# Computer Architecture HW1

Fabian Wüthrich

October 5, 2020

1. Critical Paper Reviews (1000 points)

   see here

2. Main Memory (150 points)

   (a) The memory access pattern of application A could induce more *row buffer misses* whereas application B has more *row buffer hits*. As a *row buffer miss* has a higher latency as a *row buffer hit*, a memory request of application A takes longer.

   (b) A *row buffer miss* consumes more energy as a *row buffer hit* (e.g. precharge of bitlines necessary). As application A has more *row buffer misses* it will also use a larger amount of memory energy.

   (c) The scheduling algorithm in the memory controller is most likely FR-FCFS. This algorithm could prioritize application B (*memory performance hog*) because it has a better row buffer locality degrading the performance of application A.

   (d) Accessing and refreshing a row in DRAM is actually the same process so this policy prevents unnecessary refreshes while still maintaining a retention time of 64ms which is good for performance and energy consumption. The downside of this policy is that additional circuitry is required to keep track of the last access time.

   (e) Application B seems to open many different so the new policy will not refresh these rows again. Application A has a good row buffer locality and doesn't open many rows which increases refresh energy consumption. It is important to note that the total memory energy consumption of application B is still higher because of the many row buffer misses.

3. DRAM Refresh - Utilization (150 points)

   (a) We have 4 channels×2 ranks×8 banks×32K rows $= 2^{21}$ rows in total and $\frac{1.024s}{64ms} = 2^4$ refreshes per row. Thus, $2^{25}$ refreshes across all four channels.

   (b) The command bus of each memory channel is occupied for $2^{23} \cdot 5ns$ by refreshes. For a time-span of $1.024s$ this gives an utilization of $\frac{2^{23} \cdot 5ns}{1.024s} = 4.096\%$

   (c) A refresh uses only the command bus so the data bus utilization is 0.

   (d) Each bank issues $2^{15}$ banks $\times$ $2^4$ refreshes/bank $= 2^{19}$ refreshes and each refresh takes $40ns$. Therefore, we have an utilization of $\frac{2^{19} \cdot 40ns}{1.024s} = 2.048\%$

   (e) We have $2^5$ rows that are refreshed 16 times, $2^9$ rows get refreshed 8 times and the remaining $2^{21} - 2^5 - 2^9$ rows are only 4 times refreshed. Therefore,

   $$2^5 \cdot 16 + 2^9 \cdot 8 + (2^{21} - 2^5 - 2^9) \cdot 4 = 8391040$$

   refreshes in total.

   Then we get an utilization of $\frac{8391040 \cdot 5ns}{4 * 1.024s} = 1.0243\%$ over all four command buses.

   (f) The new distribution still uses only the command bus so the data bus utilization is zero.

   (g) We have 64 banks so we get a bank utilization of $\frac{8391040 \cdot 40ns}{64 \cdot 1.024} = 0.5121\%$

   (h) The system has to track three different states so we use 2 bits per row. To track the state of all rows we need 2 bits $\times 2^{21}$ rows $= 4$ Mbit

(i) When using bloom filters the number of refreshes per interval stays the same but we have to add the false-positives to the number of rows which are refreshed in each interval. For the $64ms$ interval the bloom filter classifies $2^5$ rows correctly plus $(2^{21} - 2^5) \cdot 2^{-20}$ false-positives so $n_{64} = 2^5 + (2^{21} - 2^5) \cdot 2^{-20}$ rows are refreshed. With a similar reasoning we get $n_{128} = 2^9 + (2^{21} - 2^9 - 2^5) \cdot 2^{-8}$ and $n_{256} = 2^{21} - n_{64} - n_{128}$. Therefore, we get $16 \cdot n_{64} + 8 \cdot n_{128} + 4 \cdot n_{256} = 8423823$ refreshes.

Similar to (e) we get a command bus utilization of $\frac{8423823 \cdot 5ns}{4 * 1.024s} = 1.028\%$

Still no data bus used so zero data bus utilization.

Similar to (g) we get a bank utilization of $\frac{8423823 \cdot 40ns}{64 \cdot 1.024} = 0.5141\%$

4. RowHammer (150 points)

(a) The given code doesn't work because the `STORE` instruction is cached in the row buffer even thought the other caches are circumvented using `CLFLUSH`.

(b) We assume that the MSB of the physical address is use to identify the channel. Then we use the next two bits to address the bank and the next six bits are use to identify the row. The remaining seven LSBs are used to address the individual bytes.

Using this address scheme only **Code 2a** can induce RowHammer errors because both addresses map to the same bank on the same channel (three MSBs are equal). RowHammer requires two aggressor rows on the same bank to circumvent the row buffer.

5. RowHammer Mitigation Mechanisms

(a) We need 18 bits to store $T$ for each row. Then the total number of bits required is $2 \cdot 8 \cdot 2^{15} \cdot 18\text{bits} = 9\text{Mbits}$.

(b) Now we have 16 banks and $2^{16}$ rows per bank so we get $2 \cdot 16 \cdot 2^{16} \cdot 18\text{bits} = 36\text{Mbits}$.

(c) To analyze mechanism A we look at a $17 \times 64ms$ interval. In this interval row `0x5FF02` had 170 K activations so with T = 164 K mechanism A issued one additional activation. Thus, including the other rows with a higher activation count we get

$$\left\lfloor \frac{17 \cdot 10K}{T} \right\rfloor + \left\lfloor \frac{17 \cdot 32K}{T} \right\rfloor + \left\lfloor \frac{17 \cdot 64K}{T} \right\rfloor + \left\lfloor \frac{17 \cdot 73K}{T} \right\rfloor = 17$$

additional activations in $17 \times 64ms$. Note that we ignored the remaining $17 \times 301$ activations because they are distributed across the rows such that $T$ does not exceed for a specific row. In summary, mechanism A produces one additional activation within a 64ms time interval.

For mechanism B we model a random variable $X$ as the number of additional activations in 64ms. $X$ has a binomial distribution with parameters $n = 480K$ and $p = 0.001$. Therefore, we get $\mathbf{E}[X] = 480K \cdot 0.001 = 480$ additional activations within a 64ms time interval.

(d) As the memory access pattern doesn't change the answer is the same as before.

(e) For both techniques the memory controller must know which rows are adjacent to each other. It is a challenge to find two adjacent rows because the memory controller cannot rely on the physical address as they are often remapped within the DRAM chip and manufacturers don't disclose the remapping.

(f) To address this challenge we need intelligent memory controllers and a better interface between the memory controller and DRAM chips.

6. VRL: Variable Refresh Latency (150 points)

(a) With VRL applied a full refresh has latency $T$ whereas a partial refresh has latency $0.6T$. 10% of the rows can tolerate one partial refresh followed by a full refresh so the refresh latency for these rows is $0.1 \times (0.5 \times 0.6T + 0.5 \times T) = 0.08T$. Similarly, we get $0.6 \times (0.75 \times 0.6T + 0.25 \times T) = 0.42T$ and $0.3 \times (0.875 \times 0.6T + 0.125 \times T) = 0.195T$ as new latency for the other two groups. Overall, the new refresh latency of a bank is

$$0.695T$$

(b) In a $128ms$ interval 90% of the rows are refreshed once and 10% of the rows are refreshed twice. A bank is busy for $0.9N \times \frac{0.2}{N} + 2 \times 0.1N \times \frac{0.2}{N} = 0.22ms$ during a refresh. Therefore, we have a refresh overhead of

$$\frac{0.22}{128}$$

(c) With VRL a partial refresh costs $0.12/Nms$ whereas a full refresh still requires $0.2/Nms$. The refresh pattern for every cell is repeated every $4 \times 128ms$. In this timespan 5% of the rows have 1 full refresh and 3 partial refreshes. 65% of the rows have 2 full refreshes and 2 partial refreshes. 10% of the rows have 4 full refreshes and 4 partial refreshes. 20% of the rows have 4 full refreshes. The total time spent for refresh is then

$$0.2\ N \times (0.05N \times 1 + 0.65N \times 2 + 0.1N \times 4 + 0.2N \times 4) + \\ 0.12\ N \times (0.05N \times 3 + 0.65N \times 2 + 0.1N \times 4) = 0.732ms$$

Therefore, we have a refresh overhead of

$$\frac{0.732}{512}$$

with a reduction of $\approx 16.8\%$ compared to the baseline.

7. BONUS: DRAM Refresh - Energy (150 points)

(a) We have a total capacity of $2^{60}$ bytes and each row has a size of $2^{13}$ bytes. Hence, the memory system has $2^{47}$ rows.

(b) The memory system has to guarantee a minimum retention time of $64ms$ so it has to refresh all rows within $64ms$. The memory controller issues a REF command every $7.8\mu s$ so $\frac{64ms}{7.8\mu s} = 8205$ REF commands reach the DRAM chip in $64ms$. Thus, each REF command must refresh $2^{47}/8205$ rows.

(c) To calculate the refresh power we use Eq. 18 from the Micron Technical Note.

$$P_{ds}(REF) = (IDD5 - IDD3N) \times V_{DD}$$

From the datasheet we get $V_{DD} = 1.5V$ and if we assume DDR3-1866x16 we get $IDD5 = 175mA$ and $IDD3N = 55mA$ from Table 19. Therefore,

$$P_{ds}(REF) = (175mA - 55mA) \times 1.5V = 180mW$$

(d) For a refresh cycle ($64ms$) we have $E_{ref} = 180mW \cdot 64ms = 11.52mJ$.
For a day we get $E_{day} = 180mW \cdot 86400s = 15552J$