Graphene

Strong yet Lightweight Row Hammer Protection

Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, Jae W. Lee

Presented by Miles Dai

6.888, Fall 2020

Motivation

Proposed Row Hammer solutions exhibit various shortcomings.

A good solution should have

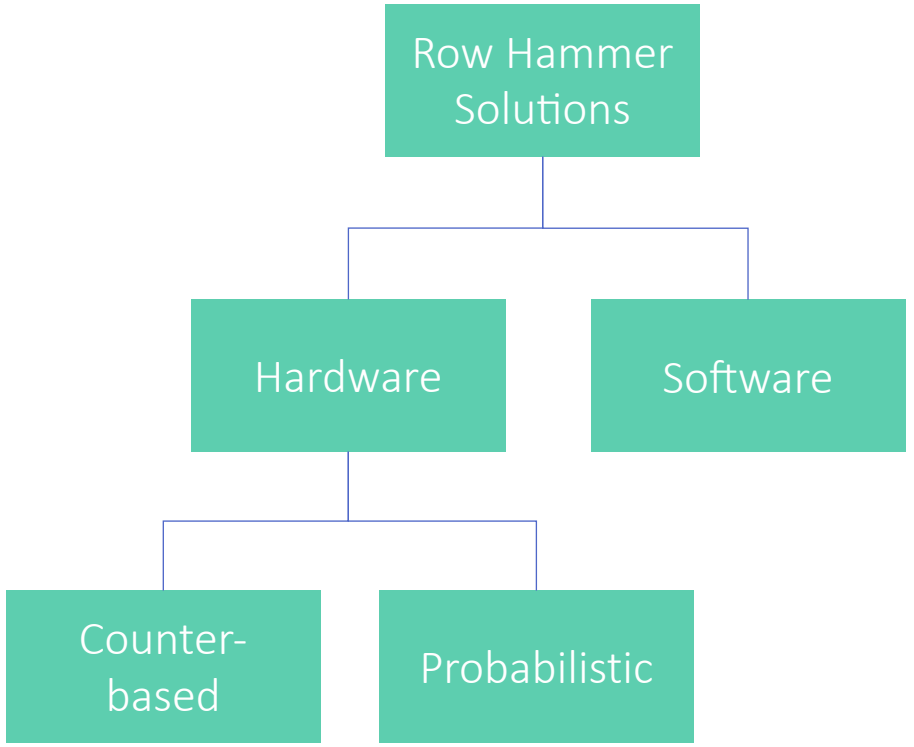
- Low performance overhead
- Low area/power overhead
- Low intrusiveness

Threat Model

- Attacker can execute a Row Hammer-style attack from any privilege level
- Hammered row can affect rows beyond neighbors (non-adjacent Row Hammer)
- As DRAM technology improves, the Row Hammer attack becomes easier to execute

Strawman Defense

- Protect against Row Hammer by refreshing rows that are in danger of being flipped accidentally
  - When a row is activated frequently, refresh its neighbors
- Software solutions?
- Can we just maintain a counter?
- What if we probabilistically refresh neighboring rows (e.g. PARA)?



Intuition

- Key difficulty: How do we track memory accesses with high precision and low cost?
- Two important insights
  - Exact counts aren't necessary
  - We only care about access frequencies above a threshold.
- Streaming algorithms have solved this problem!

Graphene Goals

- Guaranteed protection (no false negatives)
- Low energy and performance overhead
- Low area overhead
- Scalability

Discussion

What are some strengths and weaknesses of Graphene?

Evaluation

Strengths

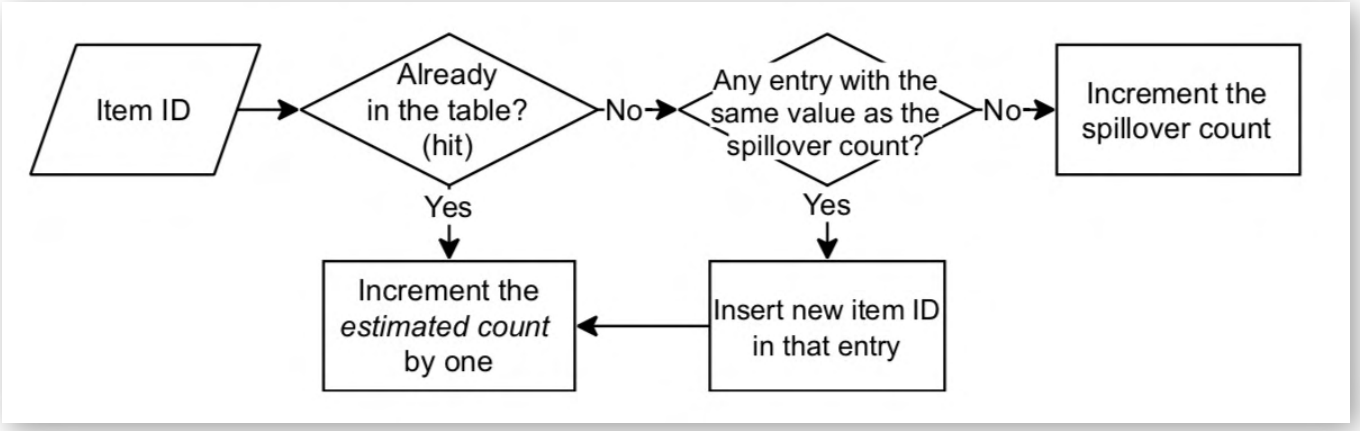
- Configurable trade off between area and energy usage
- Scalable and adaptable for a generalized Row Hammer attacks
- Simple hardware with low overheads
- Deterministic protection

Weaknesses

- Isolation: Performance degradation not considered in the presence of a Row Hammer attack
- Introduction of new MC/DRAM side channels?
- More discussion needed on the Row Hammer Threshold

Misra-Gries Algorithm

- Goal: count the most frequent elements in a stream
- Returns an associate array (map or Python dictionary) mapping most frequent elements to their estimated frequency



Misra-Gries in DRAM

Row Addr	Count
0x1010	5
0x2020	7
0x3030	3
Spillover	2

0x1010 →

Row Addr	Count
0x1010	6
0x2020	7
0x3030	3
Spillover	2

0x4040 →

Row Addr	Count
0x1010	6
0x2020	7
0x3030	3
Spillover	3

0x5050 →

Row Addr	Count
0x1010	6
0x2020	7
0x5050	4
Spillover	3

Timing Details

- Refresh Window (tREFW): cells must be refreshed every 64 ms.
- Refresh Interval (tREFI): Within the 64 ms, refresh cells in batches.
  - With 8,192 refreshes within one rank (8-cell batch), issue a refresh command every 64ms/8,192 = 7.8 us
- Refresh Command time (tRFC): how long the command is actually active.
- tRC: minimum interval between two ACTs to the same bank

Table Sizing

- Misra-Gries provides guarantees about the frequencies of the entries that appear in the final table.

$$N_{entry} > \frac{W}{T} - 1$$

- W is the maximum number of ACTs we receive within a timing window
- T is determined experimentally and takes into account double-sided Row Hammer and the phases of the refresh waves

Discussion Question

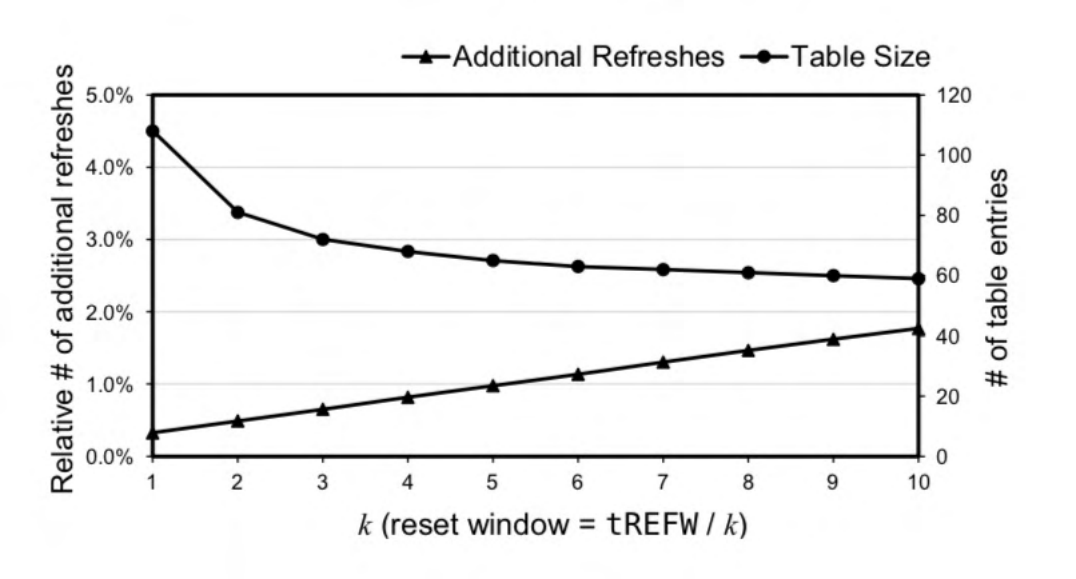
Adjustments and Optimizations to Graphene

Non-adjacent Row Hammer

- Non-adjacent ($\pm n$) Row Hammer assumes that an ACT can affect rows up to n rows away
- Can be handled by reducing T. Instead of 2 possible rows affecting the victim, 2n rows can now affect the victim.
- This can be bounded if we assume that the impact a row has and its distance from the victim follows an inverse power law.

Optimizations

- Overflow bits reduce the size of the table
  - An exact count is not needed above the threshold T
- Table size can be traded off against duration of the Reset Window



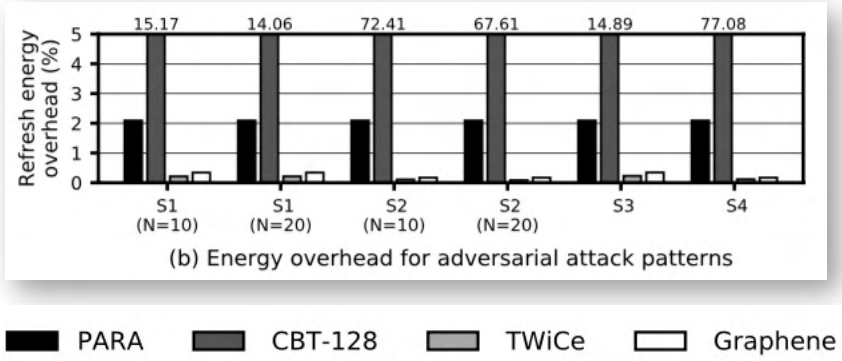
Evaluation - Area

- Graphene needs comparatively little area compared to other counter-based techniques
- The efficiency comes from leveraging streaming algorithms rather than trying to maintain per-structure counts
- SRAM is smaller than CAM, but SotA CAM technologies only present a 7% overhead over SRAM

	Table size (bits/bank)	Memory type
CBT-128 (10 levels)	3,824	SRAM
TWiCe	20,484 + 15,932	CAM + SRAM
Graphene	2,511	CAM

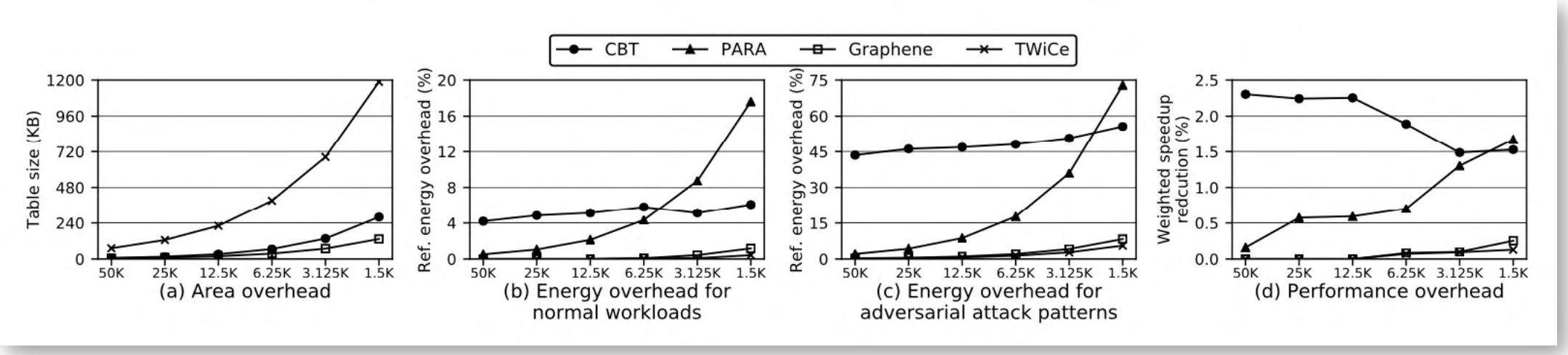
Evaluation - Energy

- Graphene hardware energy consumption is negligible
- No energy or performance overhead unless an active Row Hammer attack is taking place.
  - Adversarial attack pattern causes an energy overhead of < 0.34%
  - PARA has a constant 2.1% overhead even without Row Hammer



Scalability

- Scalability modeled by exponentially reducing the Row Hammer threshold (50k, 25k, 12.5k, …, 1.56k)



Discussion Questions - Implementation

- This relies on CAM (content addressable memory) to store data, which requires many transistors per bit. Could the (already existing) dram words be extended by a few bits as a counter or some other status tracker for row hammer prevention, which could save on the large external memory structures for saving metadata?
- Could Graphene + targeted row refreshes be utilized to execute timing side channel attacks?
- Is there still space for compressing the number of bits tracked (for instance using approximate counters in the CAM)?

Discussion Questions - Miscellaneous

- Why was targeted row refresh present in the DDR3 standard, but then removed in DDR4?
- Is there any reason counter-based solutions such as Graphene couldn't be built straight into future RAM chips?