On-Chip Randomization for Memory Protection Against Hardware Supply Chain Attacks to DRAM

Brett Meadows*, *Member, IEEE*
Department of Computer Science
University of Colorado
Colorado Springs
Colorado Springs, CO, USA
hmeadows@uccs.edu

Nathan Edwards*, *Member, IEEE*
Resilience and Systems Security Engineering
The MITRE Corporation
Colorado Springs, CO, USA
nedwards@mitre.org

Sang-Yoon Chang
Department of Computer Science
University of Colorado
Colorado Springs
Colorado Springs, CO, USA
schang2@uccs.edu

***Abstract*—Dynamic Random Access Memory (DRAM) is widely used for data storage and, when a computer system is in operation, the DRAM can contain sensitive information such as passwords and cryptographic keys. Therefore, the DRAM is a prime target for hardware-based cryptanalytic attacks. These attacks can be performed in the supply chain to capture default key mechanisms enabling a later cyber attack or predisposition the system to remote effects. Two prominent attack classes against memory are the Cold Boot attack which recovers the data from the DRAM even after a supposed power-down and Rowhammer attack which violates memory integrity by influencing the stored bits to flip. In this paper, we propose an on-chip technique that obfuscates the memory addresses and data and provides a fast detect-response to defend against these hardware-based security attacks on DRAM. We advance the prior hardware security research by making two contributions. First, the key material is detected and erased before the Cold Boot attacker can extract the memory data. Second, our solution is *on-chip* and does not require nor depend on additional hardware or software which are open to additional supply chain attack vectors. We analyze the efficacy of our scheme through circuit simulation and compare the results to the previous mitigation approaches based on DRAM write operations. Our simulation and analysis results show that purging key information used for address and data randomization can be achieved much faster and with lower power than with typical DRAM write techniques used for sanitizing memory content. We demonstrate through circuit simulation of the key register design a technique that clears key information within 2.4*ns* which is faster by more than two orders magnitude compared to typical DRAM write operations for 180*nm* technology, and with a power consumption of 0.15 picoWatts.**

***Index Terms*—DRAM, memory protection, supply chain protection, Cold Boot attack, Rowhammer attack**

I. Introduction

Dynamic Random Access Memory (DRAM) is a fundamental component within computing architectures and is used as main memory in many consumer electronic products including cell phones and computers to store encryption keys, program code and sensitive data. When a computer system is in operation, the DRAM can contain sensitive information such as passwords and cryptographic keys. DRAM is typically considered to be *volatile* as it is assumed that once power is removed from the DRAM, data contained within the memory is immediately erased. However due to the capacitive charge within each DRAM storage cell, a remnant voltage associated with the binary data state is maintained for a brief time after power is removed due to the intrinsic RC time constant. This remnant data storage presents a vulnerability to Cold Boot attack [1], which leverages and extends the memory cell retention time and provides an attacker the opportunity to recover memory content. Similarly, current DRAM architecture also makes it susceptible to Rowhammer attack which can allow a malicious actor elevated privileges and access to protected memory content through inter-row/cell coupling within the DRAM. Therefore, the DRAM is a prime target for hardware-based cryptanalytic attacks of data modification, exfiltration, or in some cases privilege escalation by local or remote malicious actors.

*The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author. © 2020 The MITRE Corporation. ALL RIGHTS RESERVED. Approved for Public Release; Distribution Unlimited. Public Release Case Number 20-0833

A. Threat Model

Both the Cold Boot and Rowhammer attacks can be executed through the supply chain to capture default key mechanisms enabling a later cyber attack or predisposition the system to remote effects. The Cold Boot attack which recovers the data from the DRAM even after a supposed power-down and Rowhammer attack which violates memory integrity by influencing the stored bits to flip.

A Rowhammer attack is more insidious as normal read operations to the DRAM are the triggering event and relies on intrinsic physical properties of DRAM. The threats presented by a Rowhammer attack include privilege escalation [2], cross-vm privilege escalation [3], [4], and have evolved to include attacks on Android based mobile devices and remotely executable attacks or root exploit [5]. The attack has demonstrated efficacy in utilizing JavaScript through WebGL on a Graphic Processing Unit (GPU) [6], additional remote attacks on hardware [7], [8], and remote direct memory access (RDMA) [9]. Extensibility of Rowhammer attack to other types of memory including multi-level cell (MLC) NAND Flash technology used in solid state drives (SSD) has been shown by [10] and [11]. With regards to supply chain, an attacker would intercept the computer system and analyze it for susceptibility to Rowhammer. If the system is susceptible then the attacker would leverage that for privilege escalation and plant an embedded or OS-level trojan or would use the system's vulnerability to remotely execute a cyber attack with greater effect.

The original Cold Boot attack in [1] from 2008 demonstrated the ability to effectively recover encryption key information associated with various encryption schemes including DES, AES, and RSA private keys using commercial, off-the-shelf multipurpose dust spray (cryogenic properties). During a supply chain intercept, an attacker would then recover the initial key material normally stored in the protected non-volatile memory and can be used as a technical basis for constructing later cyber attacks on deployed computer systems. This early work focused on DDR and DDR2 SDRAM memory technology, and requires a computer system to be intercepted in the supply chain then subsequently powered on which loads initialization key material into memory. This is the case with modern computer systems using trusted boot, TPMs or other UEFI cryptographic mechanisms. Recent and ongoing research indicate that Cold Boot vulnerabilities still exist and are still a relevant concern. Newer memory technology has also been shown to be susceptible to Cold Boot attacks [12]–[15] and have even been shown successful to recover encryption keys from Android OS [16], [17].

B. Our Contributions

In this paper, we advance the prior hardware security research by making two contributions. First is a novel technique for mitigating Cold Boot and Rowhammer attacks by an on-chip obfuscation technique that provides a fast detect-response to defend against these hardware-based security or supply chain attacks on DRAM. Second, our solution is *on-chip* and does not require nor depend on additional hardware or software which are open to additional supply chain attack vectors and instead relies on integration of the schema during chip design and manufacturing.

Various mitigation techniques have been proposed for these vulnerabilities but these techniques rely on system level modifications, including the addition or modification of hardware and/or software, while the associated overhead impacts system performance and chip area. Industry has produced a few technologies at the computer architecture level that help reduce the risks of these attacks, such as Address Space Layout Randomization (ASLR) or a chipset such as Intel's ME that scrambles the memory content external to the DRAM so that it provides a balance of electrical current utilization across the DRAM, yet both techniques are external to DRAM modules and do not address risks of physical access to the memory.

The novelty of our approach lies in on-chip scrambling of the system-provided address and the on-chip data scrambling, utilizing a randomization key stored on the DRAM chip but external to the actual memory array. The randomization is unique per every power-on event to mitigate the nature of Rowhammer tactics. The on-chip mechanism is also designed using a circuit layout that allows for fast detection of a power glitch-response and clearing of data to mitigate Cold Boot variety of attacks. The terms *randomization key* or *key* are used within this work to indicate the key used for address and data scrambling at the DRAM chip level and is distinguished from the term encryption which is used for other cryptographic measures that may occur outside of the memory chip.

Our proposed methodology is intended to support a defense-in-depth posture within a computing system. By combining information in the on-chip key register with data being written to and from the DRAM, our scheme can provide confidentiality of the memory data against a system-insider or supply chain threat accessing the memory even if the application-layer encryption is compromised. There are also opportunities to combine with other system level security technologies.

The organization of this paper is as follows: an overview of DRAM architecture and relevant circuits is presented in Section II followed by descriptions of Cold Boot and Rowhammer attacks and existing mitigation techniques in Section III. Our proposed mitigation technique is presented in Section IV, followed by a brief discussion on the physical implementation, or layout, of the key register. Circuit simulation and intrinsic delays of the key register are presented in Section V, followed by simulation of Cold Boot attack detection and key purge timing results in Section VI. The efficacy of our proposed mitigation solution on Rowhammer attack is presented in Section VII, followed by discussion of integrating the key register on-chip in Section VIII and conclusion in Section IX.

II. DRAM Architecture

A DRAM consists of an array of memory cells in which data is stored; an address decoder (row and column selection); and peripheral circuitry of sense amplifiers for data refresh as well as input/output buffers for writing and reading data to and from the memory cells. For the purpose of understanding the physical nature of the classes of memory attacks and mitigations, a brief overview of the memory array circuitry and address decoder architecture is provided. It should be noted that the term DRAM is a generic term and all references to various generations of double data rate (DDR) DRAM are denoted as synchronous DRAM (SDRAM).

The memory array contains numerous memory cell structures as shown in the upper left corner of Figure 1 comprised of a single transistor and a capacitor for storing the charge associated with a binary data state. When a voltage is applied to the wordline (WL) select signal, the transistor turns on to allow data to be written to the memory cell capacitor based on the bitline (BL) voltage representing a binary state of "0" or "1". When the wordline select signal is turned off, the transistor isolates the storage capacitor at which point the information is stored as an electric charge on the capacitor. When data is read from the memory cell, the wordline and associated transistor is turned on and the charge stored on the capacitor is transferred to the bit line. It should be noted that a sense amplifier (SA) also plays a key role in refreshing data within the DRAM with a standard refresh time of 64 milliseconds. This refresh time ensures that sufficient charge is available to detect the charge state.

The address decoding for accessing individual memory cells is performed in two parts: through row decoding and column decoding. Rows and columns are selected through address pins on the DRAM which are multiplexed and decoded because the number of cells residing on a given wordline are quite large. The purpose of the row decoder (Figure 2) is to turn on the desired wordline and access a row of cells for read or write operations. Column decoders operate similarly but select which bitlines are directed to the input/output buffers and data to be read or written to the memory. Internal critical elements to the DRAM are sense amplifiers which provide the refreshing of the data state in the memory cells through amplification of signal on the bitlines.

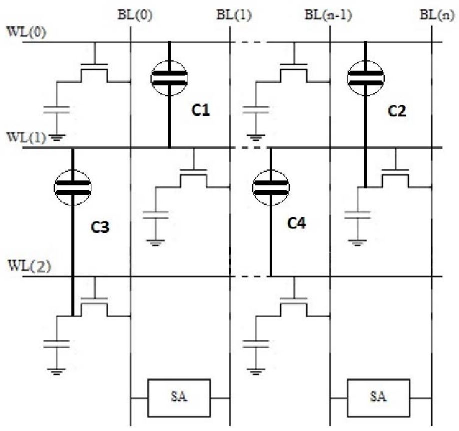


Fig. 1. DRAM Memory Array showing wordline (WL) and Bitline (BL) control signals with each memory cell, associated storage capacitor, and parasitic coupling capacitance (e.g. C1-C4).

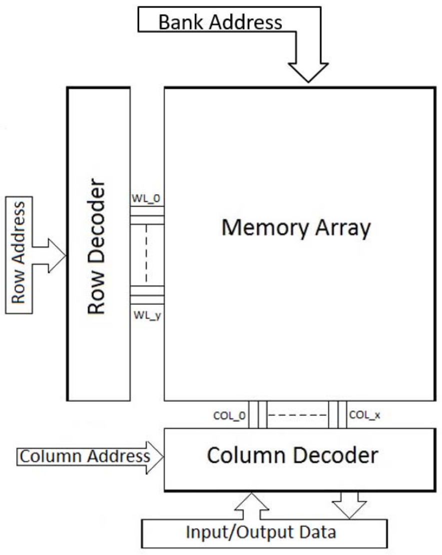


Fig. 2. Typical DRAM addressing schema.

III. Cold Boot and Rowhammer Attacks

A. Cold Boot Attack

Halderman et al. [1] demonstrated the ability to effectively increase the capacitive retention time of DRAM memory cells using commercial, off-the-shelf multipurpose dust spray (cryogenic properties) to chill memory in use. For the original research, the memory was preloaded with precomputed key schedules associated with various encryption schemes (e.g. DES, AES, and RSA). Identification of keys within memory was accomplished through a search for blocks within memory which exhibited characteristics associated with the combinatorial properties of a valid key schedule. Since the original research in 2008, SDRAM architectures have evolved to DDR3 and DDR4 with increased speed performance. More recent research has focused on validation of the original results and extrapolation of the Cold Boot attack to these higher performance parts of DDR3 and DDR4 SDRAM [12]–[15] as well as additional cryptographic key structures [18] or using FPGA-accelerated key search mechanisms [19].

B. Rowhammer Attack

The Rowhammer attack was introduced while investigating disturbance-induced errors in DRAM and discovered widespread susceptibility with 139K adjacent row activations [20]. A Rowhammer attack relies on intrinsic physical properties and capacitive coupling mechanisms of DRAM during normal read operations. Assuming wordline WL(1) in Figure 1 is chosen as the target, the attacker then begins continuously accessing an adjacent row; either WL(0) or WL(2) for a singlesided attack, or, both WL(0) and WL(2) for a double-sided attack. The vulnerability within DRAM which enables this attack is based on the physical proximity of adjacent wordlines and memory cells within the memory array and associated capacitive coupling mechanisms (C1, C2, C3, C4 of Figure 1). Through repetitive row activations “hammering”, targeted DRAM cells in proximity to the activated aggressor wordline collects a small amount of charge from each row activation and may enable data state changes within the memory cell resulting in data integrity issues. The actual coupling mechanisms between adjacent rows and adjacent cells is quite complex and is highly dependent on the specific memory fabrication technology used to create the DRAM and the physical proximity of rows and cells. Newer DRAM technologies tend to exacerbate the rowhammer problem due to increased coupling capacitance associated with smaller geometries, emphasizing the need for mitigation.

C. Related Work in Mitigation Techniques

Existing techniques for mitigation of Rowhammer and Cold Boot have significant impact to circuit Size, Weight, Power, and Cost (SWaP-C) as well as latency of purge time. Some mitigations also require system-level architectural changes. Table I below provides a summary of the mitigation techniques. A challenge in comparison of techniques is that most do not consider SWaP-C, and the performance times are relative software benchmarking rather than clock cycle performance metrics which allow for a normalized comparison of hardware-based solutions. While some solutions also address the key issues of the Cold Boot and Rowhammer attacks they add complexity or create subsequent system constraints:

TABLE I
SUMMARY OF MEMORY ATTACK MITIGATION TECHNIQUES

Mitigation Technique	**Performance Time of technique**	**Additional info**
Cold Boot Mitigations		
Overwrite [21]	Comparable to cache miss latency (estimate)	–
uProcessor SW encryption AES [22]	21.081 ns (AES128)	70.2 clock cycles, 3.33GHz
uProcessor HW encryption [23]	20.42 ns (AES128)	4.2 clock cycles per byte, 3.33GHz
Memory scramble [14], [15]	System device speed	–
Monitor state transitions [24], [25]	Comparable to cache miss latency (estimate)	–
Rowhammer Mitigations		
Circuit Modification [4]	Circuit layout to reduce capacitive coupling	Increased area
ECC [4]	Reduced performance	Increased area
Reduce row refresh freq [2]	Reduced performance	Increased area
Increased row refresh freq [26]–[28]	Increased on-chip area and power overheads	–
Targeted row refresh [4], [29]–[32]	Minimal impact to performance, increased on-chip area and power overheads for additional counters	–

IV. Proposed Scheme - On-chip Address and Data Randomization with Transient Power Detection

In this paper, we propose a novel mitigation technique to Cold Boot attacks through implementation of on-chip address and data randomization which scrambles the data stored in the memory. The method proposed in this work relies upon storing the *scrambling key* in a register fabricated on the DRAM chip, but external to the actual memory array, thus effectively hiding the key from any external access and allowing for quick clearing of the key from the register.

The proposed method for obfuscating memory content relies on randomizing address and data utilizing an on-chip generated key for scrambling row, column, and bank addresses as well as data, as shown in Figure 3. The terms $S(KEY_R, Row)$, $S(KEY_C, Column)$, $S(KEY_B, Bank)$ and $S(KED_D, Data)$ represent scramble function of the row, column, bank addresses and data buffer with their respective keys KEY_R, KEY_C, KEY_B, and KEY_D. The actual key generation would be based on true random or pseudorandom generators using existing semiconductor related noise techniques which are beyond the scope of this paper. The stored key resides in a

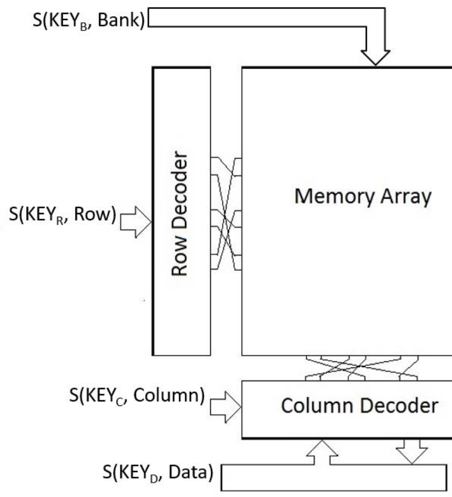


Fig. 3. Address and data randomization schema with the scramble key operating on address and data.

static register external to the actual DRAM array. Storing the key in such a manner allows for rapid clearing (less than $2.4ns$) of the stored key upon power event detection with the circuit shown in Figure 5, and requires substantially less power when compared to basic writing of DRAM cells to a known value for clearing memory cell contents.

While the ideal scrambling mechanism for the memory would be equivalent to a one time pad, such implementation would require a single bit of encryption for each memory cell (the length of the key is the same as the memory being protected) and would limit its practicality since the overhead on die area effectively doubles the die size. Yet another method of scrambling the memory content would consist of implementing a secure cryptographic encryption (e.g., AES) on the DRAM chip, but the overhead associated with both the chip area requirement as well as the overhead with scrambling would be too large, impacting both chip cost and timing performance. With this overhead in mind, a minimum key length is proposed based on DRAM architecture. The example key size of 44 bits is based on integration of latch circuitry into the existing address and data buffers in order to save valuable chip area, a critical cost factor in DRAM chip manufacturing. If a larger key space is desired, the additional key size can be implemented on-chip, at the expense of increasing chip area.

A. DRAM Write Timing and Power Consumption

One method for mitigating Cold Boot attack is to effectively purge data from the DRAM by writing, or clearing, the data of interest before it can be captured and stored for analysis. Utilizing the timing parameters in Table II, the write time and associated power were derived from a Micron Technology datasheet [33] and the MICRON DDR3 SDRAM System-Power Calculator [34], [35] for writing all memory cells, a single row of memory cells, and a single column (4 memory

TABLE II
DRAM TIMING PARAMETERS

Speed Grade	Data Rate (MT/s)	Target ^tRCD - ^tRP - CL	^tRCD (ns)	^tRP (ns)	^tCL (ns)
-093	2133	14-14-14	13.13	13.13	13.13

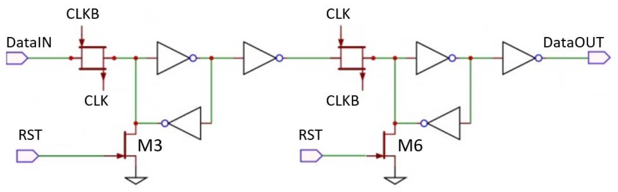


Fig. 4. Typical scramble key register cell, 44 total in schema.

cells) with results shown in Table III based on a typical 4Gb DDR3 DRAM in three I/O configurations: x4, x8 and x16.

B. Key Register

Unlike other proposed techniques which store the encryption key in actual DRAM memory cells, the proposed method relies on storing the randomization key in a register which is comprised of 2 data latches, is shown in Figure 4. All latches within the register are reset by the active RST signal from the power detect circuit (Figure 5) driving the gates of transistors M3 and M6. The measurement of time to purge is measured from the rising edge of the RST signal at half the voltage supply level to the output of the register being driven to 0.01 Volts. The power detect circuit uses a CMOS differential amplifier with additional amplification stages in order to generate the RST signal. Additionally, the specified power for purging data is associated with the current required to drive the gates of transistors M3 and M6, for all 44 register cells. Simulation results of the power detect circuit with two stages of buffering between RESET and RST are shown in are shown in Figure 8.

The individual register cells are serially connected such that the output of one cell drives the input of the next, forming a shift register which enables a serial load of the key from the first cell in the register. Using the maximum number of total bits to account for row and column addressing, a register consisting of 44 bits is created, with the output of

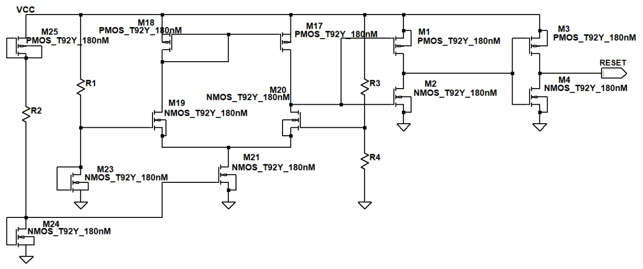


Fig. 5. Transient power detect circuit.

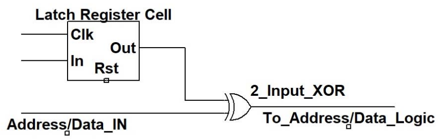


Fig. 6. Typical XOR gate used o provide the scramble obfuscation when combining a key register cell DataOUT and memory I/O data .

the register randomizing row addresses with KEY_R, column addresses with KEY_C, and I/O with KEY_D, as shown in Figure 3. Bank addresses are also randomized with key register information, KEY_B. The overall output of the key register bits are connected to XOR gates as shown in Figure 6 which provides the data obfuscation yet only imposes a 1.5ns delay during memory R/W operations and is significantly faster than the performance hit of encryption operations on the $180nm$ technology. With newer technology (e.g. $22nm$) the speed up will scale in comparison.

C. Key Register Physical Design

As previously noted, a primary constraint of the on-chip key register is the chip area requirement cost, which translates directly to the price of the DRAM. In order to quantify the cost of security associated with each bit of the on-chip key register, a physical layout of the key register was additionally created using VLSI design software. Rules governing the spacing and overlaps of the polygons associated with the different layers within the layout are specific to a given semiconductor technology. The theoretical minimum size limit of any layout is based on using minimum design rules as well as efficient layout utilizing all layers of the technology. The layout of the register cell, while not necessarily optimized, is representative of layout associated with the $180nm$ technology and requires an area of $110.22\mu m^2$, or a total required chip area of $4850\mu m^2$ for all 44 register cells. Comparing this value to a chip size of $640mm^2$ based on a 4Gb DDR3 DRAM also designed in a $130nm$ technology, yields a die area increase of 1.7222x10^{-5} percent. It should be noted that current, state of the art DRAM technologies will allow for a scalable, but relative decrease in die area.

V. SIMULATION AND PERFORMANCE

Timing simulations for purging the 44-bit key register were obtained using the LTSpice circuit simulator, from Linear Technology, Inc. with an open-source, $180nm$ technology library. The simulation methodology included loading all '1's, or the high data state, into the register and then applying an active high to the reset signal which discharged the data from all latches within the register. The relevant parameters for this analysis include the required time and associated power to discharge the stored state within the register. The rationale for analyzing the required power is derived from a threat scenario in which the DRAM is removed from the system and an on-chip detect circuit senses the loss of power. Under this scenario, any voltage required for purging must be provided by

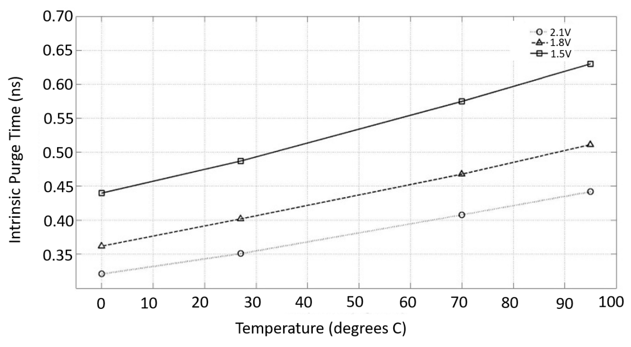


Fig. 7. Key register purge time vs. temperature per supply voltage.

TABLE III
REGISTER PURGE TIMES AND POWER COMPARED TO DRAM WRITES

Data Purge Method	Key Register Reset (44 bits)	DRAM Write (4 bits)	DRAM Write (1 row)	DRAM Write (all rows)
Number of Bits	44	4	8192	4294967296
Purge Time (sec)	6.3x10e-10	3.0x10e-8	7.7x10e-6	4
Normalized Purge Time	1	47.6	12222	6.349x109
Power (Watts)	3.68x10e-12	300x10e-9	2.4 x10e-6	1.05
Normalized Power	1	81,500	652,173	285x10e9

on-chip capacitance. This value of capacitance, however, does not account for the actual power requirements of the detect circuit in Figure 5 which is used to model interaction with the key register. Design optimization and characterization of the detect circuit is left for future research.

Simulation of the latch and register design was performed over voltage and temperature corners, which is an industry standard practice for ensuring functionality and performance. Voltage corners for simulation were 1.5 and 2.1 volts, with a nominal value of 1.8 volts, while temperature corners were run at 0, 70, and 95 degrees Celsius, with a nominal value of 27 degrees Celsius. Results for the purge time over the simulation corners are shown in Figures 7. The purge times shown are based on the noted voltage and temperature simulation corners and apply only to the intrinsic key register. An additional simulation is described in Section VI which models the key register behavior under the influence Cold Boot attack conditions.

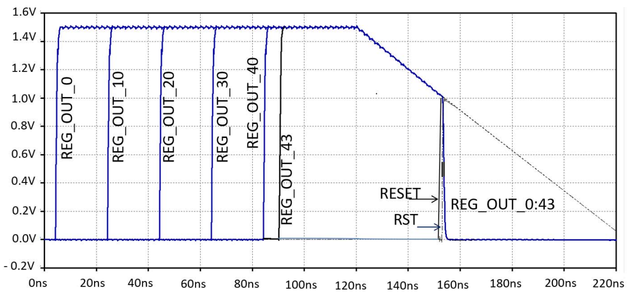
The data shown in Figure 7 reflects the key purge time as a function of the operating temperature, with the maximum purge time of 630 picoseconds occurring at 95 degrees Celsius. The data represented indicates the linearity of power over the operating temperature for a given operating voltage. Figure 7 also indicates the simulated operating bounds of power and purge time over temperature and voltage. These results are based on $180nm$ process models, which are an older technology. Newer technologies can yield faster purge times and reduced power, depending on the specific technology and design.

The results in Table III for typical DRAM write time and power are normalized in scale to our key register purge time and power. The Key Register Reset values were chosen from the slowest time value and highest power value. The results indicate the Key Register purge is significantly faster and consumes substantially less power than the standard DRAM writes used for purging data, and is effective at attack temperatures.

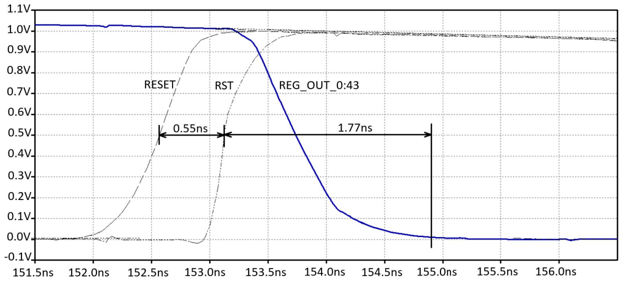
VI. COLD BOOT ATTACK DETECTION AND KEY PURGE

In the event that a Cold Boot class attack occurs in which the DRAM is cooled and removed from the system, a means of detecting the removal from the initial system is required in order to signal the key register to purge the key content before remaining power is dissipated. The detect method assumed here relies on a loss of power on the DRAM chip through either system power down or extraction of the DRAM from the system while in operation. The example power detect circuit shown in Figure 5 was created in order to detect the power level within the DRAM chip and was adjusted such that the RST signal would be asserted when the power supply (VCC) falls below 1.00 volts as can be seen in Figure 8. This value was chosen to provide sufficient turn-on voltage of the n-channel pull-down devices (M3 and M6 in Figure 4) such that any charge within the register is purged. Simulation of the power detect and key register purging was performed with the power supply ramping down from 1.5 V at a temperature of -25 degrees Celsius which is in the range of Difluoroethane, a chemical typically used in AirDust and whose cryogenic properties can be used for Cold Boot attacks.

The waveforms shown in Figure 8 include the key register reset, RST, samples of the key register, REG_OUT_0 to 42, and the final register cell, REG_OUT_43. The simulation was run clocking all "1's", or high data state, into all 44 register cells. The rising edge of REG_OUT_43 at $90ns$ reflects the final cell being written to the high data state. The DRAM is then powered down starting at $120ns$. At $152ns$, the output of the power detect circuit, RESET, goes high which, after two inverter delays, then drives the RST signal to the key register high at $153ns$. Assertion of the RST signal purges the prior written high state of all 44 bits, as shown at $154.9ns$, with all register data states, REG_OUT_0 to 43, discharging to the ground state. All register bits purge at the same time and decay rate, and are overlapping in the plot. The key register cells have similar timing, but delays associated with routing and parasitic elements of the RST signal can be expected, resulting in minor timing differences between key register bits. The simulation results indicate that a combined delay for the power detect and key register is $2.32ns$ with a consumed power of $0.15pW$, with both values being considerably less than the values associated with overwriting the data state of the DRAM cells to a known value, or erasing the DRAM, as shown in Table III.



(a) Overall simulation



(b) Power-down event (attack) and key purge response time

Fig. 8. Power-down detect simulation.

VII. The Rowhammer Attack and Mitigation

In order for an attacker to execute an attack against the Rowhammer vulnerability, he must have knowledge of how the logical row address maps to a physical row in the DRAM memory array. This mapping must be done in order to identify required memory addresses which will select rows adjacent to the targeted row. While it is possible to determine the row mapping to address through brute force methods [2], the act of doing so may corrupt any desired memory content. Thus, the most efficient method of implementing a Rowhammer attack is with a-priori knowledge of the mapping [3], [6].

Our proposed mitigation technique relies on a power down event between the time an attacker determines the correlation between rows and cells, and the use of the DRAM. If the DRAM is powered down and then powered back up, using our proposed mitigation method, a new key will be loaded into the key register and a new scramble of row addresses will have occurred, thereby corrupting the prior row and cell correlation. If the attacker wishes to launch a Rowhammer attack on the DRAM with the new row scramble, he will need to once again perform a brute force correlation. In addition, the scrambling helps disassociate the coupling between rows (required physics of the attack) by providing random physical locations which separate system-level data that would normally be adjacent.

VIII. Discussion

While this body of work presented is focused on a novel memory data and address scramble scheme as well as on-chip and purging-efficient key protections to mitigate Cold Boot and Rowhammer attacks, it is also just as important to discuss the integration considerations and practical use in a targeted computer system. This section describes relevant challenges and considerations of implementing the proposed scheme in such systems, an introductory discussion on key generation for the proposed scheme, and chip layout considerations.

A. Functional Integration

The use of our proposed memory data and address scramble scheme is compatible with any existing off-chip data encryption and address obfuscation provided either through discrete memory controllers or as an integrated memory controller on a microprocessor. The main reason for the compatibility is that the proposed scheme maintains the computer systems architecture principle of self-encapsulation, which minimizes exposures of interfaces and reduces the dependencies on external subsystem components only to those which are necessary. While there may be benefits to do so for the purpose of having other system-level trigger events to purge the old key and generate a new one, or to expose a read-only interface to ingest the generated key as part of system-level cryptographic purposes such as an HMAC initialization vector or cryptographic seed, the external exposure must be carefully evaluated for risk or impact on the memory protection.

Since the proposed mechanism can be designed to be entirely self-encapsulated, the integration of it has only a few system level considerations. The functional integration considerations are foundationally based on the system-level sequence of operations which can be generally described as: 1. System powers on; 2. A minimum threshold voltage is obtained at the scramble key generation hardware elements; 3. Key is successfully generated and shifted into the scramble register; 4. DRAM scramble hardware is appropriately switched and automatically configured based on the logic values in the register (much like an instruction decoder logic hardware within instruction set architecture); 5. DRAM is ready to be used by the system. By understanding the sequence of operations, key integration considerations emerge: timing and latency until first use of DRAM, threshold voltages necessary to enable critical functions, digital logic gates that enable system integration, indication to the system that DRAM is ready to receive data, and indication that DRAM security features and key purge have been activated along with a state-based model toward system recovery.

One of the important integration details to consider is interaction with the existing DRAM design and manufacturing process that manages memory cell defects post-manufacturing. DRAM is designed with additional rows of cells and columns to replace fabrication related defective rows and columns, both of which are enabled by laser ablation of fuses or one-time programmable fuses [22]. The fuses are set as a result of the pre-fuse test that identifies the defects and add no latency to the power-on sequence. Implementation of the proposed method requires the ability to disable address scrambling during pre-fuse test after fabrication of the DRAM in order to accommodate row and column repair. One method of disabling the schema hardware with minimal impact on current manufacturing processes includes a fuse circuit which disables the scramble circuitry by default when the fuse is intact, and would enable scheme when the fuse is blown after the normal row and column defect replacement. It should be noted that the former method is a more permanent method, and prevents any possibility of intentional (via attack) or accidental disabling of scramble scheme once the fuse is blown.

Another integration consideration of the proposed method is the impact on other techniques for mitigation of the memory attacks. Any system-level method which tracks the frequency of row activations to help mitigate the Rowhammer attack, such as those implemented in the memory controllers or chipsets (e.g. targeted row refresh), will be impeded by the on-chip address scramble. In addition, there are other considerations on the measurement of memory when used with a trusted boot environment or trusted execution as the measurements may not necessarily be as deterministic as these root of trust methods assume. Further investigation of the proposed scramble scheme's impact on other system-level security features will be performed in future work.

B. Key Generation and Entropy

The emphasis of this work is on the implementation of a key register and purging of the scramble key, yet it is important to describe the important characteristics of the key generation to establish an overall system security. The key generation should be from high entropic sources where the measured Shannon entropy is nearing the probability of chance as discussed in existing research or industry solutions for true random number generation [36]. Other considerations need to include entropy gains resulting from key size as well as protection and transportation of the key from its generation to the hardware register that will utilize it.

While the details of design considerations and process for generating the scramble key used in this proposed scheme will be addressed in future work, it is important to discuss possible methods and benefits of on-chip versus off-chip key generation. One possible method of key generation includes using on-chip (DRAM) noise sources with sufficient entropy. These might include reverse biased transistors, thermal noise, or a combination of multiple mechanisms [36] which meet the NIST standards for pseudorandom number generation. By generating the key on chip, the key value is never exposed off chip unless a designed external interface is desired such as that of commercial trusted platform modules (TPM) over a circuit board low pin count (LPC) interface. Another solution to key generation relies on generating the key off-chip and providing a serial or parallel load into the key register, using the DRAM I/O pins. The action of loading the key from an external source, however, is not within normal system behavior and would require timing and functional modifications to the memory system design.

C. Layout Considerations

In this work, we created a sample physical layout of the register cell. The actual integration of the data and address scrambling requires the implementation of XOR logic gates which are the combinatorial logic hardware that takes the values in the scramble key register and directly enables the scrambling permutation. While these XOR cells will need additional die area, efficiently designed cells can be implemented with close proximity to the key register cells although they may be distributed in order to minimize area impact. Another possible implementation would use the values in the key register as a seed for a linear feedback shift register (LFSR). LFSRs are a standard hardware technique for generating pseudo random binary sequences (PRBS). By integrating the key register with the LFSR, chip area would be conserved. Additional routing of signals associated with the feedback lines would be required, but would be of minimal impact to chip area.

D. Additional Discussion

On-chip scrambling of data and address can also mitigate preferred states in memory cells which may occur when the same data state is stored in the same memory cell for an extended period of time [37], [38]. In implementing both data and address scrambling on chip for both substitution and transposition, we realize Shannon's confusion and diffusion properties, which increases the work effort required to reverse engineer and identify the data contained in the memory array. In addition to the randomization key scrambling the address and the data on chip, the data entering the chip could have used a separate key for encryption using application-layer algorithms such as AES for defense in depth.

IX. Conclusion

DRAM is a potential target of data attack due to its inherent nature of storing important system and security-critical information. This information can remain in memory for a period of time even after power has been removed (yielding Cold Boot attack), or the control of bits in certain parts of the memory can overspill and influence the other parts of the memory (yielding Rowhammer attack). Though this phenomena has been known and researched, methods of mitigation have been limited. This paper presents a method of mitigating Cold Boot attack and Rowhammer attack. Our scheme constructs the key randomization on chip for increased security especially against an attacker having physical access to the chip/device and outperforms the existing designs by two orders of magnitude or more. We achieve such properties while minimizing the overhead and providing the design so that it only adds minimal size to the chip.

References

[1] J. A. Halderman, S. D. Schoen *et al.*, "Lest we remember: Cold-boot attacks on encryption keys," *Commun. ACM*, vol. 52, no. 5, pp. 91–98, May 2009. [Online]. Available: http://doi.acm.org/10.1145/1506409.1506429

[2] M. Seaborn and T. Dullien, "Exploiting the DRAM rowhammer bug to gain kernel privileges," *Black Hat*, vol. 15, 2015.

[3] Y. Xiao, X. Zhang *et al.*, "One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 19–35. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/xiao

[4] K. Razavi, B. Gras *et al.*, "Flip feng shui: Hammering a needle in the software stack," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 1–18. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/razavi

[5] V. van der Veen, Y. Fratantonio *et al.*, "Drammer: Deterministic rowhammer attacks on mobile platforms," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1675–1689. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978406

[6] P. Frigo, C. Giuffrida *et al.*, "Grand pwning unit: Accelerating microarchitectural attacks with the gpu," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 195–210.

[7] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer. js: A remote software-induced fault attack in javascript," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 300–321.

[8] A. Tatar, R. K. Konoth *et al.*, "Throwhammer: Rowhammer attacks over the network and defenses," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, 2018, pp. 213–226. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/tatar

[9] M. Lipp, M. T. Aga *et al.*, "Nethammer: Inducing rowhammer faults through network requests," *CoRR*, vol. abs/1805.04956, 2018. [Online]. Available: http://arxiv.org/abs/1805.04956

[10] Y. Cai, S. Ghose *et al.*, "Characterizing, exploiting, and mitigating vulnerabilities in MLC NAND flash memory programming," *CoRR*, vol. abs/1805.03291, 2018. [Online]. Available: http://arxiv.org/abs/1805.03291

[11] A. Kurmus, N. Ioannou *et al.*, "From random block corruption to privilege escalation: A filesystem attack vector for rowhammer-like attacks," in *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. Vancouver, BC: USENIX Association, 2017. [Online]. Available: https://www.usenix.org/conference/woot17/workshop-program/presentation/kurmus

[12] M. Gruhn and T. Müller, "On the practicability of cold boot attacks," in *Proceedings of the 2013 International Conference on Availability, Reliability and Security*, ser. ARES '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 390–397. [Online]. Available: https://doi.org/10.1109/ARES.2013.52

[13] S. Lindenlauf, H. Höfken, and M. Schuba, "Cold boot attacks on ddr2 and ddr3 sdram," in *Proceedings of the 2015 10th International Conference on Availability, Reliability and Security*, ser. ARES '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 287–292. [Online]. Available: https://doi.org/10.1109/ARES.2015.28

[14] J. Bauer, M. Gruhn, and F. C. Freiling, "Lest we forget: Cold-boot attacks on scrambled ddr3 memory," *Digital Investigation*, vol. 16, pp. S65–S74, 03 2016, doi = 10.1016/j.diin.2016.01.009.

[15] S. F. Yitbarek, M. T. Aga *et al.*, "Cold boot attacks are still hot: Security analysis of memory scramblers in modern processors," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2017, pp. 313–324.

[16] T. Müller, F. C. Freiling, and A. Dewald, "Tresor runs encryption securely outside ram," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028084

[17] C. Hilgers, H. Macht *et al.*, "Post-mortem memory analysis of cold-booted android devices," in *Proceedings of the 2014 Eighth International Conference on IT Security Incident Management & IT Forensics*, ser. IMF '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 62–75. [Online]. Available: http://dx.doi.org/10.1109/IMF.2014.8

[18] C. Maartmann-Moe, S. E. Thorkildsen, and A. fRnes, "The persistence of memory: Forensic identification and extraction of cryptographic keys," *Digit. Investig.*, vol. 6, pp. S132–S140, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2009.06.002

[19] H. Riebler, T. Kenter *et al.*, "Fpga-accelerated key search for cold-boot attacks against aes," in *2013 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2013, pp. 386–389.

[20] Y. Kim, R. Daly *et al.*, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Proceeding of the 41st Annual International Symposium on Computer Architecture*, ser. ISCA '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 361–372. [Online]. Available: http://dl.acm.org/citation.cfm?id=2665671.2665726

[21] M. C. Parris, "DRAM security erase," Apr. 15 2014, US Patent 8,699,263.

[22] B. Gladman, "Implementation experience with aes candidate algorithms," in *Second AES Conference*, 1999.

[23] K. D. Akdemir, M. Dixon *et al.*, "Breakthrough AES performance with intel ® AES new instructions." Intel Corporation, 2010.

[24] P. McGregor, T. Hollebeek *et al.*, "Braving the cold: New methods for preventing cold boot attacks on encryption keys," in *Black Hat Security Conference*, 2008.

[25] N. P. Adams, M. S. Brown *et al.*, "System and method for hindering a cold boot attack," Feb. 11 2014, US Patent 8,650,639.

[26] S. M. Seyedzadeh, A. K. Jones, and R. Melhem, "Counter-based tree structure for row hammering mitigation in DRAM," *IEEE Computer Architecture Letters*, vol. 16, no. 1, pp. 18–21, 2016.

[27] ——, "Mitigating wordline crosstalk using adaptive trees of counters," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ser. ISCA '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 612–623. [Online]. Available: https://doi.org/10.1109/ISCA.2018.00057

[28] S. M. Seyedzadeh Delcheh, "Architectural techniques for disturbance mitigation in future memory systems," Ph.D. dissertation, University of Pittsburgh, 2019.

[29] K. S. Bains and J. B. Halbert, "Distributed row hammer tracking," Mar. 29 2016, US Patent 9,299,400.

[30] D. E. Fisch and W. C. Plants, "DRAM adjacent row disturb mitigation," Nov. 7 2017, US Patent 9,812,185.

[31] J.-M. Oh and H.-y. Song, "Refresh controller and memory device including the same," May 15 2018, US Patent 9,972,377.

[32] D. S. Kim and J. I. Kim, "Refresh control device and semiconductor device including the same," Nov. 14 2017, US Patent 9,818,469.

[33] "Micron 4Gb DDR3 Datasheet," Micron Technology Inc, Boise, ID, USA, 2017. [Online]. Available: www.micron.com/∼/media/documents/products/datasheet/dram/ddr3/4gb_ddr3_sdram.pdf

[34] "DDR3 SDRAM system-power calculator," Micron Technology Inc, Boise, ID, USA, 2018. [Online]. Available: https://www.micron.com/-/media/client/global/documents/products/power-calculator/ddr3_ddr3l_power_calc.xlsm

[35] "Calculating memory system power for DDR3," Micron Technology Inc, Boise, ID, USA, 2007, TN-41-01.

[36] G. Taylor and G. Cox, "Digital randomness: Behind intel's new random-number generator," in *IEEE Spectrum*, 2011.

[37] P. Gutmann, "Secure deletion of data from magnetic and solid-state memory," in *Proceedings of the 6th Conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6*, ser. SSYM'96, 1996.

[38] ——, "Data remanence in semiconductor devices," in *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, ser. SSYM'01, 2001.

Appendix

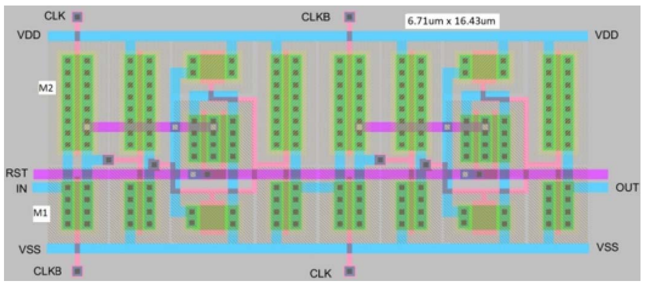


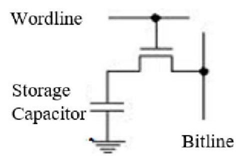
Fig. 9. Physical layout (scaled) of the 2-bit key register cell, typical of proposed architecture. A pmos transistor (M2) is formed when polysilicon (pink) crosses a P-active region (green with dotted background) and an nmos transistor (M1) is formed when polysilicon crosses a N-active region (green with striped background). Connection is made between the active regions using metal-1 (light blue) and source-drain contacts (black squares). Metal-1 can also connect to polysilicon using a metal-1 to poly contact (gray square surrounding a black square). Another layer of metal, designated as metal-2 (purple) allows connection between metal-1 lines using VIAs (white squares).

TABLE IV
DRAM CONFIGURATIONS

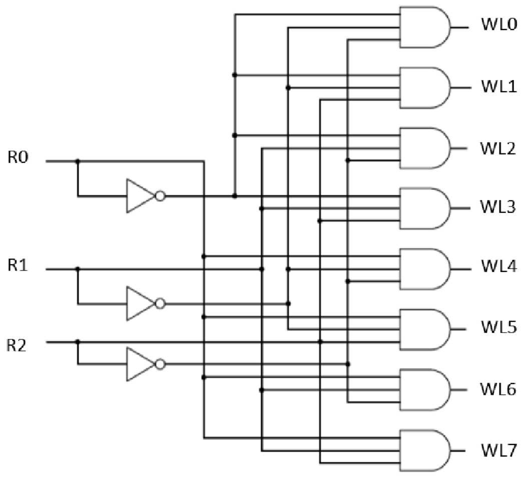
Parameter	**1G x 4**	**512M x 8**	**256M x 16**
Configuration	128M x 4 x 8 banks	64M x 8 x 8 banks	32M x16 x 8 banks
I/Os	4	8	16
Row Address	64K (A[15:0])	64K (A[15:0])	32K (A[14:0])
Bank Address	8 (BA[2:0])	8 (BA[2:0])	8 (BA[2:0])
Column Address	2K (A[11,9:0])	1K (A[9:0])	1K (A[9:0])
Total Key Register bits needed	34	37	44

TABLE V
KEY REGISTER PURGE TIMES AND REQUIRED POWER

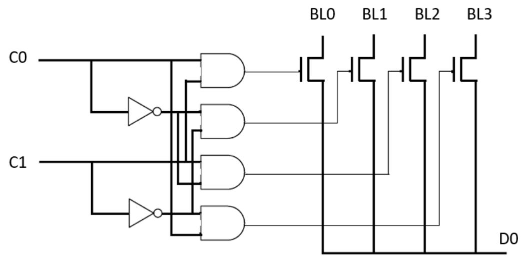
Voltage	Temp	Purge Time (vdd/2,0.01v)	Power (Watts)
2.1 V	95 C	0.442 ns	3.683E-12
2.1 V	70 C	0.408 ns	3.675E-12
2.1 V	27 C	0.351 ns	3.674E-12
2.1 V	0.0 C	0.321 ns	3.667E-12
1.8 V	95 C	0.511 ns	2.665E-12
1.8 V	70 C	0.468 ns	2.644E-12
1.8 V	27 C	0.402 ns	2.635E-12
1.8 V	0 C	0.362 ns	2.641E-12
1.5 V	95 C	0.630 ns	1.786E-12
1.5 V	70 C	0.575 ns	1.770E-12
1.5 V	27 C	0.487 ns	1.752E-12
1.5 V	0 C	0.440 ns	1.755E-12



(a) DRAM Memory Cell



(b) Row decoder



(c) Column decoder and bitline multiplexor

Fig. 10. DRAM architectural subcircuits.