Solutions to Midterm

1.A. FALSE. Minimal routing algorithms are subject to traffic patterns that exploit their minimality. For example, no minimal algorithm performs well on the tornado traffic pattern for the torus. This is not the case for non-minimal oblivious routing algorithms which can balance load through randomization, such as Valiant's algorithm.

1.B. STAYS THE SAME. The zero-load latency of virtual cut-through is not affected by buffer depth.

1.C. TRUE. All routes use the links in order: links up the tree followed by links down the tree.

1.D. For the network to be strictly non-blocking, $m > 2n - 1 = 3$. The middle stage switches are $r \times r = 4 \times 4$ crossbars (there are $m = 3$ of these switches).

1.E. FALSE. Hop-by-hop routing requires a route computation per hop, which *increases* the hop delay t_r.

1.F. FALSE. The point of wormhole flow control is that the buffer do not have to be as large as the packets. However, in store-and-forward flow control the buffers are as large as the packets.

1.G. For the throughput Θ to be 1Gbit/sec,

$$b_c = \gamma_c \Theta = (k/8) \cdot (1\text{Gbit/sec}) = 1\text{Gbit/sec}.$$

1.H. The tightest bound is found by counting the average number of hops on the two types of channels. We only have to perform this calculation for one source because the topology is vertex symmetric. So, for simplicity we choose the origin as the source:

Destination	Hops on +/-1 links	Hops on +/-3 links
0	0	0
1	1	0
2	1	1
3	0	1
4	1	1
5	0	1
6	1	1
7	1	1

This balances load equally on the long and short links, giving

$$\gamma_{\max} \geq \frac{|N|}{|\text{\# of short channels}|} \cdot \frac{5}{8} = \frac{5}{16}.$$

Thus, the throughput is at most $\Theta_{\text{ideal}} \leq 16b/5$.

2.A. The time-space diagram should be identical to Figure 12.7 of the book, except for two fewer stall cycles on the request and just one packet for the circuit.

2.B. This flow control method only requires one flit of buffering (for the head) at each input of each node.

2.C. (i) Latency can be either larger or smaller depending on the size of the packet — for larger packets, store-and-forward looses out. This was not meant to be a trick question and an extra point was given to students who identified both possible solutions. (ii) Throughput goes down because this flow control method idles the channels while a circuit is being established. Store-and-forward only uses channel bandwidth when sending a packet.

3.A.

k	n	$w = W_n/\delta$	$\Theta_{\text{ideal}} = \frac{8wf}{k}$	$T_s = L/b$	$T_h = \frac{10nk}{4}$	T_0
256	1	32	1Gbit/sec	32ns	640ns	672ns
16	2	16	8Gbit/sec	64ns	80ns	144ns
4	4	8	16Gbit/sec	128ns	40ns	168ns
2	8	4	16Gbit/sec	256ns	40ns	296ns

The best choice is a 4-ary 4-cube.

3.B. The addition the module constraint can only restrict the width of the channels. Thus, serialization latency increases and throughput decreases (both could stay the same if the constraint is not very tight).

3.C. We start by examining packagings that exactly fit x of the 4 dimensions of the 4-ary 4-cube into a single module:

Dimensions per module x	Nodes per module $M = k^x$	Signals leaving module $4wM(n-x)$
0	1	128
1	4	384
2	16	1024
3	64	2048
4	256	0

Since the 2-dimensional packaging the largest number of nodes per module does not exceed our module pin constraints, it is the best choice (if the 2-dimensional packaging didn't exactly meet the module constraint, we would have to explore more options between 2 and 3 dimensions).

4.A. The zig-zag routing algorithm is not deadlock free — examining the turn-model reveals that all eight turns are allowed by the algorithm. To make it deadlock free, one virtual channel can be used for preferred directions of $(+, +)$ and $(-, -)$ and another virtual channel is used for preferred directions of $(+, -)$ and $(-, +)$.

4.B. Again referring to the turn model, the modified algorithm allows only these five turns: $+x$ to $+y$, $+y$ to $+x$, $-x$ to $+y$, $-x$ to $-y$, and $+x$ to $-y$. This does not create any cycles in the turn model. In fact, we could also allow zigzagging in either the $(+, -)$ or $(-, +)$ directions while still avoiding deadlock.