Architecture Note

Abstract

**Design Overview**

This design shall consist of a simple network-on-chip with a 4x4 torus topology, oblivious dimension-order routing (nodal table based routing on the next rev), and virtualized wormhole flow control. Each network node shall be fully connected to each of its neighbors. Nodes shall be able to handle deadlock avoidance and recovery (next architectural revision). Nodes shall be input queued and organized with statically sized FIFOs for buffers with dynamically allocated buffers for virtual channels. Routing engine shall (subject to architectural revision) consist of one state machine per input port. Virtual channel and switch arbitration will be using round robin arbitration.

**Clocking Resources**

Max clock on Virtex US+ is 891MHz

**Topology**

4x4 torus network

Port bandwidth = .5GBps @ 100MHz

Aggregate peak bandwidth = 2GBps @ 100MHz

Port width = 16 bits input/16 bits output

Port count: 4 inputs/4 outputs per node

**Routing**

Dimension-order routing. Each node has an absolute physical address, and shortest path is determined algorithmically. If multiple shortest paths exist, routing algorithm will choose route based off of its phase (next rev – for now route selection between equivalent shortest paths will be done pseudorandomly). Each node shall be initialized with staggered phased based off of its grid position. Routing at this rev will have no knowledge of global congestion and channel loading, e.g. not capable of adaptive routing.

Route Computer Mapping

|  |  |  |
| --- | --- | --- |
| Preferred Direction Vector [3:0] | Cardinal Direction | VID Channels |
| [X, X, 0, 1] | E | 0:11 |
| [0, 1, X, X] | N | 12:23 |
| [X, X, 1, 0] | W | 24:35 |
| [1, 0, X, X] | S | 36:47 |
| [0, 0, 0, 0] | X | 48:59 |

**Flow Control**

Virtual wormhole, credit based flow control. Statically allocated buffer count and depth. No deadlock avoidance. No speculation. Credit stall avoidance by right sizing #VID’s per port.

#VID’s per port = RTT + T-pipeline + T-crediting + T-startNextSA

* Maybe 9, overprovision to 12 for margin

VID buffer depth = (RTT + think time at src + think time at dest) \* bandwidth

* Maybe 6, overprovisioning to 12 for margin

Header flit length = 4 bits VID + 2 bits type + 4 bits credit + 4 bits route info (2 bits for each dimension)

* 14 bits, ~~fits within 16 bit port width~~
* Change VID field width to 6 bits to accommodate switch allocation without the use of [R] to improve pipelining, and change port width to 32 bits to reduce [VID] and [TYPE] overhead %

Header flit format = [VID, Type, Length/Credits, Route]

Flit types => [11] Header, [10] body, [01] tail, [00] credit

How to handle allocation and deallocation of resources?

Delimitation symbols? Length field embedded in header flit? Or type field include tail in its set

Additionally should packets be able to arrive back to back in a single VID?

**Routing Engine**

Use a single routing function with FCFS arbitration.

**Unit Crossbar**

Use bit-slicing to distribute critical paths penalty of expanded data width across bits in crossbar. Additionally explore how a butterfly network could be built on crossbars in order to reduce quadratic penalty of crossbar scaling. Goal would be to have a fully connected butterfly network with no risk of collisions.

Xilinx 7-series FPGAs CLB LUT6 architecture can implement 4:1, 8:1, or 16:1 mux’s. Up to 4 4:1 mux’s, 2 8:1 mux’s, or 1 16:1 mux can be implemented in a single slice.

If using 16:1 mux in one slice, and a 4:1 mux in other slice, one CLB (2 slices) would be used to implement a 64:1 mux.

**Credit System**

Routing unit tracks credit counters for its local input unit buffers and the input unit buffers of its neighboring nodes. Routers update their neighbors of their current credit counts by way of issuing credits periodically either by piggybacking on outgoing flits or by sending standalone credit flits.

Case 1: router sends credits on outgoing flits

E.g.

Flit arrives in router-dest assigned to VID-in.

~~It would be trivial to send credits corresponding to router-dest from router-source as the credit count managed by router-source for router-dest of the respective VID would be a stale copy.~~

This would actually be useful because it tells the destination router the source router’s credit count for the same VID but for the source’s input unit buffers.

e.g. source router sends flit on VID42 and credit value denotes source’s available space on VID42.

This method becomes reliable the more uniform bidirectional traffic becomes.

Two other options would be useful:

[1] the credits are being sent for the most critical VID (implicitly a different VID than the one used for the respective flit) with the VID being correspondingly denoted along with the credits (requires another VID field and thus more header space)

[2] the routers are all using shared memory, perhaps one per bank port such that VIDs are being allocated dynamically, and thus the credit being sent corresponds to the remaining credits for that bank (no additional VID field needed)

Case 2: router sends standalone credit flits

When bidirectional traffic is less uniform either because minimal traffic is being generated between the routers or backpressure is being applied on the source router.

E.g.

Flit is stuck in source router because it’s out of credits. Destination router has no traffic scheduled for source router. Destination router’s buffers eventually free up and then sends a credit flit to the source router thereby alleviating the credit stall.

Note: if credit stalls are occurring on both sides of the VID, e.g. destination’s buffers are full and source’s buffers are full, then we have a deadlock. Our design has no deadlock detection, avoidance, or recovery features.

If no outgoing flits are scheduled for source router, when and how will destination router send credit flits?

* Create a credit management module: for any VID’s that have fallen below 50% availability, have yielded any gain in availability since, and have no pending flits on the outgoing VID, generate a credit flit

**Queueing Arbitration**

Let number of inputs to arbiter, n = 60

Maximum service time, T-max = 60

Arrival interval t-a = 1

So Max difference in arrival time ∆t = 2nT-max = 7200

And width of time stamp w-t = log2(∆t/t-a) = log2(7200) => 13 bits