EECS 570 End Term Exam
Winter 2025

Name: _____ Uniqname: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

#	Question	Points
1	Short Answers	/ 25
2	Systolic Array	/ 15
3	On-Chip Network Topology	/ 20
4	GPU Architecture	/ 15
5	GPU Programming	/ 9
6	L2 Cache Coherence	/ 16
Total		/ 100

NOTES:

- **Calculators are allowed, but no PDAs, portables, cell phones, etc.**
- **Don't spend too much time on any one problem.**
- **You have 120 minutes for the exam.**
- **There are 12 pages in the exam (including this one). Please ensure you have all pages.**
- **Be sure to show work and explain what you've done when asked to do so.**

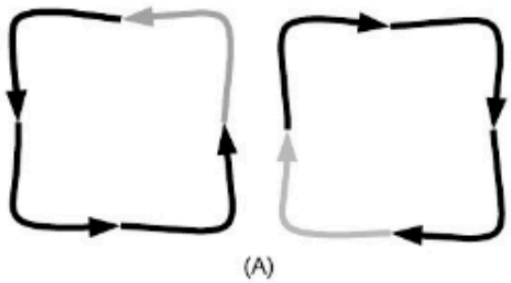
1. Short Answers [25 points]

a) State one advantage and one disadvantage of adaptive routing compared to deterministic routing. [3 points]

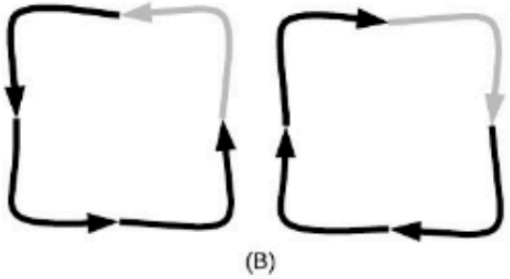
b) State two uses for virtual channels. [3 points]

c) Consider a 16-node 2D torus interconnect with minimal (shortest-path) routing. Assume a uniform random traffic pattern, where each node sends packets to all nodes, including itself, with equal probability. What is the average number of hops a packet traverses, counting 0 hops for messages sent to itself? [4 points]

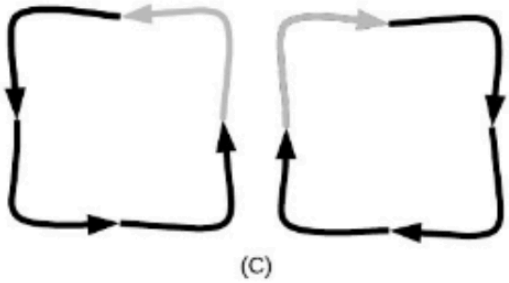
c) The following four figures show various routing turn models, in which the grayed-out turns are forbidden. Classify each of the following turn models(s) as deadlock or deadlock-free (circle the right answer) [8 points]



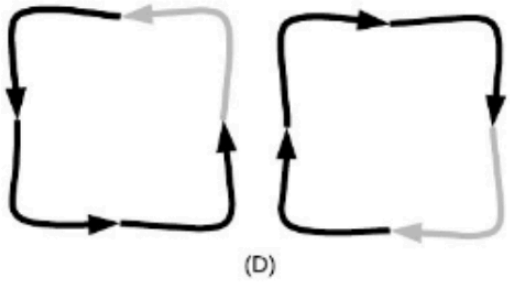
(A)



(B)



(C)



(D)

A: Deadlock / Deadlock-free

B: Deadlock / Deadlock-free

C: Deadlock / Deadlock-free

D: Deadlock / Deadlock-free

d) Briefly explain a non-predictive mechanism that GPUs use to hide memory latency. [4 points]

e) Explain one solution to reduce the number of pipeline stages in a router microarchitecture [3 points]

2. Analyze a 3×3 Systolic Array [15 points]

You are given a 3×3 systolic array that can multiply two 3×3 matrices, A and B.

$$C_{i,j} = \sum_{k=0}^{2} a_{i,k} \cdot b_{k,j}$$

Dataflow:

Each element in the input matrix moves from one PE to another as follows:

- Each row of A flows **horizontally** across the array (from left to right).
- Each column of B flows **vertically** down the array (from top to bottom).

Processing Elements (PEs):

Every cycle,

- Each PE **multiplies** the two new input values it receives from its neighbours.
- Each PE **accumulates** the product into a local register that holds the partial sum.
- The **partial sum** stays **inside** the same PE (output stationary)

A PE does not compute in a cycle when it doesn't receive valid inputs.

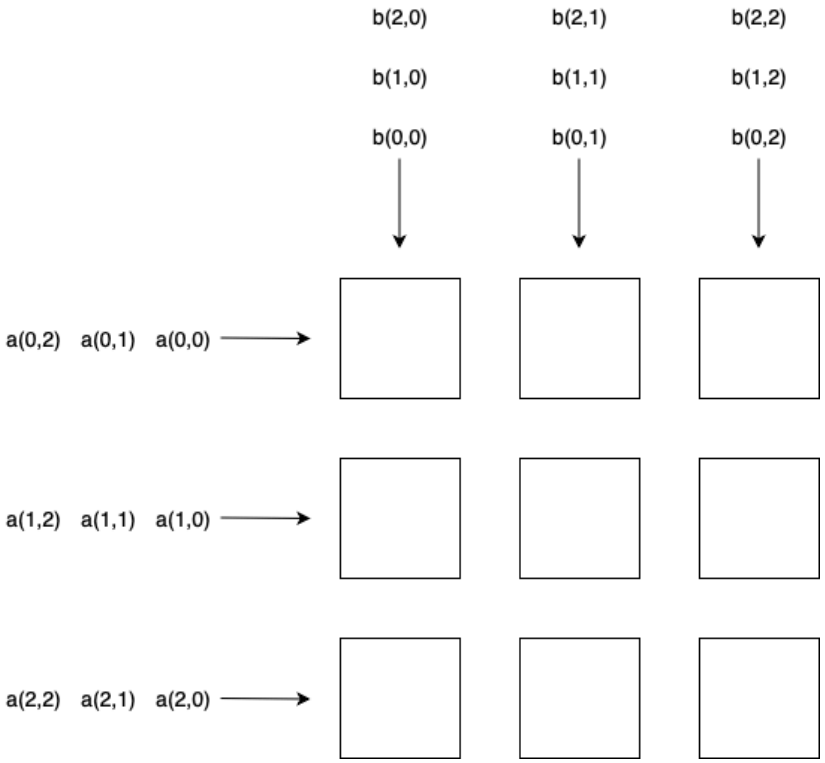
Completion:

Once all the input matrix data has passed through the systolic array, PE(i,j) contains the final sum for C(i,j).

Assume:

- At time t = 0, the systolic array is completely empty.
- Example: At t=1, PE(0,0) computes A(0,0) * B(0,0), and adds it to its partial sum.

(a) Determine the cycle in which PE (i,j) will compute the first valid product and accumulate it to its partial sum. Fill the following grid with those cycle numbers. Briefly explain. [8 points]

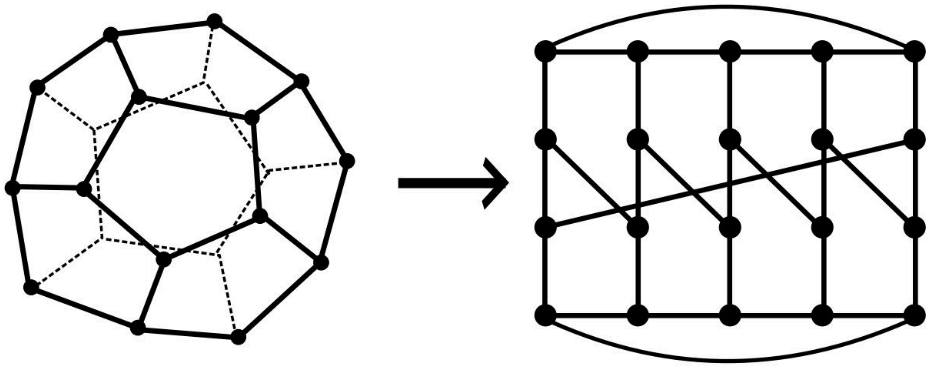


(b) Compute the latency of one matrix multiplication operation. Explain. [3 points]

(c) Compute the throughput of the systolic array in terms of MAC (multiply-accumulate) per cycle. [4 points]

3. On-Chip Network Topology [20 points]

Several topologies have been investigated for on-chip networks. One interesting proposal calls for designing NoCs based on a dodecahedron.



A dodecahedron is a polyhedron (three-dimensional polygon) with twelve faces, wherein each face forms a regular pentagon. The left figure shows a 3D dodecahedron as seen from above, dotted lines indicating edges on the hidden faces. A router can be placed at each of the 20 vertices of the shape. The right figure illustrates an equivalent planar topology–how the topology can be laid out in a regular grid on a 2D plane. This is how an actual chip using such a topology would be connected.

Compare the 20-node dodecahedral NoC against a 20-node 5×4 mesh network.

(a) List two advantages that the dodecahedral NoC has over the mesh network. [4 points]

(b) Calculate the longest paths (in terms of hops) for both topologies and explain how this impacts each design. [4 points]

(c) Design a deadlock-free routing algorithm for the dodecahedral topology. Assume you have only one message class. You can use at most two virtual channels.

Hint: Reason using following directions for packets in 3 dimensions: top, bottom, clockwise, counter-clockwise. [12 points]

Algorithm:

Argue why your algorithm is deadlock-free

4. GPU [15 points]

(a) Why do GPUs have thousands of small cores instead of a few powerful cores like Intel CPUs? [3 points]

(b) State one performance metric where GPU underperforms a CPU [2 points]

(c) Describe briefly how GPUs enable fast warp-level switching [3 points]

(d) Specify **two main** reasons why a similar functionality (fast thread context switch) is less beneficial for CPUs, so much so that none of the commercially available CPU hardware have that support. [3 points]

(e) A GPU has a memory bandwidth of 900 GB/s and can perform 20 TFLOPS (20×10^12 floating-point operations per second). Calculate the arithmetic intensity threshold (in FLOPS/byte), above which an application is compute-bound. [4 points]

5. CUDA Programming [9 points]

(a) Consider the following CUDA kernel that processes an array of integers:

```cuda
__global__ void conditionalVectorAddition(int idx, int* data, int* result) {
    int idx =      // assume idx is correctly
                   // initialized using blockDim, blockIdx, threadIdx
                   //  such that thread_i will have idx = i

    if (idx < data[idx]) {
        result += data[idx];
    }
}
```

Assume this kernel launches a single block with a size of 8 threads (smaller than actual warps) and the following values in the `data` array:

[5, 3, 8, -6, 11, 9, -1, 14]

a) Calculate the SIMD utilization efficiency for this warp. Briefly explain.
(SIMD utilization is the percentage of threads that are actively doing useful work during execution) [4 points]

(b) Consider a CUDA kernel that processes a matrix of dimensions width × height, where each thread computes a single element of the matrix.

Write an expression for the corresponding row and column indices (i, j) in the matrix that this thread should process. Show your work. [5 points]

Notes:
- You may refer to the CUDA variables: blockDim, blockIdx, threadIdx
- The formula should ensure each thread processes a unique matrix element

6. L2 Cache Coherence in a 4-Core System [16 points]

Consider a 4-core system with a 3-level inclusive memory hierarchy, consisting of L1, L2, and Main Memory. Each core has a private L1 cache, while L2 caches are shared as follows:

- Core 0 and Core 1 share one L2 cache
- Core 2 and Core 3 share another L2 cache

The system employs a bus-based snooping MSI protocol at each cache level to maintain coherence. There are two buses in the system:

- L1 Bus: Connects the L1 caches within the same L2 domain.
- L2 Bus: Connects the L2 caches and interfaces with main memory.

To differentiate bus transactions occurring at each level, we use suffixes:

`_L1` for L1 bus transactions;
`_L2` for L2 bus transactions.

For example

`BusRd_L1` represents a Bus Read (BusRd) request on the L1 bus
`BusRd_L2` represents a Bus Read (BusRd) request on the L2 bus.

List of Possible Bus Actions

L1 Bus Transactions

FwdBusRd_L1

- Forwarded request due to BusRd on L2. If an L1 cache line is in the Modified state, it transitions to Shared and provides the data on the bus.

Following transactions are the same as taught in lectures:

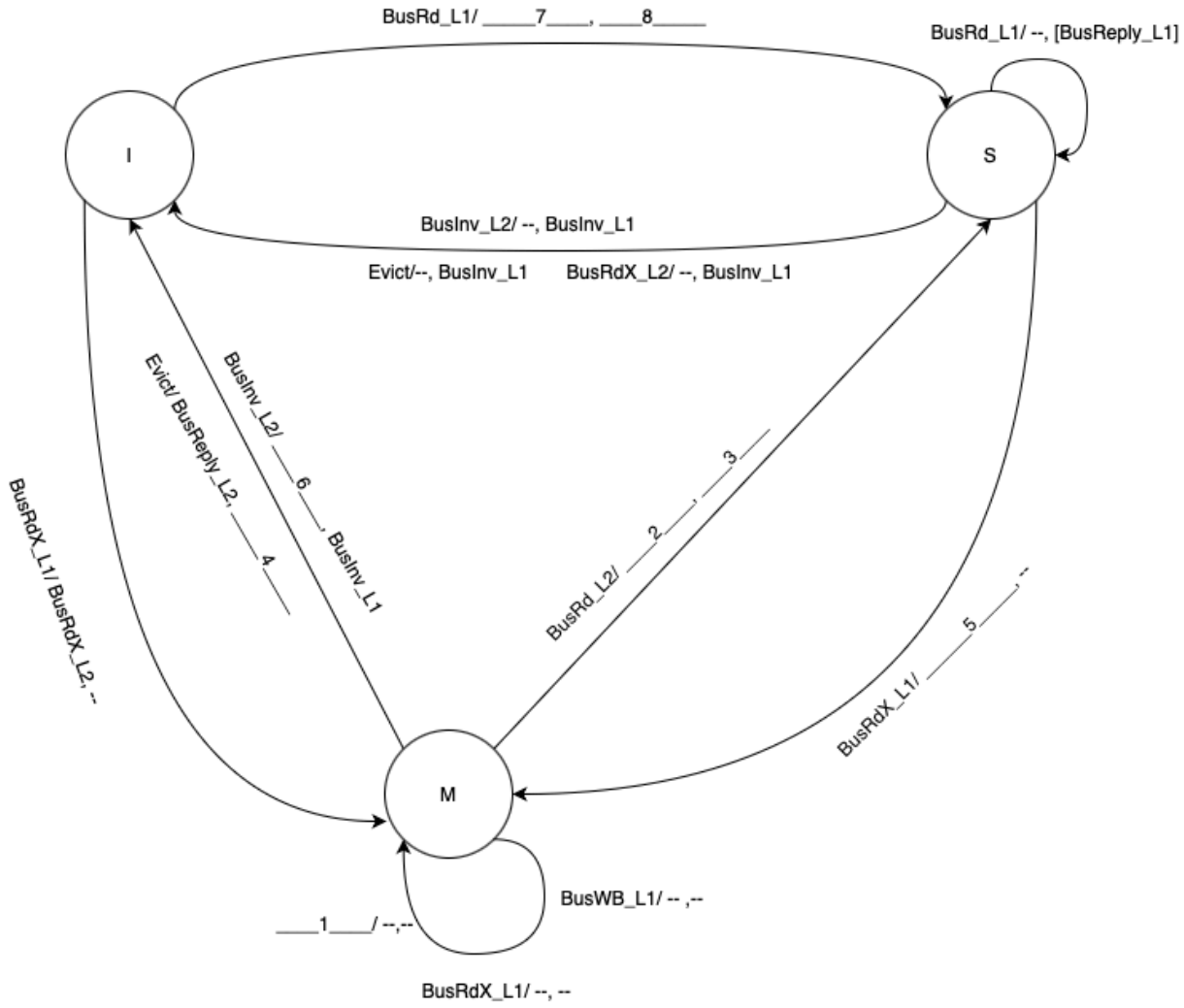
BusRd_L1, BusRdX_L1, BusWB_L1, BusInv_L1, BusReply_L1

L2 Bus Transactions

Following transactions are the same as taught in lectures

BusRd_L2, BusRdX_L2, BusWB_L2, BusInv_L2, BusReply_L2

Fill in the blanks for the state transition diagram of an **inclusive** L2 cache. Consider each transition to be atomic. Each transition in the state diagram is labeled as: ***Action / L2 bus transaction, L1 bus transaction***. '--' is used if no bus transaction occurs.



1: _____ 2: _____

3: _____ 4: _____

5: _____ 6: _____

7: _____ 8: _____