Controlling DRAM

Consider a computer with a single DIMM containing eight 1 MB DRAM chips (8 MB total capacity). Each DRAM chip has one bank and row size 2 kilobits (256 bytes). As discussed in class, the DIMM is connected to the memory controller via a 64-bit bus, with 8-bits per cycle transferred from each chip. Assume that:

- Contiguous 1 MB regions of the physical address space are mapped to a single DRAM chip. 256 consecutive physical address space bytes are in a row. 1048576 consecutive bytes fill a DRAM chip.
- Physical address 0 maps to chip 0, row 0, column 0. Physical address 1048576 maps to chip 1, row 0, column 0, etc.

Given these assumptions, reading a 64-byte cache line beginning at address X requires the following memory-controller logic, presented in C code below: (the data ends up in cache_line)

```c
char cache_line[64];

// compute DRAM clip, row, col for address X
int chip = X / 1048576;
int row = (X % 1048576) / 256;
int col = (X % 1048576) % 256

for (int i=0; i<64; i++) {
    // Read one byte from each DRAM chip at given row and column (eight in total)
    // so that the byte from chip j ends up in ‘from_dram[j]’. Assume necessary
    // DRAM row and column activations are performed inside DIMM_READ_FROM_CHIPS.
    char from_dram[8];
    DIMM_READ_FROM_CHIPS(row, column, from_dram);
    cache_line[i] = from_dram[chip];
    column++; // move to next byte in column
}
```

Questions are on next page...
A. (2 pts) Explain why 64 iterations (64 reads from the DRAM chips) are required to populate the buffer cache_line.

B. (4 pts) Now assume the address space is **byte-interleaved across the DRAM chips** as discussed in class and shown in the Figure below. (Byte X in the address space is stored on chip X % 8.) Please provide C-like pseudocode for reading the 64-byte cache line at address X from DRAM into cache_line. Your code should make a series of calls to DIMM_READ_FROM_CHIPS.

![Diagram showing memory layout and DRAM chip organization](image)

**Hint:** Recall that each DRAM chip row is 256 bytes.
C. (2 pts) How much higher “effective bandwidth” is achieved using the interleaved mapping from part B than the original blocked mapping from part A?

D. (2 pts) Imagine the byte interleaved memory system from part B is connected to a dual-core CPU. The memory controller uses a naive round-robin policy to schedule incoming memory requests from the cores (it services a request from core 0, then core 1, then core 0, etc.) All requests from the same core are processed in FIFO order.

Both cores execute the following C code on different 4MB arrays. Simply put, each thread is linearly scanning through different regions of memory.

```c
int A[N]; // let N = 1M, so this array is 4MB
int sum = 0; // assume 'sum' is register allocated
for (int i=0; i<N; i++)
    sum += A[i];
```

Assume that the cores request data from memory at granularity of 8 bytes. On this system, you observe that when running two threads, the overall aggregate bandwidth from the memory system is lower than when one thread is executing the same code. Why might this be the case? (Hint: we are looking for an answer the pertains to DRAM chip behavior: consider locality, but which kind?)

E. (2 pts) Why might performance improve if the granularity of each memory access was increased from 8 to 64 bytes?
Building a Mesh Interconnect

You are building a **packet-switched mesh interconnect** for a nine-core processor. The cost of the network is $W$ units per wire, $S$ units per switch, and 1 unit for every four bytes of buffering in each switch. Assume that the network packet size is 64 bytes and that a packet header is only 4 bytes (the payload and tail are 60 bytes).

A. (2 pts) What is the **minimum possible cost** of the network if it is designed to use **store-and-forward routing**? (Assume the network cannot drop packets under heavy congestion.)

B. (2 pts) What is the latency of transmitting a packet between cores that are **farthest apart** on the network? Assume a link can transmit 4 bytes per cycle (A packet can be communicated over one link in 16 cycles.)

C. (2 pts) What is **minimum possible cost** of the network if it is designed to use **cut-through routing**? (Assume the network cannot drop packets under heavy congestion.)

D. (2 pts) Using **cut-through routing**, what is the latency of transmitting a packet between cores that are **three hops** apart on the network? Assume a link can transmit 4 bytes per cycle, and assume there is no network contention, and no extra delays inside the switches.