DDR Memory Errors caused by Row Hammer

What you don't know CAN hurt you, why this failure mechanism is important to understand

Barbara Aichinger
Vice President New Business Development
FuturePlus Systems Corporation
Bedford, NH
Barb.Aichinger@FuturePlus.com

***Abstract*—DDR3 memory is at the heart of almost all cloud computing servers today. A recently publicized failure mechanism in DDR3 memory, coined Row Hammer, has been shown to not only be a reliability issue but also a security risk. No industry standards group, government agency or trade association has signed up to address this issue. Data Centers and end users are on their own. This paper will discuss briefly the problem, mitigation strategies and a unique testing tool to determine what applications have the potential to create these types of failures.**

Keywords—DDR3 Failures, DDR4, Row Hammer, Data Center down time, DDR3 memory.

I. Introduction

Computer architecture relies on three basic building blocks, the CPU or central processing unit, the I/O, Input and Output and the Memory. When it comes to the memory the dominate technology is DRAM or Dynamic Random Access Memory. Today's most prevalent version of memory is called DDR3 which stands for the 3rd generation of Double Data Rate Memory. In the quest to get memories smaller and faster memory vendors have had to make very small physical geometries. These small geometries put memory cells very close together and as such one memory cell's charge can leak into an adjacent one causing a bit flip. It has come to the attention of the industry that this is indeed happening under certain conditions. Very simply the problem occurs when the memory controller under command of the software causes an ACTIVATE command to a single row address repetitively. If the physically adjacent rows have not been ACTIVATED or Refreshed recently the charge from the over ACTIVATED row leaks into the dormant adjacent rows and causes a bit to flip. This failure mechanism has been coined 'Row Hammer' as a row of memory cells are being 'hammered' with ACTIVATE commands. Additionally double sided Row Hammering has also been proven. This involves two 'aggressor' rows on either side of a 'victim' row. This double sided hammering produces failures faster and causes more bits to flip[1]. Once this failure occurs a Refresh command from the Memory Controller solidifies the error into the memory cell. Current understanding is that the charge leakage does not permanently damage the physical memory cell which makes repeated memory tests trying to find the failing device useless.

DDR3 memory is pervasive today and used in nearly all cloud server systems, many embedded applications and military applications. Most critical applications *do use* error detection and correction, ECC. However ECC is a single bit detection and correction and double bit detection. In the case of more than two bit errors, which has been demonstrated with Row Hammer failures, ECC falls short. Our dependence on DDR3 memory and this known failure mechanism should be a wake up call for the industry. So far the most common workaround is to double the refresh rate to the memory. This is an attempt to 'charge up' the dormant memory cells so that they do not fall victim to adjacent rows that might become 'hammered'. This reduces performance and increases power consumption and the problem is not going away. This workaround just reduces the statistical probability.

II. Why does this happen?

Simply put, the memory controller's job is to read and write information to and from the memory under program control. If the software running executes certain commands that cause repeated accesses to a single location the memory controller will generate excessive ACTIVATE commands. Currently there is nothing in the DDR3 memory controller designs to prevent this from happening. Software often uses repetitive accesses to check to see if a task has been completed. This is a very common occurrence in software architecture and referred to as a Semaphore. Several tasks or threads will communicate with each other using a shared location in the memory. Thus they all need to repeatedly access these shared locations in order to communicate.

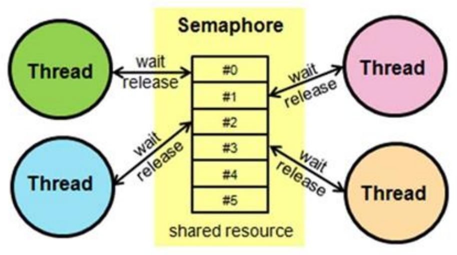

Figure 1: The use of a semaphore can cause repeated accesses to a single location in memory

[1] http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

Individual software instructions associated with Row Hammer have come to light in recent studies[2] namely the CLFLUSH command. This forces the processor to not store the information in cache. Thus the memory controller initiates the page open ACTIVATE command to the main memory. As of May 2015 several open source programs on the internet can be downloaded, which within a few lines of code, can create Row Hammer failures fairly quickly.[3]

III. IS THE PROBLEM REAL?

Although curiously void from the JEDEC[4] meeting minutes prior to 2012 we do see evidence of the problem being mentioned in the press and on the internet in early 2014. Electronic Design has mentioned this phenomenon and IBM has a field update to its firmware to try to deal with it. A search of recent patent applications reveals that in January of 2014 Intel submitted two patent applications that deal with Row Hammer. The first is a technique to detect excessive ACTIVATES to a single row address.

Row Hammer Condition Monitoring: US 20140006704 A1. A system monitors data accesses to specific rows of memory to determine if a Row Hammer condition exists. The system can monitor accessed rows of memory to determine if the number of accesses to any rows exceeds a threshold associated with risk of data corruption on a row of memory physically adjacent to the row with high access. Based on the monitoring, a memory controller can determine if the number of accesses to a row exceeds the threshold, and indicate address information for the row whose access count reaches the threshold.

The second Intel patent application deals with a targeted row refresh. That is if the memory controller sees the excessive ACTIVATE commands, below the error threshold, it can tell the DRAM the address of that Hammered Row and the DRAM can refresh and restore the charge to the physically adjacent rows to avoid the problem.

Row hammer refresh command: WO 2014004748 A1. A memory controller issues a targeted refresh command. A specific row of a memory device can be the target of repeated accesses. When the row is accessed repeatedly within a time threshold (also referred to as "hammered" or a "row hammer event"), physically adjacent row (a "victim" row) may experience data corruption. The memory controller receives an indication of a row hammer event, identifies the row associated with the row hammer event, and sends one or more commands to the memory device to cause the memory device to perform a targeted refresh that will refresh the victim row.

In 2014 Samsung divulged the issue in a recent investors presentation touting that its DDR4 “in-DRAM” solution is “most efficient for Row Hammer operation”.

In the summer of 2014 the most conclusive information was published by Carnegie Mellon University researcher Yoongu Kim. His paper ‘Flipping Bits Without Accessing them’ gave the industry its first conclusive information proving the failure was actually quite wide spread.[5] Kim and his team found the failures were across all vendors tested and were found more frequently in DRAMs manufactured after 2010.

In March 2015 Google stepped into the Row Hammer arena with its blog post *Exploiting the DRAM rowhammer bug to gain kernel privileges.*[6] This ignited a fire storm of technical media press on the topic but it died out after a few weeks. The industry now waits for a response from the memory vendors but there has been very little.

IV. CONFUSION AND MISUNDERSTANDINGS

Given that this is a fairly technical topic those weighing in who do not have a complete understanding have added unreliable and misleading comments to the conversation. First is the notion that ECC will correct this problem. The commonly used ECC technique for DRAM memory, SECDED is a single error bit detection and correction and a double error bit detection. The notion that all memory errors can be magically corrected by ECC is unfortunately a common misconception. The uninformed think they can buy substandard memory and motherboards as long as they have ECC. The CMU study showed repeated row hammer failures with multiple bits per transfer. Though the number of failures beyond two bits was much less common it proved that ECC could not prevent undetected data corruption for this failure mechanism.

The Google Blog post focused on the security risk for this problem. Since their exploits were picked up by the technical media the reliability consequences of this failure mechanism has taken a back seat. However the reliability issues are much more likely to be of concern to the industry. With the number of servers increasing daily, with some estimates saying that there are over 50M servers world wide, it is a significant statistical probability that DDR3 Row Hammer failures will occur in cloud computing applications. How large? Hard to gauge but if we go with a low end estimate of each server having 10 DIMMs and each DIMM has 2GB capacity (relatively small) using the CMU study failure rate of ~ 10^5per 10^{9th} cells[7] (10,000/1G) for the 3 major memory manufacturers gives bit failure rates in the millions per system. As an example a 2GB DIMM has $2x10^9$ x8 cells which equals $16x10^9$ and if each system has 10 DIMMs then you have 160 $x10^9$ memory cells per system. ***If the error potential is 10^4 per 10^9 you have a potential for 1.6M errors per system if Row Hammering were to occur.***

This author is fairly certain that millions of memory failures per system would be unacceptable for almost any

[2] Both the Google Blog and CMU papers referenced later in this paper report using the CLFLUSH command to create the failure.

[3] https://github.com/google/rowhammer-test

[4] JEDEC is the industry standards group that controls the DDR Specification

[5] http://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

[6] http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

[7] Figure 3 http://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

application. With over 50 million servers world wide the memory failure potential goes into the billions.

Lest us not fear monger! What is unknown is does the system have an application running that creates the Row Hammer event. In order to understand that we would need to study the code of every application, which would seem impossible, or run the application on systems that employed hardware counters to monitor the memory and count the number of ACTIVATE commands which lead to the problem.

V. Detecting if Applications Create the Row Hammer Event

It would be physically impossible to examine the actual code running on all servers to look for the sequence of instruction that have the potential to cause Row Hammer failures. However if the memory could be observed while critical applications were running or observed when applications that have shown to be running when mysterious memory failures occur, the analysis becomes more manageable. Such a tool has been designed for this exact application. A general purpose protocol analyzer, the DDR Detective®, has been repurposed using its programmable FPGA to count the number of ACTIVATE commands to unique row addresses.

Some background on DDR3 memory is in order to understand the basis for the testing. DDR3 has 8 banks per rank and each bank containing rows and columns. For a 2GB DIMM there are 16,384 Rows per bank with a total of 131,072 (16, 384 * 8) unique row addresses.

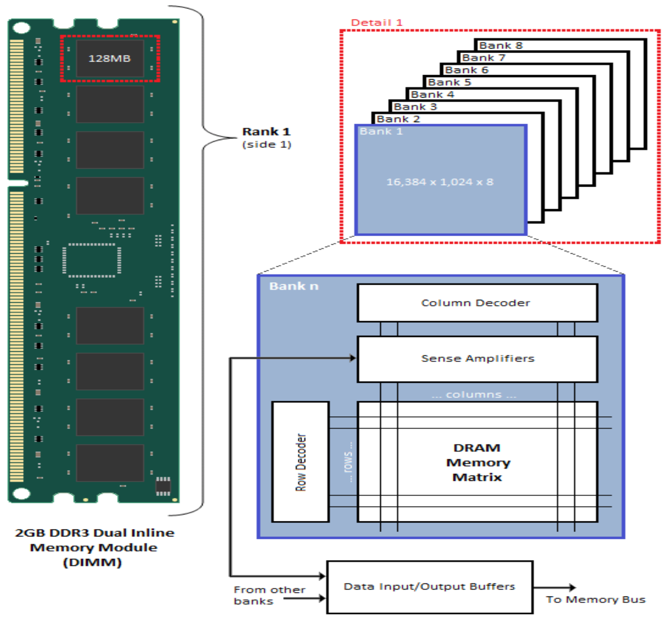

Figure 2: DDR3 DRAM Configuration [8]

Using current FPGA technology it is impossible to create 131,072 counters thus a statistical approach to the problem must be taken. Examining the first 1000+ unique row addresses that are issued by the memory controller is reasonable given the other timings that must be obeyed for a compliant DDR3 memory channel to work. In addition only the unique addresses that occur during a 64ms time period need to be accumulated. After 64ms the counters can be cleared and the hunt for new unique row addresses can be restarted.

The repurposed DDR protocol analyzer[9] was programmed to use 2400 counters divided into 2, 1200 counter Row Hammer Detection Units to track every ACTIVATE by row address that occur within an interval of 64mS, which relates to the minimum retention period of a single location in a synchronous DRAM. These counters are reset and reassigned at the end of each retention period. Theses counters run continuously and never miss an ACTIVATE command. If the number of unique addresses exceeds the available counters a remainder count is incremented to indicate that the traffic is highly variable. In addition, the retention period can be lowered to 32ms by selecting Hi Temp as this is the retention period for high temperature operation. The retention period of the tool is fixed in that it is not a 'rolling' window. To help compensate for this the tool actually has a duplicate set of counters offset by ½ the retention period to give better coverage.

Figure 3: The repurposed DDR Protocol Analyzer with its DIMM interposer

There are 2 variable threshold limits that the user can set to flag Row Addresses that cross those limits. The output graphic shows the rows that have seen ACTIVATE commands exceed the threshold. Blue for Threshold 1 and red for Threshold 2. Threshold1 and Threshold2 are defaulted to 100K and 300K respectively and can be adjusted. The exact address of the row address that crosses these thresholds is also listed in the right hand side scroll pane.

The Row Hammer setup page allows the user to set an Address Range across slots, Ranks and Banks for which the tool will only count ACTIVATE commands to rows in this range. By narrowing down the range it helps keep the counters from overflowing and can give more accurate results.

[8] http://www.anandtech.com/show/3851/everything-you-always-wanted-to-know-about-sdram-memory-but-were-afraid-to-ask/2

[9] The DDR Detective® has its own web site www.DDRDetective.com

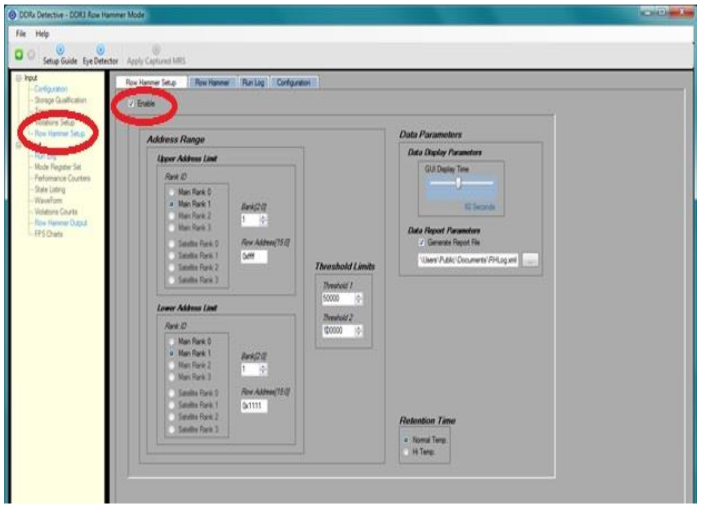

Figure 4: Row Hammer Detection setup screen

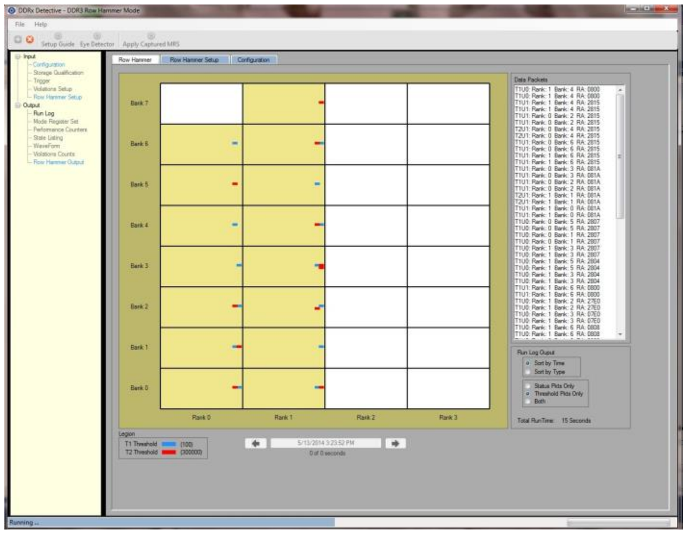

Figure 5: Row Hammer Detection output screen shows when a row has been 'hammered'

An XML based report file for all ACTIVATES that cross the thresholds can be generated.

The tool will give a trigger out 7200 cycles following an ACTIVATE command which reaches Threshold 2 just in case the user wants to trigger a logic analyzer to capture returning read data to then pinpoint the exact failing bits. The size of the Threshold counters are 21 bits.

Now the tool can look for the potential of Row Hammer events while the server is running ANY application. The presence of the tool is virtually invisible to the system and the software running. The tool connects to the system using an interposer which intercepts the signals to and from the memory controller to the DIMM. The probing can be changed to address embedded or memory down applications.

The Row Hammer output graphic shows a mapping by Bank and Rank of each Row Address location that had a total number of ACTIVATES occur over the defined Thresholds (T1 & T2). This display is updated every second.

The Row Hammer output graphic shows Ranks in columns and Banks in Rows, which form cells. The specific Row Address is represented in that cell area by the first 2 nibbles (last 2 are don't cares) of the Address as shown below. If the exact Address is needed it can be seen in the report window on the right, or from a generated report.

The output graphic is refreshed every second. After the tool is stopped each previous or successive 1 second period can be paged through using the arrow buttons below the output graphic. A maximum of a 120 seconds worth of traffic can be stored.

Each Row Hammer Detection Unit also reports status at the end of each retention cycle. This status would indicate how many ACTIVATES were not counted (remainder count) and if an overflow did not occur how many counters remained unused.

An xml output report can also be generated that writes out to a file all of the row addresses that exceed the threshold for each Row Hammer detection unit.

Now critical applications can test for the presence of excessive ACTIVATE commands caused by their application. If applications do not create excessive ACTIVATE commands the urgency to address the Row Hammer failure mechanism is greatly reduced.

VI. WHAT ARE THE MITIGATION STRATEGIES?

If it can be shown that applications are creating the Row Hammer events, mitigation strategies can now be investigated. As previously mentioned the most common mitigation strategy being employed today is a doubling of the Refresh rate. In DDR3 Memory the REFRESH command is issued to the entire RANK (half the DIMM for a 2 rank implementation). The DRAM device itself does not refresh every row in the device upon receiving this command. Rather it kicks off a scheduler of sorts that has the task of making sure that all rows in the Rank are refreshed at least once in a 64ms retention period. Once a refresh command is issued the entire Rank becomes unavailable for Write and Read operations. Doubling the occurrence of refreshes not only burns additional power but is a performance hit as the memory becomes unavailable twice as often. This method, although widely employed, reduces the statistically probability but does not prevent the failures[10]. The changing of the refresh rate is usually a BIOS selection.

[10] Figure 4 http://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

The other mitigation strategy that has been discussed for DDR3 has been what is referred to as a pseudo targeted row refresh or pTRR. Since targeted row refresh commands are being discussed for inclusion into DDR4 and LPDDR4 (Low Power DDR4) in the JEDEC committees, the phrase 'psuedo' was derived for a backwards method to be used for DDR3 since no specific command exists in DDR3. This 'psuedo' technique involves the memory controller issuing a REFRESH command and placing the 'hammered' row address on the address lines. Its then up to the memory device to refresh the victim rows which are the ones physically adjacent to the 'aggressor' row. Intel put this feature into their Ivy Bridge Processor families[11] but it is not clear if any other memory control vendor has implemented this feature as only Intel has revealed its implementation. This feature can only work if the DRAMs understand this command and execute it accordingly.

Pseudo Target Row Refresh is certainly not a strategy that will be adopted for any pre Ivy Bridge servers or servers using a non Intel memory controller. In addition the millions if not billions of embedded DRAM implementations cannot be retrofitted to use either the doubling of the refresh rate or the pTRR. This should be a wakeup call for all critical applications using DDR3 DRAMs. Other mitigation strategies include using DDR3 DIMMs that have been specifically tested to be row hammer free a feat that no DRAM vendor has yet signed up for. The most widely touted mitigation strategy by the DRAM and system vendors is a total machine swap and upgrade to DDR4. A profitable choice for them of course but not at all practical for everyone else. One caveat on this last strategy is that no test data showing that DDR4 is immune to this problem has ever been published. In addition the Target Row Refresh command is not part of the JEDEC DDR4 specification however it is part of the LPDDR4 specification which does not help servers.

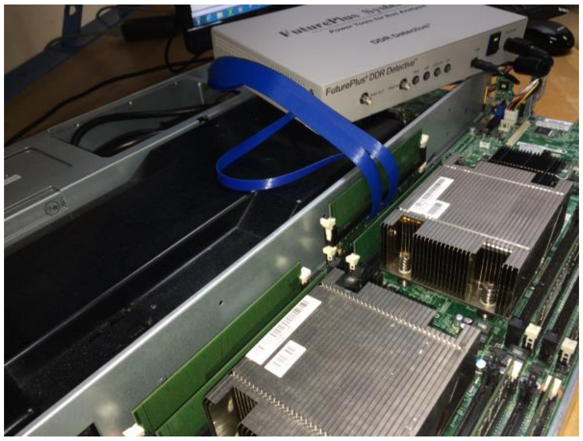

Figure 3: Open Compute Server being tested for Row Hammer failures

Lastly engineers may choose to identify the source code creating the excessive ACTIVATE commands and rewrite the code to remedy the situation. Not using the CFLUSH command or lengthen loops that accessed shared semaphores are a few strategies that can be employed.

VII. SUMMARY

DDR3 memory is a critical part of the world's cloud computing strategy and today's servers have an extensive amount of DDR3 memory. The studies have shown a potential for millions of Row Hammer failures per system. Given the vast amount of DDR3 memory in today's systems failures should clearly be a concern. This known failure mechanism can lead to undetected data corruption, reliability issues and security breaches. Current mitigation strategies, for deployed systems, are impractical, expensive or just reduce the statistical likelihood. A strategy to determine if applications even create the Row Hammer failure should be considered. Understanding if an application is at risk can reduce the pressure to implement unneeded, expensive and time consuming mitigation strategies saving organizations millions of dollars. If applications are shown to be at risk then steps can be taken to upgrade hardware, rewrite the application and provide warnings to the field that such failures might occur.

[11] http://infobazy.gda.pl/2014/pliki/prezentacje/d2s2e4-Kaczmarski-Optymalna.pdf page 13