EECS 570 *Final Exam*

Winter 2024

Name: _____ unique name: _____

Sign the honor code:

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

_____

Scores:

#	Points
1	**/ 20**
2	**/ 30**
3	**/ 30**
4	**/ 20**
Total	***/ 100***

NOTES:

- Closed book. One 8.5" x 11" page of notes is allowed.
- Calculators are allowed
- Don't spend too much time on any one problem.
- You have about 120 minutes for the exam (avg. 30 minutes per problem).
- There are 10 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 [20 points]

a. A memory system is coherent if there is a total order on all __stores___ to any given address.

b. This isn't enough for consistency. Sequential Consistency (SC) assumes there is a total order among _all operations to all addresses_____

c. Sun's "Relaxed Memory Order" (RMO) allows unordered, coalescing post-retirement store buffers, but the compiler cannot reorder loads between stores.

○ True ● False

d. Sequential Consistency (SC) makes nearly all compiler optimizations illegal, but (in c/c++ programming model) operations within an expression may be reordered.

● True ○ False

e. Order the following four network domains in terms of their latency requirements:

	low latency ==========> High latency			
LANs(Local Area Network)	○	○	●	○
WAN(Wide Area Networks)	○	○	○	●
SANs(Storage Area Networks)	○	●	○	○
OCNs/NoCs(Network on Chip)	●	○	○	○

2. Murphi for MSI [30 points]

Answer the questions on the following page about verifying the MSI protocol using Murphi. You should make the same assumptions as PA2, listed here:

1. It uses an interconnect network that supports only point-to-point communication. All communication is done by sending and receiving messages. **The interconnect network may reorder messages arbitrarily.** It may delay messages, but it will always deliver messages eventually. Messages are never lost, corrupted or replicated. Message delivery cannot be assumed to be in the same order as they were sent, even for the same sender and receiver pair.
2. At the receiving side of the interconnect system, messages are delivered to a receive port. Once a message has been delivered to the receive port, it will block all subsequent messages to this port until the message is read. Consider this behavior equivalent to that of a mailbox with room for only one letter: you have to remove the letter from the mailbox before you can receive the next one. On the sending side, there is no such restriction: you can always send messages. The interconnect system has enough buffer space to queue messages.
3. For the purpose of this assignment, you may assume that there is no limit on the buffer space in the interconnect system. However, your protocol will be considered broken if there is a way to generate an infinite number of undelivered messages. Besides, you will not be able to verify your protocol in this case.
4. You may assume that the interconnect network supports multiple lanes. For each lane, you have a separate set of send- and receive-ports for each unit. Traffic on one lane is independent of traffic in the other lanes. Messages will never switch lanes. Note that using fewer lanes is better.
5. Each processor has a dedicated cache that is **not shared with any other processor**. All caches must be kept coherent by your cache coherency protocol. Processors may issue load and store operations only. Because this assignment only deals with cache coherency and not with consistency issues, **you will be concerned with only one storage location** (address). However, you need to model cache conflicts. To do this, you need to model a third operation besides load and store: a cache write-back. Write-backs normally arise from a cache conflict if the old line is dirty. Write-back operations may occur at any time between any pair of load/store operations. If the cache is in a clean state, you may simply set it to be invalid or take the appropriate action according to your CC protocol. Cache replacements of dirty lines must obviously write the line back to memory.
6. You should assume that the coherency unit is equal to one word and that all loads and stores read or write the entire word.
7. Besides processors with their caches, there is one memory unit in your system. The memory unit has a directory-based cache-consistency controller which ensures that only one processor can write to the memory block at a time (exclusive-ownership style protocol). The directory representation is unimportant for this assignment. You should assume that you have a full directory (bit vector) that can keep track of all sharers.
8. The interconnect system can send messages from any unit to any other unit. It is OK if your protocol requires that a cache controller has to send a message to another cache controller.

a. Assume core 0 is in the S state and core 1 wants to transition from I state to S state. Only core 0 is on the sharer list. What messages are sent if this is a 4-hop system? (from whom to whom, message name, and what information is included)

   1) GetS request Core 1 to Home with address.

   2) Response Home to Core 1 with data.

b. What about a 3-hop system? (Note: PA2 implements a 3-hop system)

   1. GetS request Core 1 to Home with address.

   2. Response Home to Core 1 with data.

c. [Race conditions] Assume core 0 and core 1 both send a GetM message to the home node. What should the Home node do to avoid race conditions?

The home node should stall the GetM message received later until it finishes the first request. For example, if GetM from core 0 arrives at the HomeNode earlier, it should process it, and grant core 0 Modified permission. It should stall the GetM request from core 1 until it receives the MAck from core 0 and confirms that core 0 is in the Modified state.

d. PA2 setup allows multiple virtual channels on the interconnect network. Is it okay if all the messages are sent on a single channel? Why and why not (use examples to explain)?

No. Sending all the messages on a single channel will cause deadlocks. For example, in the situation of question c), stalling the GetM message from core1 will block the Ack message from core0, and cause a deadlock.

e. Processor-initiated rules. Specify all possible rules a processor can fire and what coherence messages are sent when it is in M, S, or I state.

   p.state = **Invalid**

   (example) rule “write request at invalid state.” Send GetM to HomeNode.

   rule “read request at invalid state.”: Send GetS to HomeNode

   p.state = **Modified**

   rule “dirty eviction”: send PutM to HomeNode

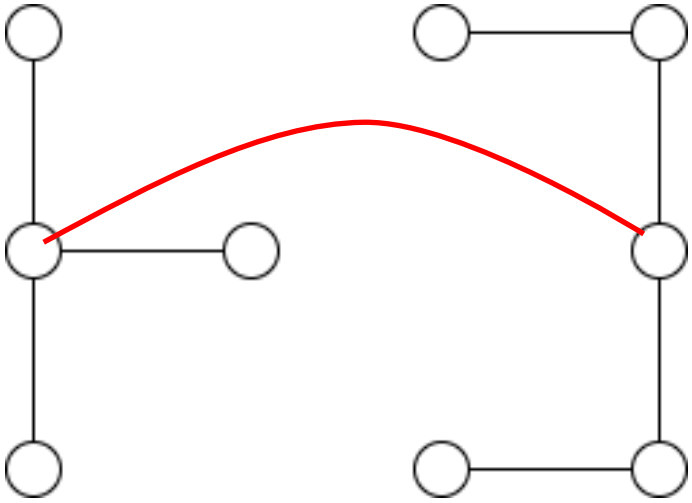
   p.state = **Shared**

   rule “upgrade request”: send GetM to HomeNode

   rule “evict cache in shared state”: send PutS to HomeNode

3) Network-on-Chip [30 points]

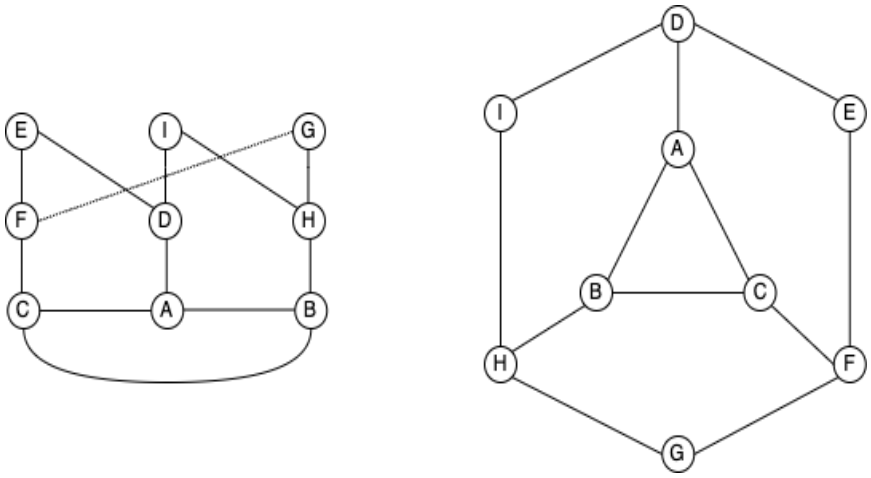
When designing topologies, there are several important properties to consider that can affect NoC performance. One of these considerations is *average hop count* which measures the average number of hops required to travel between any two processors on the chip. Consider the following *incomplete* NoC topology where each node represents a processor and each line represents a link.



a. As a NoC topology designer, what *single link* would you add between *any two processors* in this network to minimize the average hop distance in the NoC? Draw the link above and give a one-sentence explanation. (Note: the length of each link is not a consideration for this question.)

For each subnetwork, we can find the "central node" that routing traffic to minimizes average hop count and then connect those two thereby minimizing average hop distance in the whole network.

In class we have covered many different topologies. One of the most basic topologies is an NxN grid. Researchers at the University of Michigan are investigating alternatives to this using 3D topologies like the one below (these are two representations of the same NoC topology).



b. In one sentence each, name an advantage and a disadvantage of this topology compared to a 3x3 grid topology.

(+) Example: the longest path between nodes is now 3
(–) Example: 3D topologies are more complex to build

Note: path diversity was not a valid advantage or disadvantage in this example. Take a 3x3 mesh vs a 3x3x3 torus topology. You can think of the mesh as being a strict subset of the torus - all of the mesh's links and nodes are contained in the torus, but the torus has far more links and a whole new dimension that messages can travel in hence it offers a strict advantage in path diversity. Both a 3x3 mesh and the 3D topology in this problem have the same number of links and nodes, they are just organized differently. This means that for certain node pairs more possible paths could be routed in the mesh, and for other pairs there are more possible paths in the 3D topology - neither is strictly better.

c. Below is the state of the link queues. Identify the cyclical resource dependency in the network that is causing a deadlock by **circling** the links involved - you can do this on either representation.

Link	A→B	A→C	A→D	B→A	B→H	C→F	C→B	F→E	E→D	D→A
Packet Destination	H	F	D	G	I	E	B	D	A	C

In one sentence explain why this is a cyclical resource dependency:

Each packet in the queue wants to advance to the next link but that link will not be free until it advances the packet currently waiting in the queue - this happens in a cycle that creates a cyclical resource dependency.

d. We now want you to design a deadlock-free routing algorithm for this topology. Assume there is only one message class and you can use at most two virtual channels (Hint: work on making routing deadlock-free within each 2D layer first, then think of how to connect them in a deadlock-free manner.)

(i) How is routing done within each 2D layer? Explain how VCs are used within a layer.

Example answer: always route traffic clockwise in a 2D layer. Select some point in the 2D layer as the dateline - all packets switch from VC0 to VC1 once they pass the dateline.

(ii) How is routing done between each 2D layer? Explain how VCs are used between layers.

` Example answer: On the bottom layer we go clockwise until we reach the first link that can take us up - we take this link then go clockwise again until we reach the destination. For the other case (starting in the top layer), we can immediately take a link down, and then go clockwise until we reach our destination. When routing from the bottom layer to the top layer we always use VC0 and when routing from the top layer to the bottom layer we always use VC1.

e. Argue why your algorithm is deadlock-free in 3-4 sentences.

My messages always travel clockwise on a 2D layer, meaning any resource dependency would have to happen in a ring spanning the entire 2D layer. My algorithm won't have a resource dependency on a 2D layer because the dateline breaks any dependency that travels one direction in a ring.

Additionally, it is not possible for there to be a deadlock between 2D layers because all messages traveling up and all messages traveling down use different VCs, meaning that independent of datelines there will never be a cycle on the same VC that goes up and then comes back down (and vice versa).

4) Consistency [20 points]

Given the following litmus test, for each given output value set, mark (by shading the box) whether it is a valid outcome or not for the different consistency models.

Core 0	Core 1	Core 2	Core 3
(i1) a = 1 (i2) r1 = a (i3) r2 = b	(i4) b = 1 (i5) r3 = b (i6) r4 = a	(i7) r5 = a (i8) r6 = b	(i9) r7 = b (i10) r8 = a

Example

r1	1
r2	1
r3	1
r4	1
r5	1
r6	1
r7	1
r8	1

		Not Valid
SC		
TSO – MCA Style (IBM 370)		
TSO – rMCA Style (Sun)		
PC – nMCA Style		
PSO		

a. Output of

r1	1
r2	1
r3	1
r4	1
r5	1
r6	0
r7	1
r8	0

	Valid	Not Valid
SC		
TSO – MCA Style (IBM 370)		
TSO – rMCA Style (Sun)		
PC – nMCA Style		
PSO		

This is the IRIW test case that shows that the PC (nMCA) model doesn't enforce write atomicity. The other models are invalid for this test case. The PSO model was not discussed this semester as to what the write atomicity model was, so will accept either answer.

b. Output of

r1	1
r2	0
r3	1
r4	0
r5	1
r6	1
r7	1
r8	1

	Valid	Not Valid
SC		
TSO – MCA Style (IBM 370)		
TSO – rMCA Style (Sun)		
PC – nMCA Style		
PSO		

This is the test case from lecture that shows how the IBM 370 does not allow for a core to read a value from the write buffer before other cores have seen it. It is invalid for SC and MCA. Valid for the others.

c. Output of

r1	1
r2	1
r3	0
r4	1
r5	1
r6	1
r7	1
r8	1

	Valid	Not Valid
SC		
TSO – MCA Style (IBM 370)		
TSO – rMCA Style (Sun)		
PC – nMCA Style		
PSO		

There is a RAW dependency in the core for i4 and i5, so this outcome is not possible for any of the models.

d. Output of

r1	1
r2	1
r3	1
r4	1
r5	0
r6	0
r7	1
r8	1

	Valid	Not Valid
SC		
TSO – MCA Style (IBM 370)		
TSO – rMCA Style (Sun)		
PC – nMCA Style		
PSO		

If we execute the instructions in this order (i7 i8 i1 i4 i2 i3 i5 i6 i9 i10) the outcome is valid under SC, so it will be valid for all models.