EECS 570 Final Exam
Fall 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	Custom Accelerators	/ 15
2	GPU	/ 30
3	Hardware Multithreaded - SMT	/ 20
4	Multiple Network-on-Chip	/ 20
5	Coherence Verification	/ 15
Total		*/ 100*

NOTES:

- **Calculators are allowed, but no PDAs, portables, cell phones, etc.**
- **You can use one page of notes (double-sided)**
- **Don't spend too much time on any one problem.**
- **You have 120 minutes for the exam.**
- **Be sure to show work and explain what you've done when asked to do so.**
- **The exam has 16 pages. Make sure you have all of them.**

1) Custom Accelerators [15 points]

Consider **matrix multiplication** of:

- Matrix **A** with dimensions **M × K**
- Matrix **B** with dimensions **K × N**
- Resulting matrix **C** with dimensions **M × N**

The accelerator uses a **4 × 4 systolic array** to compute **4 × 4 output tiles of matrix C**.

Architecture assumptions:

- Output-stationary dataflow
- One MAC per processing element (PE) per cycle
- Matrix A enters from the left, matrix B enters from the top
- No stalls
- Clock frequency is 1 GHz

Q1. In output-stationary dataflow, each PE primarily stores:

A) row of A B) column of B C) partial sum of one C element D) output tile

Q2. A 4×4 systolic array has _____ PEs.

Q3. Which dimension of matrices is the reduction dimension streamed over time?

A) M B) N C) K D) all

Q4. Peak compute throughput of a 4×4 array (MACs/cycle)?

A) 4 B) 8 C) 16 D) 32

Q5. Peak throughput at 1 GHz is _____ GMAC/s based on Q4.

Q6. Primary cause of pipeline fill latency in a systolic array?

A) decode B) cache miss C) data propagation D) register read

Q7. Total number of MAC operations required to compute one 4×4 output tile when K = 3:

_____

Q8. For A (64×128) × B (128×64), number of 4×4 output tiles:
A) 64 B) 128 C) 256 D) 512

Q9. Key reason systolic arrays fit ML workloads:
A) large caches B) regular local data movement C) no DRAM access D) OoO exec

Q10. In output-stationary dataflow, each output element is written back to memory _____ time(s).

Q11. A key advantage of domain-specific accelerators over CPUs is:
A) higher single-thread latency
B) better efficiency for targeted workloads
C) support for all workloads without recompilation
D) better memory system performance

Q12. Accelerators are least effective for workloads that:
A) have regular data access patterns
B) have high arithmetic intensity
C) are memory-bound with low data reuse
D) use low-precision arithmetic

Q13. Which program characteristic most strongly determines whether the one-time overheads of using a custom accelerator (e.g., data movement, setup, kernel launch) can be effectively amortized?

A) Size of the input data
B) Degree of control-flow divergence
C) Amount of computation per invocation
D) Use of low-precision arithmetic

Q14. Which statement best captures a fundamental tradeoff in designing domain-specific accelerators?
A) Specialization improves efficiency but increases design, verification, and software cost
B) Efficiency mainly comes from smaller caches and lower miss rates
C) Specialization improves both efficiency and generality
D) Accelerators are always smaller because control logic is eliminated

Q15. What limitation of custom accelerators does processing-in-Memory (PIM) primarily aims to address?

2) GPU architecture [30 points]

You are given a modern SIMT GPU with the following simplified parameters:

- Warp size: **32 threads**
- SM resources:
    - Max warps per SM: **64**
    - Max threads per SM: **2048**
    - Register file per SM: **256 KB**, registers are **4 bytes each**
    - Shared memory per SM: **64 KB**
- Memory system:
    - Global memory latency: **400 cycles**
    - L2 cache hit latency: **100 cycles**
    - Peak DRAM bandwidth: **800 GB/s**
- Clock frequency: **1.5 GHz**
- Each SM can issue **1 warp instruction per cycle**
- Memory system supports **perfect coalescing** when accesses are aligned and contiguous

A kernel is launched with **256 threads per block.** Each thread uses:

- **40 registers**
- **2 KB shared memory per block**

The kernel executes the following loop:

```
for (int i = 0; i < 128; i++) {

    A[idx] = B[idx] + C[idx];

}
```

Assume:

- `idx` is thread-unique
- All arrays are in global memory
- All accesses are **coalesced**
- No cache reuse across iterations

(a) Occupancy Analysis (10 points)

What is the **maximum number of active warps per SM** for this kernel? Clearly show which resource is the limiting factor.

(b) Latency Hiding (4 points)

Assume **all global memory accesses miss in L2**.

Is the achieved occupancy from part (a) **sufficient to fully hide global memory latency**? Justify numerically.

(c) Bandwidth vs Compute Bound (6 points)

Assume:

- One floating-point add per loop iteration
- Global memory access requires **2 loads + 1 store per iteration**
- Ignore instruction overhead other than memory and FP add

Arithmetic intensity (FLOPs per byte of global memory traffic, counting all loads and stores, and assuming one floating-point add = 1 FLOP): _____

Max achievable performance limited by memory bandwidth (in GFLOPs/s, counting all global loads and stores): _____

Determine whether this kernel is **memory-bound or compute-bound**.

(d) Memory-Level Parallelism and Warp Scheduling (10 points)

Assume the same kernel and occupancy from **part (a)**

Additional GPU details:

- Each warp can have at most **one outstanding global memory request**
- Each SM supports at most **16 concurrent outstanding global memory requests**
- Global memory latency is **400 cycles**
- The warp scheduler can issue one ready warp per cycle

(i) What is the **maximum number of concurrent global memory requests per SM** for this kernel? [3 points]

(ii) Given this limit, how many cycles of the **400-cycle memory latency** can be hidden by the SM? [4 points]

(iii) Would adding more active warps make this kernel faster? Why or why not? [3 points]

3) Hardware Multi-threading -- SMT (20 points)

Simultaneous Multithreading (SMT) is a processor technique in which multiple hardware threads share a single core and can issue instructions in the same cycle, allowing the core to execute instructions from different threads simultaneously to improve utilization of execution resources.

a) SMT can improve performance for workloads that are limited by memory bandwidth. **True / False [1 pt]**

b) Even a wide out-of-order core cannot fully utilize all execution resources when running a single thread. What fundamental limitation of out-of-order execution does SMT address? **(3 pts)**

c) Does enabling SMT usually increase or decrease throughput of a set of threads? Why? **(3 pts)**

d) Does enabling SMT usually increase or decrease latency of a thread? Why? **(3 pts)**

e) In an SMT processor, the instruction fetch unit must choose one hardware thread to fetch from each cycle. Consider a policy that always selects the thread with the fewest instructions currently in the pipeline (ICOUNT).

Why can this policy improve overall throughput compared to fetching threads in a round-robin manner? **(3 pts)**

f) A GPU warp scheduler typically selects the next ready warp in a round-robin fashion, rather than prioritizing warps with fewer in-flight instructions.

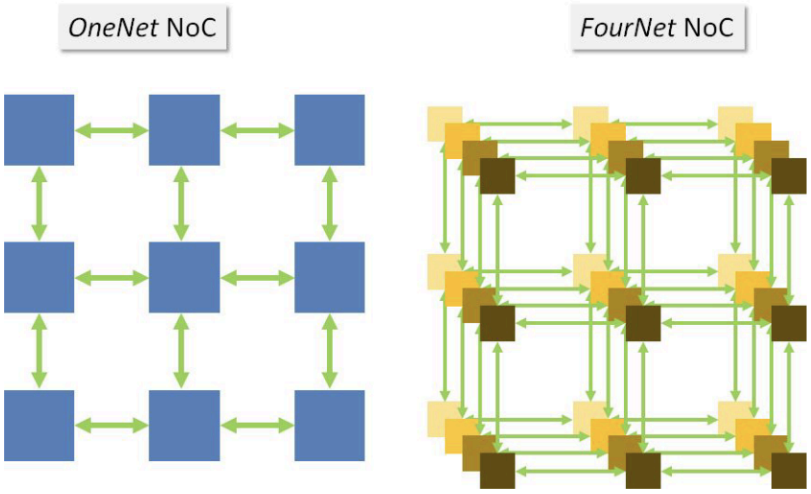
Why does round-robin scheduling work well for GPUs, even though a policy like the one described above is beneficial for SMT CPUs? **(3 pts)**

g) In a power-constrained (dark silicon) system where area is plentiful, does enabling SMT always improve energy efficiency? Why or why not? **(4 pts)**

4) Multiple Network-on-Chip Design [20 pts]

Consider a 3×3 mesh NoC (**OneNet**) with a conventional 5-stage router pipeline, 4 virtual channels (VCs), and 32-bit links.

We extend this to **FourNet**, which consists of four independent 3×3 mesh subnets. FourNet has four times as many routers as OneNet, but the total router buffer capacity and link wires are the same and are evenly partitioned across subnets (e.g., a 4 KB buffer in OneNet becomes 1 KB per subnet router). A node may inject packets into any subnet, and once injected, all flits of a packet remain in the same subnet until reaching the destination. Assume both designs use the same packet size.



a) OneNet uses 32-bit links. FourNet divides total link wires equally among 4 subnets, so each subnet link is 1/4 the width of OneNet. Assume packets are 32 bits in both OneNet and FourNet (packet size fixed in bits), and assume flit size equals phit size.
(flit: flow-control unit; phit: physical transfer unit) **[3 pts]**

i) What is the link width of a subnet in FourNet? _____

ii) What is the flit size? OneNet: _____ FourNet _____

iii) What is the number of flits in a packet? OneNet: _____ FourNet _____

b) **OneNet** uses **4 virtual channels (VCs)** solely to ensure **protocol-level deadlock freedom** by assigning dependent message classes to different VCs.

In **FourNet**, packets can be injected into any of four physically separate subnets and remain in the same subnet end-to-end.

Do you still need **virtual channels** in FourNet to ensure **protocol-level deadlock freedom**? Why or why not? **[3 pts]**

c) In **FourNet**, a node may inject a packet into any of the four subnets. Routers can sense congestion locally and exchange congestion information **only with their immediate neighbors**.

Assume the source node selects the subnet that is predicted to be least congested.

Describe **two distinct, concrete heuristics** a router can use to estimate congestion in a subnet using **only local and neighbor information**. Clearly state what metric is measured and how it reflects congestion. **[4 pts]**

d) Consider a node in **FourNet** sending two packets, **P1** and **P2**, to the same destination. To balance load, **P1** is injected into **subnet 0**, and one cycle later **P2** is injected into **subnet 1**. **[4 pts]**

(i) Explain how **P2** can arrive at the destination before **P1**, even though it was injected later.

(ii) Describe **one mechanism** to guarantee **in-order delivery** without adding reordering buffers at the destination.

(e) Components that remain idle can be **power-gated** to save energy, but waking a gated component incurs a latency of **tens of cycles**.

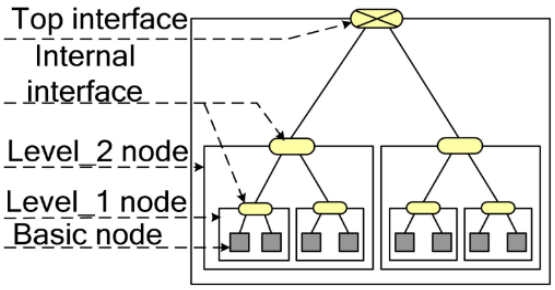
Compare OneNet and FourNet in terms of their suitability for power-gating.

Which configuration is better suited for effective power-gating, and why? **[5 pts]**

5) Cache Coherence Verification [15 pts]

A major challenge in designing cache coherence protocols is scaling their verification to modern many-core systems. As the number of processors increases, the system state space grows explosively. An even more fundamental issue is that adding more nodes can expose new behaviors that do not arise in smaller systems. As a result, a protocol is often verified only for instances up to **N** nodes.

Design for scalable verifiability. One approach is to view the system as a logical tree structure, defined as follows:



- **Basic Node:** A leaf in the tree (grey squares), consisting of a set of cores along with their caches and memories.

- **Interface Node:** An internal node (yellow ellipses) that logically composes the behavior of its child nodes.

A top interface logically composes the behavior of its child basic nodes; this composition forms a Level_1 node. Recursively, two or more Level_{n−1} nodes joined by a top or internal interface compose a Level_n node, where **n** is the height of the tree. We assume a binary tree.

The key idea is to identify a minimal system structure that can be composed to form any larger system by expanding the basic node(s) into another instance of a component of the minimal system. If this minimal system is verified to be cache coherent, and if the interface preserves self-equivalence, then the overall system is formally verified to be cache coherent for an arbitrary number of nodes.

a) Design the interface behavior for ***self-equivalence*** for MSI coherence protocol by answering the following.

For a cache block, how should the logical coherence protocol state (M/S/I) for an interface node be assigned based on the coherence state of its children? It should capture the equivalent coherence states of all its child nodes.

Hint: What should be the logical protocol state of a Level_1 interface node if its left child node is Modified and its right child node is Invalid? Think through other possible combinations. [5 pts]

Fill in the following table to specify the interface state for each combination of child states.

Child 1	Child 2	Interface

Explain:

b) Assume this is a directory-based protocol. For a cache block, what additional information (if any) must be stored in the interface node for each coherence state? (M/S/I) [5 pts]

Composite state of the interface

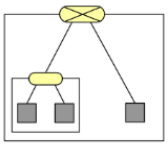
M: _____

S: _____

I: _____

c) For a system using binary trees, reason why the following structure is the ***minimal system***. You should argue (i) how it is possible to compose an arbitrarily large system using only components of this structure, and (ii) why that is not possible using any smaller structure.

Hint: How would you compose a system with 6 nodes? Drawing this may help your reasoning. [5 pts]



<<<< BLANK SHEET >>>>