Multicast in router

microarchitecture

Joji Philip

Multicast in routers

[1 Introduction 2](#_Toc497686992)

[1.1 Restrictions and assumptions 2](#_Toc497686993)

[1.2 NocStudio 2](#_Toc497686994)

[2 Bridge updates 3](#_Toc497686995)

[3 Router updates 4](#_Toc497686996)

[3.1 Multicast route lookup 4](#_Toc497686997)

[3.2 Packet replication 4](#_Toc497686998)

# Introduction

Multicast requires a single packet from Source Bridge to be replicated within the NoC and delivered to multiple destinations.

## Restrictions and assumptions

* Multicast support is currently planned only for Crux or streaming NoC
* NocStudio limits multicast traffic to a single NoC layer
* A single bridge interface generates and receives multicast traffic
* There is no multicast from a NoC layer to multiple host interfaces at the destination
* No upsizing/downsizing is required for multicast packets
* No Clock crossing supported in VC buffers for multicast layers. Link clock crossing might still be used
* Weighted barrier based QoS will not be supported on multicast layer
* Multicast layer can support unicast traffic as well
* Multi-flit multicast packets must be supported, though restricting to a maximum supported size is acceptable.

## NocStudio

Currently, Nocstudio imposes some topological and route/turns restrictions on the multicast layers to allow network level deadlock free routes between any source and any destination on the multicast layer. Based on multicast traffic specification, it then creates unicast traffic routes to each destination.

# Bridge updates

* Unicast traffic destination is currently specified using

{Destination node ID, destination interface ID, class of traffic}.

* Multicast host interface and NoC layer will use a bit vector to specify the destination nodes
* P\_NUM\_MULTICAST\_DEST will decide the number of destination ports on the MC layer. This parameter decides the width of the MC vector
* Multicast interface only talks to the multicast NoC layer so no route lookup is needed on multicast interface
* New port \*multicast\_destinations\* is to be added to host interfaces. On the noc layers side of streaming TX bridge, multicast destination list is multiplexed with the unicast route\_info pins
* No port changes should be required on streaming RX bridge
* On Tx bridge, NocStdudio will provide and new parameters P\_HST\_MULTICAST\_ENB, P\_NOC\_MULTICAST enable to designate host and noc layers with multicast support. Related logic should be generated off.

# Router updates

## Multicast route lookup

* Each router input port has a constant input port specifying multicast lookup table. Note that this is an input pin and not parameter to allow common instance stamping.

Input [8\*P\_NUM\_MULTICAST\_DEST-1:0] \*\_multicast\_lookup\_tbl.

* The input ports maintain destination bit maps for each output port. This table is setup by noctsudio after it creates multicast routing trees. Note that from a given input port on a router, a particular destination can be reached through only one output port of the router.
* On an input port, incoming packet’s MC destination list is ANDed with each output port’s bit map and OR reduced. If a non-zero value occurs, then one of the multicast destination is reached through that port. This operation provides the outputs ports of the router on which the input MC packet has to be sent.

### Combined MC and UC

Multicast layer can carry both multicast and unicast traffic. Following should be done to support this

* Each packet has a single bit identifying routing type. \*\_mc\_pkt: 1’b1 for multicast, 1’b0 for unicast
* Sideband field carrying routing information is either carrying the existing unicast turn based routing information or the destination port list for MC routing.
* Currently output port is binary encoded \*\_oport[2:0]. This needs to be changed to be a vector encoding the output port (unicast) or ports (for multicast). Note that this vector encoding is only for multicast layer. Unicast-only layers must continue to use binary encoding for output port.

### Precomputing port list

For multicast routing, it is desirable to precompute the output port list one hop early. So a router computes for each of its output ports, the list of output ports on the next router.

Each output port has a set of 8 destination lists, one for each of the output ports on the downstream router at that output port. At each input port, these tables are used along with the incoming packet’s MC destination list to come up with the next hop port list for each output port.

Note that this lookup only needs to be performed for the directional output ports. Host output ports don’t need the next hop output port list as they never connect to another router.

* Providing these tables in the form of pins on the router can be expensive as it can make the IO count and hence area large
* An alternative is to provide these tables in the form of a superset parameter table. A single common parameter specifies the tables on all the routers in the noc. Appropriate section of these tables in used within a router using the router ID.

{router ID,

{N-output, 8\*destination list},

{E-output, 8\*destination list},

{W-output, 8\*destination list},

{S-output, 8\*destination list}}

## Packet replication

A multicast packet in an input VC buffer needs to be sent on multiple output ports before it can be removed from the input buffer. Implementation can introduce dependencies which can lead to deadlocks. To avoid deadlocks, a multicast packet on an output port must be able to make progress independent of its progress on other output ports. To achieve this, one of the following can be done.

1. A multicast packet is picked from the head of an input buffer only if each and every output port has full credit to accept the entire packet.
2. Each output port can independently send flits from the input buffer, irrespective of other destination ports being blocked.

### Micro-architecture options

#### Multiple input buffers:

In this option each input port maintains a buffer for each output port. As a packet arrives from the input link, multicast address lookup is performed and packet is copied into all input buffers corresponding to the targeted output ports. These buffers now arbitrate for respective output ports independently.

#### Input buffer with multiple read pointers

In this scheme input VC buffers will have a read pointer for each output port. All the read pointers move together like a single pointer for normal packets in the input VC buffer. When head of input buffer is a MC packet, requests are raised to the target destination ports using logic similar to exiting request logic. Now each read pointer can move independently based on grants from output ports. Once multicast packet finishes on one of the output ports, its read pointer doesn’t advance further. Once all output ports have finished their MC packet, their read pointers align and a new packet from the input buffer can be serviced.

#### Multicast output buffer

This scheme will need to know the max size of multicast packet. The intended application needs a size of 2 flits. Every output port has a dedicated multicast buffer of the maximum packet size for every input port that can multicast to it. A multicast packet from an input buffer requests only when MC buffer on all its destination output ports are free. This allows all the destination output ports to unload the packet simultaneously from the input buffer. Output ports then arbitrates among the MC buffers to transmit the flits on the output link.

Options #1 and #2, multiply the number of data wires within the router, crossing from input ports to outputs. This speedup is needed for the multicast packet replication, but wiring congestion can become a physical design challenge if the data path is wide.

#### Multicast for single flit packets

There is significant simplification of the packet replication logic if multicast packets are restricted to be be single flit packets.

* Input VC buffer does not need multiple read pointers to allow each output port to make independent progress on a multi-flit packet
* Multiple data busses from input to output are no longer required for the same reason.

Exiting input VC buffer can be used as it is currently. Multicast related logic can just be handled in the ivcctrl.

* Each IVC controller generates requests to all the ready target output ports simultaneously. This changes the exiting request vector from one-hot to multi hot.
* Multiple output ports can grant in a given cycle and transfer out the multicast pkt/flit at the head of the VC buffer. However the packet is not read/popped out from the buffer.
* Granted ports have to be removed from the active multicast vector.
* This step has to be repeated till grant has been received from all target output ports for that multicast packet.
* With the final grant, the multicast packet can now be removed from the head of the VC buffer by issuing a read.