HN-SN CHI Bridge

[1 HN-SN CHI Bridge 3](#_Toc476223942)

[1.1 Features 3](#_Toc476223943)

[1.2 ID Renaming 3](#_Toc476223944)

[1.3 Retry management 4](#_Toc476223945)

[1.4 LPID 4](#_Toc476223946)

[1.5 Snoop management 5](#_Toc476223947)

| Revision # | Author | Date | Notes |
| --- | --- | --- | --- |
| 0.1 | Joji Philip | 16 February 2017 | Initial Draft |

# HN-RN CHI Bridge

This bridge has a CHI HN-F interface to iCCCs in a coherent noc and provides a unified CHI RN-F port to an external coherent CHI interconnect.

## Block diagram



## Fucntions

* ID name space management
* Snoop management
* Retry management
* LPID management
* Coherency connection SYSCOREQ/SYSCOACK
* Direct cache transfer
* DVM snoop handling

## ID Renaming

The bridge can support up to a maximum of 256 outstanding requests. Each outgoing request needs to have a unique transaction ID. Also all outgoing requests need to go out with a single srcid. Returning responses need to retrieve the original transaction and source IDs to forward upstream.

1. Configurable max outstanding requests
2. Table sized to max outstanding
3. Every request from the NoC is allocated first free entry
4. Original request’s txnid and srcid will be recorded in this entry
5. Allocated entry ID is used as outgoing txnid
6. Value on input port Bridge\_SRCID is used as outgoing srcid
7. How outgoing target ID is determined: carry forward original target ID?
8. Response txnid is used to index this table and retrieve original source where the response packet is to be forwarded and the txnid of the original request
9. DBID field in response packet has to be recorded and swapped out with the entry ID
10. Response packet also records the SrcID field in the allocated location to be used as target ID for WriteData
11. In CHI-E mode, For CompData response, HomeNID field needs to be stored instead of SRCid to be used as target ID for CompACK
12. Responses in the opposite direction, CompACK, WriteData, index the table with their txnid field.
13. Original DBID is retrieved and replaces txnid in the outgoing response packet. Target ID in the external noc is also retrieved and replaced in the outgoing response
14. Entry remains allocated for the duration of the flow. Last message in the sequence de-allocates the txnid.
15. Allocate deallocate
16. There is a need to keep track of number data packets a request can generate

|  |  |  |
| --- | --- | --- |
| Allocate | Deallocate | intermediate |
| ReadReq | ReadReceipt (if order)  &  (RXDAT: Last CompData (no expcompack)  Or  TXRSP: CompAck (if expcompack)) | RXRSP: ReadReceipt |
| Dataless | RXRSP: Comp (no expcompack)  Or  TXRSP: CompAck (if expcompack) | - |
| CopyBack | TXDAT: Last WriteData |  |
| WriteNoSnp | RXRSP: CompDBIDResp &  TXDAT: Last WriteData  Or  RXRSP: Comp & DBIDResp &  TXDAT: Last WriteData |  |
| WriteUnique | TXDAT: Last WriteData  Or  RXRSP: Last WriteData & CompAck (if expcompack) |  |
| DVMOps | RXRSP: Comp |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hn RXREQ | Sn RXRSP | Sn RXDAT | Hn RXRSP | Hn RXDAT |
| ReadNoSnp, ReadOnce  Alloc Index  Store: Hn SrcID  Store: Hn TxnID  Index -> Sn TxnID | ReadReceipt  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID | CompData  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID  Store: Sn HomeNID  Store: Sn DBID | CompACK  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn HomeNID -> Sn TgtID | - |
| Snoopable Reads  <same as above> | - | CompData  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID  Store: Sn HomeNID  Store: Sn DBID | CompACK  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn HomeNID -> Sn TgtID | - |
| Dataless  <same as above> | Comp  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID  Store: Sn SrcID  Store: Sn DBID | - | CompACK  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn SRCID -> Sn TgtID | - |
| CopyBack  <same as above> | CompDBIDResp  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID  Store: Sn SrcID  Store: Sn DBID | - | - | WriteData  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn SRCID -> Sn TgtID |
| WriteNoSnp  <same as above> | Comp, CompDBIDResp,  DBIDResp  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID  Store: Sn SrcID  Store: Sn DBID | - | - | WriteData  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn SRCID -> Sn TgtID |
| WriteUnique | Comp, CompDBIDResp,  DBIDResp  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID  Store: Sn SrcID  Store: Sn DBID | - | CompACK  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn SRCID -> Sn TgtID | WriteData  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn SRCID -> Sn TgtID |
| DVMOps | DBIDResp, Comp  Index: Sn TxnID  Get: Hn SrcID -> Hn TgtID  Get: Hn TxnID  Store: Sn SrcID  Store: Sn DBID |  |  | WriteData  Index: Hn TxnID  Get: Sn DBID -> Sn TxnID  Get: Sn SRCID -> Sn TgtID |

## Retry management

1. Only requests with AllowRetry set can be retried
2. Every outgoing request is allocated an entry and stored
3. A normal response deallocates it from the retry table
4. A RetryAck for that transaction makes the request enter retry state for that {destination, credit\_type} pair
5. Request can exit retry state only when PCrdGrant is received with {destination, credit\_type} match
6. Note that PCrdGrant can arrive before RetryAck
7. Note that multiple requests can be in the retry state for the same {destination, credit\_type}. In this case, when the PCrdGrant of that type is received from the destination, any one of retry state requests can be retried through an arbitration
8. Note that PCrdGrant is not associated with a transaction ID. It is only identified by the {destination, credit\_type} pair.
9. Since RSP channel cannot be blocked all PCrdGrant must be accepted even if a retry request cant be immediately issued
10. All PCrdGrants are accepted into a credit FIFO as deep as MAX outstanding. Head of the FIFO is used to select one of the pending requests that have received retryAck for the same {destination, credit\_type}.
11. Once a retry request is issued out, the entry transitions back to REQ pending state.

## LPID

1. Each processor can issue requests from multiple logical processor IDs. However outgoing LPID from HN-SN can only have up to 32 unique values.
2. NocStudio will have a host property through which the user can specify number of logical processors in a given request node.
3. Each {Srcid[P\_AGN\_ID\_WIDTH-1:0], LPID[5:0]} forms a unique identifier for logical processors in the NoC.
4. If the sum of logical processors in all the RNs exceeds 32, nocstudio must error out as LPID range on Hn-Sn output will be exceeded.
5. NocStudio will create a static parameter table listing every logical processor ID of every RN in the format {Srcid[P\_AGN\_ID\_WIDTH-1:0], LPID[5:0]}.
6. For example:

RN0 has LPID 0, 1, 2 and 3

RN1 has LPID 0, 1

RN2 has LPID 0

Table should look like (for P\_AGN\_ID\_WIDTH of 8-bits)

{{{8’d2, 5’d0}},

{{8’d1, 5’d1}, {8’d1, 5’d0}},

{{8’d0, 5’d3}, {8’d0, 5’d2}, {8’d0, 5’d1}, {8’d0, 5’d0}}}

Position of each pair indicates its outgoing LPID, maximum number of pairs cannot exceed 32.

1. Hn-Sn bridge will search incoming requests {srcid, LPID} in this table, and bin encoding of the location of match will be used as outgoing LPID.

## Snoop management

* Incoming snoops will perform an address lookup to locate the target ID for the SNP
* This will be embedded in the SrcId field to allows the slave bridge to perform a target ID based lookup for snoop route
* Snoop will allocate an entry in the snoop tracking table
* This will be used to record the incoming snoop txnid
* Outgoing snoop txnid will be the allocated entry ID
* Original srcid will also be recorded at this entry
* A snoop response will index the table with its txnid
* Retrieved srcid will be used as outgoing tgtid and outgoing txnid will also be restored

## Direct cache transfers

Forwarding snoops are converted to non-forwarding type snoops to iCCC and snoop responses are appropriately forwarded to appropriate source RN-F and external home nodes.

## DVM snoops

Currently the NoC does not handle incoming DVM snoops. HnRn accepts DVM snoop pair and returns a dummy DVM snoop response.