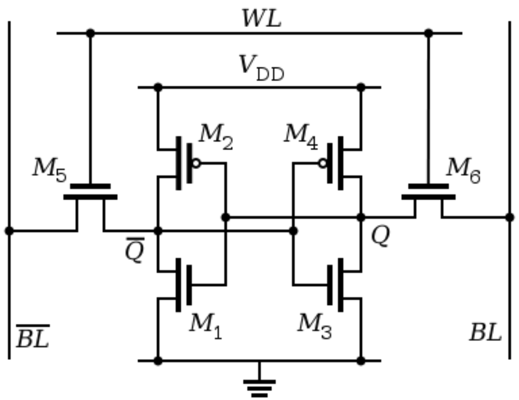
RowHammer in 15'

Nicolas RUFF
nruff+sstic15@google.com

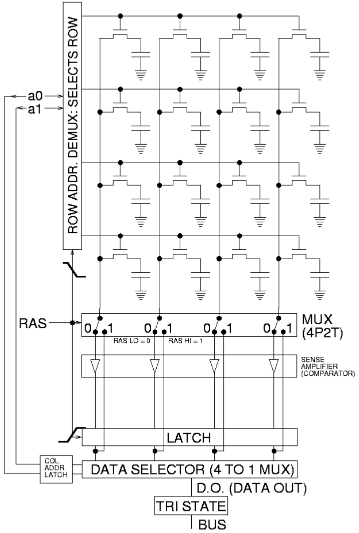
Life of an electron

SRAM: static RAM



http://en.wikipedia.org/wiki/Static_random-access_memory#/media/File:
SRAM_Cell_(6_Transistors).svg

DRAM: dynamic RAM



http://en.wikipedia.org/wiki/Dynamic_random-access_memory#/media/File:
Square_array_of_mosfet_cells_read.png

Life of an electron

SRAM

Uses a lot of die space (4 to 6 transistors per bit)

Fast random access time

Static (conserve state unless powered off)

Used for L-1 L-2 caches

DRAM

Excellent storage density (1 capacitor + 1 transistor per bit)

Slow access (full row access)

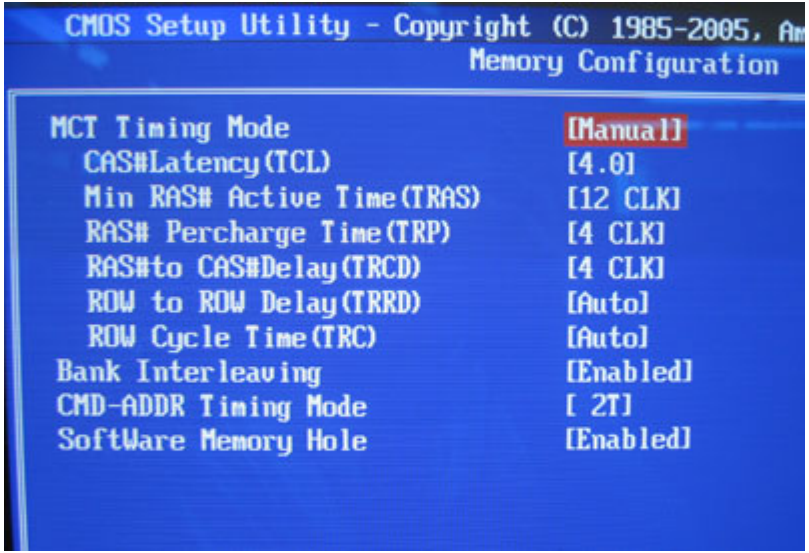
Leaky (capacitor discharges in ~N ms)

Used for external memory (Synchronous DRAM)

Life of an electron

DRAM discharge: mitigated by regular refresh

- Usually every 64ms



```
CMOS Setup Utility - Copyright (C) 1985-2005, Am
                           Memory Configuration

MCT Timing Mode                         [Manual]
  CAS#Latency(TCL)                      [4.0]
  Min RAS# Active Time(TRAS)            [12 CLK]
  RAS# Percharge Time(TRP)              [4 CLK]
  RAS#to CAS#Delay(TRCD)                [4 CLK]
  ROW to ROW Delay(TRRD)                [Auto]
  ROW Cycle Time(TRC)                   [Auto]
Bank Interleaving                       [Enabled]
CMD-ADDR Timing Mode                    [ 2T]
SoftWare Memory Hole                    [Enabled]
```

What if?

You access a value too often? Bit-flip(s)!

- Including in adjacent rows

Why? Nobody knows for sure ...

- Condenser discharge. Power glitch. Tunnel effect. You name it.

What if?

Known for years for the hardware industry

- Cf. JEDEC specifications

Re-discovered by software people

- https://github.com/CMU-SAFARI/rowhammer

Eventually exploited by Google as a generic privilege escalation

- http://googleprojectzero.blogspot.ch/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

Exploitation

Short version

- Fill memory
- Flip a PTE bit
- Profit!

Flipping *fast*

- CLFLUSH (userland, cannot be disabled by CRx/MSR or microcode update - as of today)

Unexplored ways

- Non-temporal hints (MOVNT*)
- Other cache-control instructions (MFENCE/SFENCE, ...)

Exploitation

The devil is in the details

- Guessing physical memory layout
- Flipping the right bit
  - Affected locations tend to be geographically stable (die defect)
- Double hammer vs. single hammer

Mitigations

ECC + Linux MCE policy

- Can correct 1-bit and detect 2-bit errors

Double refresh rate

Software monitoring cache miss with perf counters

pTRR / TRR: [pseudo] Targeted Row Refresh

- Specified by DDR3/DDR4 standards

MAC (Maximum Activate Count)

TODO

Other memory access vectors?

- DMA
- GPU memory
- Hidden cache-bypassing instructions?

Vendor-specific mitigations?

- Dell RMT ("Reliable Memory Technology")

Embedded devices?

- ARM, MIPS, PPC, microcontrollers, ...

Damaging physical memory?

- http://en.wikipedia.org/wiki/Hot-carrier_injection

References

Original research

- https://github.com/CMU-SAFARI/rowhammer

Google research

- http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html
- https://github.com/google/rowhammer-test

Vendor(s) statements

- http://support.lenovo.com/us/en/product_security/row_hammer
- http://azure.microsoft.com/blog/2015/03/16/microsoft-azure-uses-error-correcting-code-memory-for-enhanced-reliability-and-security/
- http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20150309-rowhammer
- http://h20564.www2.hp.com/hpsc/doc/public/display?docId=emr_na-c04593978