Cody Martin

18-447 Homework 6

1.

a) (2^9)\*8 = 4kb

b) 1 in the VPN because we have 10 bits for the page offset and we have 3 for the BiB and 8 for the index, so 11 - 10 = 1.

c)

2.

a) Minimum N = 2, assuming N is the number of banks per chip in the rank;

b) Minimum N = 4, assuming N is the number of banks per chip in the rank;

c) It is not possible because even if you have many banks, the initial values will never be able to have a hit because nothing is in the row buffer for each bank prior to the sequence. If each bank’s row buffer was pre loaded, then it would be possible.

3.

a) To get 5/15, we need 32 banks because we need 4, 8, 16, 32, and 64 to all go into different banks so that when we access them again, they all have a row hit because they were the last accessed row in their respective banks.

b) To get 7/15, we need 128 banks, because now we need every number to go into their own bank, so when we access them all again, their rows hit. The row for 256 does not hit because it is only accessed once, whereas every other number, when accessed the second time, causes a row hit, making 7 hits out of the 15.

4.

4.1

a) An alternative way of allowing parallel access is true multi-porting, where you have multiple ports to the memory. We could also have two copies of the cache, or virtual multi-porting where you timeshare one port.

b) This is called a Non-Blocking Cache.

c) The structure that keeps the bookkeeping for the cache misses is the Miss Status Handling Register.

d) DRAM has higher density, therefore it has a smaller cell.

e) For SRAM, each cell requires 6 transistors, whereas DRAM requires one transistor and one capacitor.

4.2

a) From the cache access time line, we can immediately narrow the block size to 8-32 bytes since 0xffbc6f hits (meaning its the same block as 0xffbc67), and 0xffbc70 misses (meaning its a different block than the previously mentioned addresses). Since 0xffbc70 is the start of a block, block sizes cannot be 32 bytes, and since 0xffbc67 and 0xffbc6f are in the same block, block size cannot be 8 bytes. Therefore, cache block size must be 16 bytes. Since it is 16 bytes per block, we need 4 bits to index all the bytes. Therefore, bits [3:0] of the address are used to index the cache block offset.

b) From the cache access time line we can see from the bank conflicts that there are 4 banks in the L1-cache. This means that in the physical address, bits [5:4] are used to index the cache banks

c) From the Memory access time line we can see from the bank conflicts that we need at least 4 banks in the main memory. However, we need 3 bits to properly represent the banks with the addresses given. So there are actually 8 banks in main memory represented by bits [12:10] in the physical address.

d) This system is implemented with row interleaving.

e) We need 2048 rows per bank Because we have 11 bits indexing the bank rows. Bits [23:13] are used to index the rows.

f) There are 64 columns per row per bank. This is represented by bits [9:4] of the physical address.

5.

5.1

a) Channel 0:

Read 15

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Read 15

Precharge + Activate + Read 45

Precharge + Activate + Read 45 Application A Done

Precharge + Activate + Read 45 Application B Done

Precharge + Activate + Read 45 Application C Done

Channel 1:

Read 15

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45

Precharge + Activate + Read 45 Application B Done

Precharge + Activate + Read 45 Application A Done

Precharge + Activate + Read 45 Application C Done

Application A is stalled for 420ns

Application B is stalled for 390ns

Application C is stalled for 465ns

` b) Application A is stalled for 290ns

Application B is stalled for 285ns

Application C is stalled for 335ns

c)

d)

5.2

a)

b)

c)

d)

6.

a) Yes, MagicRAM is non-volatile. So the memory controller does not have to precharge the rows which helps access time quite a bit.

b) Yes, MagicRAM costs about that of DRAM, which means it costs much less than SRAM. Therefore it is a cheaper solution. Also, the density will be much higher since the density of DRAM is higher than that of SRAM.

c) Yes it does. You could add it in parallel with the DRAM. Now when we have an application running, if power is lost, the MagicRAM will hold its values, so the application is not lost.

d) ??