from ITRS.
V_{pass} (O)	Pass voltage during program for un-selected pages. Defaults to 10V.
V_{era} (O)	Erase voltage to bias the substrate. Same as V_{pgm}.
$V_{bl,pre}$ (O)	Bit-line precharge voltage. Defaults to 3/5th of V_{dd}.
V_{step} (O)	Step voltage used for program and erase. Defaults to 0.3V

Policy Parameters	
$N_{read_verify_cycles}$ (R)	Number of read verify cycles for a program operation
N_{erase_cycles} (R)	Number of erase pulses required to erase a block.
$\Delta V_{th,slc}$ (O)	Threshold voltage difference between programmed and erased state for SLC flash. Defaults to 3V.
$\Delta V_{th,mlc}$ (O)	Threshold voltage difference between adjacent programmed states for MLC flash. Defaults to 0.9V.
$Optimize_Write$ (O)	Boolean flag enabling optimization to certain threshold levels transitions in MLC flash. Defaults to false.
$Optimize_Erase$ (O)	Boolean flag enabling optimization if the threshold level of a FGT is unchanged after programming. Defaults to false.

unselected pages to V_{read} equals:

$$E_{selected-page,r} = 0.5 * C_{wl} * (0 - V_{wl,pre})^2 \quad (10)$$

$$E_{unselected-pages,r} = 0.5 * C_{wl} * (V_{read} - V_{wl,pre})^2 *(N_{pages} - 1). \quad (11)$$

For the read operation, the bitlines remain at $V_{bl,pre}$. Depending upon the threshold voltage of the FGT, the FGT is on or off which impacts the voltage drop in the bitline. Assuming N_{bits_1} to be the number of bits corresponding to logical "1" and N_{bits_0} to be the number of bits corresponding to logical "0", $V_{bl,drop_1}$ and $V_{bl,drop_0}$ to be the drop in bitline voltages for a 1-read and 0-read, the resulting energy dissipation equals:

$$E_{bl-1,r} = 0.5 * C_{bl} * (V_{bl,pre} - V_{bl,drop_1})^2 * N_{bits_1} \quad (12)$$

$$E_{bl-0,r} = 0.5 * C_{bl} * (V_{bl,pre} - V_{bl,drop_0})^2 * N_{bits_0} \quad (13)$$

$$E_{sl,r} = E_{ssl,r} + E_{gsl,r} + E_{srcline,r} \quad (14)$$

where $V_{bl,drop_1}$ is the drop in bitline voltage when FGT is on and $V_{bl,drop_0}$ is the drop in bitline voltage when FGT is off and $E_{ssl,r}$, $E_{gsl,r}$ and $E_{srcline,r}$ are given by:

$$E_{ssl,r} = E_{gsl,r} = 0.5 * C_{ssl} * (V_{read} - 0)^2$$

$$E_{srcline,r} = 0.5 * C_{srcline} * (V_{read} - 0)^2. \quad (15)$$

We detect the state of the cell using the sense amplifier connected to each bitline. The sense amplifier is a part of the page buffer and contains a latch unit [15]. It detects the voltage changes in the bitline and compares it with a reference voltage. Since this is very similar to DRAM sense amplifier [16], we use CACTI's DRAM sense amplifier model to determine the energy dissipated for sensing and CACTI's SRAM model to model the energy dissipated to store the sensed data in the page buffer.

In order to read slow pages, we repeat the above steps, but instead of biasing the wordline of the selected page to ground, we bias the wordline of the selected page to $V_{read,slow}$. If the threshold voltage of the FGT is less than $V_{read,slow}$, then the FGT is off and the current through the bitline is small. Otherwise, the FGT is on and the current is high. Combining the first and the second read operation helps to distinguish between four threshold stages, thus helping to read the contents of the slow page.

Once the read operation is complete, the system transitions from the read state to the precharged state. This means that we bias bitlines back to the precharge voltage $V_{bl,pre}$, while we bias the select lines back to ground from V_{read} and the wordlines back to $V_{wl,pre}$. However, the biasing for the transition from read operation to precharge state is equal but

opposite to the biasing for the transition from the precharge state to read operation. Hence the energy to transition from the read to the precharge state (E_{r-pre}) equals:

$$E_{r-pre} = E_{bl-1,r} + E_{bl-0,r} + E_{selected-page,r} + E_{unselected-pages,r} + E_{sl,r}. \quad (16)$$

We calculate the energy dissipated by the decoders for the read operation, $E_{dec,r}$, using CACTI. We modify the CACTI decoder model to perform two levels of decoding (block and page decode) for each read and program operation. Hence the total energy dissipated per read operation for SLC flash and fast page of MLC flash equals:

$$E_r = E_{selected-page,r} + E_{unselected-pages,r} + E_{bl-1,r} + E_{bl-0,r} + E_{sl,r} + E_{r-pre} + E_{senseamp,r} + E_{dec,r}. \quad (17)$$

The energy for a slow page read operation in MLC flash is equal to

$$E_{r,slow} = E_r + E_{selected-page,r} + E_{unselected-pages,r} + E_{bl-1,r} + E_{bl-0,r} + E_{sl,r} + E_{r-pre} + E_{senseamp,r} + E_{dec,r} \quad (18)$$

where $E_{selected-page,r}$ is calculated by biasing the wordline to $V_{read,slow}$ instead of ground.

5) *The Program Operation:* To perform a read operation for both SLC and MLC flash, the block decoder selects one of the blocks to be programmed while the page decoder selects one of the N_{pages} inside a block. We transfer the data from the controller to the page buffers. Since the decoding for the program operation is the same as that of the read operation, we estimate the energy for decoding during the program operation to be equal to $E_{dec,p} = E_{dec,r}$, where $E_{dec,r}$ is the energy for the decode operation estimated using CACTI.

FlashPower adopts the ISPP model to estimate energy dissipation during program operation [11]. In this paper, we refer to each pulse as a subprogram operation. For each subprogram operation, we increase the program voltage of the selected page by a step factor V_{step}. After each subprogram operation, we perform a verify program operation to check if the correct data is written to the page. If not, we repeat the subprogram operation, but with a higher program voltage. $N_{read_verify_cycles}$ indicates the total number of subprogram operations performed during the write operation. The duration of each subprogram pulse is a function of the program latency $t_{program}$ and $N_{read_verify_cycles}$.

We now calculate the energy for each subprogram operation. Let $V_{pgm,i}$ be the program voltage of the selected wordline in the ith iteration. We bias the selected wordline to $V_{pgm,i}$ and the unselected wordlines to V_{pass} to inhibit them from programming. Since the program voltage for the selected page varies for every iteration, we present the energy dissipated by this step as a function of voltage. Thus the energy to bias the selected page and the unselected page equals:

$$E_{selected-page,p}(V) = 0.5 * C_{wl} * (V_{pgm,i} - V_{wl,pre})^2 \quad (19)$$

$$E_{unselected-page,p} = 0.5 * C_{wl} * (V_{pass} - V_{wl,pre})^2 * (N_{pages} - 1). \quad (20)$$

FlashPower adopts the the self-boosted program inhibit model [11] to prevent cells corresponding to logical 1 from being programmed. According to the self-boosted program inhibit model, the channel voltage is boosted to about 80% of the applied control gate voltage by biasing the bitlines corresponding to logical "1" at $V_{bl,ip}$ and setting the SSL to V_{dd}. The resulting electric field across the oxide is not high enough for tunneling and the cells are inhibited from being programmed. Assuming N_{bits_1} to be the number of bits corresponding to logical 1, the energy dissipated by bitlines to inhibit programming is equal to:

$$E_{bl,ip} = 0.5 * (C_{bl} - C_{d,mc} * N_{pages}) * V_{ch_boost}^2 * N_{bits_1} \quad (21)$$

where V_{ch_boost} is the boosted channel voltage and is a fraction of applied control gate voltage.

We bias the bitlines corresponding to logical 0 to ground. The resulting high electric field across the tunnel oxide facilitates FN tunneling resulting in charges tunneling from the channel onto the floating gate. The energy dissipation of bitline corresponding to logical 0 is equal to:

$$E_{bl,p} = (0.5 * (C_{bl} - C_{d,mc} * N_{pages}) * (0 - V_{bl,pre})^2 + E_{tunnel,mc}) * N_{bits_0} \quad (22)$$

where the term $E_{tunnel,mc}$ is the tunneling energy per cell. $E_{tunnel,mc}$ is calculated as

$$E_{tunnel,mc} = \Delta V_{th} * I_{FN} * t_{subprogram} \quad (23)$$

where I_{FN} is the tunnel current and $t_{subprogram}$ is the duration of subprogram operation. I_{FN} is calculated as $I_{FN} = J_{FN} * A_{fgt}$, where J_{FN} is the tunnel current density calculated using [17] and A_{fgt} is the area of the floating gate.

Then the energy dissipated in charging the select lines is equal to

$$E_{sl,p} = E_{ssl,p} + E_{gsl,p} + E_{srcline,p} \quad (24)$$

where

$$E_{ssl,p} = E_{gsl,p} = 0.5 * C_{ssl} * (V_{dd} - 0)^2 E_{srcline,p} = 0.5 * C_{srcline} * (V_{dd} - 0)^2. \quad (25)$$

Once the subprogram operation completes, NAND flash chips perform a program-verify operation to check if write operation is successful. *FlashPower* models the verify-program operation as a read operation. Hence, the total energy per subprogram operation is:

$$E_{subp}(V) = E_{selected-page,p}(V) + E_{unselected-page,p} + E_{bl,ip} + E_{bl,p} + E_{sl,p} + E_{vp} \quad (26)$$

where E_{vp}, the energy spent in program verification and is given by $E_{vp} = E_r - E_{dec,r}$. E_r is defined by equation (17). Since the entire process of subprogram and verify-program is repeated a maximum of N_{loop} number of times, the maximum energy for programming is given by:

$$E_{pgm} = \sum_{i=0}^{N_{read_verify_cycles}} E_{subp}(V_{pgm,i}). \quad (27)$$

$N_{read_verify_cycle}$ is obtained from datasheets like [18] or can be fed as a input to the model.

Since a program operation concludes with a read operation, the transition from the Program to Precharge is same as the transition from a read to precharge.

$$E_{p-pre} = E_{r-pre} \quad (28)$$

where E_{r-pre} is given by equation (16).

Hence the total energy dissipated in the program operation is equal to:

$$E_p = E_{dec,p} + E_{pgm} + E_{p-pre}. \quad (29)$$

6) *The Erase Operation:* Since erasure happens at the block granularity, the controller sends the address of the block to be erased. The controller uses only the block decoder and the energy for block decoding ($E_{dec,e}$) is calculated using CACTI. To aid block-level erasing, the blocks are physically laid out such that all pages in a single block share a common P-well. Moreover, multiple blocks share the same P-well [11] and therefore it is necessary to prevent other blocks sharing the same P-well from being erased. *FlashPower* assumes the self-boosted erase inhibit model [11] to inhibit other blocks sharing the same P-well from getting erased.

Once we select an erase block, NAND flash memory uses multiple erase pulses to erase a block. Each such operation is referred to as a suberase operation. The bias voltage for the P-well of the selected block, V_{era}, is set to the initial program voltage, V_{pgm}. After each suberase operation, a read-verify operation determines whether all FGTs in the block are erased or not. If all the FGTs in the block are not erased, we repeat the erase operation (maximum of N_{erase_cycles} times) after incrementing V_{era} by the step voltage V_{step}. We denote the erase voltage used for each suberase operation to be $V_{suberase,i}$, where i indicates the iteration count of the suberase operation. The duration of each suberase operation, $t_{suberase}$, is a function of the maximum erase latency and N_{erase_cycles}.

For each suberase operation, we bias the control gates of all the wordlines in the selected block to ground. We bias the P-well for the selected block to erase voltage $V_{suberase,i}$ and the SSL and the GSL to $V_{suberase,i} * \beta$, where β is the capacitive coupling ratio of cells between control gate and the P-well. A typical value of β is 0.8 [10]. We bias the SL to $V_{bl,e}$. Here $V_{bl,e}$ is equal to $V_{suberase,i} - V_{bi}$ where V_{bi} is the built-in potential between the bitline and the P-well of the cell array [19]. Adding up the charging of the SSL, GSL and SL, we get the energy dissipated in the select lines to be:

$$E_{sl,e} = E_{ssl,e} + E_{gsl,e} + E_{srcline,e} \quad (30)$$

where $E_{ssl,e}$, $E_{gsl,e}$ and $E_{srcline,e}$ are calculated using the SSL, GSL and SL capacitance and the bias condition explained above. We bias the bitlines to $V_{bl,e}$ which dissipates energy that is modeled as:

$$E_{bl,e}(V) = 0.5 * C_{bl} * (V_{bl,e} - V_{bl,pre})^2 * N_{bitlines-perblock}. \quad (31)$$

We parameterize $E_{bl,e}(V)$ as a function of applied erase voltage, since the erase voltage changes for each suberase operation.

With the P-well biased to $V_{suberase,i}$, cells that have high threshold voltage undergo FN tunneling. Electrons tunnel off the floating gate onto the substrate and the threshold voltage of the cell is reduced. To effect FN tunneling, the depletion layer capacitance between the P-well and the N-well should be charged. This capacitance of the junction between the P-well and the N-well, which form a P-N junction, is a function of the applied voltage V_{era}, the area of the p-well A_{well}. The dynamic power to charge this capacitance as the voltage across the junction raises from $0V$ to $V_{suberase,i}$ is determined as:

$$E_{junction,e}(V) = \left(\frac{C_{j0}}{(1 - V_{suberase,i}/\phi_0)^m} * A_{fgt}\right) * V_{suberase,i}^2 \quad (32)$$

where C_{j0} is the capacitance at zero-bias condition and ϕ_0 is the built-in potential of the P-N junction. m, the grading coefficient is assumed to be 0.5. We parameterize $E_{junction,e}$ as a function of applied erase voltage since the erase voltage changes for each suberase operation.

Once this P-N junction capacitance is fully charged, all cells in the block are erased. The tunneling energy for the block, $E_{tunnelerase,mc}$, is calculated using equation (23) and a suberase pulse whose duration is $t_{suberase}$. After each suberase pulse, a verify erase operation is performed to ensure that all FGTs in the block are erased [11]. The verify erase operation is a single read operation which consumes energy equivalent to a read operation. This is modeled as, $E_{block,ve} = E_r - E_{dec,r}$ where E_r is given by equation (17).

Since the erase voltage changes for each suberase operation, the total energy consumed in each suberase operation, $E_{suberase}$ is parameterized as a function of the voltage and is equal to

$$\begin{aligned} E_{suberase}(V) &= E_{sl,e} + E_{bl,e}(V) + E_{tunnelerase,mc}(V) \\ &\quad + E_{junction,e}(V) + E_{block,ve}. \end{aligned} \quad (33)$$

Since an erase operation ends with a read operation, the transition from the erase to precharge is same as the transition from read to precharge. The energy dissipated during this transition is equal to

$$E_{e-pre} = E_{r-pre} \quad (34)$$

where E_{r-pre} is given by (16).

Hence, the total energy dissipated in erase operation E_{erase} is equal to

$$\begin{aligned} E_{erase} &= E_{dec,e} + \sum_{i=0}^{N_{erase_cycles}} E_{suberase}(V_{suberase,i}) \\ &\quad + E_{e-pre}. \end{aligned} \quad (35)$$

We can observe that if all the cells in the block are programmed and are at a low threshold voltage stage, then an erase operation is not necessary. Some controllers can identify this scenario and prevent an erase operation even when they receive such a command, thereby reducing the power consumption and the latency of the erase operation.

7) *Charge Pumps:* Each read, subprogram, or suberase operation requires that the charge pump supply a high voltage (5 V to 20 V) to the flash memory array. Ishida *et al.* [20] specify that the energy consumed for each high voltage pulse

TABLE IV
NAND FLASH CHIPS USED FOR EVALUATION

Chip Name	**Feature Size** (in *nm*)	**Page Size** (in KB)	**Spare Area** (in bytes)	**Pages/Block**	**Blocks/Plane**	**Planes/Die**	**Dies/Chip**	**Total Capacity** (in *Gb*)
B-MLC8	72	2KB	64	128	2048	2	1	8.25
D-MLC32	80	4KB	128	128	2096	2	2	33.77
E-MLC64	51	4KB	128	128	2048	4	2	66.00
A-SLC4	73	2KB	64	64	2048	2	1	4.13
B-SLC4	72	2KB	64	64	2048	2	1	4.13

from the charge-pumps is 0.25 μJ for chips operating at 1.8 V and 0.15μJ for chips operating at 3.3 V for the read and program operation. In *FlashPower*, we use these constant values but since *FlashPower* is parameterized, a detailed charge-pump model can be incorporated in lieu of the current one.

8) *Multiplane Operation:* The power model described thus far corresponds to single-plane operations. However modern NAND flash chips allow multiple planes to operate in parallel. These are referred to as multiplane operations. Since *FlashPower* models single-plane operations, energy consumption for multiplane operations can be determined by multiplying the results of single-plane operations with the number of planes operating in parallel.

So far, we have provided a detailed analytical model that estimates the energy dissipation of NAND flash for individual operations. We now validate our model with measurements from real flash chips from various manufacturers.

IV. EXPERIMENTAL SETUP

To validate the model's usability and accuracy, we measured the latency and energy consumption of read, program, and erase operations from real flash chips. Using a custom-built board, we acquired fine-grain measurements of a diverse set of flash chips. In this section, we describe the hardware setup and the test procedure.

A. Hardware Setup

Fig. 6 depicts our flash characterization board that consists of two sockets for testing flash chips. The board connects to a Xilinx XUP development kit with a Virtex-II FPGA and an embedded PowerPC processor. The processor runs Linux and connects to a custom flash controller that gives the user complete access to the flash chips. The flash controller can issue operations to the chips and record latency measurements with a 10 ns resolution. For more details about the characterization board, please refer [21].

To measure the energy consumption of individual flash operations, we use a high-bandwidth current probe attached to a mixed-signal oscilloscope. The Agilent 1147A current probe attaches to a jumper that provides power to a single flash chip. We trigger the oscilloscope to capture current measurements when the flash controller sends a command to the flash chip.

B. Test Procedure

Table IV summarizes the system-level properties of the flash chips we used for validation. We selected a diverse set of chips in terms of feature size, capacity, array structure, and manufacturer to depict the compatibility and adaptability of *FlashPower*.

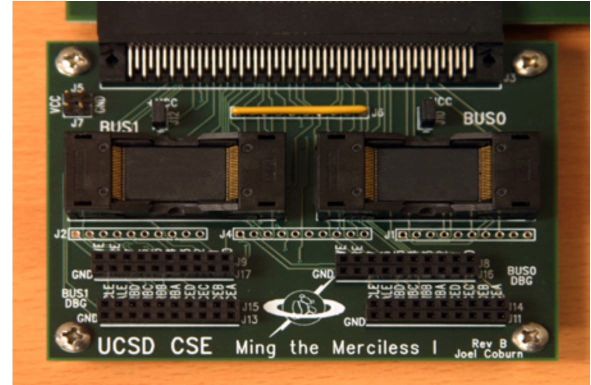

Fig. 6. **Flash test board** The custom-built flash test board connects to an FPGA-based development kit with an embedded processor running Linux and a flash controller that issues operations to up to two Flash chips at a time.

We measured the latency and power consumption for read, program, and erase operations on each chip. The types of operations we performed varied for SLC- and MLC-based flash chips.

SLC chips We tested four states for a single-page read operation on SLC-based chips: unprogrammed (freshly erased), all 0's, 50% 0's, and all 1's. For the program operation, we acquired data while programming a single page with all 0's, 50% 0's, and all 1's. Finally, for erase operations, we measured the energy while erasing an entire block of pages programmed with all 0's and pages programmed with all 1's.

MLC chips To quantify energy consumption for read operations of MLC-based chips, we measured the energy consumption of reading a page in four states:

1) *Fast Page Programmed*: reading a programmed fast page;
2) *Slow Page Programmed*: reading a programmed slow page;
3) *Fast Page Unprogrammed*: reading a freshly-erased fast page;
4) *Slow Page Unprogrammed*: reading a freshly-erased slow page.

For program operations, we measured the energy consumption of the following six transitions caused by programming a page:

1) *Fast Page 0* - Programming a freshly-erased fast page with all 0's;

2) *Fast Page 1* - Programming a freshly-erased fast page with all 1's;
3) *Slow Page 01* - Programming a slow page with all 1's and the fast page counterpart is of type *Fast Page 0*;
4) *Slow Page 11* - Programming a slow page with all 1's and the fast page counterpart is of type *Fast Page 1*;
5) *Slow Page 00* - Programming a slow page with all 0's and the fast page counterpart is of type *Fast Page 0*;
6) *Slow Page 10* - Programming a slow page with all 0's and the fast page counterpart is of type *Fast Page 1*.

For erase operations, we measured the energy consumption of erasing an entire block of fully programmed pages of all 0's and all 1's.

For each chip and each type of operation, we performed ten measurements and reported the average. We summarize the results of these experiments in conjunction with the model's approximations in Section V.

V. Validation of *FlashPower*

We now compare the outputs of the analytical model with power measurements from real SLC and MLC flash chips. The sources used to obtain values for parameters used in our model (listed in Table III) are as follows. Except for feature size of FGTs, values for all microarchitectural parameters are obtained from manufacturer datasheets. The feature size of FGTs used in our chips are obtained from [22], [23]. Values for device-level parameters are obtained from [19], [10], [14]. The operating voltage of the chip is obtained from datasheets, while for other bias parameters we use information available in Chapter 6 of [10]. The value for timing parameters and policy parameters are deduced from datasheets, by measuring the latency of flash operations, and from [10]. We used the measured latencies of each flash operation (see Section IV) because we found that the values for timing parameters and program verify cycles in datasheets were significantly different (as high as 50% in some cases) from actual behavior. While datasheets contained average latencies of flash operation, the actual latencies depended on factors like the type of page and in some cases on the data pattern being used. Finally, since we measure the energy consumption of the chips by varying the input bit pattern, we control the values for the workload parameters.

We now present the results for read, program, and erase operation for both MLC flash followed by SLC flash.

A. Validation for SLC Flash

Fig. 7(a)–(c) shows the results for the read, program, and erase operations for SLC flash. The y-axis represents the energy consumption of the particular operation, while the x-axis represents the state of a page in SLC flash as explained in Section III-B. From Fig. 7(a), we can observe that the variation between the estimated and the measured read energy is about 40% on average (at most 62% or $8\mu J$) for A-SLC4, while it is less than 10% for B-SLC4. For the program operation, the difference between estimates and measurements is 22% on average (at most 26% or $2\mu J$) for B-SLC4, while the difference is 40% on average (at most 62% or $8\mu J$) for A-SLC4. For the erase operation, it is 24% on average (at most 27% or $8\mu J$) for B-SLC4, while it is 36% on average (at most 37% or 24μJ) for A-SLC4.

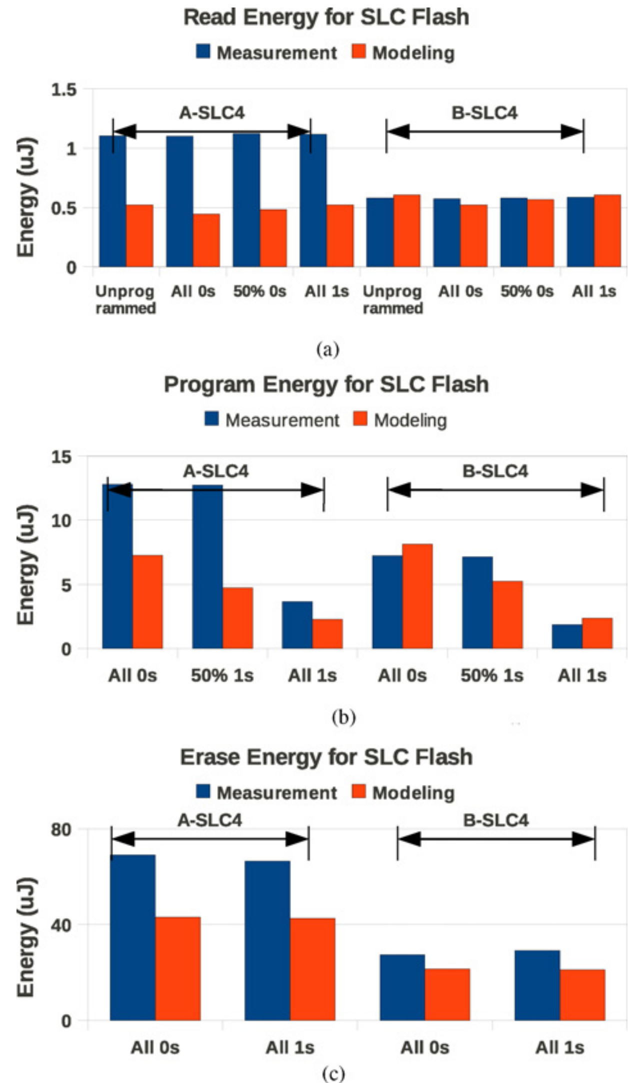

Fig. 7. Energy comparison for SLC Flash. (a) Read energy comparison for SLC Flash. (b) Program energy comparison for SLC Flash. (c) Erase energy comparison for SLC Flash.

These results show that the *FlashPower* can estimate the energy consumption of B-SLC4 with high accuracy, but for A-SLC4, the accuracy is low. It should be noted that the differences among the the *measured* values of the two chips is similar. In other words, the *estimates* of A-SLC4 and B-SLC4 are comparable among themselves, while the differences in *measurements* between A-SLC4 and B-SLC4 are high.

To understand why such variations occur, we compare the parameters of the two chips. We find that the microarchitectural parameters for the two chips are almost the same (please refer to Table IV). Since the feature size and the operating voltage of the two chips are almost the same, our model assumes that the device level parameters are same. As far as the timing parameters are concerned, our experiments show that the read latency of A-SLC4 is about 20% higher than B-SLC4. The program latency for the two chips is similar,

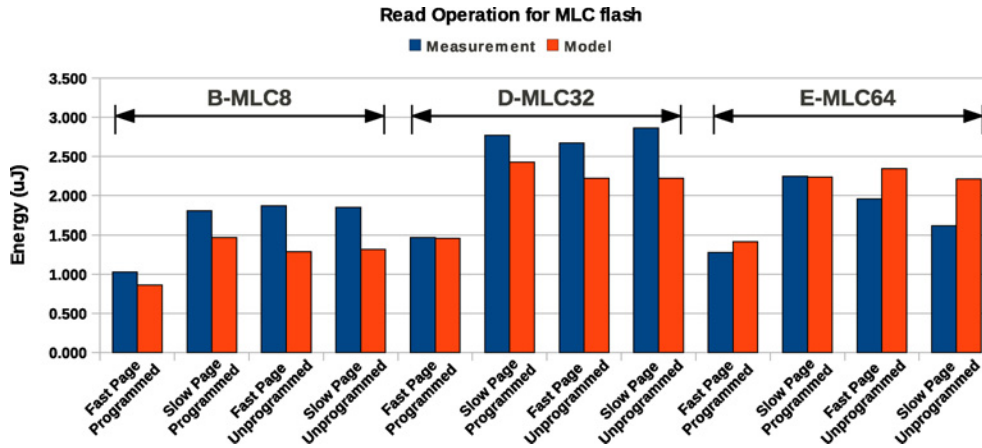

Fig. 8. Read energy comparison for MLC Flash.

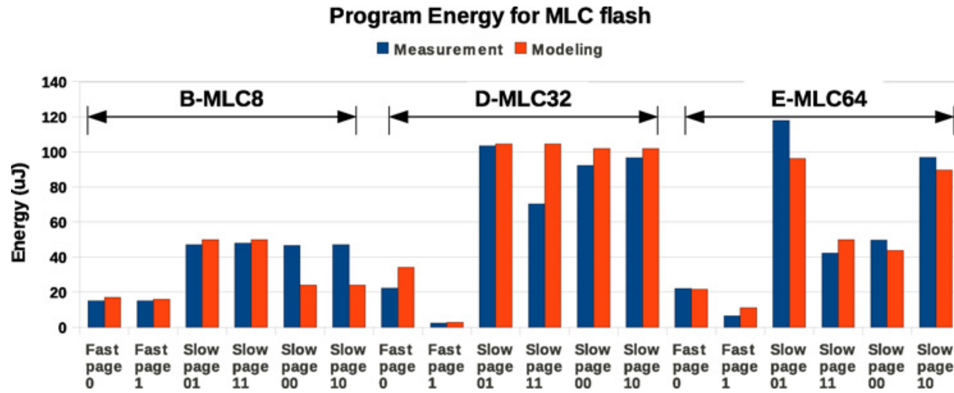

Fig. 9. Program energy comparison for MLC Flash.

while the erase latency is about 30% higher for A-SLC4 compared to B-SLC4. With respect to the policy parameters mentioned in Table IV, values for $\Delta V_{th,slc}$ is set based on [14] and since the chips have the almost same feature size, their values are same. The values of program-verify and erase cycles are set using latency measurements from the chips and they are similar. Since the bit patterns are constant across both the chips, we hypothesize that the differences in energy are due to changes in specific policy parameters like $\Delta V_{th,mlc}$ and $N_{read_verify_cycles}$, device parameters like β and t_{ox} or due to manufacturer-specific optimizations of the individual operations. Better knowledge about the values for these parameters can significantly increase the accuracy of the model. *FlashPower* is developed based on [10], [15], [12], [11] and describes the canonical operation of NAND flash memory and does not capture manufacturer-specific designs. This results in a significant variation between estimates and measurements for A-SLC4 while the variation is significantly lower for B-SLC4. However, since *FlashPower* is extensively parameterized, any manufacturer-specific implementations can be used for estimating flash power consumption.

B. Validation for 2-Bit MLC Flash

Fig. 8 shows the results for the read operation for MLC flash. The x-axis represents the page being read during the operation while the y-axis represents the total energy dissipated during the read. Each column represents the average energy consumed by the flash chip when it is at one of the various states described in Section III-B. From the figure, we can observe that the difference between estimated and measured energy is less than 22.3% (or $0.64\mu J$) for all types of pages. We found that the difference in estimated and measured energy values were significantly higher for unprogrammed pages than for programmed pages. Since the threshold voltage states of cells in unprogrammed pages are nondeterministic, modeling the energy consumption for an unprogrammed page is challenging. It should be noted that reading an unprogrammed page does not happen often in practice. If we just consider the programmed pages, the difference drops to less than 12.3% (or $0.35\mu J$) and in most cases, the difference between measured and estimated values are negligible.

Fig. 9 shows the result for the program operation for MLC flash. The y-axis represents the energy consumption of a program operation, while the x-axis represents the various states of a flash page. Each column in the figure represents the energy consumed by the flash chip when it is at one of states described in Section III-B. We find that the accuracy of the model is high while programming fast pages for all three flash chips. For slow pages, the accuracy depends on the chip being measured and the bit pattern being used.

Many factors like the number of program verify cycles, the duration of each subprogram operation, the thickness of the tunnel oxide (which affects the tunneling energy), the program and step voltages used, the difference between the

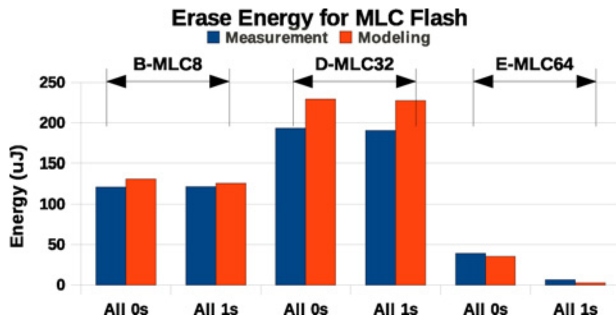

Fig. 10. Erase energy comparison for MLC Flash.

TABLE V

DIFFERENT 16 GB CHIP CONFIGURATIONS

Chip Name	Page Size(KB)	Spare Bytes	Pages/ Block	Blocks/ Plane	Planes/ Die	Dies/ Chip
X1-M16	2	64	128	2048	2	2
X2-M16	2	64	128	4096	2	1
X3-M16	4	128	128	2048	2	1
X4-M16	2	64	256	2048	2	1
X5-M16	2	64	128	2048	4	1

starting and the final threshold voltage states affect the energy dissipation during a program operation. Moreover, the values for these factors change with the type of bit pattern being written. Since programming a slow page involves carefully controlling the threshold voltage distribution of the FGT, an accurate value for each of these parameters is even more important. For example, a linear difference in program and step voltage can cause quadratic difference in energy consumption. While *FlashPower* identifies values for some factors like number of program-verify cycles used, and the duration of each subprogram operation by reverse engineering the chips, values for V_{pgm}, V_{step}, $\Delta V_{th,mlc}$ for each chip that is being evaluated are not publicly available. Since values for such parameters are not publicly available, the variation between estimates and measurements are high for some bit patterns. For example, in the case of B-MLC8, increasing V_{pgm} and $\Delta V_{th,mlc}$ reduced the difference between the measurement and the estimates for the Slow Page 00 and Slow Page 11 states from 49% to nearly 8%. In the case of D-MLC32, even a small decrease in V_{pgm} and $\Delta V_{th,mlc}$ reduces the variation between estimates and measurement for the Slow Page 11 state from 49% to nearly 10%. Since *FlashPower* has been extensively parameterized, any user who has detailed information on individual parameters can feed those parameters into *FlashPower* to get an accurate estimate of flash energy consumption.

Fig. 10 shows the results for the erase operation for the three chips for two different bit patterns. For B-MLC8 and D-MLC32, changing the bit pattern did not affect the erase energy significantly. For E-MLC64, when all pages in the block were in the lowest threshold state (11), the erase energy was about 5X lower than the energy consumption for all other bit patterns. We call this the *optimize erase operation*, where the controller does not erase a block where all cells already have the lowest threshold voltage. While two chips, B-MLC8 and D-MLC32, did not perform this optimization, E-MLC64 had this optimization enabled. From this figure, we can also see that the estimates correspond well to the measurements. The variation between estimates and measurement was less than 10% for B-MLC8 and E-MLC64, while the difference was less than 20% for D-MLC32.

TABLE VI

ENERGY DISSIPATION FOR VARIOUS 16 GB CHIP CONFIGURATIONS

Configuration	Read Energy Fast Page Programmed(μJ)	Program Energy-Fast Page 0 (μJ)	Erase Energy (μJ)
X1-MLC16	1.085	15.418	117.602
X2-MLC16	1.629	18.297	232.577
X3-MLC16	1.632	29.747	232.908
X4-MLC16	1.59	18.205	227.719
X5-MLC16	1.277	15.562	117.595

So far, we have examined the accuracy of *FlashPower* by comparing the results with measurements from a range of chips across various manufacturers. We now use the tool to perform some design space exploration studies.

VI. DESIGN SPACE EXPLORATION USING *FlashPower*

FlashPower enables us to study the effect of various flash parameters (listed in Table III) on the power consumption of the chip. In this section, we show the utility of *FlashPower* through two case studies to identify power optimal design configurations.

Case Study 1: Let us say a chip architect decides to design a 16 Gb NAND flash chip and wants to estimate the energy consumption for various design scenarios. Table III provides a list of all parameters that can be adjusted by the architect to identify the relative benefits of each design scenario. Assuming that the designer has the flexibility to change the memory array configuration of flash, we show how to use *FlashPower* to compare some configurations. For this study, we vary the microarchitecture parameters listed in Table III, except the feature size, while the values for all other parameters are assumed to be the same as the B-MLC8 chip. Table V provides a list of configurations for designing a 16 Gb memory chip.

Table VI shows the read, write and erase energy for different memory configurations while reading and programming a fast page and erasing a block. Note that, in the case of X3-M16, read and program operations occur at 4KB granularity, while for other configurations, they occur at a 2KB granularity. The block size is 512KB for X3-M16 and X4-M16, while it is 256KB for the other configurations. Table VI can be used by the architect to estimate the energy dissipation for various memory configurations. From Table VI we can see that X1-M16 is the chip that dissipates the least energy for read and program operations (for erase operation, the energy dissipation is almost the same as X5-M16). However, the page size of X1-M16 is just 2KB. The page size for X3-M16 is 4KB, while the energy consumption of reads is only 60% more than X1-M16. Depending upon whether the architect is optimizing for total energy consumption or energy dissipated per byte processed, they can choose one design over the other. In systems that run on battery power and have small data access granularity like consumer electronics devices and wireless sensors, where the architect is limited by maximum energy dissipation of the chip, X1-M16 is better option that X3-M16. In systems where energy efficiency is of higher importance than maximum energy dissipation, X3-M16 is a better option. Some designers may choose X5-M16 over the rest because the

number of planes in X5-M16 is higher than the rest, which allows for more operations to happen in parallel. Note that for X3-M16, the write and erase energy, is still ~2X higher than X1-M16, because their pages are larger.

Case Study 2: Let us say a chip architect decides to design a NAND flash chip that can consume less energy while storing specific data patterns. This scenario could be motivated by an application behavior or due to flash aware encoding schemes that seek to increase flash lifetime by favoring one bit pattern over an another. For example, a flash aware encoding scheme can choose to write more logical 1s than logical 0s in order to reduce the probability of tunneling of cells, thereby increasing flash lifetime. *FlashPower* allows the architect to estimate the impact of energy dissipation due to different bit patterns. Table VII shows the energy dissipation of X1-M16 for read, write and erase operations for various bit patterns. These results are based on the assumption that there is 100% bitline discharge for logical 1s and no discharge for logical 0s during the read operation.

From Table VII, we can see that for a read operation, a page filled with 0s consume less energy than a page with 1s because bitlines corresponding to logical 0 do not discharge while bitlines corresponding to logical 1 discharges fully and hence dissipates more energy. However, in case of write operations, writing logical 1s consumes less energy because programming is inhibited for logical 1s, while tunneling happens when logical 0s are written, consuming more energy. For erase operations, if the chip supports optimize erase feature, then the erase energy is just $2.5\mu J$. Otherwise, the erase energy does not vary significantly with the data pattern since the entire P-well of the erase block is biased to the erase voltage V_{era} and hence dissipate almost the same energy.

These two case studies show the utility of *FlashPower* in evaluating the design tradeoffs for NAND flash memory. Because *FlashPower* is parameterized, it is possible to analyze such what-if scenarios and identify power optimal flash chip configurations.

TABLE VII

ENERGY DISSIPATION FOR VARIOUS BIT PATTERNS FOR X1-M16. IF X1-M16 OPTIMIZES ERASE, THEN THE ENERGY DISSIPATED IS $2.566\mu J$. OTHERWISE, IT IS $115.218\mu J$

Configuration	Read Energy Fast Page Programmed(μJ)	Program Energy-Fast Page (μJ)	Erase Energy (μJ)
All 1s	1.182	8.385	2.566/115.218
75% 1s	1.023	10.144	115.221
50% 1s	0.863	11.902	115.224
25% 1s	0.703	13.660	115.226
All 0s	0.543	15.418	115.229

VII. RELATED WORK

CACTI is an integrated timing, area, leakage and power modeling tool from HP Labs [9]. CACTI supports evaluating SRAM and DRAM memory technologies. Our paper uses the underlying infrastructure available in CACTI and extends it to model NAND flash memories. Research interest in nonvolatile memories has spawned development of analytical models for many memory technologies, most of which use CACTI as their base. Smullen *et al.* [24] provide an analytical model to characterize the behavior of spin-transfer torque RAM (STT-RAM). Dong *et al.* [25] provide a similar analytical model for phase change memories. Xu *et al.* [26] extend CACTI to model resistive RAM (RRAM) memory technology. While analytical models form one end of the spectrum, characterizing memory technologies with actual measurements form the other end. Grupp *et al.* [22] provide a characterization of physical properties using actual measurements from flash chips. Yaakobi *et al.* [27] use results from actual measurements to characterize the reliability of flash chips. Boboila *et al.* [28] perform a similar study, but focus mainly on write endurance. *NVSim* presents analytical models for flash with validations at a datasheet level [29]. *FlashPower* uses measurements from actual chips and combines it with a detailed parameterized analytical model for NAND flash memories. NANDFlashSim is a microarchitecture level tool that simulates only the performance characteristics of NAND flash chips NANDflashsim. NANDFlashSim can be used on top of circuit-level analytical modeling tools like *FlashPower* to provide a flash simulation infrastructure that models both energy and performance.

VIII. CONCLUSION

In this paper, we presented *FlashPower*, a detailed power model for the two most popular variants of NAND flash namely, the single-level cell (SLC) and 2-bit multilevel cell (MLC) based NAND flash memory chips. *FlashPower* modeled the energy dissipation of flash chips from first principles and was highly parameterized to study a large design space of memory organizations. We validated *FlashPower* against measurements from 5 different chips from different manufacturers, and showed that our model accurately estimated the energy dissipation of several real chips.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their valuable comments.

REFERENCES

[1] C. Dirik and B. Jacob, "The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device architecture, and system organization," in *Proc. Int. Symp. Computer Architecture (ISCA)*, 2009, pp. 279–289.

[2] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," in *Proc. USENIX Tech. Conf. (USENIX)*, 2008, pp. 57–70.

[3] T. Kgil, D. Roberts, and T. Mudge, "Improving NAND Flash based disk caches," in *Proc. Int. Symp. Computer Architecture*, 2008, pp. 327–338.

[4] A. M. Caulfield, L. M. Grupp, and S. Swanson, "Gordon: Using Flash memory to build fast, power-efficient clusters for data-intensive applications," in *Proc. Int. Conf. Architectural Support Programming Languages Operating Systems (ASPLOS)*, 2009, pp. 217–228.

[5] J. K. Kim, H. G. Lee, S. Choi, and K. Il Bahng, "A PRAM and NAND Flash hybrid architecture for high-performance embedded storage subsystems," in *Proc. 8th ACM Int. Conf. Embedded Software*, 2008, pp. 31–40.

[6] E. Gal and S. Toledo, "Algorithms and data structures for Flash memories," *ACM Comput. Surveys*, vol. 37, no. 2, pp. 138–163, 2005.

[7] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: A Flash translation layer employing demand-based selective caching of page-level address mappings," in *Proc. 14th Int. Conf. Architectural Support Programming Languages Operating Syst.*, 2009, pp. 229–240.

[8] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *Proc. IEEE 17th Int. Symp. HPCA*, Feb. 2011, pp. 50–61.

[9] S. Wilton and N. Jouppi, "Cacti: An enhanced cache access and cycle time model," *IEEE J. Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, May 1996.

[10] J. Brewer and M. Gill, Eds., *Nonvolatile Memory Technologies With Emphasis on Flash*. New York, USA: IEEE Press, 2008.

[11] K.-D. Suh, B.-H. Suh, Y.-H. Um, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, J.-H. Choi, and J.-R. Kim, H.-K. Lim, "A 3.3V 32 Mb NAND Flash memory with incremental step pulse programming scheme," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, Nov. 1995.

[12] T. Tanaka, Y. Tanaka, H. Nakamura, K. Sakui, H. Oodaira, R. Shirota, K. Ohuchi, F. Masuoka, and H. Hara , "A quick intelligent page-programming architecture and a shielded bitline sensing method for 3V-only NAND Flash memory," *IEEE J. Solid-State Circuits*, vol. 29, no. 11, pp. 1366–1373, Nov. 1994.

[13] K. Takeuchi, T. Tanaka, and T. Tanzawa, "A multipage cell architecture for high-speed programming multilevel NAND Flash memories," *IEEE J. Solid-State Circuits*, vol. 33, no. 8, pp. 1228–1238, Aug. 1998.

[14] *Process Integration and Device Structures, ITRS 2011 Edition* [Online]. Available: http://www.itrs.net/Links/2011ITRS/2011Tables/PIDS_2011Tables.xlsx

[15] J. Lee, S.-S. Lee, O.-S. Kwon, K.-H. Lee, D.-S. Byeon, I.-Y. Kim, K.-H. Lee, Y.-H. Lim and B.-S. Choi and J.-S. Lee, W.-C. Shin, J.-H. Choi, and K.-D. Suh, "A 90-nm CMOS 1.8-V 2-Gb NAND Flash memory for mass storage applications," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 1934–1942, Nov. 2003.

[16] D. T. Wang, "Modern DRAM memory systems: Performance analysis and a high performance, power-constrained DRAM scheduling algorithm," Ph.D. dissertation, Univ. Maryland, College Park, MD, USA, 2005.

[17] M. Lenzlinger and E. Snow, "Fowler-Nordheim tunneling into thermally grown SiO_2," *IEEE Trans. Electron Devices*, vol. 15, no. 9, pp. 686–686, Sep. 1968.

[18] Samsung Corporation, "K9XXG08XXM Flash memory specification," K9XXG08XXM Datasheet.

[19] K. Sakui, Intel Corporation/Tohoku University, May 2009, private correspondence.

[20] K. Ishida, K. Yasufuku, T. Miyamoto, S. Nakai, H. Takamiya, M. Sakurai, T. Takeuchi, "A 1.8V 30nJ adaptive program-voltage (20V) generator for 3D-integrated NAND Flash SSD," in *Proc. Solid-State Circuits Conf. Dig. Tech. Papers*, 2009, pp. 238–239, 239a.

[21] T. Bunker, M. Wei, and S. Swanson, "Ming II: A Flexible platform for NAND Flash-based research," Dept. Comput. Sci. Eng., Univ. California, San Diego, CA, USA, Tech. Rep. CS2012-0978, 2012.

[22] L.M. Grupp, A.M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P.H. Siegel, J.K. Wolf, "Characterizing Flash memory: Anomalies, observations, and applications," in *Proc. 42nd IEEE/ACM MICRO*, 2009, pp. 24–33.

[23] H. Schmidt, D. Walter, F. Gliem, B. Nickson, R. H.-Sorensen, A. Virtanen, "Tid and see tests of an advanced 8 gbit NAND-FLASH memory," in *Proc. IEEE Radiation Effects Data Workshop*, Jul. 2008, pp. 38–41.

[24] C. W. Smullen, IV, A. Nigam, S. Gurumurthi, and M. R. Stan, "The STeTSiMS STT-RAM simulation and modeling system," in *Proc. Int. Conf. Comput.-Aided Design*, 2011, pp. 318–325.

[25] X. Dong, N. P. Jouppi, and Y. Xie, "PCRAMsim: System-level performance, energy, and area modeling for phase-change RAM," in *Proc. Int. Conf. Comput.-Aided Des.*, 2009, pp. 269–275.

[26] C. Xu, X. Dong, N. Jouppi, and Y. Xie, "Design implications of memristor-based RRAM cross-point structures," in *Proc. DATE*, 2011, pp. 1–6.

[27] E. Yaakobi, J. Ma, L. Grupp, P. Siegel, S. Swanson, and J. Wolf, "Error characterization and coding schemes for flash memories," in *Proc. IEEE GLOBECOM Workshops*, Dec. 2010, pp. 1856–1860.

[28] S. Boboila and P. Desnoyers, "Write endurance in Flash drives: Measurements and analysis," in *Proc. USENIX Conf. File and Storage Technol.*, 2010.

[29] NVSim [Online]. Available: http://www.rioshering.com/nvsimwiki/index.php?title=Main_Page

[30] M. Jung, E. Wilson, D. Donofrio, J. Shalf, and M. Kandemir, "NANDFlashSim: Intrinsic latency variation aware NAND Flash memory system modeling and simulation at microarchitecture level," in *Proc. IEEE 28th Symp. Mass Storage Syst. Technol.*, Apr. 2012, pp. 1–12.

Vidyabhushan Mohan received the B.E. degree from the College of Engineering, Anna University, Guindy, Chennai, India, in 2006, and the M.S. degree from the University of Virginia, Charlottesville, VA, USA, in 2010, both in computer science and engineering. He is currently pursuing the Ph.D. degree in computer science at the University of Virginia.

His current research interests include solid state storage, nonvolatile memories, and computer architectures.

Trevor Bunker received the B.S. degree in computer engineering from Brigham Young University, Provo, UT, USA, in 2009. He is currently pursuing the Ph.D. degree at the University of California, San Diego, CA, USA.

His current research interests include the hardware/software co-design of scalable systems for solving large-scale, data-intensive applications.

Laura Grupp characterizes Flash chips to find unpublished behaviors and trends, and then designs ways to exploit these characteristics to address application-specific requirements for performance, energy efficiency, and reliability. Her current research interests bridge the divide between hardware and software, with a focus on Flash-based storage systems.

Sudhanva Gurumurthi (SM'10) received the B.E. degree from the College of Engineering Guindy, Chennai, India, in 2000, and the Ph.D. degree from the Pennsylvania State University, University Park, PA, USA, in 2005, both in computer science and engineering.

He is currently an Associate Professor at the Computer Science Department, University of Virginia, Charlottesville, VA, USA. His current research interests include architecture reliability, storage systems, nonvolatile memory, and data center architectures.

Dr. Gurumurthi was the Associate Editor-in-Chief of *Computer Architecture Letters* and currently serves as an Associate Editor. He is a recipient of the NSF CAREER Award and several research awards from NSF, Google, Intel, and HP. He is a Senior Member of the ACM.

Mircea R. Stan (SM'XX) received the Diploma degree from Politehnica University, Bucharest, Romania, and the M.S. and Ph.D. degrees from the University of Massachusetts, Amherst, MA, USA.

He teaches and does research in the areas of high-performance low-power VLSI, temperature-aware circuits and architecture, embedded systems, and nanoelectronics at the ECE Department, University of Virginia, Charlottesville, VA, USA, where he is currently a Professor.

Dr. Stan received the NSF CAREER Award in 1997. He was a co-author on Best Paper Awards at ISQED 2008, GLSVLSI 2006, ISCA 2003, and SHAMAN 2002. He is an Associate Editor for the IEEE TRANSACTIONS ON NANO and was an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS SYSTEMS I from 2004 to 2008 and for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS from 2001 to 2003. He is currently a Distinguished Lecturer for the IEEE Circuits and Systems (CAS) Society, was also for 2004 to 2005, and for the IEEE Solid-State Circuits Society in 2007 to 2008. He is a member of the ACM, and also of Eta Kappa Nu, Phi Kappa Phi, and Sigma Xi.

Steven Swanson received the Ph.D. degree from the University of Washington, Seattle, WA, USA, in 2006.

He is currently an Associate Professor at the Department of Computer Science and Engineering, University of California, San Diego, CA, USA, and the Director of the Non-Volatile Systems Laboratory. His current research interests include systems, architectures, security, and reliability issues surrounding nonvolatile, solid-state memories. Previously, he has worked on scalable dataflow architectures, ubiquitous computing, and simultaneous multithreading.