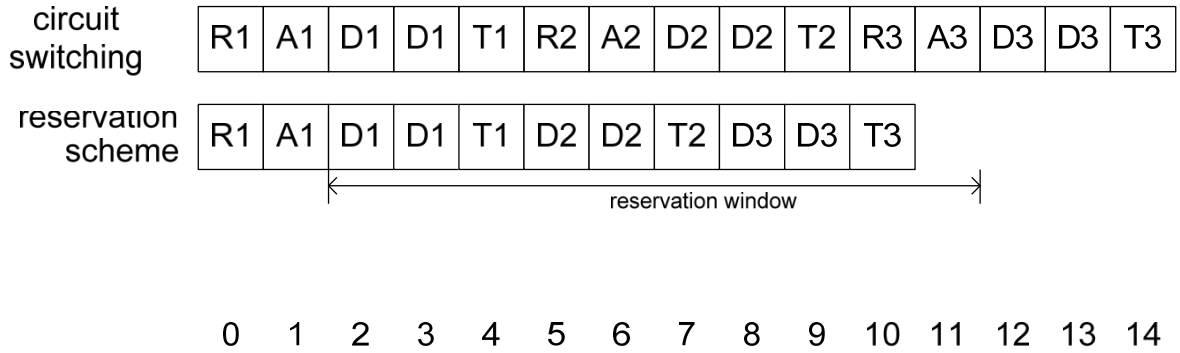
Problem M12.1: Networks-on-Chip

Problem M12.1.A

Consider a flow control method similar to circuit switching but where the request message 'reserves' each channel for a fixed period of time in the future (for example, for 10 cycles since a reservation is made). At each router along the path, a reservation is made if a request from a neighbor can be accommodated. If the request cannot be accommodated a NACK is sent that cancels all previous recommendations for the connection, and the request is retired. If a request reaches the destination, an acknowledgement is sent back to the source, confirming all reservations.

Draw a time-space diagram of a situation that demonstrates the advantage of reservation circuit switching over conventional circuit switching.

Clearly, this scheme eliminates the overhead of establishing connections for every packet. For example, if a source is sending out short packets (two data flits per packet) and the reservation window is 10 cycles, the time-space diagram looks like the following:

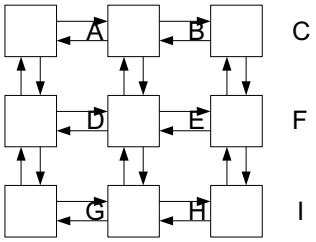


Note that tail flits may be able to get eliminated in this new scheme if they are used only to indicate when channels can be deallocated.

Problem M12.1.B

(a) Randomized dimension-order: All packets are routed minimally. Half of the packets are routed completely in the X dimension before the Y dimension and the other packets are routed Y before X.

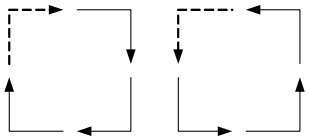
No, this generates a cycle in CDG.



In the CDG corresponding to the mesh network above, for example, EF→FC→CB→BE→EF generates a cycle (Flow E-F-C, flow F-C-B, flow C-B-E, and flow B-E-F will generate a deadlock).

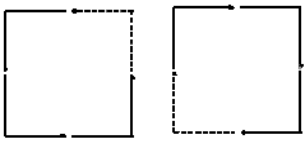
(b) Less randomized dimension-order: All packets are routed minimally. Packets whose minimal direction is increasing in both X and Y, always route X before Y. Packets whose minimal direction is decreasing in both X and Y, always route Y before X. All other packets randomly choose between X before Y and vice versa.

Yes. This effectively eliminates the following two turns.



This corresponds to the third turn model, *'negative-first'* model, which is

deadlock-free. (c) All packets are prohibited to take the two turns in dash:



No. In the 3-by-3 mesh network in part (a), EB->BC->CF->FE->ED->DG->GH->GE->EB generates a cycle.

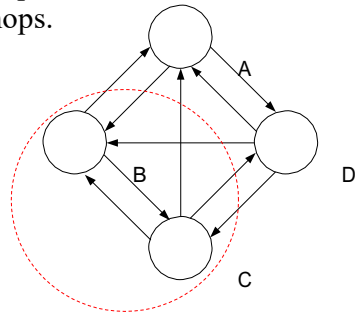
Problem M12.2: Non-mesh Networks

Problem M12.2.A

Fill in the following table of the properties of this network.

Diameter	**2**
Average Distance	**7/8**
Bisection Bandwidth	**6 flit/cycle**

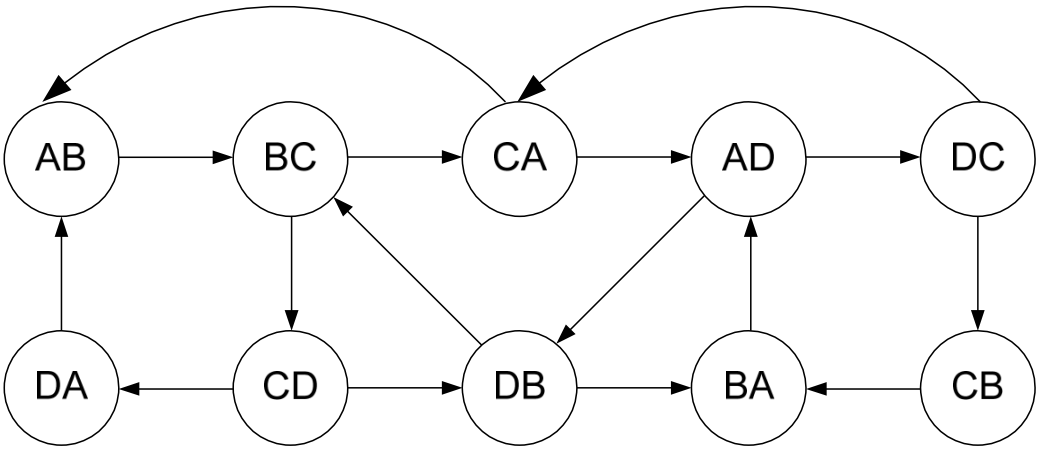
Including self-loops, there are 16 unique (source, destination) pairs. Among them, only the routing distance of B-to-D and A-to-C is 2 hops (which is the diameter of this network), four of them are 0 hops, and all others are 1. Therefore, the average distance is (0*4 + 1*10 + 2*2)/16 = 7/8 hops.



And the bisection bandwidth is 6 flit/cycle.

Problem M12.2.B

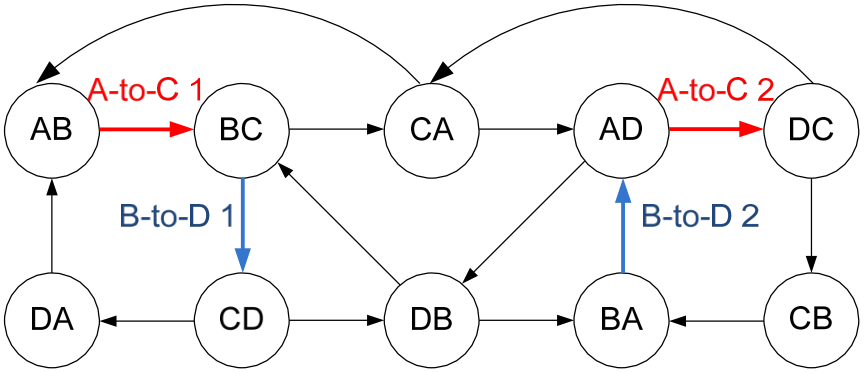
Draw the channel dependency graph of this network.



Problem M12.2.C

Is a **minimal** routing on this network deadlock-free? Show your reasoning and give a deadlock scenario if it is not deadlock -free.

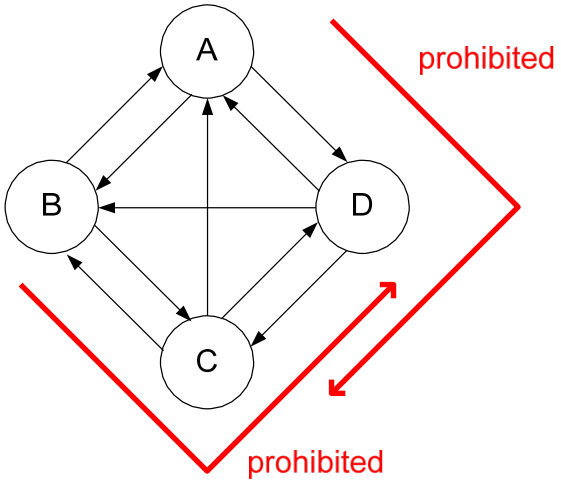
Yes. In minimal routing, all flows except for the ones from B to D and from A to C have 1-hop distance, which is represented by a single node in CDG; they are not holding resources while waiting for another because they need only one resource. The dependencies of flow from B to D and from A to C are represented in the CFG as following (note that each flow can take two possible minimal routes):



There are no cycles in the CDG, thus the routing is deadlock-free.

Problem M12.2.D

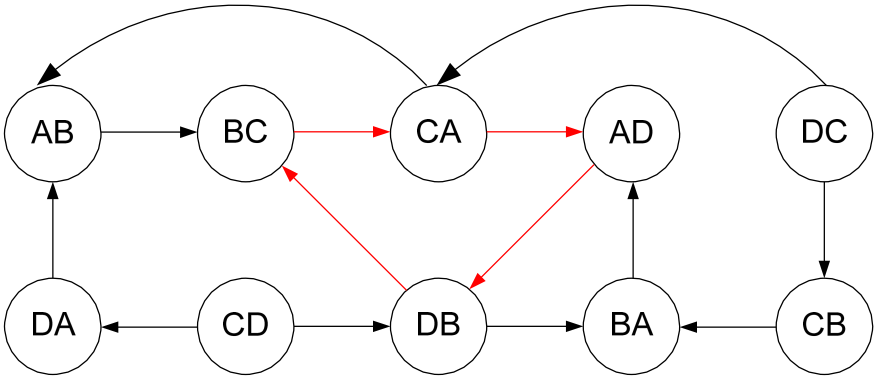
Now, we use a possibly **non-minimal** routing on this network. Plus, we prohibited the following two movements on the non-minimal routing: 1) A to D then D to C and 2) B to C then C to D.



Is this routing deadlock-free? Show your reasoning and give a deadlock scenario if it is not deadlock -free.

No.

Prohibiting those movements, the CDG becomes:



However, there are still cycles in this CDG. For example, if flow 1 from B to D is routed through B-C-A-D, flow 2 from C to B is routed through C-A-D-B, and flow 3 from A to C is routed through A-D-B-C, there can be deadlock by these three flows.

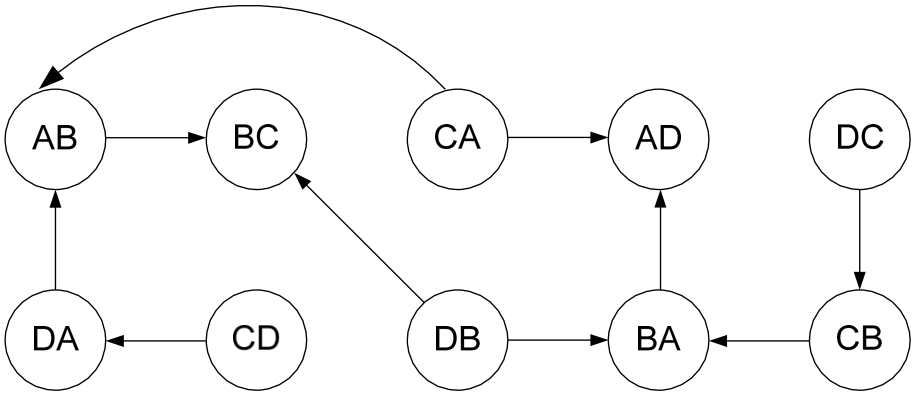
Problem M12.2.E

Still having the two paths in M12.2.D prohibited, we added another restriction in routing: the link from C to A can be used only by packets generated at C, before the packets are transferred to any other nodes (it should be the first link those packets ever take). Also, the link from D to B can be used only by packets generated at D with the same condition (however, routes may be non-minimal).

Is this routing deadlock-free? Show your reasoning and give a deadlock scenario if it is not deadlock -free.

Yes.

These conditions effectively eliminate any turns cornered at C and moving out to A, and any turns cornered at D and moving out to B. Then the CDG becomes:

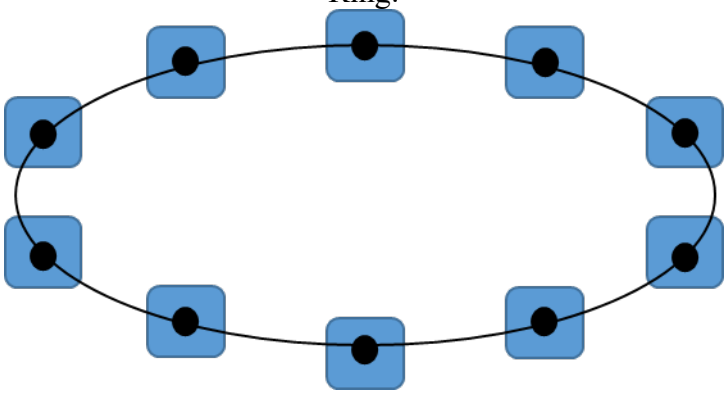


This CDG is acyclic, thus this routing is deadlock-free.

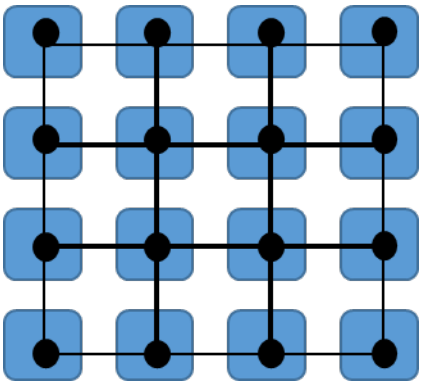
Problem M12.3: Network Effects (Spring 2014 Quiz 4, Part A)

You are choosing between several network topologies for your on-chip network, shown below.

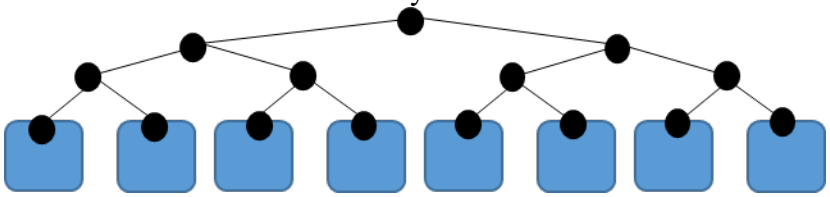
Ring:



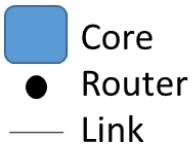
Mesh:



Binary tree:



Legend:



Problem M12.3.A

Your first task is to evaluate these topologies along several important dimensions. Fill in the table below as a function of the number of nodes in the network, *N*. You can safely assume *N* is an even power of 2, giving a complete mesh and binary tree. *For partial credit, give the asymptotic growth instead.*

	Ring	Mesh	Tree
Number of links	N	$2(N-\sqrt{N})$	$2N-2$
Diameter	$\frac{N}{2}$	$2(\sqrt{N}-1)$	$2\log_2 N$
Average distance	$\frac{N}{4}$	$\frac{2(\sqrt{N}-1)}{3}$	$\approx \log_2 N$
Bisection bandwidth	2	$\sqrt{N}$	1

To compute the number of links, consider how many outgoing channels leave each router, and divide by 2 since they are bidirectional links. Then compute how many routers you need for *N* cores. In all cases, the number of links grows proportional to N since we are considering topologies where the number of channels per router is constant with respect to N.

Ring: With a bidirectional ring, you have *N* routers and 2 outgoing channels for each, giving *N* bidirectional links. At worst you have to go half way around the ring to reach another node, for a diameter of $N/2$. On average you will need to do half of this, or $N/4$. To split the network in half, you must cut 2 links, for a bisection bandwidth of 2.

Mesh: You have 4 outgoing channels for each router, except those on the perimeter of the mesh. So you expect the answer to be $2N$ minus those on the perimeter, or the number of channels on the edge divided by 2, $2\sqrt{N}$. The diameter of a mesh is the distance from the top-left to bottom-right, which is twice the number of links in each dimension. The average distance is complicated because, unlike the ring, the starting position makes a difference. Without going into details, this gives you length/3, and the length here is the length of a dimension. You have to traverse this for each dimension, so we multiply by two. Finally the bisection bandwidth is just the bandwidth across an *even* split of nodes in the network, or the length of a dimension, $\sqrt{N}$.

Tree: The tree is the most complicated to compute the number of links. To form a tree of size N, we combine two trees of size $N/2$. This gives a recurrence relation: the number of links for a tree of size N is twice the number of links for $N/2$ nodes, plus 2. Or in other words, $L(N) = 2L(N/2) + 2$. This yields the solution $L(N) = 2N - 2$. The diameter is the distance traversing

up to the top of the tree and back down again—twice the logarithm of the number of nodes. The average distance is similar, since *half the nodes are contained in the other side of the tree*, so the distance is roughly proportional to $\log N$. The exact answer for this is very complicated and we were generous in awarding points. Finally, the bisection bandwidth is just a single link, since cutting any link at the root divides the network.

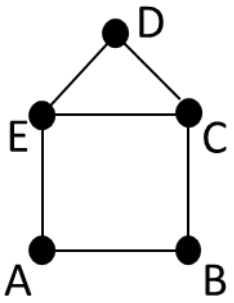
Another question we could have asked, and is worth thinking about, is scenarios when each of these topologies could be better than the others.

The ring requires the fewest links and simplest routers of the three topologies here. For small systems, it is probably the best choice, since the difference between diameter and average distance only becomes substantial as N grows large. (At small N, rings can have lower distances actually.)

A mesh is probably best for large systems since distances grow slowly—with the square root of N—and bisection bandwidth also grows with N—although not proportionally. The mesh requires the most complicated routers, however, so there are non-trivial costs in adopting a mesh topology.

Finally the tree has the best asymptotic growth of distance and so might be beneficial for large systems, but only if bandwidth requirements are low since the bisection bandwidth is so low. Also note that we have computed average distance assuming uniform traffic. A well-designed parallel algorithm should account for locality, in which case the picture gets a lot murkier.

In a sudden flash of inspiration, you decide to use the following topology:



Having decided upon a topology, you now want to make sure your system works properly. All links are bidirectional.

Problem M12.3.B

Show how deadlock could arise in the network by drawing an example on the graph above. Explain your answer in one or two sentences.

If B is sending a message to C through BAEC, and E is sending a message to A through ECBA, then B can grab BA and E can grab EC and the system is deadlocked.

Problem M12.3.C

Draw the channel dependency graph (CDG) for your topology.

The CDG connects *links* on the network to each other if one link can route to another. Links can be grouped into four categories on this topology based on their incoming/outgoing links.

Incoming/Outgoing	Links
1/1	AB, BA
1/2	AE, BC, DE, DC
2/1	EA, CB, ED, CD
2/2	EC, CE

The adjacency list for this CDG is below. (Just think: from this link, where can I go?) Note the symmetry.

AB: BC
BA: AE
AE: ED, EC
BC: DC, CE
DE: EC, EA
DC: CE, CB
EA: AB
CB: BA
ED: DC
CD: DE
EC: CD, CB
CE: ED, EA

Show an example of how to eliminate routes to prevent deadlock on the CDG.

There are many possible answers to this. You must eliminate edges until the CDG is a DAG.

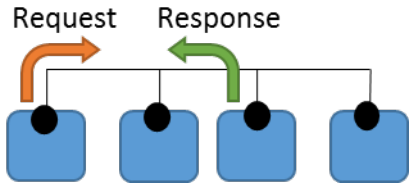
Problem M12.4: The Truth Will Set You Free (Spring 2014 Quiz 4, Part III)

Problem M12.4.A — Peanuts

Snoopy coherence protocols rely on broadcast communication to detect sharing and updates. These are conventionally implemented using bus networks that allow for one message to be sent at a time to all nodes on the network.

Ben Bitdiddle is implementing a bus-based snoopy coherence protocol. One fifth of instructions access memory, and one quarter of these miss in the core's local cache (either because the line is invalid or doesn't have necessary permissions). Assuming each memory operation consists of a request and acknowledgement, the network traffic per core is therefore: $\frac{1}{5} \times \frac{1}{4} \times 2 = \frac{1}{10} \frac{\text{messages}}{\text{instruction}}$. Assume all messages fit within a single network flit.

Assuming a fixed IPC of 1, perfect bus arbitration, and infinite buffers, how many cores can the bus support?



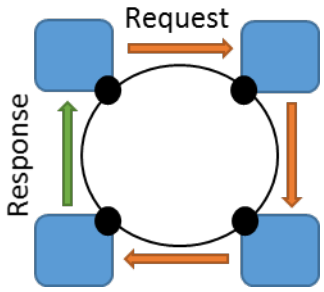
A bus has an aggregate throughput of 1 message per cycle.

A memory operation requires 2 messages on 1/20 of instructions, or 1/10 messages per cycle.

The number of cores this system can support is $1 = N/10$ so $N = 10$.

Problem M12.4.B **... To rule them all**

Ben needs to build a larger system than the bus network will allow, so he changes the system to use a unidirectional ring network. In this design, the core issuing the memory operation sends the request around the ring, and each node along the way either forwards the request or replaces it with its response. Assuming fixed IPC of 1 and a single-cycle per hop in the network, at how many cores will this design saturate?



The ring with N cores has an aggregate throughput of N messages per cycle. (It is a unidirectional ring.)

Each memory operation requires one circuit around the ring, or N messages. Each core produces one request every 20 instructions, so the messages generated per core is $N/20$.

Thus the number of cores is $N = N/(N/20) = 20$.

Maybe a simpler way to see this is that with the bus, each memory operation required global communication twice (for the request and response). In the ring, each memory request requires global communication only once—since ~half the nodes see the request and the rest simply forward the response. Since we are placing half the demand on the network, we can support twice as many cores.

Problem M12.4.C **Matryoshka**

Ben next explores the tradeoffs in cache design between an inclusive cache, where the parent always has a copy of every line in the child's cache, and non-inclusive caches, where this isn't guaranteed.

Give one advantage and one disadvantage of a non-inclusive cache design.

Non-inclusive caches allow the parent cache to get rid of a copy of a line without invalidating the child's copy. This essentially increases the capacity of the parent cache, since it can use the space to store new lines instead of copies of the child's contents.

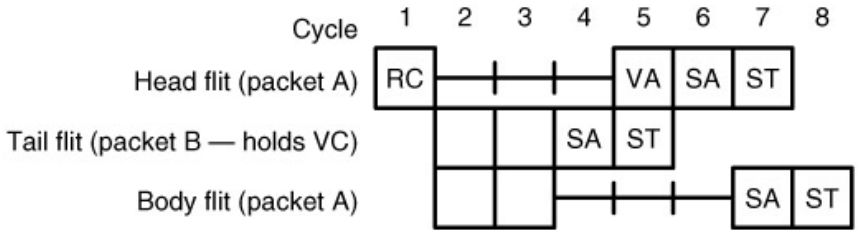
The downside of this is that the parent no longer knows from looking at its own contents whether or not a child has a line. This makes coherence more expensive since the parent must now check with the children to see if they have a line, for example to process an invalidate when the line is written elsewhere.

One way to try to get the best of both worlds is to separate coherence tracking and data storage into separate structures. So rather than having the directory in the cache tags, the directory is a separate structure. This directory can be kept inclusive with the cache non-inclusive.

Problem M12.5: Network-on-chip (Spring 2015 Quiz 3, Part A)

Problem M12.5.A

Consider the router in Handout 16. Assume this router **has one virtual channel per physical link**. Suppose two packets, A and B, are traversing the router. Both are routed to output unit 2, as shown in the following waterfall diagram.



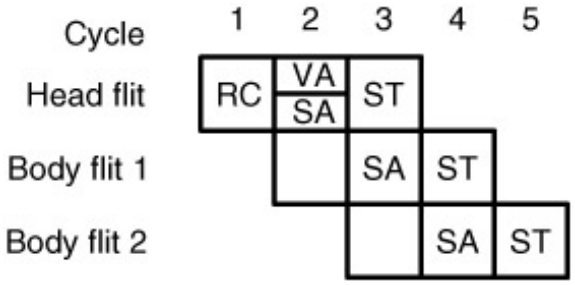
Before cycle 1, packet B's head flit has finished RC and VA. In the following cycles, packet B's four flits traverse SA and ST without stalls. Packet A's head flit completes routing computation at cycle 1 and tries to allocate an output virtual channel starting at cycle 2. Unfortunately, the only output virtual channel compatible with its route is occupied by packet B, so packet A's head flit fails to allocate a VC and is unable to make progress until packet B's tail flit releases the VC.

Fill in the following table showing the state of packet A's input virtual channel.

Cycle	G	R	O
1	R	-	-
2	V	Output 2	-
3	V	Output 2	-
4	V	Output 2	-
5	V	Output 2	-
6	A	Output 2	VC1
7	A	Output 2	VC1
8	A	Output 2	VC1

Problem M12.5.B

Suppose the router in Handout 16 is improved with **speculative switch allocation.** Head flits attempt VC and switch allocation in the same cycle. If both succeed, the head flit traverses the switch on the next cycle, as shown in the waterfall diagram below.



Consider the same scenario as in question 1, with packets A and B going to the same output unit. Assume that **non-speculative switch allocation requests are always prioritized over speculative ones** (i.e., those from flits without a VC). Fill in the following waterfall diagram to show how packet A is routed.

Cycle	1	2	3	4	5	6	7	8
A: Head Flit	RC	-	-	-	VA SA	ST		
A: Body Flit 1						SA	ST	
A: Body Flit 2							SA	ST
B: Body Flit 1	SA	ST						
B: Body Flit 2	-	SA	ST					
B: Body Flit 3	-	-	SA	ST				
B: Tail Flit	-	-	-	SA	ST			

Problem M12.5.C

Consider the same speculative switch allocation optimization as in question 2. Unfortunately, always prioritizing non-speculative switch allocation requests over speculative ones increases the critical path too much, so we opt for a **simpler switch allocator that is oblivious to whether requests are speculative**.

We want to analyze the performance of this simpler design under the following scenario:

- All packets in the router are single-flit packets.
- The probability that a packet successfully obtains a VC on its first try is 75%.
- The probability that a flit successfully allocates the switch on its first try is 80%.
- If a packet fails either virtual channel or switch allocation on its first try, it always succeeds on its second try.

1) What percentage of allocated timeslots on the switch goes unused?

   The switch is unused when the packet get the switch but not VC.

   $0.25 * 0.8 = 0.2$

2) What is the average latency to go through this speculative router?

   If both VA and SA succeed, the latency is 3 cycle. Otherwise, it is 4 cycles

   Average latency $= 3 * 0.75 * 0.8 + 4 * (1 - 0.75 * 0.8) = 3.4$ cycles

3) Briefly explain the effect of this optimization on network performance at both very low loads and very high loads (near saturation).

   For very low loads, the speculation almost always succeeds, so the average latency is lower. For very high loads, the speculation fails frequently so the switch is not highly utilized, and the average latency is higher.

Problem M12.5.D

Ben Bitdiddle wants to implement the Valiant routing algorithm, which routes each packet through a randomly-chosen intermediate node. He uses routers with two virtual channels per physical link. He decides to use X-Y routing between the source node and intermediate node, and Y-X routing between the intermediate node and the destination node. However, Alyssa points out this routing algorithm will not work without further modification. Explain why this is the case and provide a solution for Ben.

Ben's routing algorithm will cause deadlock

To solve this problem, Ben should allocate one virtual channel for X-Y routing, and another for Y-X routing. Note that using X-Y only still causes deadlock since there will be a forbidden turn when passing through the intermediate node.

(Source -X-Y-intermediate node-X-Y-destination)
^^^^^^^^^^^^^^^^^^^^^^^^
This is turn in Y-X routing

Unless the intermediate node has infinite buffer, using X-Y or Y-X only still deadlocks.

Problem M12.6: Network-on-chip (Spring 2016 Quiz 3, Part D)

Problem M12.6.A

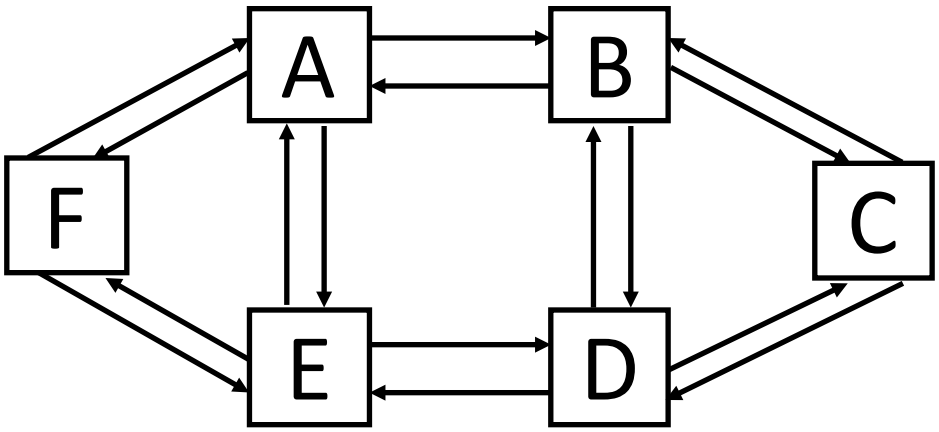
Determine whether the following routing algorithms are deadlock-free for a 2D-mesh. State your reasoning.

a) (3 points) Randomized dimension-order: All packets are routed minimally. Half of the packets are routed completely in the X dimension before the Y dimension, and the other packets are routed in the Y dimension before the X dimension.
Not deadlock-free. All turns in the turn model are allowed, and hence not deadlock-free. Alternatively, you may argue that the CDG has a cycle.

b) (3 points) Less randomized dimension-order: All packets are routed minimally. Packets whose minimal direction is increasing in both X and Y always route **X before Y**. Packets whose minimal direction is decreasing in both X and Y always route **Y before X**. All other packets choose randomly between X before Y and vice-versa.
Deadlock-free. In essence, this prevents north-to-east and west-to-south turns in the turn model which is sufficient to prevent deadlock.

Problem M12.6.B

Consider the following topology:



(a) (2 points) What is the diameter of this topology?

3

(b) (2 points) What is the bisection bandwidth (in flits/cycle) of this topology?

4

(c) (5 points) Assume that 180-degree turns are prohibited. No other turns are prohibited. Show how deadlock could arise in the given topology.

AB → BD → DE → EA → AB is a cycle in the CDG

(d) (10 points) We now restrict all routes to be minimal and disallow the following turns on the mesh (among the nodes A, B, E, D): north-to-east, north-to-west, south-to-east, south-to-west. Is the routing strategy deadlock-free? Draw the CDG to justify your answer.

Deadlock-free as shown by the CDG below.