**CMI Bridge Checkers**

# **1. Block** diagram

The block diagram depicted in Fig 1 shows CMI bridge checkers in NoC verification environment. The CMI checkers are bound to RTL bridge design top using System Verilog bind. The System Verilog bind allows to monitor the RTL interface and probe RTL signals inside the design by hierarchical referencing. The CMI Bridge checkers are based on CMI HAS version 1.01 and CMI Master and slave bridge MAS

DUT

**SV Bind (bridge top)**

**CMI Bridge Checker**

*Figure 1:* Block Diagram of CMI Bridge Checkers

# **2. Checker Controls**

CMI Bridge checkers can be enabled and disable though value based simv plus arguments during the runtime. There is no capability to disable individual checkers during the compile time.

The following table shows available bridge checker plusargs and value to enable or disable:

|  |  |
| --- | --- |
| Plusarg | Behavior |
| +ns\_cmim\_checker\_en=0 | Disable CMI master bridge checker |
| +ns\_cmim\_checker\_en=1 | Enable CMI master bridge checker |
| +ns\_cmis\_checker\_en=0 | Disable CMI slave bridge checker |
| +ns\_cmis\_checker\_en=1 | Enable CMI slave bridge checker |

*Table 1:* CMI Bridge Checkers Control plusargs

# **3. Unknown Checks**

When design is in non-reset phase, control signals should not be in unknown or hi-impedance state. Table 2-8 depicts XZ checks performed on the individual channel control signals based on the corresponding qualifier, default checks on valid, credit and stall signals, finally the CSRs.

|  |  |
| --- | --- |
| Property name | RTL Signal |
| REQ\_STALL\_CTL\_NOT\_XZ | req\_stall |
| REQ\_VALID\_CTL\_NOT\_XZ | req\_valid |
| REQ\_VALID\_EARLY\_CTL\_NOT\_XZ | req\_valid\_early |
| REQ\_WDATA\_VALID\_CTL\_NOT\_XZ | req\_wdata\_valid |
| REQ\_WDATA\_VALID\_EARLY\_CTL\_NOT\_XZ | req\_wdata\_valid\_early |
| RD\_CPL\_VALID\_CTL\_NOT\_XZ | rd\_cpl\_valid |
| RD\_CPL\_VALID\_EARLY\_CTL\_NOT\_XZ | rd\_cpl\_valid\_early |
| RD\_CPL\_DATA\_VALID\_CTL\_NOT\_XZ | rd\_cpl\_data\_valid |
| RD\_CPL\_DATA\_VALID\_EARLY\_CTL\_NOT\_XZ | rd\_cpl\_data\_valid\_early |
| RSP\_VALID\_CTL\_NOT\_XZ | rsp\_valid |
| RD\_CREDIT\_PUT\_CTL\_NOT\_XZ | rd\_credit\_put |
| WR\_CREDIT\_PUT\_CTL\_NOT\_XZ | wr\_credit\_put |
| REQ\_FAB\_CREDIT\_PUT\_CTL\_NOT\_XZ | req\_fab\_credit\_put |
| RD\_CPL\_CREDIT\_PUT\_CTL\_NOT\_XZ | rd\_cpl\_credit\_put |
| RSP\_CREDIT\_PUT\_CTL\_NOT\_XZ | rsp\_credit\_put |
| RSP\_FAB\_CREDIT\_PUT\_CTL\_NOT\_XZ | rsp\_fab\_credit\_put |
| RD\_CPL\_FAB\_CREDIT\_PUT\_CTL\_NOT\_XZ | rd\_cpl\_fab\_credit\_put |
| REQ\_STALL\_ACK\_NOT\_XZ | req\_stall\_ack |
| RSP\_RD\_CPL\_STALL\_ACK\_NOT\_XZ | rsp\_rd\_cpl\_stall\_ack |
| RSP\_RD\_CPL\_STALL\_NOT\_XZ | rsp\_rd\_cpl\_stall |

*Table 2:* Non-reset XZ Assertions

The following assertions are qualified by req\_valid:

|  |  |
| --- | --- |
| Property name | RTL Signal |
| REQ\_ADDR\_NOT\_XZ | req\_address |
| REQ\_CHID\_NOT\_XZ | req\_chid |
| REQ\_PRIORITY\_NOT\_XZ | req\_priority |
| REQ\_OPCODE\_NOT\_XZ | req\_opcode |
| REQ\_TUNNEL\_NOT\_XZ | req\_tunnel |
| REQ\_NO\_ADDR\_NOT\_XZ | req\_no\_addr |
| REQ\_PSC\_NOT\_XZ | req\_psc |
| REQ\_SRCID\_NOT\_XZ | req\_srcid |
| REQ\_TID\_NOT\_XZ | req\_tid |
| REQ\_DSTID\_NOT\_XZ | req\_dstid |
| REQ\_LBINFO\_NOT\_XZ | req\_lbinfo |
| REQ\_PMEM\_REGION\_NOT\_XZ | req\_pmem\_region |
| REQ\_SECURE\_REGION\_NOT\_XZ | req\_secure\_region |
| REQ\_SRSP\_NOT\_XZ | req\_srsp |
| REQ\_DATA\_TYPE\_NOT\_XZ | req\_fdata\_type |
| REQ\_FDATA\_ID\_NOT\_XZ | req\_fdata\_id |
| REQ\_CHAIN\_NOT\_XZ | req\_chain |
| REQ\_NO\_ALLOCATE\_NM\_NOT\_XZ | req\_no\_allocate\_nm |
| REQ\_NO\_FETCH\_FM\_NOT\_XZ | req\_no\_fetch\_fm |
| REQ\_SPECRD\_NOT\_XZ | req\_specrd |
| REQ\_LENGTH\_NOT\_XZ | req\_length |
| REQ\_UNCACHEABLE\_NOT\_XZ | req\_uncacheable |
| REQ\_CLOS\_NOT\_XZ | req\_clos |
| REQ\_NO\_LOOKUP\_NOT\_XZ | req\_no\_lookup |
| REQ\_NO\_DIR\_ONLY\_HINT\_NOT\_XZ | req\_dir\_only\_hint |
| REQ\_DIRECTORY\_UPDATE\_NOT\_XZ | req\_directory\_update |
| REQ\_GT\_NOT\_XZ | req\_GT |
| REQ\_MIRRO\_NOT\_XZ | req\_mirror |
| REQ\_PRIMARY\_NOT\_XZ | req\_primary |
| REQ\_MIRROR\_FAILOVER\_NOT\_XZ | req\_mirror\_failover |
| REQ\_ADDR\_PARITY\_NOT\_XZ | req\_address\_parity |
| REQ\_ADDR\_PARITY\_VALID\_NOT\_XZ | req\_address\_parity\_valid |
| REQ\_TID\_PARITY\_NOT\_XZ | req\_tid\_parity |
| REQ\_TID\_PARITY\_VALID\_NOT\_XZ | req\_tid\_parity\_valid |
| REQ\_META\_ADATA\_NOT\_XZ | req\_metadata |

*Table 3:* Request Channel XZ Assertions

The following assertions are qualified by wdata\_valid:

|  |  |
| --- | --- |
| Property name | RTL Signal |
| REQ\_WDATA\_BE\_NOT\_XZ | req\_wbe |
| REQ\_WDATA\_DIR\_NOT\_XZ | req\_wdata\_dir |
| REQ\_WDATA\_META\_DATA\_NOT\_XZ | req\_wdata\_metadata |
| REQ\_WDATA\_POISON\_NOT\_XZ | req\_wdata\_poison |
| REQ\_WDATA\_PARITY\_NOT\_XZ | req\_wdata\_parity & req\_wdata\_parity\_valid |
| REQ\_WDATA\_PARITY\_VALID\_NOT\_XZ | req\_wdata\_parity\_valid |
| REQ\_WDATA\_ECC\_VALID\_NOT\_XZ | req\_wdata\_ecc\_valid |
| REQ\_WBE\_PARITY\_NOT\_XZ | req\_wbe\_parity |
| REQ\_WBE\_PARITY\_VALID\_NOT\_XZ | req\_wbe\_parity\_valid |

*Table 4:* Write Data Channel XZ Assertions

The following assertions are qualified by rsp\_valid

|  |  |
| --- | --- |
| Property name | RTL Signal |
| RSP\_CHID\_NOT\_XZ | rsp\_chid |
| RSP\_TYPE\_NOT\_XZ | rsp\_type |
| RSP\_SRCID\_NOT\_XZ | rsp\_srcid |
| RSP\_TID\_NOT\_XZ | rsp\_tid |
| RSP\_LBINFO\_NOT\_XZ | rsp\_lbinfo |
| RSP\_PSR\_NOT\_XZ | rsp\_psr |
| RSP\_ERR\_TYPE\_NOT\_XZ | rsp\_error\_type |
| RSP\_TID\_PARITY\_NOT\_XZ | rsp\_tid\_parity |
| RSP\_TID\_PARITY\_VALID\_NOT\_XZ | rsp\_tid\_parity\_valid |
| RSP\_MEDA\_DATA\_NOT\_XZ | rsp\_metadata |

*Table 5:* Response channel XZ Assertions

The following assertions are qualified by rd\_cpl\_valid

|  |  |
| --- | --- |
| Property name | RTL Signal |
| RD\_CPL\_DATA\_POISON\_NOT\_XZ | rd\_cpl\_data\_poison |
| RD\_CPL\_META\_DATA\_NOT\_XZ | rd\_cpl\_metadata |
| RD\_CPL\_DIR\_STATE\_NOT\_XZ | rd\_cpl\_dir\_state |
| RD\_CPL\_DATA\_ERROR\_TYPE\_NOT\_XZ | rd\_cpl\_data\_error\_type |
| RD\_CPL\_ERROR\_NOT\_XZ | rd\_cpl\_error |
| RD\_CPL\_PARITY\_NOT\_XZ | rd\_cpl\_parity |
| RD\_CPL\_ECC\_VALID\_NOT\_XZ | rd\_cpl\_ecc\_valid |
| RD\_CPL\_PARITY\_VALID\_NOT\_XZ | rd\_cpl\_parity\_valid |

*Table 6:* Read completion channel XZ Assertions

|  |  |
| --- | --- |
| CMI\_CFG\_SCRATCH\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmim\_scratch |
| CMI\_CFG\_BRIDGE\_ID\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmim\_bridge\_id |
| CMI\_CFG\_ERR\_INT\_STATUS\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmim\_error\_interrupt\_status\_int\_wrdata |
| CMI\_CFG\_ERR\_INT\_MASK\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmim\_error\_interrupt\_mask |
| CMI\_CFG\_RD\_WR\_REQ\_MAX\_CREDITS\_NOT\_XZ\_AFTER\_RESET | cfg\_cmimb\_rd\_wr\_req\_max\_credit |
| CMI\_CFG\_RD\_CPL\_MAX\_CREDITS\_NOT\_XZ\_AFTER\_RESET | cfg\_cmimb\_rd\_cpl\_rsp\_max\_credit |
| CMI\_CFG\_ADDRS\_BASE\_MEM\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_addrs\_base\_mem |
| CMI\_CFG\_ADDRS\_MASK\_MEM\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_addrs\_mask\_mem |
| CMI\_CFG\_READ\_ALLOWED\_MEM\_CSR\_NOT\_XZ\_AFTER\_RESET | ext\_cfg\_read\_allowed\_mem |
| CMI\_CFG\_ADDR\_RELOC\_SLV\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_addrs\_reloc\_slv\_mem |
| CMI\_CFG\_ADDR\_RELOC\_SYS\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_addrs\_reloc\_sys\_mem |
| CMI\_CFG\_HASH\_FUNCTIONS\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_hash\_functions\_mem |

*Table 7:* CMI Master CSR XZ Assertions

|  |  |
| --- | --- |
| CMI\_CFG\_ERR\_INT\_STATUS\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_err\_status\_int\_wrdata |
| CMI\_CFG\_ERR\_INT\_MASK\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_err\_mask |
| CMI\_CFG\_ERR\_INJ\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_err\_inj |
| CMI\_CFG\_RD\_WR\_REQ\_MAX\_CREDITS\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_rd\_wr\_req\_max\_credit |
| CMI\_CFG\_RD\_CPL\_RSP\_MAX\_CREDITS\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit |
| CMI\_CFG\_ISM\_IDLE\_STATUS\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_idle\_status |
| CMI\_CFG\_ISM\_IDLE\_THRESHOLD\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_ism\_idle\_threshold |
| CMI\_CFG\_CG\_OVERRIDE\_CSR\_NOT\_XZ\_AFTER\_RESET | cfg\_cmisb\_cg\_override |

*Table 8:* CMI Slave CSR XZ Assertions

# **4. Reset Checks**

When design is in reset phase, control signals such as valid and credit puts must be low:

|  |  |
| --- | --- |
| Property name | RTL Signal |
| REQ\_VALID\_LOW\_IN\_RESET | req\_valid |
| REQ\_VALID\_EARLY\_LOW\_IN\_RESET | req\_valid\_early |
| REQ\_WDATA\_VALID\_LOW\_IN\_RESET | req\_wdata\_valid |
| REQ\_WDATA\_VALID\_EARLY\_LOW\_IN\_RESET | req\_wdata\_valid\_early |
| RD\_CPL\_VALID\_LOW\_IN\_RESET | rd\_cpl\_valid |
| RD\_CPL\_VALID\_EARLY\_LOW\_IN\_RESET | rd\_cpl\_valid\_early |
| RD\_CPL\_DATA\_VALID\_LOW\_IN\_RESET | rd\_cpl\_data\_valid |
| RD\_CPL\_DATA\_VALID\_EARLY\_LOW\_IN\_RESET | rd\_cpl\_data\_valid\_early |
| RSP\_VALID\_LOW\_IN\_RESET | rsp\_valid |
| RD\_CREDIT\_PUT\_LOW\_IN\_RESET | rd\_credit\_put |
| WR\_CREDIT\_PUT\_LOW\_IN\_RESET | wr\_credit\_put |
| REQ\_FAB\_CREDIT\_PUT\_LOW\_IN\_RESET | req\_fab\_credit\_put |
| RD\_CPL\_CREDIT\_PUT\_LOW\_IN\_RESET | rd\_cpl\_credit\_put |
| RSP\_CREDIT\_PUT\_LOW\_IN\_RESET | rsp\_credit\_put |
| RSP\_FAB\_CREDIT\_PUT\_LOW\_IN\_RESET | rsp\_fab\_credit\_put |
| RD\_CPL\_FAB\_CREDIT\_PUT\_LOW\_IN\_RESET | rd\_cpl\_fab\_credit\_put |

*Table 9:* Reset Phase Checks

# **5. Onehot Checks**

The onehot checks are performed on request routing, read/write request VC arbitration and error arbiter logic of the CMI master bridge as shown in table 10.

|  |  |
| --- | --- |
| Property name | RTL Signal |
| CMIM\_TX1\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx1\_ch\_noc\_req |
| CMIM\_TX3\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx3\_ch\_noc\_req |
| CMIM\_TX5\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx5\_ch\_noc\_req |
| CMIM\_TX7\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx7\_ch\_noc\_req |
| CMIM\_TX0\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx0\_ch\_noc\_req |
| CMIM\_TX2\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx2\_ch\_noc\_req |
| CMIM\_TX4\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx4\_ch\_noc\_req |
| CMIM\_TX6\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx6\_ch\_noc\_req |
| CMIM\_RX0\_CH\_NOC\_REQ\_NOT\_ONEHOT | noc\_rx0\_ch\_req |
| CMIM\_RX1\_CH\_NOC\_REQ\_NOT\_ONEHOT | noc\_rx1\_ch\_req |
| ROUTE\_FOUND\_ONEHOT\_WHEN\_REQ\_VALID\_HIGH | route\_found\_arr |
| ERR\_ARB\_OUT\_RW\_ONEHOT\_WHEN\_REQ\_VALID\_HIGH | err\_arb\_out\_rw |

*Table 10:* Onehot Checks on Master Bridge

The onehot checks are performed on response and read completion VC arbitration logic of the CMI slave bridge as shown in table 11.

|  |  |
| --- | --- |
| Property name | RTL Signal |
| CMIS\_TX1\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx1\_ch\_noc\_req |
| CMIS\_TX3\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx3\_ch\_noc\_req |
| CMIS\_TX5\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx5\_ch\_noc\_req |
| CMIS\_TX7\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx7\_ch\_noc\_req |
| CMIS\_TX0\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx0\_ch\_noc\_req |
| CMIS\_TX2\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx2\_ch\_noc\_req |
| CMIS\_TX4\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx4\_ch\_noc\_req |
| CMIS\_TX6\_CH\_NOC\_REQ\_NOT\_ONEHOT | tx6\_ch\_noc\_req |
| CMIS\_RX0\_CH\_NOC\_REQ\_NOT\_ONEHOT | noc\_rx0\_ch\_req |
| CMIS\_RX1\_CH\_NOC\_REQ\_NOT\_ONEHOT | noc\_rx1\_ch\_req |

*Table 11:* Onehot Checks on Slave Bridge

# **6. Flow Control checks**

There are multiple flow control checks performed on both CMI Master and Slave Bridges and as follows:

## 6.1. Credit Initialization Checks

The credit init checks are performed for no short-init. When both requester and responder are in CREDIT\_INIT state, 8 cycles credit put count per VC plus 8 cycles of Qdepth count going out of the bridge is compared against the value programmed in the corresponding credit CSR. If the comparison fails then assert a failure error. The bridge checker natively supports on the fly credit init as the checks are performed a cycle after credit initialization is completed.

## 6.2. Overflow and Underflow Checks

CMI bridge credit counters are initialized as depicted in table 12 and 13. The individual credit counter is debited whenever a request is sent out of the bridge and counter gets incremented on receiving corresponding credit put. Please note that credits work on per virtual channel basis. The rd\_cpl\_credit and rsp\_credits are initialize with value 8’hff on the CMI Master Bridge checker and rd\_credit and wr\_credits are initialize with value 8’ff on the CMI Slave Bridge checker.

The following table depicts VC based credit counter mapping with CSR on CMI master bridge:

|  |  |
| --- | --- |
| Credit Signal | CSR |
| rd\_credit\_vc0 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[7:0] |
| rd\_credit\_vc1 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[15:8] |
| rd\_credit\_vc2 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[23:16] |
| rd\_credit\_vc3 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[31:24] |
| wr\_credit\_vc0 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[39:32] |
| wr\_credit\_vc1 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[47:40] |
| wr\_credit\_vc2 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[55:48] |
| wr\_credit\_vc3 | cfg\_cmimb\_rd\_wr\_req\_max\_credit[63:56] |

*Table 12:* CMI Master Bridge Credit Signal, CSR mapping

The following table depicts VC based credit counter mapping with CSR on CMI Slave Bridge:

|  |  |
| --- | --- |
| Credit Signal | CSR |
| rd\_cpl\_credit\_vc0 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[7:0] |
| rd\_cpl\_credit\_vc1 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[15:8] |
| rd\_cpl\_credit\_vc2 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[23:16] |
| rd\_cpl\_credit\_vc3 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[31:24] |
| rsp\_credit\_vc0 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[39:32] |
| rsp\_credit\_vc1 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[47:40] |
| rsp\_credit\_vc2 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[55:48] |
| rsp\_credit\_vc3 | cfg\_cmisb\_rdcpl\_rsp\_vc\_max\_credit[63:56] |

*Table 13:* CMI Slave Bridge Credit Signal, CSR mapping

# **7. Decode Error**

The following 3 types of decode errors are checked in the CMI master bridge today:

## 7.1. Address Decode or Route not found

The RTL route\_found\_arr signal is used to decide decode error and create a response expectation based on route\_found\_arr.

## 7.2. Address Parity Error

Address parity checks are performed on every valid input address along with req\_addr\_parity\_valid signal and NocStudio param P\_CMI\_PARITY\_EN. If the input parity value doesn’t match with the calculated address parity then a decode error response is expected by the CMI master bridge checker

## 7.3. DST\_ID Based Routing Decode Error

The RTL signal route\_found\_arr of address decode block is used to determine routing of packets over the NoC. The logic is same as address decode error and it expects P\_ID\_BASED\_ROUTE\_LKUP\_MEM NocStudio parameter to be programmed to value 1.

# **8. Packet Checks**

## 8.1. SOP EOP Checks

The **S**tart **O**f **P**acket and **E**nd **O**f **P**acket are generated based on the interface width, request type and request length and these are performed on all transmitting (Tx) and receiving (Rx) ports. Table 14 and 15 depicts the CMI bridge checker expectations.

|  |  |  |
| --- | --- | --- |
| Interface Width | SOP | EOP |
| 32B | Set for every flit | Set for every flit |
| 16B | Set for every flit | Set for every flit |
| 8B | Set for every flit | Set for every flit |

*Table 14:* SOP, EOP Checks on Request and Response flits

|  |  |  |  |
| --- | --- | --- | --- |
| Request Length | Interface Width | SOP | EOP |
| 0 | 32 | Set for the first flit of 64B | Set for the last flit of 64B |
| 0 | 16 | Set for the first flit of 64B | Set for the last flit of 64B |
| 0 | 8 | Set for the first flit of 64B | Set for the last flit of 64B |
| 1 | 32 | Set for every flit of 32B | Set for every flit of 32B |
| 1 | 16 | Set for the first flit of 32B | Set for the last flit of 32B |
| 1 | 8 | Set for the first flit of 32B | Set for the last flit of 32B |

*Table 15:* SOP, EOP Checks on write data and read completion flits

## 8.2. Side Band Packet checks

### 8.2.1. Request SB Packets

CMI fabric supports read and write requests with up-to 4 virtual channel support for each request type. Table 16 and 17 shows the request channel transmitting/receiving ports mapping. Please refer to CMI Spec and CMI Bridge MAS for packet structure.

|  |  |
| --- | --- |
| CMI Master Interface | Switch Interface |
| CMI request channel(Read, Channel 0) | tx0\_ch\_noc\_sb |
| CMI request channel(Read, Channel 1) | tx2\_ch\_noc\_sb |
| CMI request channel(Read, Channel 2) | tx4\_ch\_noc\_sb |
| CMI request channel(Read, Channel 3) | tx6\_ch\_noc\_sb |
| CMI request channel(Write, Channel 0) | tx1\_ch\_noc\_sb |
| CMI request channel(Write, Channel 1) | tx3\_ch\_noc\_sb |
| CMI request channel(Write, Channel 2) | tx5\_ch\_noc\_sb |
| CMI request channel(Write, Channel 3) | tx7\_ch\_noc\_sb |

*Table 16:* CMI Master request and transmitter interface mapping

|  |  |
| --- | --- |
| CMI Slave Interface | Switch Interface |
| CMI request channel(Read) | noc\_rx0\_ch\_sb |
| CMI request channel(Write) | noc\_rx1\_ch\_sb |

*Table 17:* CMI Slave request and receiver interface mapping

### 8.2.2. Response SB Packets

Table 18 and 19 depicts response channel mapping with transmitting/receiving interface on CMI Master or Slave Bridge. Please refer to CMI Spec and CMI Bridge MAS for packet structure.

|  |  |
| --- | --- |
| CMI Master Interface | Switch Interface |
| CMI Response channel | noc\_rx1\_ch\_sb |

*Table 18:* CMI Master response and receiver interface mapping

|  |  |
| --- | --- |
| CMI Slave Interface | Switch Interface |
| CMI Response Channel 0 | tx1\_ch\_noc\_sb |
| CMI Response Channel 0 | tx3\_ch\_noc\_sb |
| CMI Response Channel 2 | tx5\_ch\_noc\_sb |
| CMI Response Channel 3 | tx7\_ch\_noc\_sb |

*Table 19:* CMI Slave response and transmitter interface mapping

### 8.2.3. Read completion SB Packets

Table 20 and 21 depicts read completion channel mapping with transmitting/receiving interface on CMI Master or Slave Bridge. Please refer to CMI Spec and CMI Bridge MAS for packet structure

|  |  |
| --- | --- |
| CMI Master Interface | Switch Interface |
| CMI Read completion channel | noc\_rx0\_ch\_sb |

*Table 20:* CMI Master read completion and receiver interface mapping

|  |  |
| --- | --- |
| CMI Slave Interface | Switch Interface |
| CMI Read completion channel(Channel 0) | tx0\_ch\_noc\_sb |
| CMI Read completion channel(Channel 1) | tx2\_ch\_noc\_sb |
| CMI Read completion channel(Channel 2) | tx4\_ch\_noc\_sb |
| CMI Read completion channel(Channel 3) | tx6\_ch\_noc\_sb |

*Table 21:* CMI Slave Read Completion and transmitter interface mapping

## 8.3. Tx/Rx port optimization

NocStudio will optimize the port width of the transmitting or receiving ports based on the valid traffic. Table 22 shows NocStudio param and value, should be used to determine if the port got optimized by the software.

|  |  |  |
| --- | --- | --- |
| NocStudio Param | Valid Traffic | No Traffic |
| P\_TX0\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_TX1\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_TX2\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_TX3\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_TX4\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_TX5\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_TX6\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_TX7\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_RX0\_NOC\_SB\_WIDTH | >1 | 1 |
| P\_RX1\_NOC\_SB\_WIDTH | >1 | 1 |

*Table 22:* CMI transmitter/ receiver port optimization

# **9. Exit Checks**

CMI bridge checkers perform fifo empty checks to ensure all transactions are drained before the simulation is terminated. In addition to that bridge checkers also perform total credit count checks: number of requests/responses and number of credits transfer/received should match.

# **10. Unsupported Features Checks**

CMI master bridge checker asserts for the following not supported features:

* SRSP must be set for all write requests
* Read completion data interleaving
* Data forwarding is not supported

Along with the above unsupported features mentioned above, the following assertions are also enabled:

* Out of range address decode errors
* Address parity errors
* Invalid opcodes

The valid opcodes are mentioned in table 23, any opcode other than the ones mentioned in table 23 are considered to be invalid.

|  |  |  |
| --- | --- | --- |
| Opcode | Enumeration | Type |
| 3'b000 | MRD | Read |
| 3'b001 | MWR | Write |
| 3'b011 | MWRPTL | Write |
| 3'b101 | MPCMT | Write |
| 3'b110 | NDTC | Read |

Table 23: CMI Valid Opcodes

# **11. Validation Collaterals and Checkers path**

## 11.1. Checkers

ns\_cmimstrbrdg/verif/ns\_cmim\_checker.sv

ns\_cmislvbrdg/verif/ns\_cmis\_checker.sv

## 11.2. Checker bind

ns\_common/verif/bench/cmi\_nocs/ns\_bind\_cmim\_checker.svh

ns\_common/verif/bench/cmi\_nocs/ns\_bind\_cmis\_checker.svh

## 11.3. Defines and Params

ns\_common/verif/bench/cmi\_nocs/ns\_cmi\_checker\_defines.vh

ns\_common/verif/bench/cmi\_nocs/ns\_cmi\_checker\_params.vh

## 11.4. Compilation file list

ns\_common/verif/ns\_cmi\_checkers.vc

**12**. **Parity checks:**

**12.1. req\_tid\_parity:**

Check if req\_tid\_parity is xor(req\_srcid, req\_tid) when req\_tid\_parity\_valid is high.

**12.2 req\_wdata\_parity:**

Check if req\_wdata\_parity is xor of req\_wdata anded with req\_wbe (based on interface widths) when req\_wdata\_parity\_valid is high.

**12.3 req\_wbe\_parity:**

Check if req\_wbe\_parity is xor of req\_wbe[31:0] when req\_wbe\_parity\_valid is high.

**12.4 rd\_cpl\_tid\_parity:**

Check if rd\_cpl\_tid\_parity is xor(rd\_cpl\_srcid, rd\_cpl\_tid) when rd\_cpl\_tid\_parity\_valid is high.

**12.5 rd\_cpl\_parity:**

Check if rd\_cpl\_parity is xor of rd\_cpl\_data (based on interface widths) when rd\_cpl\_parity\_valid is high.

**12.6 rsp\_tid\_parity:**

Check if rsp\_tid\_parity is xor(rsp\_srcid, rsp\_tid) when rsp\_tid\_parity\_valid is high.

**13. Short credit initialization support:**

Bridge checkers are updated to support short credit initialization support. Credits accumulation logic during credit initialization state is modified to support short credit initialization as well.

Master and slave bridges do not initiate short credit initialization sequence but will honor the short credit initialization from agents. So checks have been added in master bridge checker to check there is no short credit initialization from responder ISM (master bridge ISM) and in slave bridge checker to check there is no short credit initialization from requester ISM (slave bridge ISM).

**14. Interrupts:**

**14.1 Interrupt details:**

**14.1.1 CMIMB:**

|  |  |  |
| --- | --- | --- |
| **Interrupts** | **Bit position for intr\_mask, intr\_svrty and intr\_status registers** | **Type of interrupt** |
| Req channel route lookup table error | 0 | Fatal/Non fatal |
| Read max credit error | 1 | Fatal/Non fatal |
| Write max credit error | 2 | Fatal/Non fatal |
| Unsupported feature error | 3 | Fatal/Non fatal |
| Req TID parity error | 4 | Fatal/Non fatal |
| Req WDATA parity error | 5 | Fatal/Non fatal |
| Req WBE parity error | 6 | Fatal/Non fatal |
| Unused | [63:7] |  |

**Default values:**

intr\_mask : 0x7F

intr\_status : 0x00

intr\_svrty : 0x00

**14.1.2 CMISB:**

|  |  |  |
| --- | --- | --- |
| **Interrupts** | **Bit position for intr\_mask, intr\_svrty and intr\_status registers** | **Type of interrupt** |
| Read completion buffer overflow error | 0 | Fatal |
| Response max credit error | 1 | Fatal/Non fatal |
| Read completion max credit error | 2 | Fatal/Non fatal |
| ID based routing lookup error | 3 | Fatal |
| Unsupported feature error | 4 | Fatal |
| Response TID parity error | 5 | Fatal/Non fatal |
| Read completion TID parity error | 6 | Fatal/Non fatal |
| Read completion data parity error | 7 | Fatal/Non fatal |

**Default values:**

intr\_mask : 0xE6

intr\_status : 0x00

intr\_svrty : 0x19

**14.2 X/Z checks:**

Bridge checkers are updated to check X/Z after reset on signals: interrupt, interrupt\_nfatal and interrupt\_fatal. X/Z checks after reset are also done on interrupt CSRs: intr\_mask, intr\_status and intr\_svrty.

**14.3 Interrupt behavior checks:**

Checks are in place to check that interrupt, interrupt\_nfatal and interrupt\_fatal signals are asserted when an error occurs depending on intr\_mask and intr\_svrty register bits.

When the status bits for any interrupt is cleared, bridge checkers check that the corresponding interrupts are low. There are also checks to make sure that interrupt signals are asserted until status bits are cleared.

There are checks to make sure that there is no unexpected interrupt signals toggling.

**15. Optional ports checks:**

CMI optional ports by default are tied by NocStudio. Only when we enable them with NS props, they get connected to the agents. We have parameters available for valid\_early and parity signals in bridges to check whether an optional port prop is enabled or not.

Bridge checkers make use of these parameters and check when the props are disabled, the corresponding optional signals are tied appropriately. Signals for which such checks are present are

CMIMB : req\_valid\_early, req\_wdata\_valid\_early, req\_address\_parity, req\_address\_parity\_valid, req\_tid\_parity, req\_tid\_parity\_valid, req\_wdata\_parity, req\_wdata\_parity\_valid, req\_wbe\_parity, req\_wbe\_parity\_valid

CMISB : rd\_cpl\_vaid\_early, rd\_cpl\_data\_valid\_early, rsp\_valid\_early, rd\_cpl\_tid\_parity, rd\_cpl\_tid\_parity\_valid, rd\_cpl\_parity, rd\_cpl\_parity\_valid, rsp\_tid\_parity, rsp\_tid\_parity\_valid