AVATAR: A VARIABLE-RETENTION TIME AWARE REFRESH FOR DRAM

DSN-45
06/24/2015
Rio de Janeiro, Brazil

Moinuddin Qureshi, Georgia Tech

Dae-Hyun Kim
Prashant Nair

Samira Khan
Onur Mutlu
Dynamic Random Access Memory (DRAM) stores data as charge on a capacitor.
Dynamic Random Access Memory (DRAM) stores data as charge on capacitor.
Dynamic Random Access Memory (DRAM) stores data as charge on capacitor.

DRAM BACKGROUND

DRAM is a volatile memory → charge leaks quickly.
Retention Time: The time for which cell/memory retains data

DRAM maintains data by “refresh” operations at row granularity

[Diagram of DRAM Chip]
Retention Time: The time for which cell/memory retains data

DRAM maintains data by “refresh” operations at row granularity
Retention Time: The time for which cell/memory retains data

DRAM maintains data by “refresh” operations at row granularity.
Retention Time: The time for which cell/memory retains data.

DRAM maintains data by "refresh" operations at row granularity.
Retention Time: The time for which cell/memory retains data

DRAM maintains data by “refresh” operations at row granularity
Retention Time: The time for which cell/memory retains data

DRAM maintains data by “refresh” operations at row granularity

Refresh period determined by “worst-case” cell: 64ms (JEDEC)

DRAM relies on refresh (64ms) for data integrity
“REFRESH WALL” FOR DRAM SYSTEMS

Refresh cost proportional to capacity ➔ Exponentially increasing

"REFRESH WALL" FOR DRAM SYSTEMS

Refresh cost proportional to capacity ➔ Exponentially increasing

“REFRESH WALL” FOR DRAM SYSTEMS

Refresh cost proportional to capacity ➔ Exponentially increasing

---

**Refresh consumes significant time and energy**

Retention time of cells vary significantly: most cells $>> 64$ ms

Retention time of cells vary significantly: most cells $>> 64\text{ms}$

Exploit variability in retention time $\Rightarrow$ Multirate Refresh

Normal Refresh (64ms) & Slow Refresh (e.g. 256ms+)

Retention time of cells vary significantly: most cells >> 64ms

Exploit variability in retention time ➔ Multirate Refresh
Normal Refresh (64ms) & Slow Refresh (e.g. 256ms+)

Row contains a cell with retention time < period of Slow Refresh

Retention time of cells vary significantly: most cells >> 64ms

Exploit variability in retention time ➔ Multirate Refresh
Normal Refresh (64ms) & Slow Refresh (e.g. 256ms+)

Row contains a cell with retention time < period of Slow Refresh

No
Use Slow Refresh

Retention time of cells vary significantly: most cells $>> 64$ms

Exploit variability in retention time $\Rightarrow$ Multirate Refresh
Normal Refresh (64ms) & Slow Refresh (e.g. 256ms+)

Row contains a cell with retention time $<$ period of Slow Refresh

Yes $\Rightarrow$ Use Normal Refresh
No $\Rightarrow$ Use Slow Refresh

Retention time of cells vary significantly: most cells >> 64ms

Exploit variability in retention time ➔ Multirate Refresh
Normal Refresh (64ms) & Slow Refresh (e.g. 256ms+)

Row contains a cell with retention time < period of Slow Refresh

Yes
Use Normal Refresh

No
Use Slow Refresh

Efficient DRAM refresh by exploiting variability

DRAM Rows

A
B
C
D
E
F
G
H
DRAM Rows

A
B
C
D
E
F
G
H

RETENTION PROFILING
MULTI RATE REFRESH: DESIGN & EFFECTIVENESS

DRAM Rows

A
B
C
D
E
F
G
H

Weak Cell

RETENTION PROFILING
MULTI RATE REFRESH: DESIGN & EFFECTIVENESS

DRAM Rows

A
B
C
D
E
F
G
H

Ref. Rate Table

0: Slow Refresh
1: Normal Refresh

Weak Cell

RETENTION PROFILING
MULTI RATE REFRESH: DESIGN & EFFECTIVENESS

DRAM Rows

A
B
C
D
E
F
G
H

Ref. Rate Table

0
0
1
0
0
0
1
0

Retention Profiling

Weak Cell

0: Slow Refresh
1: Normal Refresh

Slow Refresh Rate=1/8x

Slow Refresh Rate=1/4x

Reduction in Refresh (%) vs. Num Rows Using Fast Refresh (%)
Multi rate refresh can reduce refresh by 70%+

DRAM Rows

A
B
C
D
E
F
G
H

Weak Cell

RETENTION PROFILING

0: Slow Refresh
1: Normal Refresh

Ref. Rate Table

0
0
1
0
0
0
1
0

Reduction in Refresh (%)

Slow Refresh Rate = 1/8x

Slow Refresh Rate = 1/4x

Num Rows Using Fast Refresh (%)
VARIABLE RETENTION TIME (VRT): THE NEMESIS

Multirate refresh relies on retention time to remain unchanged.

Retention time can vary at runtime due to VRT.

### DRAM Rows
- A
- B
- C
- D
- E
- F
- G
- H

### Ref. Rate Table

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Weak Cell**

**RETENTION PROFILING**
VARIABLE RETENTION TIME (VRT): THE NEMESIS

Multirate refresh relies on retention time to remain unchanged.

Retention time can vary at runtime due to VRT.

<table>
<thead>
<tr>
<th>DRAM Rows</th>
<th>Ref. Rate Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
</tr>
</tbody>
</table>

Weak Cell

RETENTION PROFILING
Multirate refresh relies on retention time to remain unchanged.

Retention time can vary at runtime due to VRT.

VRT renders multi-rate refresh unusable in practice.
VRT considered one of the biggest impediment to DRAM scaling
-- [Samung & Intel, Memory Forum 2014]

Our study investigates the following questions:

1. Can we analyze VRT using architecture level models?
2. Can we overcome VRT simply by using ECC DIMM?
3. If not, what is a low cost solution to mitigate VRT?
Background

VRT: mechanism, measurement, model

Can’t we fix VRT by simply using ECC DIMM?

AVATAR

Results

Summary
WHY DOES VRT OCCUR? WHEN IS IT HARMFUL?

VRT caused by fluctuations in Gate Induced Drain Leakage.

External factors: mechanical stress, high temperature etc.
WHY DOES VRT OCCUR? WHEN IS IT HARMFUL?

VRT caused by fluctuations in Gate Induced Drain Leakage.

External factors: mechanical stress, high temperature etc.

Not all VRT is harmful

VRT problematic when strong cell becomes weak
EXPERIMENTAL SETUP

**Test platform:** DDR3 testing platform Xilinx ML605 FPGA development board in temperature controlled setting

**Slow Refresh:** Studied refresh of 4s at 45C, corresponds to 328ms at 85C [khan+ SIGMETRICS’14, Liu+ ISCA’13]

**Test:** Write specific pattern, read pattern, log id of erroneous cell Statistics collected every 15 minutes, over 7 days (672 rounds)

Three (2GB) modules, one each from different DRAM vendor
Even after several days of testing, VRT causes new (previously unidentified) cells to cause failures
2: VRT-CELLS CAN SWITCH RANDOMLY

A VRT cell can randomly and frequently transition between strong and weak states.

Cell with retention time $< 328$ ms $\Rightarrow$ Weak Cell, else Strong Cell
3: SIZE OF ACTIVE-VRT POOL VARIES

Active-VRT Cell: Cell that failed during the given 15-min round

Active-VRT Pool (AVP): Group of Active VRT Cells

The size of AVP varies dynamically for all modules

Avg=347

Avg=492

Avg=388
Predicting the exact AVP size is difficult, but it can be modeled using a lognormal distribution. 

**Observation:** AVP size tends to follow a lognormal distribution.

**AVP size modeled using lognormal distribution**
Active-VRT Injection (AVI) Rate

The rate at which new cells become Active-VRT cells

AVP reduces to ~1 new cell per 15-min period
ARCHITECTURE MODEL FOR CELL UNDER VRT

Strong Cell

AVI

Active-VRT Cell

Weak Cell

AVP

Dormant VRT Cell
ARCHITECTURE MODEL FOR CELL UNDER VRT

Two key parameters:

**Active-VRT Pool (AVP):** How many VRT cells in this period?

**Active-VRT Injection (AVI):** How many new (previously undiscovered) cells became weak in this period?

Model has two parameters: AVP and AVI
ARCHITECTURE MODEL FOR VRT

Input: $\mu$, $\text{Sdev}$, for the logn of Active–VRT pool
Input: $K$, rate of discovering new VRT cells
Input: \( \mu, \sigma \), for the logn of Active-VRT pool
Input: \( K \), rate of discovering new VRT cells

\[
\text{PoolSize} = \text{Rand} (\text{LogNormDist}[\mu, \sigma])
\]

Insert \( K \) new elements in Pool
Remove \( K \) elements from Pool

\[
P[\text{TimePeriod}] = \text{System Failure Probability}
\]
ARCHITECTURE MODEL FOR VRT

Input: \( \mu, \sigma \), for the logn of Active-VRT pool
Input: \( K \), rate of discovering new VRT cells

\[
\text{PoolSize} = \text{Rand} (\lognormal(\mu, \sigma)) \\
\text{Insert } K \text{ new elements in Pool} \\
\text{Remove } K \text{ elements from Pool} \\
P[\text{TimePeriod}] = \text{System Failure Probability}
\]

TimePeriod++
ARCHITECTURE MODEL FOR VRT

Input: $\mu, S_{dev}$, for the logn of Active-VRT pool
Input: $K$, rate of discovering new VRT cells

1. $\text{PoolSize} = \text{Rand} \left( \text{LogNormDist}[\mu, S_{dev}] \right)$
2. Insert $K$ new elements in Pool
3. Remove $K$ elements from Pool
4. $P[\text{TimePeriod}] = \text{System Failure Probability}$
5. $\text{TimePeriod}++$
ARCHITECTURE MODEL FOR VRT

Input: Mu, Sdev, for the logn of Active-VRT pool
Input: K, rate of discovering new VRT cells

1. PoolSize = Rand (LogNormDist[Mu, Sdev])
2. Insert K new elements in Pool
3. Remove K elements from Pool
4. P[TimePeriod] = System Failure Probability
5. TimePeriod++

Parameter scaling for larger systems: 2GB DIMM to 8GB DIMM
AVP size increased by 4x: from ~400 to ~1600
AVI rate increased by 4x: from 1 to 4
OUTLINE

- Background
- VRT: mechanism, measurement, model
- Can’t we fix VRT by simply using ECC DIMM?
- AVATAR
- Results
- Summary
BACKGROUND ON ECC DIMM

ECC DIMM can tolerate 1 error per word (8 bytes)

Typically used to tolerate soft error but can also be used to fix a bit error due to VRT

A multi-bit error per word ➔ uncorrectable error

What is time to double error per word under VRT?
W words in memory (strong rows only)
P words have 1 bit error already (AVP)
K new weak cells get injected in given time quanta

\[ P(\text{DIMM has no uncorrectable error}) = \left(1 - \frac{P}{W}\right)^K \]
W words in memory (strong rows only)
P words have 1 bit error already (AVP)
K new weak cells get injected in given time quanta

\[ P(DIMM \text{ has no uncorrectable error}) = \left(1 - \frac{P}{W}\right)^K \]

For T time quanta, and D DIMMS
ANALYTICAL MODEL FOR ECC DIMM

W words in memory (strong rows only)
P words have 1 bit error already (AVP)
K new weak cells get injected in given time quanta

\[
P(DIMM \text{ has no uncorrectable error}) = \left(1 - \frac{P}{W}\right)^K
\]

For T time quanta, and D DIMMS

\[
P(\text{System has no uncorrectable error}) = \left(1 - \frac{P}{W}\right)^{K \cdot T \cdot D}
\]
EVEN WITH ECC-DIMM, ERROR RATE IS HIGH

System: Four channels, each with 8GB DIMM

VRT still causes an error every ~6-8 months
Background

VRT: mechanism, measurement, model

Can’t we fix VRT by simply using ECC DIMM?

AVATAR

Results

Summary
Insight: Avoid forming Active VRT Pool ➔ Upgrade on ECC error

Observation: Rate of VRT >> Rate of soft error (50x-2500x)
Insight: Avoid forming Active VRT Pool ➔ Upgrade on ECC error
Observation: Rate of VRT >> Rate of soft error (50x-2500x)
**Insight:** Avoid forming Active VRT Pool ➔ Upgrade on ECC error

**Observation:** Rate of VRT >> Rate of soft error (50x-2500x)
Insight: Avoid forming Active VRT Pool ➔ Upgrade on ECC error
Observation: Rate of VRT >> Rate of soft error (50x-2500x)
Insight: Avoid forming Active VRT Pool ➔ Upgrade on ECC error
Observation: Rate of VRT >> Rate of soft error (50x-2500x)
**Insight:** Avoid forming Active VRT Pool ➔ Upgrade on ECC error

**Observation:** Rate of VRT >> Rate of soft error (50x-2500x)

### DRAM Rows

<table>
<thead>
<tr>
<th>ECC</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
</table>

### Ref. Rate Table

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Weak Cell**

**RETENTION PROFILING**
Insight: Avoid forming Active VRT Pool ➔ Upgrade on ECC error
Observation: Rate of VRT >> Rate of soft error (50x-2500x)
AVATAR

Insight: Avoid forming Active VRT Pool ➔ Upgrade on ECC error

Observation: Rate of VRT >> Rate of soft error (50x-2500x)

DRAM Rows

<table>
<thead>
<tr>
<th>ECC</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC</td>
<td>B</td>
</tr>
<tr>
<td>ECC</td>
<td>C</td>
</tr>
<tr>
<td>ECC</td>
<td>D</td>
</tr>
<tr>
<td>ECC</td>
<td>E</td>
</tr>
<tr>
<td>ECC</td>
<td>F</td>
</tr>
<tr>
<td>ECC</td>
<td>G</td>
</tr>
<tr>
<td>ECC</td>
<td>H</td>
</tr>
</tbody>
</table>

Ref. Rate Table

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Weak Cell

Row protected from future retention failures
**AVATAR**

**Insight:** Avoid forming Active VRT Pool ➔ Upgrade on ECC error

**Observation:** Rate of VRT >> Rate of soft error (50x-2500x)

AVATAR mitigates VRT by breaking AVP Pool
Insight: Avoid forming Active VRT Pool ➔ Upgrade on ECC error
Observation: Rate of VRT >> Rate of soft error (50x-2500x)

AVATAR mitigates VRT by breaking AVP Pool
Insight: Avoid forming Active VRT Pool ➞ Upgrade on ECC error
Observation: Rate of VRT >> Rate of soft error (50x-2500x)

AVATAR mitigates VRT by breaking AVP Pool
**Insight:** Avoid forming Active VRT Pool ➞ Upgrade on ECC error

**Observation:** Rate of VRT >> Rate of soft error (50x-2500x)

AVATAR mitigates VRT by breaking AVP Pool

<table>
<thead>
<tr>
<th>DRAM Rows</th>
<th>Ref. Rate Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>1</td>
</tr>
<tr>
<td>ECC</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>0</td>
</tr>
<tr>
<td>ECC</td>
<td>1</td>
</tr>
</tbody>
</table>

AVATAR

Scrub (15 min)

Weak Cell

RETENTION PROFILING
Only errors injected between scrub can clash with each other

Instead of 1000+ weak cells (AVP), deal with 4 errors (AVI)
Only errors injected between scrub can clash with each other

Instead of 1000+ weak cells (AVP), deal with 4 errors (AVI)

\( W \) words in memory, \( K \) errors in time quanta (AVI Rate)
AVATAR: ANALYTICAL MODEL

Only errors injected between scrub can clash with each other.

Instead of 1000+ weak cells (AVP), deal with 4 errors (AVI)

W words in memory, K errors in time quanta (AVI Rate)

\[
\text{Prob(DIMM has no uncorrectable error)} = (1 - \frac{1}{W}) \times (1 - \frac{2}{W}) \times \ldots \times (1 - \frac{K - 1}{W})
\]
Only errors injected between scrub can clash with each other.

Instead of 1000+ weak cells (AVP), deal with 4 errors (AVI)

$W$ words in memory, $K$ errors in time quanta (AVI Rate)

$$\textit{Prob(DIMM has no uncorrectable error)} = (1 - \frac{1}{W}) \times (1 - \frac{2}{W}) \times \ldots \times (1 - \frac{K - 1}{W}) = e^{-\frac{K^2}{2W}}$$
Only errors injected between scrub can clash with each other

Instead of 1000+ weak cells (AVP), deal with 4 errors (AVI)

\( W \) words in memory, \( K \) errors in time quanta (AVI Rate)

\[
\text{Prob(DIMM has no uncorrectable error)} = (1 - \frac{1}{W}) \times (1 - \frac{2}{W}) \times \ldots \times (1 - \frac{K - 1}{W}) = e^{-\frac{K^2}{2W}}
\]

For, \( T \) time quanta, and \( D \) DIMMS
Only errors injected between scrub can clash with each other.

Instead of 1000+ weak cells (AVP), deal with 4 errors (AVI).

\( W \) words in memory, \( K \) errors in time quanta (AVI Rate).

\[
\text{Prob(DIMM has no uncorrectable error)} = (1 - \frac{1}{W}) \times (1 - \frac{2}{W}) \times \ldots \times (1 - \frac{K - 1}{W}) = e^{\frac{-K^2}{2W}}
\]

For, \( T \) time quanta, and \( D \) DIMMS.

\[
\text{Prob(System has no uncorrectable error)} = e^{\frac{-DTK^2}{2W}}
\]
AVATAR: TIME TO FAILURE

System: Four channels, each with 8GB DIMM

* We include the effect of soft error in the above lifetime analysis (details in the paper)
System: Four channels, each with 8GB DIMM

AVATAR increases time to failure to 10s of years

* We include the effect of soft error in the above lifetime analysis (details in the paper)
Background
VRT: mechanism, measurement, model
Can’t we fix VRT by simply using ECC DIMM?
AVATAR

Results

Summary
RESULTS: REFRESH SAVINGS

AVATAR reduces refresh by 60%-70%, similar to multi rate refresh but with VRT tolerance
RESULTS: REFRESH SAVINGS

Retention Testing Once a Year can revert refresh saving from 60% to 70%

AVATAR reduces refresh by 60%-70%, similar to multi rate refresh but with VRT tolerance
AVATAR gets $\frac{2}{3}$rd the performance of NoRefresh. More gains at higher capacity nodes.
AVATAR reduces EDP,
Significant reduction at higher capacity nodes
OUTLINE

- Background
- VRT: mechanism, measurement, model
- Can’t we fix VRT by simply using ECC DIMM?
- AVATAR
- Results

- Summary
SUMMARY

Multirate refresh ➔ retention profiling to reduce refresh

Variable Retention Time ➔ errors with multirate refresh

✓ Architecture model of VRT based on experiments
✓ We show ECC DIMM alone is not enough
✓ AVATAR (upgrade refresh rate of row on ECC error)

AVATAR increase the time to failure from 0.5 years to 500 years and incurs the same storage as ECC DIMM