EECS578 – Bertacco
Fall 2015

EECS 578 – Interconnect Mini-project

Assigned 09/17/15 (Thu) − Due 10/02/15 (Fri)

Introduction

In this mini-project, you are asked to answer questions about issues relating to interconnect (*i.e.*, network on chip). There are four parts of questions: disconnected topology, performance evaluation, routing deadlock and hardware design. Throughout this assignment, we make two assumptions for the sake of simplicity.

(1) Each failure affects both directions of a bi-directional link. Note that a bi-directional link contains two uni-directional links.

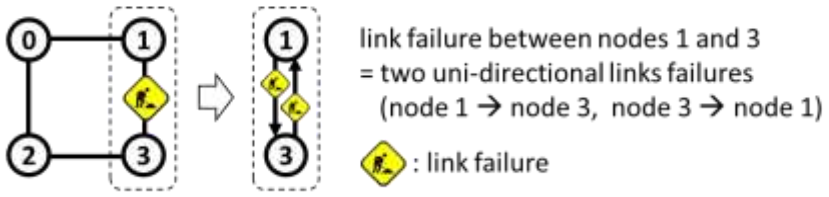
0
1
2
3
link failure between nodes 1 and 3
= two uni-directional links failures
(node 1 → node 3, node 3 → node 1)
: link failure

Figure 1. Bi-directional link failure

(2) No two failures may occur on a same link. For instance, a network with three link failures must have precisely three broken links.

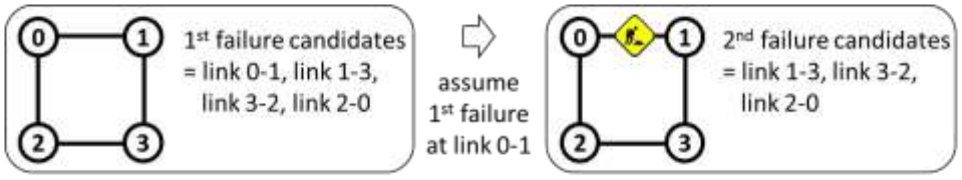

Figure 2. No failure recurrence

Part 1. Disconnected topology

In this part, we need to measure how resilient an interconnect is against link failures. Specifically, we consider it disconnected, if there is at least one source and destination that cannot communicate with each other.

For instance, let's consider a 2x2 mesh with 1 failing link. There are four different failure patterns, $\binom{4}{1}$= 4. However, in all cases the faulty mesh is still connected because all possible source-destination pairs have at least one valid route.

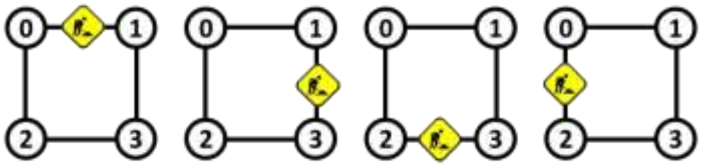

Figure 3. Four possible faulty patterns for a 2x2 mesh with 1 link failure

However, a 2x2 mesh with 2 failing links is always disconnected regardless of the failure pattern. You may want to try some patterns to check that this is true. Note that there are 6 different patterns in this case, $\binom{4}{2}$= 6.

In the questions below, you need to **calculate the probability that a network becomes disconnected after experiencing the given number of link failures**. You must provide your reasoning for each answer. For questions 1.1 through 1.5, the paper can be solved on paper. The extra-credit question 1.6 may require you to write a simple program or other tool to analyze the situations that may arise. If you do so, please submit it, along with your findings.

Question 1.1. (8 pts) 4x4 mesh with 2 link failures.

Question 1.2. (8 pts) 4x4 mesh with 3 link failures.

Question 1.3. (8 pts) 4x4 mesh with 4 link failures.

Question 1.4. (4 pts) 8x8 mesh with 2 link failures.

Question 1.5. (4 pts) 8x8 mesh with 3 link failures.

Question 1.6. (Extra credit, 20 pts) 8x8 mesh with 10 failing links.

Part 2. Interconnect performance evaluation

Saturation throughput and packet latency are performance metrics that are often used to evaluate a network on chip. Computing these metrics often requires network simulation because many parameters can affect the results (*e.g.*, congestion). In this part of the assignment, you need to calculate two simpler metrics: link utilization and hop count. These two metrics will allow you to estimate saturation throughput and packet latency.

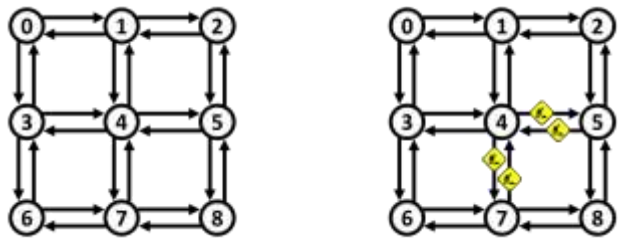

Figure 4. (a) A 3x3 mesh network (b) A faulty 3x3 mesh network

Consider the five pairs of source and destination in Table 1. Each pair has a different communication frequency in flits per clock cycle, and a deterministic route from the source to the destination. For instance, in the first pair, packets from node 0 (source) visit intermediate nodes 1, 2 and 5 in that order, before reaching node 8 (destination). The table also tells us that the first pair transfers 3 times as many packets as the second pair.

Table 1. Communication frequency and routes for 5 source-destination pairs

No.	Source	Destination	Frequency (flits / cycle)	Route
1	0	8	0.03	0→1→2→5→8
2	2	6	0.01	2→5→4→7→6
3	8	1	0.025	8→5→4→1
4	3	2	0.015	3→4→1→2
5	4	5	0.02	4→5

Question 2.1. (8 pts) For the regular 3x3 mesh network of Figure 4(a) and the patterns of Table 1, calculate the utilization of each uni-directional link in the network in flits/cycle. (Draw a network similar to Figure 4(a), and write your answer near the links in your network.)

Question 2.2. (8 pts) Consider the faulty network of Figure 4(b). Note that some routes in Table 1 are broken, and should be rerouted to alternative ones. For instance, the last pair should be rerouted because of the faulty link 4-5. Provide two different sets of alternative routes that are not using any faulty link. (You can use the table below for your answer. Simply fill out the blanks for alternative routes 1 and 2.) For this question, you do not need to worry about link utilization. For pairs that do not require rerouting, please use the routes in Table 1.

No.	Source	Destination	Frequency (flit / cycle)	Alternative route 1	Alternative route 2
1	0	8	0.03		
2	2	6	0.01		
3	8	1	0.025		
4	3	2	0.015		
5	4	5	0.02		

Question 2.3. (8 pts) For the two route sets you found in Question 2.2, calculate link utilizations of the route sets, similarly to your answer in Question 2.1. You must provide one using all the alternative routes 1, and the other using all the alternative routes 2. (You will need to draw two networks at this time.)

Question 2.4. (8 pts) Which route set is better in terms of load balancing? Explain why you think so.

Question 2.5. (8 pts) Compute the average hop count for each alternative route set. Note that the average hop count does NOT take into account utilization to weight each route. Based on average hop count, which route set is better in terms of packet latency?

Part 3. Routing and deadlock

In the lecture, we have studied a deadlock in the context of networks-on-chip. To remind you what the deadlock problem is, Figure 5 shows a simple example. In the figure, all 4 nodes in the network are trying to send packets a neighbor router in a cyclic manner. For instance, a packet in node 0 needs to go to node 3 via node 1, while another packet in node 1 wants to go to node 2 via node 3, etc.

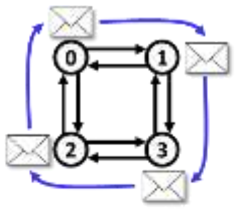

Figure 5. Deadlock situation in a 2x2 mesh

While it looks simple, identifying deadlock situations can be actually difficult in a large network. A popular way of detecting deadlocks is to check if there exists any cyclic resource dependency. A resource dependency is defined as a situation where one resource waits for another resource. In this assignment, you can simply consider a link as a resource. Let's find a cyclic resource dependency in Figure 5:

Dependency 1: Link 0→1 waits for link 1→3.
Dependency 2: Link 1→3 waits for link 3→2.
Dependency 3: Link 3→2 waits for link 2→0.
Dependency 4: Link 2→0 waits for link 0→1. These four dependencies form a cycle.

Question 3.1. (8 pts) Is the route set in Question 2.1 deadlock-free?

Question 3.2. (8 pts) Are the two route sets you designed in Question 2.2 deadlock-free?

Question 3.3. (4 pts) Provide an example of up/down link assignment for the network in Figure 4(a). (You need to draw the same network in your answer, and mark "up" or "down" for each link.)

Question 3.4. (4 pts) Provide an example of up/down link assignment of the network in Figure 4(b).

Part 4. Designing a route-computation module

The up/down routing algorithm is an example of deadlock-free routing that uses turn restrictions. Turn restrictions are placed in a way to remove all cyclic resource dependencies existing in the network. In this assignment, a turn is defined as a change of direction (*i.e.*, X → Y or Y → X), as shown in Figure 6(a). In the figure, there is one bi-directional turn restriction at node 3: west-north. This bi-directional turn restriction disables two uni-directional turns: (1) a turn from west to north and (2) a turn from north to west.

Extending this idea, we can place turn restrictions on any mesh network. For instance, Figure 6(b) shows an example of turn-restriction placement for a 3x3 mesh. Note that neither deterministic X-Y nor deterministic Y-X routing are allowed with the turn restrictions placed in the figure.

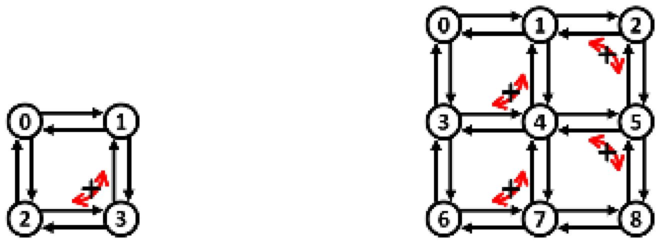

Figure 6. (a) Turn restriction (b) Deadlock-free turn restrictions for a 3x3 mesh

Question 4. (32 pts) You are designing a route-computation hardware module that will be deployed in each router of a regular mesh network. Your module should perform hop-by-hop route-computation (*i.e.*, the route computation only decides the next hop, not the entire route to the destination) based on locations (current and destination) and turn restrictions at neighboring nodes. You are allowed to use Boolean gates (e.g., AND, OR, NOT, etc) or blocks such as adders and comparators, but you cannot use routing tables. Figure 7 shows the detailed inputs and outputs for your module. Your module should be applicable to any size mesh network. In other words, your module should not depend on the size of the network. (The only thing that changes is the bit-width of X/Y locations.) **Show a schematic of your route-computation module.**

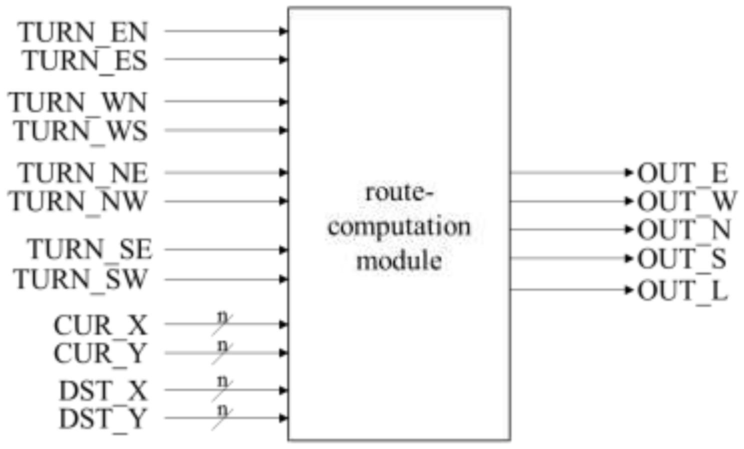

Figure 7. Inputs and outputs of route-computation module

Inputs

- Eight 1-bit turn-restriction flags (TURN_EN, TURN_ES, TURN_WN, TURN_WS, TURN_NE, TURN_NW, TURN_SE and TURN_SW). If asserted, the turn is restricted.
  - TURN_EN(TURN_ES): The north(south) turn at the east router is restricted.
  - TURN_WN(TURN_WS): The north(south) turn at the west router is restricted.
  - TURN_NE(TURN_NW): The east(west) turn at the north router is restricted.
  - TURN_SE(TURN_SW): The east(west) turn at the south router is restricted.
- X and Y location of current router (CUR_X and CUR_Y)
- X and Y location of packet destination (DST_X and DST_Y)

Outputs

- Five 1-bit output direction flags (OUT_E, OUT_W, OUT_N, OUT_S and OUT_L): each 1-bit for east, west, north, south and local. If asserted, the output direction is taken.

Special note: Be aware of the fact that the input turn-restriction flags indicate the restrictions at the "neighboring" routers. For instance, in Figure 6(b), router 4 receives the flags for the turn restrictions at routers 1 (north), 3 (west), 5 (east) and 7 (south). This router takes the inputs as follows: TURN_EN = 0, TURN_ES = 1, TURN_WN = 0, TURN_WS = 0, TURN_NE = 0, TURN_NW = 0, TURN_SE = 0 and TURN_SW = 1.