# CMI Master bridge

|  |  |  |
| --- | --- | --- |
| Ver | Date | History |
| 0.0 | 14 Feb 2019 | 1. Initial draft |
| 0.1 | 21 Feb 2019 | 1. Added Req channel FIFO port diagram |
| 0.2 | 28 Feb 2019 | 1. Updated on proposal confirmation for NoC chanelling. |
| 0.3 | 11 Mar 2019 | 1. Updated feature list for 1904 release. 2. Updated block diagram where each CMI VC per channel connects to separate switch interface. |
| 0.4 | 25 Mar 2019 | 1. Revised feature list for 1904 release. 2. Listed CMI channel widths. 3. Listed NoC Studio bridge props for CMI and for port enables. 4. Added sections on packetization and flow control |
| 0.5 | 27 Mar 2019 | 1. Added porten for data forward related CMI ports. |
| 0.6 | 11 Apr 2019 | 1. Added packetization section. 2. Added section for CHANCT=1 support. |
| 0.7 | 15 Apr 2019 | 1. Reduced Response channel’s sideband width by 1-bit. |
| 1.0 | 16 Apr 2019 | 1. Updated block diagram to support more than 2 VCs. |
| 1.1 | 26 Apr 2019 | 1. Moved route lookup to before req FIFOs. 2. Added input FIFO for shared req credits. |
| 1.2 | 17Jun 2019 | 1. Updated feature list. 2. Added ports *req\_metadata, rsp\_metadata* and *req\_wbe\_parity\_valid* to optional NocStudio enabled port list. 3. Updated cell size to 91 (+ due to wbe\_parity\_valid addition) 4. Updated CMI channel SB widths, bridge props and porten parameter list to accommodate new req and rsp metadata widths. |
| 1.3 | 20 Jun 2019 | 1. Added note on using MS-bit of loopback info fields by bridges. |
| 1.4 | 25 Jun 2019 | 1. Added global parameters and parameter restrictions to CMI parameter table. 2. Added note CFG BYPASS in ISM and flow control section. |
| 1.5 | 1 Jul 2019 | 1. Updated tid\_width max to 12 (to support IDI2CMI) |
|  | 2 Jul 2019 | 1. Updated sideband widths as we are transporting both sender and receiving CHID (for now) |
|  | 9 Jul 2019 | 1. Added further clarity on rsp loopback info for decode errors. |
| 1.6 | 12 Jul 2019 | 1. Updated variable width support for PSC, PSR, LBINFO and AGNT\_ID widths. |
| 1.7 | 26 Jul 2019 | 1. Add bubble avoidance FIFO for Rdcpl in block diagram. 2. Remove fab credit FIFO and parity check/create from block diagram as they are planned for post 1907. 3. Updated cmi addr width parameter restriction. |
| 1.8 | 24 Sep 2019 | 1. Corrected position of hash mask and compare bits in route table entry. |
| 1.9 | 04 Nov 2019 | 1. Updated block diagram to reflect RSSB supported configuration. 2. Updated inband, sideband and acc\_sb widths for each switch interfaces 3. Added credit re-initialization section and Known issues section. |

Contents

[CMI Master bridge 1](#_Toc23843905)

[Features: 3](#_Toc23843906)

[Mapping CMI channels to NoC channels 5](#_Toc23843907)

[Block diagram 6](#_Toc23843908)

[NoC Channel packet widths 7](#_Toc23843909)

[CMI parameters 9](#_Toc23843910)

[Port-en parameters 11](#_Toc23843911)

[Req channel FIFOs 13](#_Toc23843912)

[Route lookup 14](#_Toc23843913)

[Address transformation 14](#_Toc23843914)

[Packetization 15](#_Toc23843915)

[Write data cell format 15](#_Toc23843916)

[Read completion data cell format 15](#_Toc23843917)

[Bubble avoidance FIFO for rd\_cpl\_data 16](#_Toc23843918)

[ISM and flow control 16](#_Toc23843919)

[CFG BYPASS for ISM INIT 16](#_Toc23843920)

[Credit re-initialization 16](#_Toc23843921)

[CHANCT=1 support 17](#_Toc23843922)

[Known issues 17](#_Toc23843923)

## Features:

Legend: Not supported yet. Not supported.

* CMI Spec 1.1.
* Supported Opcodes: MRd, MWr, MWrPtl, MPCmt, NDTC.
* Configurable VCs (1, 2-4).
* Cross mapping of CMI VCs between Master and Slave bridges.
* Parameterized FIFO depths for credit FIFOs. Configurable register value to exercise a depth lesser than hardware configured config depth.
* ~~Abort response (even if set\_srsp=0) for route lookup error, or address parity error~~. Host is expected to not send a transaction that can cause route lookup or address parity error.
* Tunnelling – Supported only if *req\_no\_addr*=0.
* 32B/64B request lengths.
* Route lookup based on either request address or destination Id (Configurable by NoC) followed by address hashing.
* Interface width = 8/16/32B.
* WDATA\_DELAY – default, fixed. Variables delay modes are for clock crossing between host and bridge, so verification not targeted for the same.
* RDATA\_DELAY – default, fixed Variables delay modes are for clock crossing between host and bridge, so verification not targeted for the same.
* Fabric credits –
  + No shared Fab credit FIFOs, so zero fab credits published to host.
  + Non-zero fab credits published to host agent only for async mode (host and bridge on async clocks).
  + Can receive and honor fab credits published by host agent.
* Unordered fabric. So all write requests have responses. (set\_srsp=0 not supported for writes).
* No interleaving of read completion data.
  + Interleaving of read completion data.
* Parity
  + Parity generation before sending transaction to the receiving host (irrespective of whether it is being sent by the transmitting host).
  + Error response if *req\_address\_parity* error.
  + Parity check and reporting for all fields.
* Data forward – Not supported.
* Dual data mode – Not supported.
  + On the secondary channel, there is no request associated with data and responses. So for wr\_data, there is no information on where to route this data since such information will be available in req channel. In that cases, the secondary request channel’s master bridge might need to get such information from primary request channel’s master bridge.
* Power management –
  + Q-channel or any power related interface.
* Upstream stall acknowledgement.
* ~~Error response if~~ *~~req\_address\_parity~~* ~~error~~.
* Out-of-band QoS support.
* Meta data forwarding.
* Clock crossing.
  + On NoC Side – Bridge and router are on different clocks.
  + CMI Interface side – Bridge and host are on different clocks.
* Early valid signals – Assert few cycles before valid signals, de-assert few cycles after valid signals de-assert.
* Loopback info.
* For below optional CMI features, no processing/generation of the related fields are done in the bridges, but they are forwarded through the bridges.
  + Tunnelling.
  + Speculative read.
  + Near Memory/Far memory
  + clos
  + Directory state.
  + ~~ECC~~.
  + Poison
  + Mirroring.

## Mapping CMI channels to NoC channels

|  |  |  |
| --- | --- | --- |
| **Proposal#1** (Chosen)  Separate NoC Req channels for Rd and Wr. | **Proposal#2** (Not Chosen)  Single NoC Req channel. | **Proposal#3** (Not Chosen)  Combined RrReq, WrReq and Wrdata channels |
|  |  |  |
| Wr Req and Wr data use same NoC channel (Wr Req in user side band) | Wr Req and Wr data are on separate NoC Channels. Wr data address lookup takes inputs from req channel. | Rd/Wr Req and Wr data use same NoC channel (Req in user side band) |
| To maintain compatibility with existing CMI ATS SB and with AXI NoC packetization. | NoC Channels not compatible with existing CMI ATS SB. | Not compatible with existing CMI ATS SB. |
| Wr/Rd Req ordering cannot be maintained in NoC routing as the Rd and Wr requests might be routed separately. Not a fully ordered Fabric. | Wr/Rd Req ordering can be maintained during in the NoC routing. | Wr/Rd Req ordering can be maintained during in the NoC routing. |
|  | Additional resources required in Slave Bridge to align Wr Req and Wr data. |  |
| CMI SB can be modified from current ATS version. | CMI SB’s request and wrdata channels have to be re-built completely. | Existing CMI SB’s request and wrdata channels have to be modified. |

Meteor lake expects un-ordered network. So Proposal#1 chosen.

Subsequent sections in this document follows Proposal#1.

## Block diagram



**Problem**:

If address decoding is done after VC arbitration, choke up of req channel can happen due to lack of the selected VC’s credit at TX switch’s input (from router).

Solution#1 (Chosen post 1904): Storage FIFO VC (CMI VC) arbitration to be done after address decoding, and the arbitration to take into account VC (NoC VC) credit availability (from router to switch). Drawback. Increased FIFO width to store decoded address output. But since the FIFO depths will be for small transactions (<10), its fine. But current switches don’t provide VC (NoC VC) credit availability for each VC (NoC VC) of its interface. Switches to be updated. Till then, we can let the choke up of channel, ~~or when valid is provided to a switch for a VC, if the grant is not available from the VC, arbitration can move to the next VC~~.

Solution#2 (Not Chosen): FIFO VC arbitration done before address decoding. But VC credit availability (from router to switch) is used for VC arbitration. But current switches don’t provide VC credit availability for each VC of its interface. Switches to be updated.

Solution#3 (Chosen for 1904): Connect each CMI VC to separate switch interface. Limits max VCs support to 2. Chosen till VCs can provide VC credit availability to host side.

## NoC Channel packet widths

(Refer Packetization section for cell size calculation)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Interface** | **Channels** | **Inband Width** | **No of Cells** | **Sideband Width** | **Accumulated Sideband width** | **Reference File** |
| MB – Tx0/Tx2/Tx4/ Tx6 INTF  SB – Rx0 INTF | Rd Req channel | Not used – (All zeros) |  | 8 + P\_SYS\_CMI\_CHID\_WIDTH + P\_SYS\_ADDR\_WIDTH +(P\_CMI\_TRPEN\_REQ\_TUNNEL ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_ADDR ? 1 :0) +  (P\_CMI\_TRPEN\_REQ\_PSC ? P\_SYS\_CMI\_PSC\_WIDTH : 0) +  (2\*P\_SYS\_CMI\_AGNTID\_WIDTH) + P\_SYS\_CMI\_TID\_WIDTH + P\_SYS\_CMI\_LBINFO\_WIDTH + (P\_CMI\_TRPEN\_PMEM\_REGION? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_CHAIN? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_ALLOCATE\_NM ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_FETCH\_FM ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_SPECRD ? 1 : 0) + (P\_CMI\_TRPEN\_32BYTE\_REQ ? 1 : 0) + (P\_CMI\_TRPEN\_UNCACHEABLE ? 1 : 0) + (P\_CMI\_TRPEN\_CLOS ? P\_SYS\_CMI\_CLOS\_WIDTH : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_LOOKUP ? 1 : 0) + (P\_CMI\_TRPEN\_DIRECTORY ? 1 : 0) + (P\_CMI\_TRPEN\_DIRECTORY ? 2 : 0) + (P\_CMI\_TRPEN\_REQ\_GT ? 1 : 0) + (P\_CMI\_TRPEN\_MIRROR ? 1 : 0) + (P\_CMI\_TRPEN\_MIRROR ? 1 : 0) + (P\_CMI\_TRPEN\_MIRROR ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_ADDRESS\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_ADDRESS\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_TID\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_TID\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_METADATA ? P\_CMI\_REQ\_METADATA\_WIDTH : 0) |  | $REPO\_ROOT/src/hw/ns\_cmn\_cmi/rtl/ns\_cmn\_cmi\_req\_params.vh |
| MB – Tx1/Tx3/Tx5/Tx7 INTF  SB – Rx1 INTF | Wr Req + Data channel | 9 \* No of cells | P\_CMI\_WR\_DWIDTH | 8 + P\_SYS\_CMI\_CHID\_WIDTH + P\_SYS\_ADDR\_WIDTH +(P\_CMI\_TRPEN\_REQ\_TUNNEL ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_ADDR ? 1 :0) +  (P\_CMI\_TRPEN\_REQ\_PSC ? P\_SYS\_CMI\_PSC\_WIDTH : 0) +  (2\*P\_SYS\_CMI\_AGNTID\_WIDTH) + P\_SYS\_CMI\_TID\_WIDTH + P\_SYS\_CMI\_LBINFO\_WIDTH + (P\_CMI\_TRPEN\_PMEM\_REGION? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_CHAIN? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_ALLOCATE\_NM ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_FETCH\_FM ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_SPECRD ? 1 : 0) + (P\_CMI\_TRPEN\_32BYTE\_REQ ? 1 : 0) + (P\_CMI\_TRPEN\_UNCACHEABLE ? 1 : 0) + (P\_CMI\_TRPEN\_CLOS ? P\_SYS\_CMI\_CLOS\_WIDTH : 0) + (P\_CMI\_TRPEN\_REQ\_NO\_LOOKUP ? 1 : 0) + (P\_CMI\_TRPEN\_DIRECTORY ? 1 : 0) + (P\_CMI\_TRPEN\_DIRECTORY ? 2 : 0) + (P\_CMI\_TRPEN\_REQ\_GT ? 1 : 0) + (P\_CMI\_TRPEN\_MIRROR ? 1 : 0) + (P\_CMI\_TRPEN\_MIRROR ? 1 : 0) + (P\_CMI\_TRPEN\_MIRROR ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_ADDRESS\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_ADDRESS\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_TID\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_TID\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_REQ\_METADATA ? P\_CMI\_REQ\_METADATA\_WIDTH : 0)  **Note**: metadata to be sent as 2nd payload once RSSB supports the feature. | (P\_CMI\_TRPEN\_DIRECTORY ? 2:0) + (P\_CMI\_TRPEN\_POISON ? 1 : 0) | $REPO\_ROOT/src/hw/ns\_cmn\_cmi/rtl/ns\_cmn\_cmi\_wdata\_params.vh (in-band)   $REPO\_ROOT/src/hw/ns\_cmn\_cmi/rtl/ns\_cmn\_cmi\_req\_params.vh (User sideband)  $REPO\_ROOT/src/hw/ns\_cmn\_cmi/  rtl/ns\_cmn\_cmi\_wdata\_acc\_sb\_params.vh (accumulated sideband) |
| MB -Rx1 INTF  SB -TX1/TX3/TX5/TX7 INTF | Rsp Channel | Not used – (All zeros) |  | 3 + P\_SYS\_CMI\_CHID\_WIDTH + P\_SYS\_CMI\_AGNTID\_WIDTH + P\_SYS\_CMI\_TID\_WIDTH + P\_SYS\_CMI\_LBINFO\_WIDTH + (P\_CMI\_TRPEN\_RSP\_PSR ? P\_SYS\_CMI\_PSR\_WIDTH : 0) + (P\_CMI\_TRPEN\_RSP\_ERROR\_TYPE ? 2 : 0) + (P\_CMI\_TRPEN\_RSP\_TID\_PARITY ? 1 : 0) +  (P\_CMI\_TRPEN\_RSP\_TID\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_RSP\_METADATA ? P\_CMI\_RSP\_METADATA\_WIDTH : 0) |  | $REPO\_ROOT/src/hw/ns\_cmn\_cmi/  rtl/ns\_cmn\_cmi\_rsp\_params.vh |
| MB-RX0 -INTF  SB-TX0/TX2/TX4/TX6 INTF | Rd cpl + Rd cpl data channel | 9 \* No of cells | P\_CMI\_RD\_DWIDTH | 1 + P\_SYS\_CMI\_CHID\_WIDTH + P\_SYS\_CMI\_AGNTID\_WIDTH + P\_SYS\_CMI\_TID\_WIDTH + P\_SYS\_CMI\_LBINFO\_WIDTH + (P\_CMI\_TRPEN\_RD\_CPL\_TID\_PARITY ? 1 : 0) + (P\_CMI\_TRPEN\_RD\_CPL\_TID\_PARITY ? 1 : 0)  **Note**: metadata to be sent as 2nd payload once RSSB supports the feature. | (P\_CMI\_TRPEN\_DIRECTORY ? 2:0) + (P\_CMI\_TRPEN\_RDCPL\_ERROR ? 3 : 0) + (P\_CMI\_TRPEN\_POISON ? 2 : 0) | $REPO\_ROOT/src/hw/ns\_cmn\_cmi/rtl/ns\_cmn\_cmi\_rdata\_params.vh (in-band)   $REPO\_ROOT/src/hw/ns\_cmn\_cmi/rtl/ns\_cmn\_cmi\_rdcpl\_params.vh (User sideband)  $REPO\_ROOT/src/hw/ns\_cmn\_cmi/rtl/ns\_cmn\_cmi\_rdata\_acc\_sb\_params.vh  (accumulated sideband) |

Master bridge props in NoC Studio

### CMI parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **1910** | Bridge property (BP) /  Interface property (IP) | Global parameter  (Common for all CMI bridges) | Parameter  Restrictions  (Same/Varied across connected bridges) |
| P\_CMI\_ADDR\_MSB | 15-59 | BP: **cmi\_addr\_width\_msb** |  | Varied |
| P\_CMI\_ADDR\_LSB | 0-6 | BP: **cmi\_addr\_width\_lsb** |  | Varied |
| P\_CMI\_ADDR\_WIDTH | 16-60 | NA | P\_SYS\_ADDR\_WIDTH >= max (P\_CMI\_ADDR\_MSB+1 of all bridges) | Varied |
| P\_CMI\_CHANCT | 1~4 | BP: **cmi\_num\_vcs** |  | Varied |
| P\_CMI\_CHID\_WIDTH | 1-2 | NA | P\_SYS\_CMI\_CHID\_WIDTH >= max (P\_CMI\_CHID\_WIDTH of all bridges) | Varied |
| P\_CMI\_NUM\_TIDS | 2-2048 | BP: **cmi\_num\_tids** |  | Varied |
| P\_CMI\_TID\_WIDTH | 1-12 | BP: **cmi\_tid\_width** | P\_SYS\_CMI\_TID\_WIDTH >= max (P\_CMI\_TID\_WIDTH of all bridges) | Varied  P\_CMI\_TID\_WIDTH of slaves >= masters talking to it |
| P\_CMI\_LINE\_SIZE | 64 | NA |  | Same |
| P\_CMI\_WR\_DWIDTH | 8, 16, 32 | IP: **data\_width** |  | Varied |
| P\_CMI\_RD\_DWIDTH | 8, 16, 32 | IP: **data\_width** |  | Varied |
| P\_CMI\_WR\_DCREDIT | 64 | NA |  | Same |
| P\_CMI\_RD\_DCREDIT | 32 | NA |  | Same |
| P\_CMI\_WDATA\_DELAY | 0-10 | BP: **cmi\_wdata\_delay** |  | Varied |
| P\_CMI\_RDATA\_DELAY | 0-10 | BP: **cmi\_rdata\_delay** |  | Varied |
| P\_CMI\_MAX\_INTRLV\_CPL | 0 | BP: **cmi\_max\_intrlv\_cpl** |  | Varied (0 for now) |
| P\_CMI\_REQ\_METADATA\_WIDTH | 1 | BP: **cmi\_req\_metadata\_width** |  | Same |
| P\_CMI\_RSP\_METADATA\_WIDTH | 1 | BP: **cmi\_resp\_metadata\_width** |  | Same |
| P\_CMI\_WR\_METADATA\_WIDTH | 1 | BP: **cmi\_wr\_metadata\_width**\_ |  | Same |
| P\_CMI\_RD\_METADATA\_WIDTH | 1 | BP: **cmi\_rd\_metadata\_width** |  | Same |
| P\_CMI\_AGNTID\_WIDTH | 3-6 | BP: **cmi\_agentid\_width** | P\_SYS\_CMI\_AGNTID\_WIDTH >= max (P\_CMI\_AGNTID\_WIDTH of all bridges) | P\_CMI\_AGNT\_ID\_WIDTH of master bridge should be wide enough to hold DST\_ID’s (CMI Agent IDs) of all Slave bridges connected to it.  P\_CMI\_AGNT\_ID\_WIDTH of slave bridge should be wide enough to hold SRC\_IDs of all Master bridges connected to it. |
| P\_CMI\_LBINFO\_WIDTH | **1**-32  (CMIM)  **2**-33  (CMIS) | BP: **cmi\_lbinfo\_width** | P\_SYS\_CMI\_LBINFO\_WIDTH >= max (P\_CMI\_LBINFO\_WIDTH of all bridges) | Varied  P\_CMI\_LBINFO\_WIDTH of slaves >= **1 +** masters talking to it |
| P\_CMI\_PSC\_WIDTH | 1, 10 | BP: **cmi\_psc\_width** | P\_SYS\_CMI\_PSC\_WIDTH >= max (P\_CMI\_PSC\_WIDTH of all bridges) | Varied.P\_CMI\_PSC\_WIDTH of slaves >= masters talking to it |
| P\_CMI\_PSR\_WIDTH | 1, 10 | BP: **cmi\_psr\_wdith** | P\_SYS\_CMI\_PSR\_WIDTH >= max (P\_CMI\_PSR\_WIDTH of all bridges) | Varied |
| P\_CMI\_FDATA\_ID\_WIDTH | 1 | NA |  | Same |
| P\_CMI\_CLOS\_WIDTH | 1, 3 | BP: **cmi\_clos\_width** | P\_SYS\_CMI\_CLOS\_WIDTH >= max (P\_CMI\_CLOS\_WIDTH of all bridges) | Varied  P\_CMI\_CLOS\_WIDTH of slaves >= masters talking to it. |

### Port-en parameters

Listed below are group of CMI Master/Slave bridge ports listed against an intended NoC Studio ***porten*** parameter/property.

If a given output port from Master host to Master bridge is not available, then the ***porten***parameter/bridge prop must be 0. The listed Master bridge input ports against the prop will be tied to 0 by NoCStudio.

Example:

If Master host cannot provide *req\_tunnel* output signal to Master Bridge, then the bridge prop *cmi\_tunnel\_enable* mut be set 0.

Master bridge’s input ports *req\_tunnel* will be tied to 0 by NoC Studio.

Exception is for *valid\_early* signals which have to be tied 1 if disabled.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **New Bridge prop** |  | **Request Channel (Master bridge inputs)** | **Req Data channel (Master bridge inputs)** | **Rd cpl (Slave bridge inputs)** | **Rd cpl data (Slave bridge inputs)** | **Rsp (Slave bridge inputs)** |
| cmi\_porten\_valid\_early\_enable |  | req\_valid\_early | req\_wdata\_valid\_early | rd\_cpl\_valid\_early | rd\_cpl\_data\_valid\_early | rsp\_valid\_early |
| cmi\_<signal name>\_enable |  | req\_tunnel | req\_wdata\_metadata |  | Rd\_cpl\_metadata | rsp\_psr |
|  | req\_no\_addr |  |  |  | rsp\_metadata |
|  | req\_psc |  |  |  |  |
|  | req\_metadata |  |  |  |  |
| cmi\_<signal name>\_enable |  | req\_chain |  |  |  |  |
|  | req\_no\_allocate\_nm |  |  |  |  |
|  | req\_no\_fetch\_fm |  |  |  |  |
|  | req\_No\_Lookup |  |  |  |  |
| cmi\_req\_specrd\_enable |  | req\_SpecRd |  |  |  |  |
| cmi\_32byte\_req\_enable |  | req\_length |  |  |  |  |
| cmi\_clos\_enable |  | req\_clos |  |  |  |  |
| cmi\_directory\_enable |  | req\_dir\_only\_hint | req\_wdata\_dir |  | rd\_cpl\_dir\_state |  |
|  | req\_Directory\_update |  |  |  |  |
| cmi\_<signal name>\_enable - (combine X\_parity and X\_parity\_valid under the same prop) |  | req\_address\_parity | req\_wdata\_parity | rd\_cpl\_tid\_parity | rd\_cpl\_parity | rsp\_tid\_parity |
|  | req\_address\_parity\_valid | req\_wdata\_parity\_valid | rd\_cpl\_tid\_parity\_valid | rd\_cpl\_parity\_valid | rsp\_tid\_parity\_valid |
|  | req\_tid\_parity | req\_wbe\_parity |  |  |  |
|  | Req\_tid\_parity\_valid | req\_wbe\_parity\_valid |  |  |  |
| cmi\_ecc\_enable |  |  | req\_wdata\_ecc |  | rd\_cpl\_ecc |  |
|  |  | Req\_wdata\_ecc\_valid |  | rd\_cpl\_ecc\_valid |  |
|  |  |  |  | rd\_cpl\_error |  |
|  |  |  |  | rd\_cpl\_data\_error\_type |  |
| cmi\_poison\_enable |  |  | req\_wdata\_poison |  | rd\_cpl\_data\_poison |  |
| cmi\_mirror\_enable |  | req\_mirror |  |  |  |  |
|  | req\_primary |  |  |  |  |
|  | Req\_mirror\_Failover |  |  |  |  |
| cmi\_req\_gt\_enable |  | req\_GT |  |  |  |  |
| cmi\_rsp\_err\_type\_enable |  |  |  |  |  | rsp\_error\_type |
| cmi\_fdata\_enable |  | req\_fdata |  |  |  | rsp\_fdata |
|  | req\_fdata\_type |  |  |  |  |
|  | req\_fdata\_id |  |  |  |  |

## Req channel FIFOs

* Separate FIFOs per VC for each of the Read and Write requests.
* NoC Studio configurable FIFO depths for each instantiated FIFOs. Additional programmable register limits (should be ≤ NoC studio configured depth) the FIFO depth to be used.
* TBD. If clock crossing between master host and master bridge, a shared FIFO (req fab credit FIFO) for all the requests is present at the CMI interface boundary. The FIFO depth programmable by NoCStudio. Credits relevant to this FIFO are exchanged as *req\_fab\_credit*s.
* If no clock crossing between master host and master bridge, no shared FIFO across rd and wr requests, hence no fab credits are published for the request channel.
* *WR\_DELAY* support.
  + Wr\_data\_fifo arrives WR\_DELAY cycles (default or fixed delay mode) after Req (Wr). Master bridge aligns these two channels before sending downstream. The slave bridge is expected to re-align these two CMI channels before sending downstream.
  + Valid outs for wr\_req\_fifo and for first flit of wr\_data\_fifo is available only if both the FIFOs are not-empty.
  + TBD. To support an offset between write req and write data along the NoC link, switches and routers may be modified. Then there will be separate <NOC\_WR\_DELAY> parameter for the NOC. FIFO read in this case must ensure that write req and write data are offset by this <NOC\_WR\_DELAY> before sending out on the switches.



* The Tx switch can provide a grant on its host side interface’s output only if the host side has a valid data to be fed to the switch. So the FIFOs are configured to prefetch an entry. An entry is popped out of FIFO only if grant is available from the downstream.

## Route lookup

* Nocstudio configures whether Master bridge uses Address based lookup or ID based lookup.
* Address based lookup
  + CMI’s dst id field is not used for any route lookup, but forwarded to the destination.
  + For multiple destinations sharing the same address range, address hashing (configurable through parameter) is used to determine which one to send to.
  + Abort response is sent to Master agent if any route lookup error (even if srsp=0). Abort response has higher priority over responses from NoC.
* ID based lookup.
  + CMI’s dst id field is used for route lookup.
  + CMI request address is forwarded to the destination.
  + For multiple destinations sharing the same destination ID, address hashing (hash function configurable through parameter) is used to determine which one to send to.
* Write and Read requests may need to take a different route on the NoC. So route lookup may also include Wr/Rd opcode.
* Requests for the same address may take a different route on the NoC. So route lookup may also include CMI VC. The CMI VC input to route look up also helps map CMI VC to a NoC VC.

Table 1 CMIM route table entry format

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **UNQID** | **Dest CMOPS** | **CMI DST VC** | **Route path** | **Outp** | **Layer and VC** | **CMD\_TYPE**  **(0: if Rd Request,**  **1: if Wr Request)** | **CMI REQ\_SECURE**  **(dummy)** | **CMI SRC VC** | **Hash Compare Bits** | **Hash Mask Bits** | **Destination Id** |
| 1’b0 | 1’b0 | P\_SYS\_CMI\_CHID\_WIDTH | P\_ROUTE\_PATH\_WIDTH | 3 | P\_LYR\_ID\_WIDTH+2 | 1 | 2 | P\_CMI\_CHID\_WIDTH | P\_HASH\_BITS | P\_HASH\_BITS | P\_CMI\_AGNTID\_WIDTH |

## Address transformation

|  |  |
| --- | --- |
| Stage↓ | **Address in sequence** |
| **Actual address** | req\_address[ADDR\_MSB:0] |
| **CMI Master agent output/**  **CMI Master bridge input** | req\_address[ADDR\_MSB:ADDR\_LSB] |
| **CMI Master bridge** | Append incoming address with 0’s in LSB, translate to system address width and use the resultant address for address route lookup, hashing, relocation etc. |
| **CMI MB output to Noc/**  **CMI SB input from Noc** | Req\_addr[ADDR\_MSB:0] upsized to (and relocated to)  System wide address width |
| **CMI Slave Bridge** | Downsize to CMI address width,  chop of ADDR\_LSB 0 bits, before sending to CMI slave |
| **CMI Slave** | Append incoming address with 0’s in LSB before any processing. |

## Packetization

* CMI spec supports interface sizes of 8B/16B/32B.
* Cell size is 9bit (8 bit data + 1 bit byte enable).
* For an 8B CMI interface, 1 cell is sent per clock cycle on the NoC. For a 16B interface, 2 cells are sent per clock cycle on the NoC.
* Write credits on CMI interface are exchanged on 64B boundaries. On the NoC channel, irrespective of the CMI interface width, a NoC transaction consists of entire packets related to one 64B (32B for *req\_length*=1 i.e., 32B transactions) CMI transaction. Eg: if CMI interface is 8B, a NoC transaction on Wr req NoC channel’s inband consists of 8 (4 if 32B transactions) valid flits/clock cycles within a pair of SOP and EOP.
* Similarly, Read credits on CMI interface are exchanged on 32B boundaries. On the NoC channel, irrespective of the CMI interface width, a NoC transaction consists of entire packets related to one 64B (32B for *req\_length*=1 i.e., 32B transactions) CMI transaction. Eg: if CMI interface is 8B, a NoC transaction on Rd cpl NoC channel’s inband consists of 8 (4 if 32B transactions) valid flits/clock cycles within a pair of SOP and EOP. NoC transaction of 64B instead of 32B is chosen to avoid interleaving when multiple slaves send read completion packets to the same master. **Note**: The response or read completion channels aren’t aware if the transactions correspond to a 64B or 32B transactions. So the Master Bridge appends req\_length to request channel lbinfo’s MS-bit. For the rsp and rdcpl channels send to Master agent, the Master Bridge removes this MS-bit and sends the rest to the Master agent. Hence the Slave Bridge’s lbinfo\_width has to be atleast 1 greater than that of connected Master bridges connected to it.

### Write data cell format

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MSB** | **CMI Req Data 8b Data cell** | | | | | | | | | | **LSB** | | |
|  |  |  |  |  |  |  |  |  |  | **Byte Enable** | | **Data** |  |
|  |  |  |  |  |  |  |  |  |  | 1 | | 8 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

### Read completion data cell format

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MSB** | **CMI Rd Cpl Data 64b Data Cell** | | | | | | | | | | **LSB** | |
|  |  |  |  |  |  |  |  |  |  | ***--*** | | **Data** |
|  |  |  |  |  |  |  |  |  |  | *1* | | 8 |

## Bubble avoidance FIFO for rd\_cpl\_data

CMI spec disallows any bubble within 32B of rd\_cpl\_data sent on CMI interface if no async crossing between connected CMI agents (host and bridge). So a FIFO is instantiated for the purpose which accumulates 32B of rd\_cpl\_data before sending over CMI interface.

## ISM and flow control

Master bridge implementation follows CMI spec 1.01 (ISM responder within Section 4) recommended timing diagrams for Credit initialization state machine.

Master bridge publishes fabric credits if req fab credit FIFO is of non-zero depth. Else Master bridge doesn’t publish fabric credits to the Master CMI agent, during ism initialization and during active transactions. For credits received from Master CMI agent (rsp and rdcpl), the Master CMI agent may or may not send fabric credits to the Master bridge. If the Master CMI agent doesn’t support fab credits, then Master agent is expected to drive its fabric credit put signals towards Master CMI bridge with 0.

### CFG BYPASS for ISM INIT

If Nocstudio configurable parameter P\_CMI\_CFG\_BYPASS=0, bridge initialization starts only after register configuration is done. For this purpose, register programming is expected to write all ISM related config registers (FIFO depths) and then write 1 to *cfg\_credits\_config\_done*. This register bit triggers the ISM initialization.

If parameter P\_CMI\_CFG\_BYPASS=1 (default), then ISM initialization starts immediately after bridge comes out of reset. FIFO depth registers shouldn’t be programmed for the first initialization for this configuration.

### Credit re-initialization

CMI bridges support the ability to re-program credit FIFO depth registers and re-trigger initialization, provided the **re-trigger happens when ISMs are IDLE state**. The fifo depth registers however have to be programmed to a value less than or equal to that of credit fifo depth set by Nocstudio.

Below is the sequence.

1. Ensure that there are no ongoing and outstanding transactions or credits on the CMI interface (CMI interface is in IDLE state).

*wait for* ***reqism\_idle\_status*** *register to be 1.*

1. Enable writing to credit FIFO depth registers.

*Write 0 to* ***credits\_config\_done*** *registers (register present in both Master and Slave bridges)*. *Similar action to be taken for CMI host agents*.

***credits\_initialized*** *status register will get cleared in the CMI bridges.*

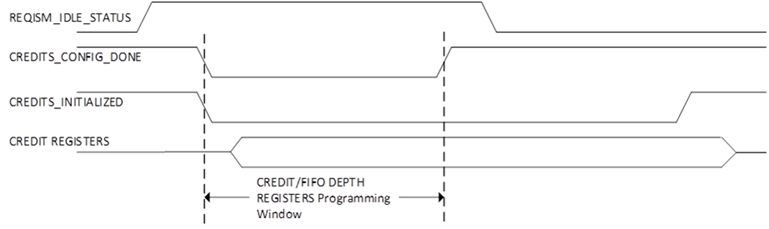
1. Write new values to credit FIFO depth registers if required.
2. Trigger credit initialization on both requestor and responder

*Write 1 to credits\_config\_done register in bridges. Similar action to be taken for CMI host agents.*

*~~If the ISMs are still in IDLE state,~~ credit initialization starts*.

**Not supported yet** ~~-~~ *~~If the ISMs are in non-IDLE state~~* ~~(ISM might have moved from IDLE to non-idle before~~ *~~credits\_config\_done~~* ~~is deasserted)~~ *~~due to incoming transactions, credit initialization starts only after ISMs are back to idle~~.*

*A value of 1 on credits\_intialized status register indicates that credit re-initialization is complete.*



### CHANCT=1 support

CMI Spec allows a values of 2-4 for CHANCT parameter. In addition the current Master bridge also supports CHANCT=1. If CHANCT=1, FIFOs and most of the logic related to VC1 will not be instantiated. For response and read completion channels, credits are expected for VC0 within first 8 cycles. For request channels, credits are published for VC0 in first 8 cycles. Queue depth credits are published in the next 8 cycles. Since only 1 VC is present if CHANCT=1, queue depths credit can also be ignored by the master agent as the values are identical to the credits for VC0.

## Known issues

<https://netspeed.atlassian.net/browse/BI-13173> - At times, even after all transactions are complete, Slave bridge incorrectly assumes that some transactions are outstanding and hence doesn’t go to IDLE state. There are no transactions lost because of this, but any power saving related to IDLE state in Slave agents might not happen.