MEMOCODE 2011 Hardware/Software CoDesign Contest

Derek Chiou
The University of Texas at Austin

February 28, 2011

1 Administration

If you are interested in entering, please send email to memocode2011@externalmail.com with the subject line "MEMOCODE 2011 Design Contest Registration" and include the following information:

Desired username:
First name:
Last name:
email address:

When you register, you will be put on an email list for immediate updates and information. Information be posted on bit.ly/memocode2011designcontest.

2 Overview

The objective of the 2011 MEMOCODE Hardware/Software CoDesign Contest is to deliver the fastest simulator of a simple Network-on-Chip (NoC). Simulators are important tools to perform architectural tradeoffs without the cost of implementing each possibility. Thus, simulator flexibility, along with simulation speed are two key traits to a good simulator.

The contest is to write a simulator that takes two inputs: a specification of the NoC to be simulated (the *target*) and a traffic pattern that will be simulated on the NoC. Your result will be judged firstly on *correctness* (that the number of cycles is accurately modeled) and secondly on *simulation speed* (the amount of time it takes to complete the entire simulation.)

3 The System Being Simulated: Network-on-Chip

Many modern chips contain an on-chip network that moves *packets* of data through *routers* that are connected to both other routers and chip components via unidirectional links that carry packets. A figure of a typical NoC is shown in Figure 1. In this figure, the NoC has six routers (squares), each router connected to one processor (circles). The figure represents two unidirectional links as a single bidirectional link. Note that the router links are not necessarily symmetric (each unidirectional link has a partner going in the opposite direction) though in this example the router links are symmetric.

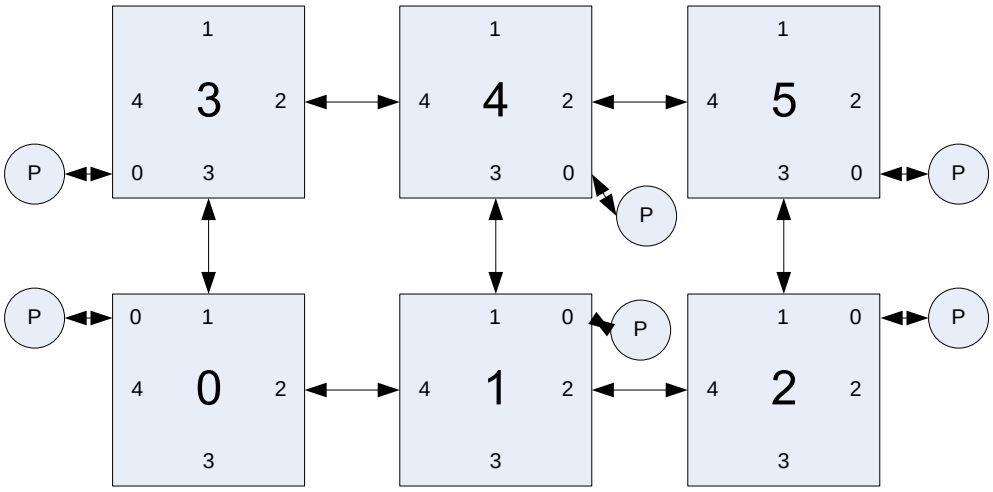

Figure 1: A typical mesh NoC with six routers and six processors. The numbers associated with each link are the in_port and out_port numbers. Each double-ended arrow stands for two unidirectional links. Note that the in_port and out_port link 0 are the injection port and extraction port from/to the processor that generates and consumes messages.

Each router has a set of N_i input ports (in_ports), numbered 0 to $N_i - 1$ and a set of N_o output ports (out_ports), numbered 0 to $N_o - 1$. The number of input ports is not necessarily equal to the set of output ports. Each uni-direction link connects an output port to an input port.

Each router also has an insertion port that allows the connected component to insert packets into the network. Likewise, each router has an extraction port that allows the connected component to extract packets from the network. The insertion in_port number is always 0 and the extraction out_port number is also always 0. There is always exactly one insertion port and exactly one extraction port.

A packet consists of one or more *flits*. In a real NoC, flits consist of both control information and data but in the simulator, we omit the data. There are three kinds of flits: (i) a head flit (the first flit of a packet), (ii) a tail flit (the last flit of a packet), (iii) a head-tail flit (only one flit in the packet), or a (iv) body flit (neither a head or a tail flit.) The type of flit is indicated by two bits: a head_p bit indicating that the flit is a head flit and a tail_p bit indicating that the flit is a tail flit. A head-tail flit would have both bits set and a body flit would have neither bit set. In addition to these two bits, a head flit also contains the destination node where all of the flits of the packet will be extracted. Flits from a single packet always travel in head, body, tail order; flits from a single packet should never get out of order.

Each in_port has *flit buffers* that store flits until they can be forwarded to the next router or extracted. Flit buffers are partitioned into one or more disjoint *virtual channels*. A packet is assigned a virtual channel as part of the traffic specification and keep the same fabric channel from insertion through extraction. Virtual channels are independent of each other; if a flit is available and a flit buffer in its virtual channel is available downstream, that flit is eligible to use that link, even though flits on other virtual channels on the same link are blocked. Thus, virtual channels make better use of links by eliminating head-of-line blocking between flits in different virtual channels.

The combination of a link and a virtual channel is called a *virtual link*. Thus, each link supports N_{vc} virtual links, where N_{vc} is the number of virtual channels. **Once a multi-flit packet starts on a virtual link it must complete before that virtual link can carry flits from another packet.** However, flits traveling down the same physical link but different virtual links do so independently of each other. Thus, not all of the flits from a packet on one virtual channel need to be moved before flits from a packet on another virtual channel starts. All router-router links and extraction links operate in this fashion. For

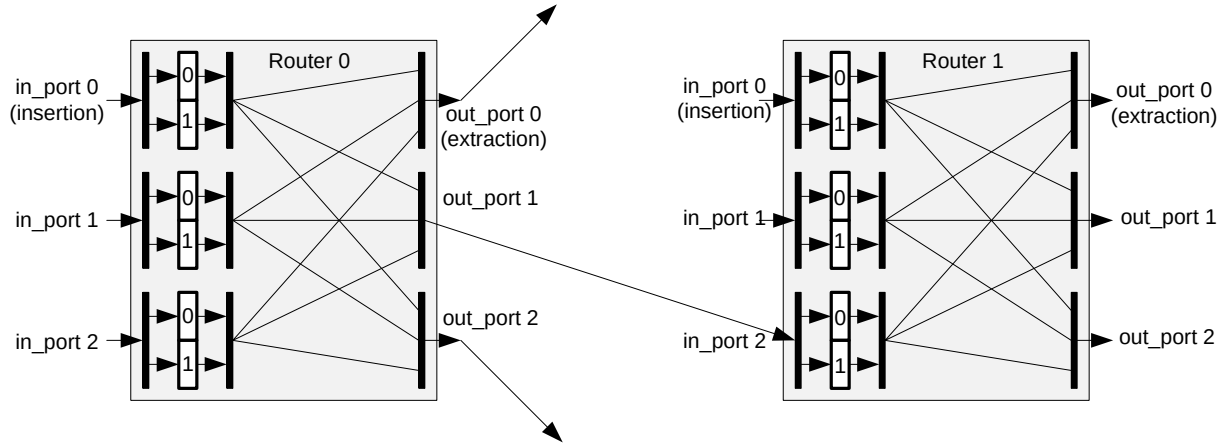

Figure 2: Two routers. Flits enter on an in_port, then get stored in the an in_port flit buffer associated with the appropriate virtual channel (in white.) This system supports two virtual channels. Note that there are two flit buffers, one per virtual channel, per in_port. Flits are then moved out of of the flit buffer, across the central crossbar out of an out_port to another router or to be extracted from the network.

simplicity, insertion links require that all the flits from one packet are inserted before the next packet can start.

Flit buffers are managed using *credits* that indicate that there is a free flit buffer available at the downstream router's in_port. Credits are specified by the virtual channel number, since each credit is returned individually. As a flit is passed downstream, a credit is simultaneously returned upstream via an implicit backwards path for each link to return credits.

No more than a single flit can be moved through a single out_port in a single cycle. In addition, no more than a single flit can be read from a single in_port in a single cycle.

Figure 2 shows two routers that are connected via the out_port of Router 0 to the in_port of Router 1. The crossbar is arbitrated using a greedy fixed priority scheme that first prioritizes virtual channels (the lower the virtual channel number, the higher the priority), then source ports (the lower the source port number, the higher the priority.) Based on that prioritization, if a flit is available and the destination port is available and the downstream flit buffer for the packet's virtual channel is available and no other packet from another in_port is in progress on that virtual link, that flit is sent out that out_port on that cycle and a corresponding credit is returned to the upstream router.

Routing is done by a routing table included in each router that is specialized for that router. The routing table contains an entry per destination that indicates which out_port to use to get to that destination. Thus, all packets to the same destination that go through a router, regardless of the in_port it arrived on, will go through the same out_port.

4 The Simulator

The objective of this contest is to predict the precise number of cycles that it takes to run the specified sets of traffic on the specified router configurations. In order for your result to be valid, your predicted number of cycles must precisely match the output of the reference code. At least four router configurations will be provided and at least four sets of traffic will be provided for each router configuration. You must be able to support any router configuration that is allowable. However, the total amount of state needed to perform

the simulation will be designed to ensure that the design will fit on-chip (in cache/scratchpad/block RAMs) for most platforms to eliminate the need for significant memory bandwidth. Thus, we will target less than 6Mbits when the design is bit optimized.

Note that no parameter is required to be a power of 2.

Parameter	Description
Credit delay	The number of cycles of delay before a credit is returned to the upstream router. Ranges from 1 to 16. Explicitly specified in the router configuration file.
Number of virtual channels	The number of virtual channels supported by the system. Ranges from 1 to 8. Explicitly specified in the router configuration file.
Max time	The maximum number of cycles that will be simulated. Up to 2^{31} cycles. Explicitly specified in the traffic configuration file.
Number of in_ports	The maximum number of in_ports of all the routers (each router can be different). Range from 2 to 16. Implicitly defined in the router configuration file by the specified links.
Number of out_ports	The maximum number of out_ports of all the routers (each router could be different). Range from 2 to 16. Implicitly defined in the router configuration file by the specified links.
Number of routers	The number of routers. Range from 4 to 256. Implicitly defined in the router configuration file.
Number of traffic entries	Each router can specify its own traffic pattern. Each router will have no more than 1024 traffic pattern entries.

In order to fit in 6Mb, when one parameter goes up, others must come down. For example, if 256 routers are specified, it's impossible that 8 virtual channels, 16 in/out ports, and 1024 traffic entries per router will also be specified.

The simulated NoC is specified by a *router configuration specification file* that defines the number of routers and the connectivity between routers and a *traffic configuration specification file* that defines the route table and the traffic.

A "%" in the left most column indicates that line is a comment and an entirely empty line is ignored.

```
% start of router configuration file

% specify the number of cycles of delay credit.
num_credit_delay_cycles=1

% the number of virtual channels
num_vcs=4

% <source router>:<out_port>-<destination router>:<in_port number>
% below is router 0, out_port 1 linked to router 2, in_port 3
0:1-2:3
0:2-1:4
```

4.1 Traffic Configuration Specification File

A "%" in the left most column indicates that line is a comment and an entirely empty line is ignored.

```
% specify whether to be verbose or not.  Verbose mode does not need to
%  be supported in contest code.
verbose=1

% terminate at max_cycle if not terminated before
max_cycle=100000

% route_table specification.
% route:<source router>-><destination router>:out_port

% below, router 0 goes to router 0 (extract) through out_port 0
route:0->0:0

% below, router 0 goes to router 1 through out_port 2
route:0->1:2

% traffic specified as follows
% node <node number>:<number of packets>.
node 0:2

% each packet specified as
% <source node>:<dest node>:<virtual channel>:<num flits>
0:3:3:2
0:1:2:1

% if the number of packets is greater than the number of packets
%  specified, then repeat until the total number of packets is reached.
```

5 The Contest

The reference code, by definition, produces the correct result. However, if bugs are found in the first week of the contest, we will correct those bugs and post new reference code that will be standard of correctness (please notify us if you find bugs.) You need to predict precisely the same results as predicted by the reference code in order for your result to be considered correct. For each of your correct results, we will divide your time by the fastest correct time. Thus, the best score for each correct result is a 1. If your correct solution takes twice the time as the best solution, you get a score of 0.5 for that correct solution. An incorrect result gets a 0. The sum of your scores is your base score.

On April 1, 2011 at 12:01AM GMT time, time trials start. You will be allowed to specify your own start time, negotiated with us (when we are all awake), within 24 hours of 12:01AM GMT. At your start time, you will be provided with the four router configurations and the *format* of the traffic file (number of traffic entries per router, maximum number of flits.) 24 hours after your start time, you will be provided with the traffic patterns, one at a time. As soon as you generate a result, you will send us that result, at which point we will immediately provide the next traffic pattern. Your times are the time between receiving the traffic pattern and the time you return a correct answer. If you get an answer incorrect, we will tell you that you were incorrect, but not provide you with the correct answer at that time.

There are two ways to win for this contest. It is possible to win both. The Unlimited Class is won by having the highest base score. The Normalized Class is won by the highest of base score divided by publically listed academic price (or retail price if an academic price isn't available) of the execution platform. If the platform is a custom one, we will attempt to generate a price based on the list retail price of the

components.

Winners will be confirmed by contest organizers or appointees which will require remote access to the winning platform and tool chain.

Hints You are free to optimize you implementation for specific router configurations to improve performance and save space. High-level synthesis techniques and highly parameterized solutions are very welcome.

Research work that has been done in the area of parallelized and accelerated simulation. You might want to consider multithreaded/time-division-multiplexed solutions. A HPCA 2011 paper entitled "HAsim: FPGA-Based High-Detail Multicore Simulation Using Time-Division Multiplexing" by Pellauer, et. al. and an ISCA 2010 paper entitled "A case for FAME: FPGA architecture model execution" by Tan, et. al. describe some of that work.

Parallelized software simulators are another way to improve performance. It may be worthwhile to scan the work of the parallel discrete event simulation community.