# Programmers Model

**Please note that starting with 1901 release, register renaming is underway in order to provide better readability and explicitly match the functionality.**

CFG NoC delivers a rich set of registers used for NoC control, debug and performance monitoring of the NoC. This section describes all available registers to SoC designers. The complete list of registers implemented for a specific design can be found in the noc\_reference\_manual.html under the <project> folder when the RTL is generated by NocStudio.

Registers are divided into the following categories:

* Router registers
* Bridge registers
* AMBA register
  + AXI Master registers
  + AXI Slave registers
* Regbus layer registers
* Protocol Converter
  + AHB2AXI
  + AXI2AHB
  + APB
* Agent registers
  + DVM Host registers
  + CCC Host registers
  + LLC Host registers
  + Configurable Slave registers
* OCP Register
  + OCP Master registers
  + OCP Slave registers

## Router Registers

### RTR\_CG\_CTRL

This register is used by coarse grained clock gating logic. This register can be set to override coarse clock gating for the entire router. Coarse clock gating for selective routers can be overridden by locally setting this register, if the user does not want incur and aggregate coarse clock gating cycle penalty over a "fast path/critical path" through the NoC.

Attribute: RW

Security: Non-secure

Bit field description:

* **FPO**[0] -   
  1'b1: Coarse clock gating is locally disabled (for fast path)  
  1'b0: Coarse clock gating is locally enabled

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FPO |

Table RTR\_CG\_CTRL register.

### RTR\_CG\_HYST\_COUNT

Programmable interval used by coarse clock gating logic in routers.This count determines the consecutive number of idle cycle after which a router output port initiates coarse clock gating of the local port clock and de-asserts the 'busy' signal to the downstream router. This signal indicates inactivity to the downstream router and allows it to initiate coarse clock gating of its corresponding input port.

Attribute: RW

Security: Non-secure

Bit field description:

* **HYSTERESIS\_COUNTER**[31:0] - Hysteresis counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HYSTERESIS\_COUNTER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table RTR\_CG\_HYST\_COUNT register.

### RTR\_ERR\_PARITY\_MASK

One mask register bit for each parity status bit in RPERR. When mask bit is set, corresponding parity error does not cause an interrupt. Default state is reset for all mask bits, allowing interrupt on any parity error event

Attribute: RW

Security: Non-secure

Bit field description:

* **RI\_3**[31] - Mask Parity Error in VC 3 Buffer Routing Information.
* **PK\_3**[30] - Mask Parity Error in VC 3 Buffer Packet Delineation Controls.
* **SB\_3**[29] - Mask Parity Error in VC 3 Buffer User Sideband.
* **D\_3**[28] - Mask Parity Error in VC 3 Buffer Data.
* **RI\_2**[27] - Mask Parity Error in VC 2 Buffer Routing Information.
* **PK\_2**[26] - Mask Parity Error in VC 2 Buffer Packet Delineation Controls.
* **SB\_2**[25] - Mask Parity Error in VC 2 Buffer User Sideband.
* **D\_2**[24] - Mask Parity Error in VC 2 Buffer Data.
* **RI\_1**[23] - Mask Parity Error in VC 1 Buffer Routing Information.
* **PK\_1**[22] - Mask Parity Error in VC 1 Buffer Packet Delineation Controls.
* **SB\_1**[21] - Mask Parity Error in VC 1 Buffer User Sideband.
* **D\_1**[20] - Mask Parity Error in VC 1 Buffer Data.
* **RI\_0**[19] - Mask Parity Error in VC 0 Buffer Routing Information.
* **PK\_0**[18] - Mask Parity Error in VC 0 Buffer Packet Delineation Controls.
* **SB\_0**[17] - Mask Parity Error in VC 0 Buffer User Sideband.
* **D\_0**[16] - Mask Parity Error in VC 0 Buffer Data.
* **CR**[4] - Mask Parity Error in Link Credit From Downstream Router.
* **RI**[3] - Mask Parity Error in Link Routing Information.
* **PK**[2] - Mask Parity Error in Link Packet Delineation Controls.
* **SB**[1] - Mask Parity Error in Link User Sideband.
* **D**[0] - MaskParity Error in Link Data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RI\_3 | PK\_3 | SB\_3 | D\_3 | RI\_2 | PK\_2 | SB\_2 | D\_2 | RI\_1 | PK\_1 | SB\_1 | D\_1 | RI\_0 | PK\_0 | SB\_0 | D\_0 | u | | | | | | | | | | | CR | RI | PK | SB | D |

Table RTR\_ERR\_PARITY\_MASK register.

### RTR\_ERR\_PARITY

There is one register for each router port capturing parity error events occurring on the port. Parity errors are monitored on router physical link and also on data read from VC buffers of the router. Error status bits are sticky. First detected error while the status bit is in cleared state sets the bit. The bit needs to be explicitly cleared using zero write, before another error can be logged for that status bit. Following fields of information transported over the NoC are monitored for error at router ports. [FATAL] all bits in this register are classified as fatal for interrupt purpose.

1. Data Parity: Parity is checked over multiple segments of data in each flit. Parity error in any segment will be recorded in the data parity status bit. Note that parity is checked on data only if parity mode error check is enabled on the router's layer. In ECC mode, data parity is not monitored on each router.
2. User sideband parity: Similar to data field above.
3. Packet control parity: Parity over start of packet, end of packet, byte valid and data valid fields of a flit.
4. Routing information parity: Parity over routing information carried in every flit.
5. Credit parity: Parity monitored over credits returned downstream port.

Attribute: WZC

Security: Non-secure

Bit field description:

* **RI\_3**[31] - 1'b1: Parity Error in VC 3 Buffer Routing Information
* **PK\_3**[30] - 1'b1: Parity Error in VC 3 Buffer Packet Delineation Controls
* **SB\_3**[29] - 1'b1: Parity Error in VC 3 Buffer User Sideband
* **D\_3**[28] - 1'b1: Parity Error in VC 3 Buffer Data
* **RI\_2**[27] - 1'b1: Parity Error in VC 2 Buffer Routing Information
* **PK\_2**[26] - 1'b1: Parity Error in VC 2 Buffer Packet Delineation Controls
* **SB\_2**[25] - 1'b1: Parity Error in VC 2 Buffer User Sideband
* **D\_2**[24] - 1'b1: Parity Error in VC 2 Buffer Data
* **RI\_1**[23] - 1'b1: Parity Error in VC 1 Buffer Routing Information
* **PK\_1**[22] - 1'b1: Parity Error in VC 1 Buffer Packet Delineation Controls
* **SB\_1**[21] - 1'b1: Parity Error in VC 1 Buffer User Sideband
* **D\_1**[20] - 1'b1: Parity Error in VC 1 Buffer Data
* **RI\_0**[19] - 1'b1: Parity Error in VC 0 Buffer Routing Information
* **PK\_0**[18] - 1'b1: Parity Error in VC 0 Buffer Packet Delineation Controls
* **SB\_0**[17] - 1'b1: Parity Error in VC 0 Buffer User Sideband
* **D\_0**[16] - 1'b1: Parity Error in VC 0 Buffer Data
* **CR**[4] - 1'b1: Parity Error in Link Credit From Downstream Router
* **RI**[3] - 1'b1: Parity Error in Link Routing Information
* **PK**[2] - 1'b1: Parity Error in Link Packet Delineation Controls
* **SB**[1] - 1'b1: Parity Error in Link User Sideband
* **D**[0] - 1'b1: Parity Error in Link Data

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RI\_3 | PK\_3 | SB\_3 | D\_3 | RI\_2 | PK\_2 | SB\_2 | D\_2 | RI\_1 | PK\_1 | SB\_1 | D\_1 | RI\_0 | PK\_0 | SB\_0 | D\_0 | u | | | | | | | | | | | CR | RI | PK | SB | D |

Table RTR\_ERR\_PARITY register.

### RTR\_EVENT\_INTERRUPT\_MASK

This register is used to select whether the interrupt events in the Router Event Interrupt Status register should send an interrupt when asserted. If the corresponding bit is set to 1, an interrupt will not be sent. This register can be read and written to.

Attribute: RW

Security: Non-secure

Bit field description:

* **MK**[16] -   
  1'b1: Mask KLU error interrupt
* **MJ**[15] -   
  1'b1: Mask JLU error interrupt
* **MI**[14] -   
  1'b1: Mask ILU error interrupt
* **MH**[13] -   
  1'b1: Mask HLU error interrupt
* **MS**[12] -   
  1'b1: Mask SLU error interrupt
* **MW**[11] -   
  1'b1: Mask WLU error interrupt
* **ME**[10] -   
  1'b1: Mask ELU error interrupt
* **MN**[9] -   
  1'b1: Mask NLU error interrupt
* **PGM**[8] -   
  1'b1: Mask PGE error interrupt
* **OVFOM**[2] -   
  1'b1: Masks or disables an interrupt from being generated by the output event count overflow status bit (RE)  
  1'b0: Enables an interrupt to be generated when event counter status bit is set
* **CSR\_PARERRM**[1] -   
  1'b1: Mask CSR parity error interrupt
* **OVFIM**[0] -   
  1'b1: Masks or disables an interrupt from being generated by the input event count overflow status bit (RE)  
  1'b0: Enables an interrupt to be generated when event counter status bit is set

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | MK | MJ | MI | MH | MS | MW | ME | MN | PGM | u | | | | | OVFOM | CSR\_PARERRM | OVFIM |

Table RTR\_EVENT\_INTERRUPT\_MASK register.

### RTR\_EVENT\_STATUS

This register tracks the interrupt or error events that can occur in the router. The only interrupt event is the event counter overflow. This register is readable, and can be cleared by performing a write with the write data bits set to 0 for the bits that should be cleared.

Attribute: WZC

Security: Non-secure

Bit field description:

* **KLU**[16] -   
  1'b1: Traffic destined for K link which is unavailable
* **JLU**[15] -   
  1'b1: Traffic destined for J link which is unavailable
* **ILU**[14] -   
  1'b1: Traffic destined for I link which is unavailable
* **HLU**[13] -   
  1'b1: Traffic destined for H link which is unavailable
* **SLU**[12] -   
  1'b1: Traffic destined for South link which is unavailable
* **WLU**[11] -   
  1'b1: Traffic destined for West link which is unavailable
* **ELU**[10] -   
  1'b1: Traffic destined for East link which is unavailable
* **NLU**[9] -   
  1'b1: Traffic destined for North link which is unavailable
* **PGE**[8] -   
  1'b1: Power gating error, traffic received after router commited to power down
* **OVFO**[2] -   
  1'b1: In this status bit indicates that the router output event counter has overflowed (32'hFFFFFFFF -> 32'dh0), this is a sticky status bit  
  1'b0: To clear
* **CSR\_PARERR**[1] -   
  1'b1: Parity error in config/status registers
* **OVFI**[0] -   
  1'b1: In this status bit indicates that the router input event counter has overflowed (32'hFFFFFFFF -> 32'dh0), this is a sticky status bit  
  1'b0: To clear

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | KLU | JLU | ILU | HLU | SLU | WLU | ELU | NLU | PGE | u | | | | | OVFO | CSR\_PARERR | OVFI |

Table RTR\_EVENT\_STATUS register.

### RTR\_ID

This register holds layer and position information for the router. It is a read-only register. It can be used for debugging software access to the NoC elements by confirming that a read has successfully targeted the correct NoC element.

Attribute: R

Security: Non-secure

Bit field description:

* **ONE**[24] - One
* **ZERO**[23:21] - Zeroes
* **POS**[20:5] - 16-bit position ID of this router in the NoC
* **LAYER**[4:0] - 5-bit identifier of the NoC layer on which this router is located

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | ONE | ZERO | | | POS | | | | | | | | | | | | | | | | LAYER | | | | |

Table RTR\_ID register.

### RTR\_IVC\_EVENT\_CONTROL

This register is used to select which hardware events will increment the event counter.

Attribute: RW

Security: Non-secure

Bit field description:

* **EVT**[9:8] -   
  11: Generates count event when VC has valid data, but is stalled  
  10: Generates count event on every flit received for the selected input port and selected input VCs, this can be used to count total flits received on a router input port  
  01: Generates count event on every EOP received for the selected input port and selected input VCs, this can be used to count packets received on a router input port  
  00: Disable
* **INP**[6:4] - Input port on which the event is captured
* **IVC**[1:0] -   
  11: Input VC 3  
  10: Input VC 2  
  01: Input VC 1  
  00: Input VC 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | EVT | | u | INP | | | u | | IVC | |

Table RTR\_IVC\_EVENT\_CONTROL register.

### RTR\_IVC\_EVENT\_COUNTER

This register holds the event counter. The value can be read to determine the current count value. The value can be written to initialize the counter. When events trigger a count, the counter will increment. When the counter increments at its highest value, it will roll over to zero and the overflow will mark the Router Event Interrupt Status register, which could trigger an interrupt.

Attribute: RW

Security: Non-secure

Bit field description:

* **EVENT\_CNTR**[31:0] - 32'bit event incrementing counter. Rollover from 32'hFFFFF -> 32'd0 sets the rollover status bit RE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EVENT\_CNTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table RTR\_IVC\_EVENT\_COUNTER register.

### RTR\_IVC\_STATUS

This register indicates the current status of a single input port of a router. Each register tracks the status of up to 4 virtual channels for the input port. There are 8 rtr\_ivc\_status per router, one for each router's input port.

Attribute: R

Security: Non-secure

Bit field description:

* **V\_3**[31] -   
  1'b1: Head flit valid (buffer ready) in VC 3
* **F\_3**[30] -   
  1'b1: Buffer full in VC 3
* **B\_3**[29] -   
  1'b1: Indicates that the head flit of the VC 3 is of the 'QoS Barrier' type  
  1'b0: Indicates that the head flit of the VC 3 is of the 'QoS Normal' type
* **S\_3**[28] -   
  1'b1: Indicates that the head flit is a start of packet. This also indicates that this input VC 3 has not yet acquired its corresponding output VC  
  1'b0: Indicates that the head flit is not a start of packet. Also indicates that this input VC 3 has already acquired the VC on the output port
* **UP\_3**[27] -   
  1'b1: Indicates that the flit accumulator on this VC 3 for upsizing to an output port is currently holding a flit  
  1'b0: Indicates that either the upsizing accumulator is empty or there is no upsizing from the VC 3
* **OUTP\_3**[26:24] - Value indicates the router output port to which the packet at the head of the VC 3 is destined to: 3'd0:N, 3'd1:E, 3'd2:W, 3'd3:S, 3'd4:H, 3'd5:I, 3'd6:J, 3'd7:K
* **V\_2**[23] -   
  1'b1: Head flit valid (buffer ready) in VC 2
* **F\_2**[22] -   
  1'b1: Buffer full in VC 2
* **B\_2**[21] -   
  1'b1: Indicates that the head flit of the VC 2 is of the 'QoS Barrier' type  
  1'b0: Indicates that the head flit of the VC 2 is of the 'QoS Normal' type
* **S\_2**[20] -   
  1'b1: Indicates that the head flit is a start of packet. This also indicates that this input VC 2 has not yet acquired its corresponding output VC  
  1'b0: Indicates that the head flit is not a start of packet. Also indicates that this input VC 2 has already acquired the VC on the output port
* **UP\_2**[19] -   
  1'b1: Indicates that the flit accumulator on this VC 2 for upsizing to an output port is currently holding a flit  
  1'b0: Indicates that either the upsizing accumulator is empty or there is no upsizing from the VC 2
* **OUTP\_2**[18:16] - Value indicates the router output port to which the packet at the head of the VC 2 is destined to: 3'd0:N, 3'd1:E, 3'd2:W, 3'd3:S, 3'd4:H, 3'd5:I, 3'd6:J, 3'd7:K
* **V\_1**[15] -   
  1'b1: Head flit valid (buffer ready) in VC 1
* **F\_1**[14] -   
  1'b1: Buffer full in VC 1
* **B\_1**[13] -   
  1'b1: Indicates that the head flit of the VC 1 is of the 'QoS Barrier' type  
  1'b0: Indicates that the head flit of the VC 1 is of the 'QoS Normal' type
* **S\_1**[12] -   
  1'b1: Indicates that the head flit is a start of packet. This also indicates that this input VC 1 has not yet acquired its corresponding output VC  
  1'b0: Indicates that the head flit is not a start of packet. Also indicates that this input VC 1 has already acquired the VC on the output port
* **UP\_1**[11] -   
  1'b1: Indicates that the flit accumulator on this VC 1 for upsizing to an output port is currently holding a flit  
  1'b0: Indicates that either the upsizing accumulator is empty or there is no upsizing from the VC 1
* **OUTP\_1**[10:8] - Value indicates the router output port to which the packet at the head of the VC 1 is destined to: 3'd0:N, 3'd1:E, 3'd2:W, 3'd3:S, 3'd4:H, 3'd5:I, 3'd6:J, 3'd7:K
* **V\_0**[7] -   
  1'b1: Head flit valid (buffer ready) in VC 0
* **F\_0**[6] -   
  1'b1: Buffer full in VC 0
* **B\_0**[5] -   
  1'b1: Indicates that the head flit of the VC 0 is of the 'QoS Barrier' type  
  1'b0: Indicates that the head flit of the VC 0 is of the 'QoS Normal' type
* **S\_0**[4] -   
  1'b1: Indicates that the head flit is a start of packet. This also indicates that this input VC 0 has not yet acquired its corresponding output VC  
  1'b0: Indicates that the head flit is not a start of packet. Also indicates that this input VC 0 has already acquired the VC on the output port
* **UP\_0**[3] -   
  1'b1: Indicates that the flit accumulator on this VC 0 for upsizing to an output port is currently holding a flit  
  1'b0: Indicates that either the upsizing accumulator is empty or there is no upsizing from the VC 0
* **OUTP\_0**[2:0] - Value indicates the router output port to which the packet at the head of the VC 0 is destined to: 3'd0:N, 3'd1:E, 3'd2:W, 3'd3:S, 3'd4:H, 3'd5:I, 3'd6:J, 3'd7:K

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| V\_3 | F\_3 | B\_3 | S\_3 | UP\_3 | OUTP\_3 | | | V\_2 | F\_2 | B\_2 | S\_2 | UP\_2 | OUTP\_2 | | | V\_1 | F\_1 | B\_1 | S\_1 | UP\_1 | OUTP\_1 | | | V\_0 | F\_0 | B\_0 | S\_0 | UP\_0 | OUTP\_0 | | |

Table RTR\_IVC\_STATUS register.

### RTR\_OVC\_EVENT\_CONTROL

This register is used to select which hardware events will increment the output event counter.

Attribute: RW

Security: Non-secure

Bit field description:

* **EVT**[10:8] -   
  100: Port stalled. Input flits are available for the port, but no output VC has credit  
  011: Generates count event when flits are available to be sent to output VC, but the VC has no credit  
  010: Generates count event on every flit sent on the selected output port and selected outpt VCs, this can be used to count total flits sent on a router output port  
  001: Generates count event on every EOP sent on the selected output port and selected output VCs, this can be used to count packets sent on a router output port  
  000: Disable
* **OP**[6:4] - Output port on which the event is captured
* **OVC**[3:0] - Bit map to select output VCs to monitor events on

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | EVT | | | u | OP | | | OVC | | | |

Table RTR\_OVC\_EVENT\_CONTROL register.

### RTR\_OVC\_EVENT\_COUNTER

This register holds the output event counter. The value can be read to determine the current count value. The value can be written to initialize the counter. When events trigger a count, the counter will increment. When the counter increments at its highest value, it will roll over to zero and the overflow will mark the Router output Event Interrupt Status register, which could trigger an interrupt.

Attribute: RW

Security: Non-secure

Bit field description:

* **EVENT\_CNTR**[31:0] - 32'bit event incrementing counter. Rollover from 32'hFFFFF -> 32'd0 sets the rollover status bit RE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EVENT\_CNTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table RTR\_OVC\_EVENT\_COUNTER register.

### RTR\_OVC\_STATUS

This register indicates the current status of one of the output ports of a router. Each register tracks the status of up to 4 virtual channels for the output port. There are 8 rtr\_ovc\_status per router, one for each router's output port (only 5 are active registers, while the other 3 are reserved).

Attribute: R

Security: Non-secure

Bit field description:

* **RSV\_3**[27] - Reserved
* **VB\_3**[26] -   
  1'b1: Indicates that this output VC 3 is currently locked to the corresponding VC on one of the input ports.  
  1'b0: Indicates that this output VC 3 is free and can be acquired by the corresponding VC on one of the input port for the transmission of a packet.
* **CE\_3**[25] -   
  1'b1: Indicates that this output VC 3 has no credit for transmission of flits to the downstream link.  
  1'b0: Indicates that credits are available for transmission to downstream link.
* **CF\_3**[24] -   
  1'b1: Indicates that the credit level with this VC 3 is at the maximum provisioned value.
* **RSV\_2**[19] - Reserved
* **VB\_2**[18] -   
  1'b1: Indicates that this output VC 2 is currently locked to the corresponding VC on one of the input ports.  
  1'b0: Indicates that this output VC 2 is free and can be acquired by the corresponding VC on one of the input port for the transmission of a packet.
* **CE\_2**[17] -   
  1'b1: Indicates that this output VC 2 has no credit for transmission of flits to the downstream link.  
  1'b0: Indicates that credits are available for transmission to downstream link.
* **CF\_2**[16] -   
  1'b1: Indicates that the credit level with this VC 2 is at the maximum provisioned value.
* **RSV\_1**[11] - Reserved
* **VB\_1**[10] -   
  1'b1: Indicates that this output VC 1 is currently locked to the corresponding VC on one of the input ports.  
  1'b0: Indicates that this output VC 1 is free and can be acquired by the corresponding VC on one of the input port for the transmission of a packet.
* **CE\_1**[9] -   
  1'b1: Indicates that this output VC 1 has no credit for transmission of flits to the downstream link.  
  1'b0: Indicates that credits are available for transmission to downstream link.
* **CF\_1**[8] -   
  1'b1: Indicates that the credit level with this VC 1 is at the maximum provisioned value.
* **RSV\_0**[3] - Reserved
* **VB\_0**[2] -   
  1'b1: Indicates that this output VC 0 is currently locked to the corresponding VC on one of the input ports.  
  1'b0: Indicates that this output VC 0 is free and can be acquired by the corresponding VC on one of the input port for the transmission of a packet.
* **CE\_0**[1] -   
  1'b1: Indicates that this output VC 0 has no credit for transmission of flits to the downstream link.  
  1'b0: Indicates that credits are available for transmission to downstream link.
* **CF\_0**[0] -   
  1'b1: Indicates that the credit level with this VC 0 is at the maximum provisioned value.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | RSV\_3 | VB\_3 | CE\_3 | CF\_3 | u | | | | RSV\_2 | VB\_2 | CE\_2 | CF\_2 | u | | | | RSV\_1 | VB\_1 | CE\_1 | CF\_1 | u | | | | RSV\_0 | VB\_0 | CE\_0 | CF\_0 |

Table RTR\_OVC\_STATUS register.

## NOC Interface registers

### NOC\_RX\_ERR\_PARITY\_0

Receive bridge parity error status register monitoring parity errors on enabled layers from 0 to 7 (noc\_rx\_err\_parity\_0), and from 8 to 15 (noc\_rx\_err\_parity\_1). Parity/ECC error are monitored and captured for physical link to the bridge on each NoC layer. Following fields are monitored.

1. Data ECC/Parity: Parity/ECC is checked over multiple segments of data in each flit. An error in any segment will be recorded in the data ECC/parity error status bit. In ECC mode, single bit errors are corrected and the event is recorded.
2. User sideband ECC/parity: Similar to data field above.
3. Packet control parity: Parity over start of packet, end of packet, byte valid and data valid fields of a flit.

This register makes use of the logical layer mapping (and not the physical layer mapping). For the physical to logical table, please refer to the Physical to Logical Layer Mapping section in the help.

Correctable errors will raise interrupt\_nfatal if fatal/nonfatal interrupt mode is configured. All other error types are considered fatal.

Attribute: WZC

Security: Non-secure

Bit field description:

* **PK0**[4] - Parity error in packet delineation controls in layer 0
* **SBC0**[3] - Correctable single bit user sideband error (only ECC) in layer 0
* **SB0**[2] - Uncorrectable User sideband ECC/parity error in layer 0
* **DC0**[1] - Correctable single bit data error (only ECC) in layer 0
* **D0**[0] - Uncorrectable Data ECC/parity error in layer 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | PK0 | SBC0 | SB0 | DC0 | D0 |

Table NOC\_RX\_ERR\_PARITY\_0 register.

### NOC\_RX\_ERR\_PARITY\_1

Receive bridge parity error status register monitoring parity errors on enabled layers from 0 to 7 (noc\_rx\_err\_parity\_0), and from 8 to 15 (noc\_rx\_err\_parity\_1). Parity/ECC error are monitored and captured for physical link to the bridge on each NoC layer. Following fields are monitored.

1. Data ECC/Parity: Parity/ECC is checked over multiple segments of data in each flit. An error in any segment will be recorded in the data ECC/parity error status bit. In ECC mode, single bit errors are corrected and the event is recorded.
2. User sideband ECC/parity: Similar to data field above.
3. Packet control parity: Parity over start of packet, end of packet, byte valid and data valid fields of a flit.

This register makes use of the logical layer mapping (and not the physical layer mapping). For the physical to logical table, please refer to the Physical to Logical Layer Mapping section in the help.

Correctable errors will raise interrupt\_nfatal if fatal/nonfatal interrupt mode is configured. All other error types are considered fatal.

Attribute: WZC

Security: Non-secure

Bit field description:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | | | | | |

Table NOC\_RX\_ERR\_PARITY\_1 register.

### NOC\_RX\_ERR\_PARITY\_MASK\_0

Mask registers for receive bridge parity error interrupts from register noc\_rx\_err\_parity\_0 and noc\_rx\_err\_parity\_1. One mask register bit for each parity status bit in noc\_rx\_err\_parity. When mask bit is set, corresponding parity error does not cause an interrupt. Default state is reset for all mask bits, allowing interrupt on any parity error event.

This register makes use of the logical layer mapping (and not the physical layer mapping). For the physical to logical table, please refer to the Physical to Logical Layer Mapping section in the help.

Attribute: RW

Security: Non-secure

Bit field description:

* **PK0**[4] - Mask Parity error in packet delineation controls in layer 0
* **SBC0**[3] - Mask Correctable single bit user sideband error (only ECC) in layer 0
* **SB0**[2] - Mask User sideband ECC/parity error in layer 0
* **DC0**[1] - Mask Correctable single bit data error (only ECC) in layer 0
* **D0**[0] - Mask Data ECC/parity error in layer 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | PK0 | SBC0 | SB0 | DC0 | D0 |

Table NOC\_RX\_ERR\_PARITY\_MASK\_0 register.

### NOC\_RX\_ERR\_PARITY\_MASK\_1

Mask registers for receive bridge parity error interrupts from register noc\_rx\_err\_parity\_0 and noc\_rx\_err\_parity\_1. One mask register bit for each parity status bit in noc\_rx\_err\_parity. When mask bit is set, corresponding parity error does not cause an interrupt. Default state is reset for all mask bits, allowing interrupt on any parity error event.

This register makes use of the logical layer mapping (and not the physical layer mapping). For the physical to logical table, please refer to the Physical to Logical Layer Mapping section in the help.

Attribute: RW

Security: Non-secure

Bit field description:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | u | | | | | | | |

Table NOC\_RX\_ERR\_PARITY\_MASK\_1 register.

### NOC\_RX\_EVENT\_STATUS

This register tracks the interrupt events in the receive portion of the streaming bridge. It resets to 0, but as these conditions occur, the corresponding bits are set to 1. This register can be read and can also be cleared by sending a write with bits set to 0 for the bits that should be cleared.

There are four events that can signal an interrupt. If the host sends more credits than the streaming bridge can take, it will signal an interrupt to indicate a protocol violation has occurred. Each interface has its own status bit. These interrupts cannot be masked.

Attribute: WZC

Security: Non-secure

Bit field description:

* **EVC1\_OFLW**[6] - Event counter1 overflow. This event can be masked so that no interrupt is sent on an overflow condition.
* **PARITY\_ERR**[5] - Register parity error interrupt
* **EVC\_OFLW**[4] - Event counter overflow. This event can be masked so that no interrupt is sent on an overflow condition.
* **CRC\_OFLW\_D**[3] - Credit counter overflow for interface D
* **CRC\_OFLW\_C**[2] - Credit counter overflow for interface C
* **CRC\_OFLW\_B**[1] - Credit counter overflow for interface B
* **CRC\_OFLW\_A**[0] - Credit counter overflow for interface A

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | EVC1\_OFLW | PARITY\_ERR | EVC\_OFLW | CRC\_OFLW\_D | CRC\_OFLW\_C | CRC\_OFLW\_B | CRC\_OFLW\_A |

Table NOC\_RX\_EVENT\_STATUS register.

### NOC\_RX\_FIFO\_STATUS

These registers track the status of the bridge's receive FIFOs from the NoC. Since there are up to 16 layers of the NoC, there are up to 16 registers, one per active layer. Each register tracks the status of the active virtual channels for the layer (up to 4 active VCs within a layer).

Attribute: R

Security: Non-secure

Bit field description:

* **F\_3**[29] - Buffer full for VC 3 Layer 0
* **B\_3**[28] - Head flit barrier state for VC 3 Layer 0
* **S\_3**[27] - Head flit sop for VC 3 Layer 0
* **V\_3**[26] - Head flit (buffer ready) for VC 3 Layer 0
* **OUTI\_3**[25:24] - Head flit output interface for VC 3 Layer 0
* **F\_2**[21] - Buffer full for VC 2 Layer 0
* **B\_2**[20] - Head flit barrier state for VC 2 Layer 0
* **S\_2**[19] - Head flit sop for VC 2 Layer 0
* **V\_2**[18] - Head flit (buffer ready) for VC 2 Layer 0
* **OUTI\_2**[17:16] - Head flit output interface for VC 2 Layer 0
* **F\_1**[13] - Buffer full for VC 1 Layer 0
* **B\_1**[12] - Head flit barrier state for VC 1 Layer 0
* **S\_1**[11] - Head flit sop for VC 1 Layer 0
* **V\_1**[10] - Head flit (buffer ready) for VC 1 Layer 0
* **OUTI\_1**[9:8] - Head flit output interface for VC 1 Layer 0
* **F\_0**[5] - Buffer full for VC 0 Layer 0
* **B\_0**[4] - Head flit barrier state for VC 0 Layer 0
* **S\_0**[3] - Head flit sop for VC 0 Layer 0
* **V\_0**[2] - Head flit (buffer ready) for VC 0 Layer 0
* **OUTI\_0**[1:0] - Head flit output interface for VC 0 Layer 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | F\_3 | B\_3 | S\_3 | V\_3 | OUTI\_3 | | u | | F\_2 | B\_2 | S\_2 | V\_2 | OUTI\_2 | | u | | F\_1 | B\_1 | S\_1 | V\_1 | OUTI\_1 | | u | | F\_0 | B\_0 | S\_0 | V\_0 | OUTI\_0 | |

Table NOC\_RX\_FIFO\_STATUS register.

### NOC\_RX\_INTERRUPT\_MASK

This register is used to decide which of the error/interrupt events specified in the noc\_rx\_event\_status register should trigger an interrupt.

Attribute: RW

Security: Non-secure

Bit field description:

* **EVC1\_OFLW\_MASK**[6] -   
  1'b1: When is set to 1, the corresponding interrupt event will not send an interrupt to the system.  
  1'b0: The corresponding interrupt event will send an interrupt to the system.
* **PARITY\_ERR\_MASK**[5] - Interrupt mask for register parity error.
* **EVC\_OFLW\_MASK**[4] -   
  1'b1: When is set to 1, the corresponding interrupt event will not send an interrupt to the system.  
  1'b0: The corresponding interrupt event will send an interrupt to the system.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | EVC1\_OFLW\_MASK | PARITY\_ERR\_MASK | EVC\_OFLW\_MASK | u | | | |

Table NOC\_RX\_INTERRUPT\_MASK register.

### NOC\_RX\_UPSIZER\_STATUS

This register tracks the status of the bridge receiver upsizer/downsize structure. It can be used with the other status registers to check for packets that are still occupying the bridge. Each of the host's receiving interfaces, up to 4, can have upsizing/downsizing logic, and this register tracks the status of all 4 interfaces.

Attribute: R

Security: Non-secure

Bit field description:

* **V\_D**[3] - Interface D upsizer/downsizer valid
* **V\_C**[2] - Interface C upsizer/downsizer valid
* **V\_B**[1] - Interface B upsizer/downsizer valid
* **V\_A**[0] - Interface A upsizer/downsizer valid

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | V\_D | V\_C | V\_B | V\_A |

Table NOC\_RX\_UPSIZER\_STATUS register.

### NOC\_TX\_ERR\_PARITY

Transmit bridge parity error status register. One register bits per layer, to monitor error in credit return signals from the downstream port. Error status bits are sticky. First detected error while the status bit is in cleared state sets the bit. The bit needs to be explicitly cleared using zero write, before another error can be logged for that status bit.

Attribute: WZC

Security: Non-secure

Bit field description:

* **L15**[15] -   
  1'b1: Credit parity error on layer 15
* **L14**[14] -   
  1'b1: Credit parity error on layer 14
* **L13**[13] -   
  1'b1: Credit parity error on layer 13
* **L12**[12] -   
  1'b1: Credit parity error on layer 12
* **L11**[11] -   
  1'b1: Credit parity error on layer 11
* **L10**[10] -   
  1'b1: Credit parity error on layer 10
* **L9**[9] -   
  1'b1: Credit parity error on layer 9
* **L8**[8] -   
  1'b1: Credit parity error on layer 8
* **L7**[7] -   
  1'b1: Credit parity error on layer 7
* **L6**[6] -   
  1'b1: Credit parity error on layer 6
* **L5**[5] -   
  1'b1: Credit parity error on layer 5
* **L4**[4] -   
  1'b1: Credit parity error on layer 4
* **L3**[3] -   
  1'b1: Credit parity error on layer 3
* **L2**[2] -   
  1'b1: Credit parity error on layer 2
* **L1**[1] -   
  1'b1: Credit parity error on layer 1
* **L0**[0] -   
  1'b1: Credit parity error on layer 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | L15 | L14 | L13 | L12 | L11 | L10 | L9 | L8 | L7 | L6 | L5 | L4 | L3 | L2 | L1 | L0 |

Table NOC\_TX\_ERR\_PARITY register.

### NOC\_TX\_ERR\_PARITY\_MASK

Mask register for transmit bridge parity error interrupts. One mask register bit for each parity status bit in noc\_tx\_err\_parity. When mask bit is set, corresponding parity error does not cause an interrupt. Default state is reset for all mask bits, allowing interrupt on any parity error event.

Attribute: RW

Security: Non-secure

Bit field description:

* **L15**[15] -   
  1'b1: Interrupt Mask Credit parity error on layer 15
* **L14**[14] -   
  1'b1: Interrupt Mask Credit parity error on layer 14
* **L13**[13] -   
  1'b1: Interrupt Mask Credit parity error on layer 13
* **L12**[12] -   
  1'b1: Interrupt Mask Credit parity error on layer 12
* **L11**[11] -   
  1'b1: Interrupt Mask Credit parity error on layer 11
* **L10**[10] -   
  1'b1: Interrupt Mask Credit parity error on layer 10
* **L9**[9] -   
  1'b1: Interrupt Mask Credit parity error on layer 9
* **L8**[8] -   
  1'b1: Interrupt Mask Credit parity error on layer 8
* **L7**[7] -   
  1'b1: Interrupt Mask Credit parity error on layer 7
* **L6**[6] -   
  1'b1: Interrupt Mask Credit parity error on layer 6
* **L5**[5] -   
  1'b1: Interrupt Mask Credit parity error on layer 5
* **L4**[4] -   
  1'b1: Interrupt Mask Credit parity error on layer 4
* **L3**[3] -   
  1'b1: Interrupt Mask Credit parity error on layer 3
* **L2**[2] -   
  1'b1: Interrupt Mask Credit parity error on layer 2
* **L1**[1] -   
  1'b1: Interrupt Mask Credit parity error on layer 1
* **L0**[0] -   
  1'b1: Interrupt Mask Credit parity error on layer 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | L15 | L14 | L13 | L12 | L11 | L10 | L9 | L8 | L7 | L6 | L5 | L4 | L3 | L2 | L1 | L0 |

Table NOC\_TX\_ERR\_PARITY\_MASK register.

### NOC\_TX\_EVENT\_STATUS

This register tracks error or interrupt conditions. It resets to 0, but as these conditions occur, the corresponding bits are set to 1. This register can be read and can also be cleared by sending a write with bits set to 0 for the bits that should be cleared. This register works in combination with the Transmit Interrupt Mask register to determine when an interrupt is transmitted.

Attribute: WZC

Security: Non-secure

Bit field description:

* **PARITY\_ERR**[8] - Register parity error interrupt.
* **FIFO\_OVERFLOW\_D**[7] - Host interface FIFO D overflow. Indicates that one of the per-interface FIFOs at the transmitting bridge to NoC has overflowed. This event will always trigger an interrupt and cannot be masked
* **FIFO\_OVERFLOW\_C**[6] - Host interface FIFO C overflow. Indicates that one of the per-interface FIFOs at the transmitting bridge to NoC has overflowed. This event will always trigger an interrupt and cannot be masked
* **FIFO\_OVERFLOW\_B**[5] - Host interface FIFO B overflow. Indicates that one of the per-interface FIFOs at the transmitting bridge to NoC has overflowed. This event will always trigger an interrupt and cannot be masked
* **FIFO\_OVERFLOW\_A**[4] - Host interface FIFO A overflow. Indicates that one of the per-interface FIFOs at the transmitting bridge to NoC has overflowed. This event will always trigger an interrupt and cannot be masked
* **EVENT\_CNTR\_OVERFLOW**[3] -   
  1'b1: Sets if the event counter overflows, this event can be masked so that no interrupt is sent on an overflow condition
* **TRANS\_ILLEGAL\_DEST\_QOS**[2] -   
  1'b1: Sets if a transaction is received from bridge for which there is no entry present in the vcmap, i.e. the destination and/or QoS is not supported, this is a decode error. This event can be masked to not send an interrupt, but the packet will be dropped in the bridge.
* **SOP\_AFTER\_SOP**[1] -   
  1'b1: Sets if a SOP is received after SOP, this event will always trigger an interrupt and cannot be masked.
* **TRANS\_WITHOUT\_SOP**[0] -   
  1'b1: Sets if a transaction is initiated w/o SOP, this event will always trigger an interrupt and cannot be masked.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | PARITY\_ERR | FIFO\_OVERFLOW\_D | FIFO\_OVERFLOW\_C | FIFO\_OVERFLOW\_B | FIFO\_OVERFLOW\_A | EVENT\_CNTR\_OVERFLOW | TRANS\_ILLEGAL\_DEST\_QOS | SOP\_AFTER\_SOP | TRANS\_WITHOUT\_SOP |

Table NOC\_TX\_EVENT\_STATUS register.

### NOC\_TX\_INTERRUPT\_MASK

This register is used to decide which of the error/interrupt events specified in the Transmit Interrupt Status register should trigger an interrupt. Since only the events in bit 2 and 3 can be masked, only bit 2 and 3 are used in this register. When one of the bits in this register is set to 1, the corresponding interrupt event will not send an interrupt to the system.

Attribute: RW

Security: Non-secure

Bit field description:

* **PARITY\_ERR\_MASK**[8] - Interrupt mask for register parity error
* **EVENT\_CNTR\_OVERFLOW**[3] - Interrupt mask for event counter overflow
* **TRANS\_ILLEGAL\_DEST\_QOS**[2] - Interrupt mask for illegal destination QoS

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | PARITY\_ERR\_MASK | u | | | | EVENT\_CNTR\_OVERFLOW | TRANS\_ILLEGAL\_DEST\_QOS | u | |

Table NOC\_TX\_INTERRUPT\_MASK register.

### NOC\_TX\_QOS\_WEIGHT

This register describes the weight value of each QoS supported at the bridge. Each byte of this register must be greater than or equal to 3. Each transmitting bridge supports up to 16 QoS profiles. Each QoS is composed of pri and weight, however only the weight is programmable, therefore is part of the registers.

QoS data is composed of four registers, noc\_tx\_qos\_weight\_0, noc\_tx\_qos\_weight\_1, noc\_tx\_qos\_weight\_2 and noc\_tx\_qos\_weight\_3, each of which contains the weight of four profiles. Depending upon how many QoS profiles are enabled, the appropriate bits in the following registers are available.

Attribute: RW

Security: Non-secure

Bit field description:

* **WT\_QOS\_0**[7:0] - Weight of QoS profile 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | u | | | | | | | | u | | | | | | | | WT\_QOS\_0 | | | | | | | |

Table NOC\_TX\_QOS\_WEIGHT register.

### NOC\_TX\_UPSIZER\_STATUS\_0

These two registers (noc\_tx\_upsizer\_status\_0 and noc\_tx\_upsizer\_status\_1) track the status of the bridge transmitter upsizer/downsize structure. They can be used with the other status registers to check for packets that are still occupying the bridge. Each NoC layer, up to 16, can have upsizing/downsizing logic, and these 2 registers track the status of all 16 layers (noc\_tx\_upsizer\_status\_0 from 0 to 7 and noc\_tx\_upsizer\_status\_1 from 8 to 15).

Attribute: R

Security: Non-secure

Bit field description:

* **L7\_D**[31] - Interface upsizer status for interface D, Layer 7
* **L7\_C**[30] - Interface upsizer status for interface C, Layer 7
* **L7\_B**[29] - Interface upsizer status for interface B, Layer 7
* **L7\_A**[28] - Interface upsizer status for interface A, Layer 7
* **L6\_D**[27] - Interface upsizer status for interface D, Layer 6
* **L6\_C**[26] - Interface upsizer status for interface C, Layer 6
* **L6\_B**[25] - Interface upsizer status for interface B, Layer 6
* **L6\_A**[24] - Interface upsizer status for interface A, Layer 6
* **L5\_D**[23] - Interface upsizer status for interface D, Layer 5
* **L5\_C**[22] - Interface upsizer status for interface C, Layer 5
* **L5\_B**[21] - Interface upsizer status for interface B, Layer 5
* **L5\_A**[20] - Interface upsizer status for interface A, Layer 5
* **L4\_D**[19] - Interface upsizer status for interface D, Layer 4
* **L4\_C**[18] - Interface upsizer status for interface C, Layer 4
* **L4\_B**[17] - Interface upsizer status for interface B, Layer 4
* **L4\_A**[16] - Interface upsizer status for interface A, Layer 4
* **L3\_D**[15] - Interface upsizer status for interface D, Layer 3
* **L3\_C**[14] - Interface upsizer status for interface C, Layer 3
* **L3\_B**[13] - Interface upsizer status for interface B, Layer 3
* **L3\_A**[12] - Interface upsizer status for interface A, Layer 3
* **L2\_D**[11] - Interface upsizer status for interface D, Layer 2
* **L2\_C**[10] - Interface upsizer status for interface C, Layer 2
* **L2\_B**[9] - Interface upsizer status for interface B, Layer 2
* **L2\_A**[8] - Interface upsizer status for interface A, Layer 2
* **L1\_D**[7] - Interface upsizer status for interface D, Layer 1
* **L1\_C**[6] - Interface upsizer status for interface C, Layer 1
* **L1\_B**[5] - Interface upsizer status for interface B, Layer 1
* **L1\_A**[4] - Interface upsizer status for interface A, Layer 1
* **L0\_D**[3] - Interface upsizer status for interface D, Layer 0
* **L0\_C**[2] - Interface upsizer status for interface C, Layer 0
* **L0\_B**[1] - Interface upsizer status for interface B, Layer 0
* **L0\_A**[0] - Interface upsizer status for interface A, Layer 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| L7\_D | L7\_C | L7\_B | L7\_A | L6\_D | L6\_C | L6\_B | L6\_A | L5\_D | L5\_C | L5\_B | L5\_A | L4\_D | L4\_C | L4\_B | L4\_A | L3\_D | L3\_C | L3\_B | L3\_A | L2\_D | L2\_C | L2\_B | L2\_A | L1\_D | L1\_C | L1\_B | L1\_A | L0\_D | L0\_C | L0\_B | L0\_A |

Table NOC\_TX\_UPSIZER\_STATUS\_0 register.

### NOC\_TX\_UPSIZER\_STATUS\_1

These two registers (noc\_tx\_upsizer\_status\_0 and noc\_tx\_upsizer\_status\_1) track the status of the bridge transmitter upsizer/downsize structure. They can be used with the other status registers to check for packets that are still occupying the bridge. Each NoC layer, up to 16, can have upsizing/downsizing logic, and these 2 registers track the status of all 16 layers (noc\_tx\_upsizer\_status\_0 from 0 to 7 and noc\_tx\_upsizer\_status\_1 from 8 to 15).

Attribute: R

Security: Non-secure

Bit field description:

* **L15\_D**[31] - Interface upsizer status for interface D, Layer 15
* **L15\_C**[30] - Interface upsizer status for interface C, Layer 15
* **L15\_B**[29] - Interface upsizer status for interface B, Layer 15
* **L15\_A**[28] - Interface upsizer status for interface A, Layer 15
* **L14\_D**[27] - Interface upsizer status for interface D, Layer 14
* **L14\_C**[26] - Interface upsizer status for interface C, Layer 14
* **L14\_B**[25] - Interface upsizer status for interface B, Layer 14
* **L14\_A**[24] - Interface upsizer status for interface A, Layer 14
* **L13\_D**[23] - Interface upsizer status for interface D, Layer 13
* **L13\_C**[22] - Interface upsizer status for interface C, Layer 13
* **L13\_B**[21] - Interface upsizer status for interface B, Layer 13
* **L13\_A**[20] - Interface upsizer status for interface A, Layer 13
* **L12\_D**[19] - Interface upsizer status for interface D, Layer 12
* **L12\_C**[18] - Interface upsizer status for interface C, Layer 12
* **L12\_B**[17] - Interface upsizer status for interface B, Layer 12
* **L12\_A**[16] - Interface upsizer status for interface A, Layer 12
* **L11\_D**[15] - Interface upsizer status for interface D, Layer 11
* **L11\_C**[14] - Interface upsizer status for interface C, Layer 11
* **L11\_B**[13] - Interface upsizer status for interface B, Layer 11
* **L11\_A**[12] - Interface upsizer status for interface A, Layer 11
* **L10\_D**[11] - Interface upsizer status for interface D, Layer 10
* **L10\_C**[10] - Interface upsizer status for interface C, Layer 10
* **L10\_B**[9] - Interface upsizer status for interface B, Layer 10
* **L10\_A**[8] - Interface upsizer status for interface A, Layer 10
* **L9\_D**[7] - Interface upsizer status for interface D, Layer 9
* **L9\_C**[6] - Interface upsizer status for interface C, Layer 9
* **L9\_B**[5] - Interface upsizer status for interface B, Layer 9
* **L9\_A**[4] - Interface upsizer status for interface A, Layer 9
* **L8\_D**[3] - Interface upsizer status for interface D, Layer 8
* **L8\_C**[2] - Interface upsizer status for interface C, Layer 8
* **L8\_B**[1] - Interface upsizer status for interface B, Layer 8
* **L8\_A**[0] - Interface upsizer status for interface A, Layer 8

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| L15\_D | L15\_C | L15\_B | L15\_A | L14\_D | L14\_C | L14\_B | L14\_A | L13\_D | L13\_C | L13\_B | L13\_A | L12\_D | L12\_C | L12\_B | L12\_A | L11\_D | L11\_C | L11\_B | L11\_A | L10\_D | L10\_C | L10\_B | L10\_A | L9\_D | L9\_C | L9\_B | L9\_A | L8\_D | L8\_C | L8\_B | L8\_A |

Table NOC\_TX\_UPSIZER\_STATUS\_1 register.

## Amba Master Bridge Registers

### AXIM\_ADDR\_BASE

These registers specify the base addresses and masks of different slave ranges accessible from this master. One base, mask, and reloc register set per address range assigned to the master. These registers can be individually designated as read-only or read-write based on NocStudio property assigned to address ranges.

Even if the register is read-only, the range can be disabled using the appropriate bits described below which are always programmable.A slave address range is specified using the above base address and mask pair. An address on the AR or AW channel has a match against a range if it satisfies the equation

AxADDRS & AXIM\_ADDR\_MASK[i] == AXIM\_ADDR\_BASE[i]

Note that programmed 'base' must already factor the 'mask'. The base should not have a 1'b1 bit where the corresponding mask bit is 1'b0. What this means is that the programmed base should already have performed a bit-wise AND operation with the 'mask'. An address which doesn't match any range results in a decode error response. Note that programming of these registers must ensure that an address matches only against one range. Match against multiple ranges is a fatal error and will raise an interrupt.

Address ranges are specified at 64B cache line boundary. Lower six bits if AXIM\_ADDR\_BASE and AXIM\_ADDR\_MASK are used for specifying access permissions on an address range.

AXIM\_ADDR\_BASE[5:0]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **5** | **4** | **3** | **2** | **1** | **0** |
| X | DI | R/Wn | I | NS | P |

AXIM\_ADDR\_MASK[5:0]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **5** | **4** | **3** | **2** | **1** | **0** |
| X | X | Valid | I | NS | P |

Bits [2:0] act as value and mask for checking against AxPROT of an incoming command. A command is allowed access to a range if

AxPROT & AXIM\_ADDR\_MASK[2:0] == AXIM\_ADDR\_BASE[2:0] & AXIM\_ADDR\_MASK[2:0]

If the above check fails, then the command is denied access to the range and decode error response is returned. The encoding is specified below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AXIM\_ADDR\_BASE[4] Disable** | **AXIM\_\_ADDR\_MASK[3] RD/WRn valid** | **AXIM\_ADDR\_BASE[3] RD/WRn** |  | **Interpretation** |
| 1 | X | X |  | range disabled |
| 0 | 1 | 1 |  | Read only |
| 0 | 1 | 0 |  | Write only |
| 0 | 0 | X |  | read/write |

Attribute: RW

Security: Secure access only

Bit field description:

* **BASE\_ADDRESS**[63:6] - Base address
* **LLC**[5] - LLC disable
* **DI**[4] -   
  1'b1: Address range disabled
* **R\_Wn**[3] -   
  1'b1: Read enabled to range  
  1'b0: Write enabled to range
* **I**[2] -   
  1'b1: Instruction
* **NS**[1] -   
  1'b1: Non-secure
* **P**[0] -   
  1'b1: Privileged

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| BASE\_ADDRESS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BASE\_ADDRESS | | | | | | | | | | | | | | | | | | | | | | | | | | LLC | DI | R\_Wn | I | NS | P |

Table AXIM\_ADDR\_BASE register.

### AXIM\_ADDR\_HASH\_FUNCTION

These registers are used for programmable hash functions. They are the size of the SYSTEM address width, minus the 6 bit offset bits. Any reprogramming of these values can require the slave to understand the new hashing function in order to successfully compress the address space.

Attribute: RW

Security: Secure access only

Bit field description:

* **HASH**[39:6] - Hash bits
* **F**[0] -   
  1'b1: Force hash inversion. When set, this will invert the hash result.  
  1'b0: Default. When clear, the hash result is un-touched.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | HASH | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HASH | | | | | | | | | | | | | | | | | | | | | | | | | | u | | | | | F |

Table AXIM\_ADDR\_HASH\_FUNCTION register.

### AXIM\_ADDR\_MASK

See AXIM\_ADDR\_BASE.

Attribute: RW

Security: Secure access only

Bit field description:

* **MASK**[63:6] - Mask
* **KO**[5] - Keep out/reject read and/or write accesses
* **TM**[4] -   
  1'b1: Enable Trusted Master behavior for secure transactions
* **VAL**[3] -   
  1'b1: R\_Wn field is valid
* **I**[2] -   
  1'b1: Instruction field is valid
* **NS**[1] -   
  1'b1: Non-secure field is valid
* **P**[0] -   
  1'b1: Privileged field is valid

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| MASK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MASK | | | | | | | | | | | | | | | | | | | | | | | | | | KO | TM | VAL | I | NS | P |

Table AXIM\_ADDR\_MASK register.

### AXIM\_AR\_OVERRIDE

AR override.

Attribute: RW

Security: Secure access only

Bit field description:

* **arqos\_enb**[23:20] - 1'b1 indicates bit positions where ARQOS value is overridden. 1'b0 indicates bit positions where ARQOS is unchanged.
* **arqos\_val**[19:16] - Value to override incoming ARQOS
* **arprot\_enb**[14:12] - 1'b1 indicates bit positions where ARPROT value is overridden. 1'b0 indicates bit positions where ARPROT is unchanged.
* **arprot\_val**[10:8] - Value to override incoming ARPROT
* **arcache\_enb**[7:4] - 1'b1 indicates bit positions where ARCACHE value is overridden. 1'b0 indicates bit positions where ARCACHE is unchanged.
* **arcache\_val**[3:0] - Value to override incoming ARCACHE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | arqos\_enb | | | | arqos\_val | | | | u | arprot\_enb | | | u | arprot\_val | | | arcache\_enb | | | | arcache\_val | | | |

Table AXIM\_AR\_OVERRIDE register.

### AXIM\_ARADDR\_ON\_ERROR

This is the address on AR channel for which a decode error was detected. This corresponds to the status register bit e0 in AXIM\_ERROR\_INTERRUPT\_STATUS.

Attribute: R

Security: Non-secure

Bit field description:

* **READ\_DECERR\_ADDRS**[63:0] - Read decerr address

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| READ\_DECERR\_ADDRS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| READ\_DECERR\_ADDRS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_ARADDR\_ON\_ERROR register.

### AXIM\_AUTOWAKE\_POWER\_DOMAIN

Configures the master bridge's support for autowake of power domains.

When set, master bridge halts a request and issues wakeup requests for power domains that need to powered up to complete the transaction. The power domains should support auto wake. When reset, master bridge issues DECERR for any transaction which has dependent power domains in sleep state.

Attribute: RW

Security: Non-secure

Bit field description:

* **AW**[0] -   
  1'b1: Autowake enabled  
  1'b0: Autowake disabled

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | AW |

Table AXIM\_AUTOWAKE\_POWER\_DOMAIN register.

### AXIM\_AW\_OVERRIDE

AW override.

Attribute: RW

Security: Secure access only

Bit field description:

* **awqos\_enb**[23:20] - 1'b1 indicates bit positions where AWQOS value is overridden. 1'b0 indicates bit positions where AWQOS is unchanged.
* **awqos\_val**[19:16] - Value to override incoming AWQOS
* **awprot\_enb**[14:12] - 1'b1 indicates bit positions where AWPROT value is overridden. 1'b0 indicates bit positions where AWPROT is unchanged.
* **awprot\_val**[10:8] - Value to override incoming AWPROT
* **awcache\_enb**[7:4] - 1'b1 indicates bit positions where AWCACHE value is overridden. 1'b0 indicates bit positions where AWCACHE is unchanged.
* **awcache\_val**[3:0] - Value to override incoming AWCACHE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | awqos\_enb | | | | awqos\_val | | | | u | awprot\_enb | | | u | awprot\_val | | | awcache\_enb | | | | awcache\_val | | | |

Table AXIM\_AW\_OVERRIDE register.

### AXIM\_AWADDR\_ON\_ERROR

This is the address on AW channel for which a decode error was detected. This corresponds to the status register bit e16 in AXIM\_ERROR\_INTERRUPT\_STATUS.

Attribute: R

Security: Non-secure

Bit field description:

* **WRITE\_DECERR\_ADDRS**[63:0] - Write decerr address

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| WRITE\_DECERR\_ADDRS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WRITE\_DECERR\_ADDRS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_AWADDR\_ON\_ERROR register.

### AXIM\_BRIDGE\_ID

Unique identifier assigned to the master bridge.

Attribute: R

Security: Non-secure

Bit field description:

* **ID**[15:0] - Unique bridge ID

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | ID | | | | | | | | | | | | | | | |

Table AXIM\_BRIDGE\_ID register.

### AXIM\_CHECK\_OUTSTANDING\_REQ\_TO\_SLAVEID

This register is used to check if there are any outstanding read/write commands to a slave specified by field slvid. NocStudio provides a table of slvids corresponding to the slave ports accessible from a master bridge. Outstanding status is reflected in AXIM\_EVENT\_STATUS.

Attribute: RW

Security: Non-secure

Bit field description:

* **SLVID**[15:0] - A slave ID associated with the current master for command outstanding status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | SLVID | | | | | | | | | | | | | | | |

Table AXIM\_CHECK\_OUTSTANDING\_REQ\_TO\_SLAVEID register.

### AXIM\_CLK\_GATING\_HYSTERESIS\_COUNT

Programmable interval used by coarse clock gating logic in master bridge.This interval is used to generate heart beat pulses using noc\_clk on that bridge. These heart beat pulses are broadcast to each local clock gating domain within the bridge where they are synchronized to the CG domain's clock. Four consecutive heart beat pulses in the CG domain is used as the inactivity/idle interval to initiate coarse clock gating of the CG domain.

Attribute: RW

Security: Secure access only

Bit field description:

* **HYSTERESIS\_COUNTER**[31:0] - Hysteresis counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HYSTERESIS\_COUNTER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_CLK\_GATING\_HYSTERESIS\_COUNT register.

### AXIM\_CLK\_GATING\_OVERRIDE

Clock gating override, when set to 1'b1 will cause the clock gating logic to be disabled. 1'b1 will allow activity based clock gating to be performed on the master bridge.

Attribute: RW

Security: Secure access only

Bit field description:

* **FPO**[0] -   
  1'b1: Clock gating override is enabled (clock gating logic is disabled).  
  1'b0: Clock gating override is disabled (clock gating logic is enabled).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FPO |

Table AXIM\_CLK\_GATING\_OVERRIDE register.

### AXIM\_COHERENCY\_CONNECT\_ACK

This is the coherency connect status (acknowledge) register. Software uses this register to confirm a particular coherent agent is part of the coherency domain.

Attribute: R

Security: Non-secure

Bit field description:

* **STAT\_REG**[0] - Stat

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | STAT\_REG |

Table AXIM\_COHERENCY\_CONNECT\_ACK register.

### AXIM\_COHERENCY\_CONNECT\_REQ

This is the coherency connect request register. For agents that receive snoops (ACE, ACE-Lite+DVM), software must use this register for agents to participate in the coherency domain.

Attribute: RW

Security: Non-secure

Bit field description:

* **REQ\_REG**[0] - Request

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | REQ\_REG |

Table AXIM\_COHERENCY\_CONNECT\_REQ register.

### AXIM\_COUNT\_FOR\_LATENCY\_0

This register is programmed with the number of commands over which latency is to be measured. When this register counts down to 0, latency measurement is complete and average latency can be computed using:

Average command latency = Value in AXIM\_EVENT\_COUNTER\_0/Value which was programmed in AXIM\_COUNT\_FOR\_LATENCY\_0

There are two sets of counters available for gathering statistics. AXIM\_EVENT\_CAPTURE\_COMMAND\_1, AXIM\_EVENT\_CAPTURE\_COMMAND\_MASK\_1, AXIM\_EVENT\_COUNTER\_1, AXIM\_COUNT\_FOR\_LATENCY\_1 constitute the second bank of counters and are similar to the above set.

Attribute: RW

Security: Non-secure

Bit field description:

* **CNTR**[31:0] - Counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_COUNT\_FOR\_LATENCY\_0 register.

### AXIM\_COUNT\_FOR\_LATENCY\_1

This register is programmed with the number of commands over which latency is to be measured. When this register counts down to 0, latency measurement is complete and average latency can be computed using:

Average command latency = Value in AXIM\_EVENT\_COUNTER\_0/Value which was programmed in AXIM\_COUNT\_FOR\_LATENCY\_0

There are two sets of counters available for gathering statistics. AXIM\_EVENT\_CAPTURE\_COMMAND\_1, AXIM\_EVENT\_CAPTURE\_COMMAND\_MASK\_1, AXIM\_EVENT\_COUNTER\_1, AXIM\_COUNT\_FOR\_LATENCY\_1 constitute the second bank of counters and are similar to the above set.

Attribute: RW

Security: Non-secure

Bit field description:

* **CNTR**[31:0] - Counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_COUNT\_FOR\_LATENCY\_1 register.

### AXIM\_ERROR\_INTERRUPT\_MASK

Interrupt mask register. Individual bit position matches the error bit positions in AXIM\_ERROR\_INTERRUPT\_STATUS. When an INTM bit is set, occurrence of the corresponding error event will not cause an interrupt to be raised. When 1'b0, error event will cause interrupt to be raised.

Attribute: RW

Security: Non-secure

Bit field description:

* **E47**[47] -   
  1'b1: Flop Structure Parity Intr Mask
* **M46**[46] -   
  1'b1: CDDATA Parity Intr Mask
* **M45**[45] -   
  1'b1: WDATA Parity Intr Mask
* **M44**[44] -   
  1'b1: AWADDR Parity Intr Mask
* **M43**[43] -   
  1'b1: AW Parity Intr Mask
* **M42**[42] -   
  1'b1: ARADDR Parity Intr Mask
* **M41**[41] -   
  1'b1: AR Parity Intr Mask
* **M40**[40] -   
  1'b1: Mask interrupt for SIB portcheck error (SIB mode only)
* **M35**[35] -   
  1'b1: Mask interrupt on csr parity errors
* **M34**[34] -   
  1'b1: Mask interrupt on traffic to PG layer
* **M33**[33] -   
  1'b1: Counter 1 overflow interrupt mask
* **M32**[32] -   
  1'b1: Counter 0 overflow interrupt mask
* **M26**[26] -   
  1'b1: Mask interrupt on security check failure for write address range
* **M25**[25] -   
  1'b1: Mask interrupt on no route found for write address range
* **M24**[24] -   
  1'b1: Mask interrupt for write channel
* **M23**[23] -   
  1'b1: Mask interrupt for write channel
* **M22**[22] -   
  1'b1: Mask interrupt for write channel
* **M21**[21] -   
  1'b1: Mask interrupt for write channel
* **M20**[20] -   
  1'b1: Mask interrupt for write channel
* **M19**[19] -   
  1'b1: Mask interrupt for write channel
* **M18**[18] -   
  1'b1: Mask interrupt for write channel
* **M17**[17] -   
  1'b1: Mask interrupt for write channel
* **M16**[16] -   
  1'b1: Mask interrupt for write channel
* **M10**[10] -   
  1'b1: Mask interrupt on security check failure for read address range
* **M9**[9] -   
  1'b1: Mask interrupt on no route found for read address range
* **M8**[8] -   
  1'b1: Mask interrupt for read channel
* **M7**[7] -   
  1'b1: Mask interrupt for read channel
* **M6**[6] -   
  1'b1: Mask interrupt for read channel
* **M5**[5] -   
  1'b1: Mask interrupt for read channel
* **M4**[4] -   
  1'b1: Mask interrupt for read channel
* **M3**[3] -   
  1'b1: Mask interrupt for read channel
* **M2**[2] -   
  1'b1: Mask interrupt for read channel
* **M1**[1] -   
  1'b1: Mask interrupt for read channel
* **M0**[0] -   
  1'b1: Mask interrupt for read channel

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | E47 | M46 | M45 | M44 | M43 | M42 | M41 | M40 | u | | | | M35 | M34 | M33 | M32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | M26 | M25 | M24 | M23 | M22 | M21 | M20 | M19 | M18 | M17 | M16 | u | | | | | M10 | M9 | M8 | M7 | M6 | M5 | M4 | M3 | M2 | M1 | M0 |

Table AXIM\_ERROR\_INTERRUPT\_MASK register.

### AXIM\_ERROR\_INTERRUPT\_STATUS

These error status bits record the first error event and have to be cleared by writing a 1'b0 before new errors are recorded. For flop structure parity error interrupts, the AXIM\_LOG\_FLOPPARITY\_ERROR register should be cleared by writing it to zero before the flop structure parity interrupt bit is cleared in this register.

Attribute: WZC

Security: Non-secure

Bit field description:

* **E47**[47] -   
  1'b1: [FATAL] Flop Structure Parity Err
* **E46**[46] -   
  1'b1: [FATAL] CDDATA Parity Err
* **E45**[45] -   
  1'b1: [FATAL] WDATA Parity Err
* **E44**[44] -   
  1'b1: [FATAL] AWADDR Parity Err
* **E43**[43] -   
  1'b1: [FATAL] AW Parity Err
* **E42**[42] -   
  1'b1: [FATAL] ARADDR Parity Err
* **E41**[41] -   
  1'b1: [FATAL] AR Parity Err
* **E40**[40] -   
  1'b1: [FATAL] Indicates that portcheck detected error (SIB mode only)
* **E35**[35] -   
  1'b1: [FATAL] Parity error in configuration/status registers
* **E34**[34] -   
  1'b1: [FATAL] Traffic sent to a noc layer which is power gate
* **E33**[33] -   
  1'b1: Capture counter1 overflow
* **E32**[32] -   
  1'b1: Capture counter0 overflow
* **E26**[26] -   
  1'b1: [FATAL] Security check failure for write address range or rejected by keepout range
* **E25**[25] -   
  1'b1: [FATAL] No route found for write address range
* **E24**[24] -   
  1'b1: [FATAL] Unexpected narrow write detected
* **E23**[23] -   
  1'b1: [FATAL] Write WRAP not equal to supported cacheline size
* **E22**[22] -   
  1'b1: Write respone timeout
* **E21**[21] -   
  1'b1: [FATAL] Write address multi-hit
* **E20**[20] -   
  1'b1: [FATAL] Write exclusive split
* **E19**[19] -   
  1'b1: Non modifiable WRAP
* **E18**[18] -   
  1'b1: Write slave error
* **E17**[17] -   
  1'b1: Write address decode error from slave
* **E16**[16] -   
  1'b1: Local write address decode error
* **E10**[10] -   
  1'b1: [FATAL] Security check failure for read address range or rejected by keepout range
* **E9**[9] -   
  1'b1: [FATAL] No route found for read address range
* **E8**[8] -   
  1'b1: [FATAL] Unexpected narrow read detected
* **E7**[7] -   
  1'b1: [FATAL] Read WRAP not equal to supported cacheline size: A WRAP command of unupported cache line size was detected
* **E6**[6] -   
  1'b1: Read response timeout: Read response timeout occurred. With timeout enabled, a response wasn't received within the expected interval
* **E5**[5] -   
  1'b1: [FATAL] Read address multi-hit: An AR command matched against multiple entries in the address table
* **E4**[4] -   
  1'b1: [FATAL] Read exclusive split: An AR command of FIXED burst type was detected
* **E3**[3] -   
  1'b1: Non modifiable WRAP: A WRAP command marked as non-modifiable (ARCACHE[0]=0) was detected
* **E2**[2] -   
  1'b1: Read slave error: A slave error response was received from a slave device
* **E1**[1] -   
  1'b1: Read address decode error from slave: A decode error response was received from a slave device
* **E0**[0] -   
  1'b1: Local read address decode error: ARADDR did not find a match in the master bridges address table and a decode error was issued

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | u | | | | E35 | E34 | E33 | E32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 | u | | | | | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |

Table AXIM\_ERROR\_INTERRUPT\_STATUS register.

### AXIM\_ERROR\_INTERRUPT\_SVRTY

Interrupt Severity Register. Individual bit position matches the error bit positions in AXIM\_ERROR\_INTERRUPT\_STATUS. When an INT severity bit is set, occurrence of the corresponding error event will cause a fatal interrupt to be raised. When not set (1'b0), error event will cause non-fatal interrupt to be raised.

Attribute: RW

Security: Secure access only

Bit field description:

* **S47**[47] -   
  1'b1: Flop Structure Parity interrupt severity is Fatal.  
  1'b0: Flop Structure Parity interrupt severity is Non-Fatal.
* **S46**[46] -   
  1'b1: CDDATA Parity interrupt severity is Fatal.  
  1'b0: CDDATA Parity interrupt severity is Non-Fatal.
* **S45**[45] -   
  1'b1: WDATA Parity interrupt severity is Fatal.  
  1'b0: WDATA Parity interrupt severity is Non-Fatal.
* **S44**[44] -   
  1'b1: AWADDR Parity interrupt severity is Fatal.  
  1'b0: AWADDR Parity interrupt severity is Non-Fatal.
* **S43**[43] -   
  1'b1: AW Parity interrupt severity is Fatal.  
  1'b0: AW Parity interrupt severity is Non-Fatal.
* **S42**[42] -   
  1'b1: ARADDR Parity interrupt severity is Fatal.  
  1'b0: ARADDR Parity interrupt severity is Non-Fatal.
* **S41**[41] -   
  1'b1: AR Parity interrupt severity is Fatal.  
  1'b0: AR Parity interrupt severity is Non-Fatal.
* **S40**[40] -   
  1'b1: SIB portcheck error (SIB mode only) interrupt severity is Fatal.  
  1'b0: SIB portcheck error (SIB mode only) interrupt severity is Non-Fatal.
* **S35**[35] -   
  1'b1: CSR parity errors interrupt severity is Fatal.  
  1'b0: CSR parity errors interrupt severity is Non-Fatal.
* **S34**[34] -   
  1'b1: Traffic to PG layer interrupt severity is Fatal.  
  1'b0: Traffic to PG layer interrupt severity is Non-Fatal.
* **S33**[33] -   
  1'b1: Counter 1 overflow interrupt severity is Fatal.  
  1'b0: Counter 1 overflow interrupt severity is Non-Fatal.
* **S32**[32] -   
  1'b1: Counter 0 overflow interrupt severity is Fatal.  
  1'b0: Counter 0 overflow interrupt severity is Non-Fatal.
* **S26**[26] -   
  1'b1: Security check failure for write address range interrupt severity is Fatal.  
  1'b0: Security check failure for write address range interrupt severity is Non-Fatal.
* **S25**[25] -   
  1'b1: No route found for write address range interrupt severity is Fatal.  
  1'b0: No route found for write address range interrupt severity is Non-Fatal.
* **S24**[24] -   
  1'b1: Write channel interrupt severity is Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S23**[23] -   
  1'b1: Mask interrupt for write channel severity is Fatal.  
  1'b0: Mask interrupt for write channel severity is Non-Fatal.
* **S22**[22] -   
  1'b1: Write channel interrupt severity is Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S21**[21] -   
  1'b1: Write channel interrupt severity is Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S20**[20] -   
  1'b1: Write channel interrupt severity is Non-Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S19**[19] -   
  1'b1: Write channel interrupt severity is Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S18**[18] -   
  1'b1: Write channel interrupt severity is Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S17**[17] -   
  1'b1: Write channel interrupt severity is Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S16**[16] -   
  1'b1: Write channel interrupt severity is Fatal.  
  1'b0: Write channel interrupt severity is Non-Fatal.
* **S10**[10] -   
  1'b1: Interrupt on security check failure for read address range severity is Fatal.  
  1'b0: Interrupt on security check failure for read address range severity is Non-Fatal.
* **S9**[9] -   
  1'b1: Interrupt on no route found for read address range severity is Fatal.  
  1'b0: Interrupt on no route found for read address range severity is Non-Fatal.
* **S8**[8] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S7**[7] -   
  1'b1: Interrupt for read channelr severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S6**[6] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S5**[5] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S4**[4] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S3**[3] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S2**[2] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S1**[1] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S0**[0] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | S47 | S46 | S45 | S44 | S43 | S42 | S41 | S40 | u | | | | S35 | S34 | S33 | S32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | S26 | S25 | S24 | S23 | S22 | S21 | S20 | S19 | S18 | S17 | S16 | u | | | | | S10 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |

Table AXIM\_ERROR\_INTERRUPT\_SVRTY register.

### AXIM\_EVENT\_CAPTURE\_ADDR

This register is part of statistics gathering on the AR and AW command channels. This is the address value which is checked against AR, AW command channels in conjunction with the mask below to filter commands for statistics gathering.

Attribute: RW

Security: Non-secure

Bit field description:

* **CAPTURE\_ADDR**[63:0] - Capture address

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| CAPTURE\_ADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAPTURE\_ADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_EVENT\_CAPTURE\_ADDR register.

### AXIM\_EVENT\_CAPTURE\_ADDR\_MASK

If command address on the AR, AW channel logically ANDed with this mask is equal to the value specified in AXIM\_EVENT\_CAPTURE\_ADDR, then an address match has occurred. Note that only lowest significant bits equal to the master's address width are used in the comparison.

Attribute: RW

Security: Non-secure

Bit field description:

* **CAPTURE\_ADDR\_MASK**[63:0] - Capture address mask

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| CAPTURE\_ADDR\_MASK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAPTURE\_ADDR\_MASK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_EVENT\_CAPTURE\_ADDR\_MASK register.

### AXIM\_EVENT\_CAPTURE\_COMMAND\_0

Values of command fields/pins that are compared against AR, AW, R, W channel interface signals to filter commands/events for statistics gathering. Two selections can be made for statistics gathering, counting filtered commands or measuring latency of filtered commands.

Attribute: RW

Security: Non-secure

Bit field description:

* **TYP**[32] -   
  1'b1: Count response latency of captured command  
  1'b0: Count captured command
* **INTFID**[30:28] -   
  001: AW (can count captured event or response latency)  
  000: AR (can count captured event or response latency)
* **VAL**[25] -   
  1'b1: Valid
* **RDY**[24] -   
  1'b1: Ready
* **LOC**[23] -   
  1'b1: Lock
* **PROT**[22:20] - Prot
* **QOS**[19:16] - QoS
* **CACHE**[15:12] - Cache
* **BAR**[9:8] - Bar
* **DOMAIN**[5:4] - Domain
* **SNOOP**[3:0] - Snoop

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TYP |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | INTFID | | | u | | VAL | RDY | LOC | PROT | | | QOS | | | | CACHE | | | | u | | BAR | | u | | DOMAIN | | SNOOP | | | |

Table AXIM\_EVENT\_CAPTURE\_COMMAND\_0 register.

### AXIM\_EVENT\_CAPTURE\_COMMAND\_1

Values of command fields/pins that are compared against AR, AW, R, W channel interface signals to filter commands/events for statistics gathering. Two selections can be made for statistics gathering, counting filtered commands or measuring latency of filtered commands.

Attribute: RW

Security: Non-secure

Bit field description:

* **TYP**[32] -   
  1'b1: Count response latency of captured command  
  1'b0: Count captured command
* **INTFID**[30:28] -   
  001: AW (can count captured event or response latency)  
  000: AR (can count captured event or response latency)
* **VAL**[25] -   
  1'b1: Valid
* **RDY**[24] -   
  1'b1: Ready
* **LOC**[23] -   
  1'b1: Lock
* **PROT**[22:20] - Prot
* **QOS**[19:16] - QoS
* **CACHE**[15:12] - Cache
* **BAR**[9:8] - Bar
* **DOMAIN**[5:4] - Domain
* **SNOOP**[3:0] - Snoop

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TYP |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | INTFID | | | u | | VAL | RDY | LOC | PROT | | | QOS | | | | CACHE | | | | u | | BAR | | u | | DOMAIN | | SNOOP | | | |

Table AXIM\_EVENT\_CAPTURE\_COMMAND\_1 register.

### AXIM\_EVENT\_CAPTURE\_COMMAND\_MASK\_0

If Command fields on AR, AW channel logically ANDed with this mask are equal to the corresponding command field values in AXIM\_EVENT\_CAPTURE\_COMMAND\_0 then a command match has occurred. Address and command value match occurring together constitute events for the statistics counters.

Attribute: RW

Security: Non-secure

Bit field description:

* **NRW**[31] -   
  1'b1: Narrow
* **SPL**[30] -   
  1'b1: Split
* **SPLHZ**[29] -   
  1'b1: SplitHaz
* **ORDHZ**[28] -   
  1'b1: OrderHaz
* **MXOUT**[27] -   
  1'b1: MaxOutst
* **PWR**[26] -   
  1'b1: Power
* **VAL**[25] -   
  1'b1: Valid
* **RDY**[24] -   
  1'b1: Ready
* **LOC**[23] -   
  1'b1: Lock
* **PROT**[22:20] - Prot
* **QOS**[19:16] - QoS
* **CACHE**[15:12] - Cache
* **DECER**[11] -   
  1'b1: DecErr
* **NOCWT**[10] -   
  1'b1: NocWait
* **BAR**[9:8] - Bar
* **DOMAIN**[5:4] - Domain
* **SNOOP**[3:0] - Snoop

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NRW | SPL | SPLHZ | ORDHZ | MXOUT | PWR | VAL | RDY | LOC | PROT | | | QOS | | | | CACHE | | | | DECER | NOCWT | BAR | | u | | DOMAIN | | SNOOP | | | |

Table AXIM\_EVENT\_CAPTURE\_COMMAND\_MASK\_0 register.

### AXIM\_EVENT\_CAPTURE\_COMMAND\_MASK\_1

If Command fields on AR, AW channel logically ANDed with this mask are equal to the corresponding command field values in AXIM\_EVENT\_CAPTURE\_COMMAND\_0 then a command match has occurred. Address and command value match occurring together constitute events for the statistics counters.

Attribute: RW

Security: Non-secure

Bit field description:

* **NRW**[31] -   
  1'b1: Narrow
* **SPL**[30] -   
  1'b1: Split
* **SPLHZ**[29] -   
  1'b1: SplitHaz
* **ORDHZ**[28] -   
  1'b1: OrderHaz
* **MXOUT**[27] -   
  1'b1: MaxOutst
* **PWR**[26] -   
  1'b1: Power
* **VAL**[25] -   
  1'b1: Valid
* **RDY**[24] -   
  1'b1: Ready
* **LOC**[23] -   
  1'b1: Lock
* **PROT**[22:20] - Prot
* **QOS**[19:16] - QoS
* **CACHE**[15:12] - Cache
* **DECER**[11] -   
  1'b1: DecErr
* **NOCWT**[10] -   
  1'b1: NocWait
* **BAR**[9:8] - Bar
* **DOMAIN**[5:4] - Domain
* **SNOOP**[3:0] - Snoop

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NRW | SPL | SPLHZ | ORDHZ | MXOUT | PWR | VAL | RDY | LOC | PROT | | | QOS | | | | CACHE | | | | DECER | NOCWT | BAR | | u | | DOMAIN | | SNOOP | | | |

Table AXIM\_EVENT\_CAPTURE\_COMMAND\_MASK\_1 register.

### AXIM\_EVENT\_CAPTURE\_ID

This register is part of statistics gathering on the AR and AW command channels. This is the ID value which is checked against AR, AW command channels in conjunction with the mask below to filter commands for statistics gathering.

Attribute: RW

Security: Non-secure

Bit field description:

* **CAPTURE\_ID**[63:0] - Capture ID

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| CAPTURE\_ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAPTURE\_ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_EVENT\_CAPTURE\_ID register.

### AXIM\_EVENT\_CAPTURE\_ID\_MASK

If command ID on the AR, AW channel logically ANDed with this mask is equal to the value specified in AXIM\_EVENT\_CAPTURE\_ID, then an ID match has occurred. Note that only lowest significant bits equal to the master's ID width are used in the comparison.

Attribute: RW

Security: Non-secure

Bit field description:

* **CAPTURE\_ID\_MASK**[63:0] - Capture ID mask

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| CAPTURE\_ID\_MASK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAPTURE\_ID\_MASK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_EVENT\_CAPTURE\_ID\_MASK register.

### AXIM\_EVENT\_COUNTER\_0

32-bit counter which is used to count the captured statistics events. This counter can hold the count of commands filtered on the AR, AW channels. When measuring command latency, this counter holds the denominator or sum of number of cycles between command and response for multiple commands over which latency is measured.

Attribute: RW

Security: Non-secure

Bit field description:

* **CNTR**[31:0] - Counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_EVENT\_COUNTER\_0 register.

### AXIM\_EVENT\_COUNTER\_1

32-bit counter which is used to count the captured statistics events. This counter can hold the count of commands filtered on the AR, AW channels. When measuring command latency, this counter holds the denominator or sum of number of cycles between command and response for multiple commands over which latency is measured.

Attribute: RW

Security: Non-secure

Bit field description:

* **CNTR**[31:0] - Counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_EVENT\_COUNTER\_1 register.

### AXIM\_EVENT\_STATUS

When reordering is disabled on the master bridge, hazard stall occurs if the master tries to access a new slave device while response from a different slave is outstanding on the same AID.

This is because the responses can arrive out of order and the bridge is not equipped to correct the order. Without re-order buffers, hazard stalls also occur if a new large command needs to be split while there are older commands outstanding, or a large command just finished sending all its split segments but all responses have not returned yet.

When reordering is enabled, stall due to hazard occurs if a new command arrives, whose NoC QoS is different from the NoC QoS of commands outstanding on that AID.

Attribute: R

Security: Non-secure

Bit field description:

* **AWO**[7] -   
  1'b1: Write commands are outstanding to the slave specified in OSSLV register
* **ARO**[6] -   
  1'b1: Read commands are outstanding to the slave specified in OSSLV register
* **AWS**[5] -   
  1'b1: AW channel is stalled on hazard
* **ARS**[4] -   
  1'b1: AR channel is stalled on hazard
* **WOE**[3] -   
  1'b1: There are no write commands outstanding from the attached master device
* **ROE**[2] -   
  1'b1: There are no read commands outstanding from the attached master device
* **WOF**[1] -   
  1'b1: Maximum supported number of write commands are outstanding waiting for response and no more requests can be accepted  
  1'b0: Master bridge can accept more write requests
* **ROF**[0] -   
  1'b1: Maximum supported number of read commands are outstanding waiting for response and no more requests can be accepted  
  1'b0: Master bridge can accept more read requests

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | AWO | ARO | AWS | ARS | WOE | ROE | WOF | ROF |

Table AXIM\_EVENT\_STATUS register.

### AXIM\_INJECT\_FLOPPARITY\_ERROR

Error injection register for flop structure parity errors. This register is readable/writeable by software, and used by hardware to inject errors.

Attribute: RW

Security: Secure access only

Bit field description:

* **R\_CPKT\_FIFO\_ERR\_INJ**[14] -   
  1'b1: Inject parity error in R channel cpkt fifo.
* **CRID\_FIFO\_ERR\_INJ**[13] -   
  1'b1: Inject parity error in CRCD channel CRID fifo.
* **RACK\_FIFO\_ERR\_INJ**[12] -   
  1'b1: Inject parity error in ACK channel RACK fifo.
* **WACK\_FIFO\_ERR\_INJ**[11] -   
  1'b1: Inject parity error in ACK channel WACK fifo.
* **RXFIFO\_ERR\_INJ**[10] -   
  1'b1: Inject parity error in rx fifo specified by RXFIFO\_LAYER and RXFIFO\_VC.
* **RXFIFO\_LAYER**[9:6] - Rx Layer to force fifo parity error on
* **RXFIFO\_VC**[5:4] - Rx Virtual Channel to force fifo parity error on
* **WROB\_ERR\_INJ**[3] -   
  1'b1: Inject parity error in wrob
* **RROB\_ERR\_INJ**[2] -   
  1'b1: Inject parity error in rrob
* **WIDTBL\_ERR\_INJ**[1] -   
  1'b1: Inject parity error in widtbl
* **RIDTBL\_ERR\_INJ**[0] -   
  1'b1: Inject parity error in ridtbl

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | R\_CPKT\_FIFO\_ERR\_INJ | CRID\_FIFO\_ERR\_INJ | RACK\_FIFO\_ERR\_INJ | WACK\_FIFO\_ERR\_INJ | RXFIFO\_ERR\_INJ | RXFIFO\_LAYER | | | | RXFIFO\_VC | | WROB\_ERR\_INJ | RROB\_ERR\_INJ | WIDTBL\_ERR\_INJ | RIDTBL\_ERR\_INJ |

Table AXIM\_INJECT\_FLOPPARITY\_ERROR register.

### AXIM\_LOG\_FLOPPARITY\_ERROR

Error logging register for flop structure parity errors. Identify the flop structure that suffered the parity error. If a flop structure parity error occurs, this register should be cleared by writing zeros to it before the interrupt register is cleared.

Attribute: WZC

Security: Non-secure

Bit field description:

* **R\_CH\_CPKT\_FIFO\_PARITY\_ERR**[14] - R Channel Cpkt Fifo Parity Error
* **CRCD\_CH\_CRID\_FIFO\_PARITY\_ERR**[13] - CRCD Channel Crid Fifo Parity Error
* **ACK\_CH\_RACK\_FIFO\_PARITY\_ERR**[12] - Ack Channel Rack Fifo Parity Error
* **ACK\_CH\_WACK\_FIFO\_PARITY\_ERR**[11] - Ack Channel Wack Fifo Parity Error
* **RXFIFO\_PARITY\_ERR\_LAYER**[10:7] - Rx Fifo Parity Error Layer
* **RXFIFO\_PARITY\_ERR\_VC**[6:5] - Rx Fifo Parity Error Virtual Channel
* **RXFIFO\_PARITY\_ERR**[4] - Rx Fifo Parity Error
* **WROB\_PARITY\_ERR**[3] - Write Response Buffer Parity Error
* **RROB\_PARITY\_ERR**[2] - Read Response Buffer Parity Error
* **WIDTBL\_PARITY\_ERR**[1] - Write Aidtbl Parity Error
* **RIDTBL\_PARITY\_ERR**[0] - Read Aidtbl Parity Error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | R\_CH\_CPKT\_FIFO\_PARITY\_ERR | CRCD\_CH\_CRID\_FIFO\_PARITY\_ERR | ACK\_CH\_RACK\_FIFO\_PARITY\_ERR | ACK\_CH\_WACK\_FIFO\_PARITY\_ERR | RXFIFO\_PARITY\_ERR\_LAYER | | | | RXFIFO\_PARITY\_ERR\_VC | | RXFIFO\_PARITY\_ERR | WROB\_PARITY\_ERR | RROB\_PARITY\_ERR | WIDTBL\_PARITY\_ERR | RIDTBL\_PARITY\_ERR |

Table AXIM\_LOG\_FLOPPARITY\_ERROR register.

### AXIM\_RESPONSE\_TIMEOUT\_CONTROL

This register is used to configure response timeouts.

AXIM\_RESPONSE\_TIMEOUT\_CONTROL[8] (En) needs to be set for timeout tracking to be enabled. When this bit is 1'b0, no timestamps are recorded to generate timeout interrupts. A 64-bit free running counter is used to time the response interval.

AXIM\_RESPONSE\_TIMEOUT\_CONTROL[5:0] (TI) specifies the lower bit index into this counter, from where 2-bits are picked up and recorded as the arrival time stamp of every incoming AR and AW command. If response for a command does not return before the current time stamp rolls to arrival time stamp minus 1, the response is assumed to have timedout and an interrupt is raised along with the slave ID to which the timed out request was sent.

When changing the TI field, first write to the register with the En field cleared, then write a second time with the TI field to its new value, then a 3rd write to restore the En field to Enabled. During this update while the En field is cleared, existing timers will cancelled, and new timer starts will be inhibited.

Attribute: RW

Security: Secure access only

Bit field description:

* **EN**[8] -   
  1'b1: Enabled timeout tracking, a 64-bit free running counter is used to time the response interval.  
  1'b0: No timestamps are recorded to generate timeout interrupts
* **TI**[5:0] - Timer index, index of a 64-bit counter from where timestamp is picked. The register value has to be 'd62 or smaller.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | EN | u | | TI | | | | | |

Table AXIM\_RESPONSE\_TIMEOUT\_CONTROL register.

### AXIM\_RESPONSE\_TIMEOUT\_SLAVEID

AR slvid and AW slvid fields indicate slave IDs to which a read, write response timeout was detected.

Note that slvid encoding is not same as the bridge ID of the slave. NocStudio provides a table mapping the slvids to the actual slave ports accessible from the master bridge.

Attribute: R

Security: Non-secure

Bit field description:

* **AW\_SLVID**[31:16] - Slave ID of timed out AW request
* **AR\_SLVID**[15:0] - Slave ID of timed out AR request

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AW\_SLVID | | | | | | | | | | | | | | | | AR\_SLVID | | | | | | | | | | | | | | | |

Table AXIM\_RESPONSE\_TIMEOUT\_SLAVEID register.

### AXIM\_SAI\_RAC

This register defines the read access control for the SAI based firewall. Using SAI as the index, if the RAC[SAI] is 1’b1 the access is granted, otherwise (1’b0) the access is denied.

Attribute: RW

Security: Secure access only

Bit field description:

* RAC[SAI] = 1’b1 – read access granted
* RAC[SAI] = 1’b0 – read access denied

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| Read Permission for SAI = 32 ~ 63 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Read Permission for SAI = 0 ~ 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 63 AXIM\_SAI\_RAC register

### AXIM\_SAI\_WAC

This register defines the write access control for the SAI based firewall. Using SAI as the index, if the WAC[SAI] is 1’b1 the access is granted, otherwise (1’b0) the access is denied.

Attribute: RW

Security: Secure access only

Bit field description:

* WAC[SAI] = 1’b1 – write access granted
* WAC[SAI] = 1’b0 – write access denied

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| Write Permission for SAI = 32 ~ 63 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Write Permission for SAI = 0 ~ 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 64 AXIM\_SAI\_WAC register

### AXIM\_SIB\_PORT\_DISABLE\_[sib\_name]\_M

Indicates which of the SIB ports are disabled. When a port is disabled, all output signals are driven low, and all input signals are ignored.

Attribute: RW

Security: Secure access only

Bit field description:

* **Port\_disable**[x] - Enable/Disable bit for connected master bridge on SIB. 1 means disabled and 0 means enabled.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | port disable | | | | | | | | | | | | | | | |

Table 65 AXI\_SIB\_PORT\_DISABLE\_[sib\_name]\_M register

### AXIM\_SECERR\_INTERRUPT\_MASK

Security Interrupt mask register. Individual bit position matches the error bit positions in AXIM\_SECERR\_INTERRUPT\_STATUS. When an INTM bit is set, occurrence of the corresponding secure error event will not cause an interrupt to be raised. When 1'b0, error event will cause interrupt to be raised. This register is secured.

Attribute: RW

Security: Secure access only

Bit field description:

* **M26**[26] -   
  1'b1: Mask security interrupt on security check failure for write address range  
  1'b0: Enable security interrupt on security check failure for write address range
* **M10**[10] -   
  1'b1: Mask security interrupt on security check failure for read address range  
  1'b0: Enable security interrupt on security check failure for write address range

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | M26 | u | | | | | | | | | | | | | | | M10 | u | | | | | | | | | |

Table AXIM\_SECERR\_INTERRUPT\_MASK register.

### AXIM\_SECERR\_INTERRUPT\_STATUS

These secure error status bits record the first security error event and have to be cleared by writing a 1'b0 before new errors are recorded. This register is secured.

Attribute: WZC

Security: Secure access only

Bit field description:

* **E26**[26] -   
  1'b1: [FATAL] Security check failure for write address range or rejected by keepout range
* **E10**[10] -   
  1'b1: [FATAL] Security check failure for read address range or rejected by keepout range

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | E26 | u | | | | | | | | | | | | | | | E10 | u | | | | | | | | | |

Table AXIM\_SECERR\_INTERRUPT\_STATUS register.

### AXIM\_SLAVE\_ADDR\_RELOCATION

Register used to relocate a master address to slave address.

Attribute: R

Security: Secure access only

Bit field description:

* **SLV\_RELOC**[39:6] - Slave Reloc

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | SLV\_RELOC | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SLV\_RELOC | | | | | | | | | | | | | | | | | | | | | | | | | | u | | | | | |

Table AXIM\_SLAVE\_ADDR\_RELOCATION register.

### AXIM\_SYSTEM\_ADDR\_RELOCATION

Register used to relocate a master address to system address.

Attribute: RW

Security: Secure access only

Bit field description:

* **SYS\_RELOC**[39:6] - System Reloc

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | SYS\_RELOC | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SYS\_RELOC | | | | | | | | | | | | | | | | | | | | | | | | | | u | | | | | |

Table AXIM\_SYSTEM\_ADDR\_RELOCATION register.

## Regbus Registers

### AXIM\_NOC\_VERSION\_ID

Version identifier for the NoC. This read-only register is available only on the regbus master. This register is not available on other master bridges and access will result in decode error response.

Attribute: R

Security: Non-secure

Bit field description:

* **NOC\_VERSION\_ID**[31:0] - NoC version ID

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NOC\_VERSION\_ID | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIM\_NOC\_VERSION\_ID register.

## AMBA Slave Registers

### AXIS\_AR\_OVERRIDE

AR Overrides.

Attribute: RW

Security: Secure access only

Bit field description:

* **arqos\_enb**[23:20] - 1'b1 indicates bit positions where ARQOS value is overridden. 1'b0 indicates bit positions where ARQOS is unchanged.
* **arqos\_val**[19:16] - Value to override incoming ARQOS
* **arprot\_enb**[14:12] - 1'b1 indicates bit positions where ARPROT value is overridden. 1'b0 indicates bit positions where ARPROT is unchanged.
* **arprot\_val**[10:8] - Value to override incoming ARPROT
* **arcache\_enb**[7:4] - 1'b1 indicates bit positions where ARCACHE value is overridden. 1'b0 indicates bit positions where ARCACHE is unchanged.
* **arcache\_val**[3:0] - Value to override incoming ARCACHE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | arqos\_enb | | | | arqos\_val | | | | u | arprot\_enb | | | u | arprot\_val | | | arcache\_enb | | | | arcache\_val | | | |

Table AXIS\_AR\_OVERRIDE register.

### AXIS\_AW\_OVERRIDE

AW Overrides.

Attribute: RW

Security: Secure access only

Bit field description:

* **awqos\_enb**[23:20] - 1'b1 indicates bit positions where AWQOS value is overridden. 1'b0 indicates bit positions where AWQOS is unchanged.
* **awqos\_val**[19:16] - Value to override incoming AWQOS
* **awprot\_enb**[14:12] - 1'b1 indicates bit positions where AWPROT value is overridden. 1'b0 indicates bit positions where AWPROT is unchanged.
* **awprot\_val**[10:8] - Value to override incoming AWPROT
* **awcache\_enb**[7:4] - 1'b1 indicates bit positions where AWCACHE value is overridden. 1'b0 indicates bit positions where AWCACHE is unchanged.
* **awcache\_val**[3:0] - Value to override incoming AWCACHE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | awqos\_enb | | | | awqos\_val | | | | u | awprot\_enb | | | u | awprot\_val | | | awcache\_enb | | | | awcache\_val | | | |

Table AXIS\_AW\_OVERRIDE register.

### AXIS\_BRIDGE\_ID

Unique identifier assigned to the slave bridge.

Attribute: R

Security: Non-secure

Bit field description:

* **ID**[15:0] - Unique bridge ID

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | ID | | | | | | | | | | | | | | | |

Table AXIS\_BRIDGE\_ID register.

### AXIS\_CLK\_GATING\_HYSTERESIS\_COUNT

Programmable intervals used by coarse clock gating logic. This interval is used to generate heart beat pulses using noc\_clk on that bridge. These heart beat pulses are broadcast to each local clock gating domain within the bridge where they are synchronized to the CG domain's clock. Four consecutive heart beat pulses in the CG domain is used as the inactivity/idle interval to initiate coarse clock gating of the CG domain.

Attribute: RW

Security: Secure access only

Bit field description:

* **HYSTERESIS\_COUNTER**[31:0] - Hysteresis counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HYSTERESIS\_COUNTER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIS\_CLK\_GATING\_HYSTERESIS\_COUNT register.

### AXIS\_CLK\_GATING\_OVERRIDE

Clock gating override, when set to 1'b1 will cause the clock gating logic to be disabled. 1'b1 will allow activity based clock gating to be performed on the slave bridge.

Attribute: RW

Security: Secure access only

Bit field description:

* **FPO**[0] -   
  1'b1: Clock gating override enabled (clock gating logic is disabled).  
  1'b0: Clock gating override is disabled (clock gating logic is enabled).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FPO |

Table AXIS\_CLK\_GATING\_OVERRIDE register.

### AXIS\_ERROR\_INTERRUPT\_MASK

Interrupt mask register. Individual bit positions match the error bit positions in AXIS\_ERROR\_INTERRUPT\_STATUS. When a mask bit is set, occurrence of the corresponding error event will not cause an interrupt to be raised. When 1'b0, error event will cause interrupt to be raised.

Attribute: RW

Security: Non-secure

Bit field description:

* **E39**[39] - Flop Structure Parity Err Intr Mask
* **M38**[38] - ACADDR parity interrupt Mask
* **M37**[37] - AC parity interrupt Mask
* **M36**[36] - BRESP parity interrupt Mask
* **M35**[35] - RRESP parity interrupt Mask
* **M34**[34] - RDATA parity interrupt Mask
* **M33**[33] - Mask interrupt on csr parity errors
* **M32**[32] - Mask interrupt on traffic to PG layer
* **M19**[19] - Mask interrupts for write channel
* **M18**[18] - Mask interrupts for write channel
* **M17**[17] - Mask interrupts for write channel
* **M16**[16] - Mask interrupts for write channel
* **M4**[4] - Mask interrupts for read channel
* **M3**[3] - Mask interrupts for read channel
* **M2**[2] - Mask interrupts for read channel
* **M1**[1] - Mask interrupts for read channel
* **M0**[0] - Mask interrupts for read channel

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | E39 | M38 | M37 | M36 | M35 | M34 | M33 | M32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | M19 | M18 | M17 | M16 | u | | | | | | | | | | | M4 | M3 | M2 | M1 | M0 |

Table AXIS\_ERROR\_INTERRUPT\_MASK register.

### AXIS\_ERROR\_INTERRUPT\_STATUS

These error status bits record the first error event and have to be cleared by writing a 1'b0 before new errors are recorded. For flop structure parity errors, the AXIS\_LOG\_FLOPPARITY\_ERROR register should be cleared by writing 0s to it, before the flop structure parity bit is cleared in this register.

Attribute: WZC

Security: Non-secure

Bit field description:

* **E39**[39] -   
  1'b1: Flop Structure Parity Err
* **E38**[38] -   
  1'b1: [FATAL] ACADDR Parity error
* **E37**[37] -   
  1'b1: [FATAL] AC Parity error
* **E36**[36] -   
  1'b1: [FATAL] BRESP Parity error
* **E35**[35] -   
  1'b1: [FATAL] RRESP Parity error
* **E34**[34] -   
  1'b1: [FATAL] RDATA Parity error
* **E33**[33] -   
  1'b1: [FATAL] Parity error in config/status registers
* **E32**[32] -   
  1'b1: [FATAL] Traffic sent to a noc layer which is power gated
* **E19**[19] -   
  1'b1: Write command modified: A write command which was marked as non-modifiable was modified by the slave bridge
* **E18**[18] -   
  1'b1: [FATAL] Unknown write response destination: BID from write response produces a destination which is not present in the routing table
* **E17**[17] -   
  1'b1: Write slave error response: Slave error response received from slave device for write command
* **E16**[16] -   
  1'b1: Write decode error response: Decode error response received from slave device for write command
* **E4**[4] -   
  1'b1: Read command modified: A read command which was marked as non-modifiable was modified by the slave bridge
* **E3**[3] -   
  1'b1: [FATAL] Interleaved read response: Interleaved read response. This can occur if interleaved read response is received from a slave device for which a de-interleaver was not specified
* **E2**[2] -   
  1'b1: [FATAL] Unknown read response destination: RID from read response produces a destination which is not present in the routing table
* **E1**[1] -   
  1'b1: Read slave error response: Slave error response received from slave device for read command
* **E0**[0] -   
  1'b1: Read decode error response: Decode error response received from slave device for read command

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | E19 | E18 | E17 | E16 | u | | | | | | | | | | | E4 | E3 | E2 | E1 | E0 |

Table AXIS\_ERROR\_INTERRUPT\_STATUS register.

### AXIS\_ERROR\_INTERRUPT\_SVRTY

Interrupt Severity Register. Individual bit position matches the error bit positions in AXIS\_ERROR\_INTERRUPT\_STATUS. When an INT severity bit is set, occurrence of the corresponding error event will cause a fatal interrupt to be raised. When not set (1'b0), error event will cause non-fatal interrupt to be raised.

Attribute: RW

Security: Secure access only

Bit field description:

* **S39**[39] -   
  1'b1: Flop Structure Parity Error interrupt severity is Fatal.  
  1'b0: Flop Structure Parity Error interrupt severity is Non-Fatal.
* **S38**[38] -   
  1'b1: ACADDR parity interrupt severity is Fatal.  
  1'b0: ACADDR parity interrupt severity is Non-Fatal.
* **S37**[37] -   
  1'b1: AC parity interrupt severity is Fatal.  
  1'b0: AC parity interrupt severity is Non-Fatal.
* **S36**[36] -   
  1'b1: BRESP parity interrupt severity is Fatal.  
  1'b0: BRESP parity interrupt severity is Non-Fatal.
* **S35**[35] -   
  1'b1: RRESP parity interrupt severity is Fatal.  
  1'b0: RRESP parity interrupt severity is Non-Fatal.
* **S34**[34] -   
  1'b1: RDATA parity interrupt severity is Fatal.  
  1'b0: RDATA parity interrupt severity is Non-Fatal.
* **S33**[33] -   
  1'b1: CSR parity errors interrupt severity is Fatal.  
  1'b0: CSR parity errors interrupt severity is Non-Fatal.
* **S32**[32] -   
  1'b1: Interrupt on traffic to PG layer severity is Fatal.  
  1'b0: Interrupt on traffic to PG layer severity is Non-Fatal.
* **S19**[19] -   
  1'b1: Interrupt for write channel severity is Fatal.  
  1'b0: Interrupt for write channel severity is Non-Fatal.
* **S18**[18] -   
  1'b1: Interrupt for write channel severity is Fatal.  
  1'b0: Interrupt for write channel severity is Non-Fatal.
* **S17**[17] -   
  1'b1: Interrupt for write channel severity is Fatal.  
  1'b0: Interrupt for write channel severity is Non-Fatal.
* **S16**[16] -   
  1'b1: Interrupt for write channel severity is Fatal.  
  1'b0: Interrupt for write channel severity is Non-Fatal.
* **S4**[4] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S3**[3] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S2**[2] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S1**[1] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.
* **S0**[0] -   
  1'b1: Interrupt for read channel severity is Fatal.  
  1'b0: Interrupt for read channel severity is Non-Fatal.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | S39 | S38 | S37 | S36 | S35 | S34 | S33 | S32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | S19 | S18 | S17 | S16 | u | | | | | | | | | | | S4 | S3 | S2 | S1 | S0 |

Table AXIS\_ERROR\_INTERRUPT\_SVRTY register.

### AXIS\_EVENT\_CAPTURE\_COMMAND

Not applicable for current release.

Attribute: RW

Security: Non-secure

Bit field description:

* **intfid**[30:28] -   
  001: AW (can count captured event or response latency)  
  000: AR (can count captured event or response latency)
* **val**[25] -   
  1'b1: Valid
* **rdy**[24] -   
  1'b1: Ready

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | intfid | | | u | | val | rdy | u | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIS\_EVENT\_CAPTURE\_COMMAND register.

### AXIS\_EVENT\_CAPTURE\_COMMAND\_MASK

Not applicable for current release.

Attribute: RW

Security: Non-secure

Bit field description:

* **val**[25] -   
  1'b1: Valid
* **rdy**[24] -   
  1'b1: Ready

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | val | rdy | u | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIS\_EVENT\_CAPTURE\_COMMAND\_MASK register.

### AXIS\_EVENT\_COUNTER

Not applicable for current release.

Attribute: R

Security: Non-secure

Bit field description:

* **CNTR**[31:0] - Counter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table AXIS\_EVENT\_COUNTER register.

### AXIS\_EVENT\_STATUS

Slave bridge status bits.

Attribute: R

Security: Non-secure

Bit field description:

* **ROE**[3] -   
  1'b1: There are no read commands outstanding to the attached slave device
* **WOE**[2] -   
  1'b1: There are no write commands outstanding to the attached slave device
* **ROF**[1] -   
  1'b1: Maximum number of supported read commands are outstanding to the attached slave device awaiting response, no more read commands will be issued to slave till responses are received.  
  1'b0: Slave bridge can accept more read commands from the NoC
* **WOF**[0] -   
  1'b1: Maximum number of supported write commands are outstanding to the attached slave device awaiting response, no more write commands will be issued to slave till responses are received.  
  1'b0: Slave device can expect more write commands from NoC

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | ROE | WOE | ROF | WOF |

Table AXIS\_EVENT\_STATUS register.

### AXIS\_INJECT\_FLOPPARITY\_ERROR

Error injection register for flop structure parity errors. Bits[7:0]. This register is readable/writeable by software, and used by hardware to inject errors.

Attribute: RW

Security: Secure access only

Bit field description:

* **R\_CH\_FLUSH\_FIFO\_ERR\_INJ**[15] -   
  1'b1: Inject parity error into R Ch Flush Fifo.
* **B\_CH\_DRAIN\_FIFO\_ERR\_INJ**[14] -   
  1'b1: Inject parity error into B Ch Drain Fifo.
* **RACK\_ROB\_ENTRY\_ERR\_INJ**[13] -   
  1'b1: Inject parity error into ACK ch RACK ROB.
* **WACK\_ROB\_ENTRY\_ERR\_INJ**[12] -   
  1'b1: Inject parity error into ACK ch WACK ROB.
* **CRCD\_CH\_ROB\_ENTRY\_ERR\_INJ**[11] -   
  1'b1: Inject parity error into CRCD ch ROB.
* **RXFIFO\_ERR\_INJ\_LAYER**[10:7] - Rx Fifo Virtual Channel to force parity error on
* **RXFIFO\_ERR\_INJ\_VC**[6:5] - Rx Fifo Virtual Channel to force parity error on
* **RXFIFO\_ERR\_INJ**[4] -   
  1'b1: Inject parity error in rx fifo specified by RXFIFO\_LAYER and RXFIFO\_VC.
* **R\_CH\_DINT\_DBUF\_ERR\_INJ**[3] -   
  1'b1: Inject parity error into R Ch Deinterleaver Data Buffer.
* **R\_CH\_DINT\_CMDTBL\_ERR\_INJ**[2] -   
  1'b1: Inject parity error into R Ch Deinterleaver Cmdtbl.
* **B\_CH\_CMDTBL\_ERR\_INJ**[1] -   
  1'b1: Inject parity error into B Ch Cmdtbl
* **R\_CH\_CMDTBL\_ERR\_INJ**[0] -   
  1'b1: Inject parity error into R Ch Cmdtbl

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | R\_CH\_FLUSH\_FIFO\_ERR\_INJ | B\_CH\_DRAIN\_FIFO\_ERR\_INJ | RACK\_ROB\_ENTRY\_ERR\_INJ | WACK\_ROB\_ENTRY\_ERR\_INJ | CRCD\_CH\_ROB\_ENTRY\_ERR\_INJ | RXFIFO\_ERR\_INJ\_LAYER | | | | RXFIFO\_ERR\_INJ\_VC | | RXFIFO\_ERR\_INJ | R\_CH\_DINT\_DBUF\_ERR\_INJ | R\_CH\_DINT\_CMDTBL\_ERR\_INJ | B\_CH\_CMDTBL\_ERR\_INJ | R\_CH\_CMDTBL\_ERR\_INJ |

Table AXIS\_INJECT\_FLOPPARITY\_ERROR register.

### AXIS\_LOG\_FLOPPARITY\_ERROR

Error logging register for flop structure parity errors. This reg is written by hardware and read/cleared by software. It has meaning only if the flop structure parity error bit in AXIS\_ERROR\_INTERRUPT\_STATUS is set. To clear a flop structure parity interrupt, this register should be cleared by software before the corresponding bit in AXIS\_ERROR\_INTERRUPT\_STATUS is cleared by software.

Attribute: WZC

Security: Non-secure

Bit field description:

* **R\_CH\_FLUSH\_FIFO\_PARITY\_ERR**[15] - R Ch Flush Fifo Parity Err
* **B\_CH\_DRAIN\_FIFO\_PARITY\_ERR**[14] - B Ch Drain Fifo Parity Err
* **RACK\_ROB\_ENTRY\_PARITY\_ERR**[13] - Ack Ch Rack Reorder Buffer Parity Err
* **WACK\_ROB\_ENTRY\_PARITY\_ERR**[12] - Ack Ch Wack Reorder Buffer Parity Err
* **CRCD\_CH\_ROB\_PARITY\_ERR**[11] - CRCD Ch Reorder Buffer Parity Err
* **RXFIFO\_PARITY\_ERR\_LAYER**[10:7] - Rx Fifo Parity Error Layer
* **RXFIFO\_PARITY\_ERR\_VC**[6:5] - Rx Fifo Parity Error Virtual Channel
* **RXFIFO\_PARITY\_ERR**[4] - Rx Fifo Parity Err
* **R\_CH\_DINT\_DBUF\_PARITY\_ERR**[3] - R Ch Deinterleaver Data Buffer Parity Err
* **R\_CH\_DINT\_CMDTBL\_PARITY\_ERR**[2] - R Ch Deinterleaver Cmdtbl Parity Error
* **B\_CH\_CMDTBL\_PARITY\_ERR**[1] - B Ch Cmdtbl Parity Error
* **R\_CH\_CMDTBL\_PARITY\_ERR**[0] - R Ch Cmdtbl Parity Error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | R\_CH\_FLUSH\_FIFO\_PARITY\_ERR | B\_CH\_DRAIN\_FIFO\_PARITY\_ERR | RACK\_ROB\_ENTRY\_PARITY\_ERR | WACK\_ROB\_ENTRY\_PARITY\_ERR | CRCD\_CH\_ROB\_PARITY\_ERR | RXFIFO\_PARITY\_ERR\_LAYER | | | | RXFIFO\_PARITY\_ERR\_VC | | RXFIFO\_PARITY\_ERR | R\_CH\_DINT\_DBUF\_PARITY\_ERR | R\_CH\_DINT\_CMDTBL\_PARITY\_ERR | B\_CH\_CMDTBL\_PARITY\_ERR | R\_CH\_CMDTBL\_PARITY\_ERR |

Table AXIS\_LOG\_FLOPPARITY\_ERROR register.

## Protocol Converter Registers

### AHBM\_CTL

Control Register.

Attribute: RW

Bit field description:

* WNP [0] - 1'b1: Force non-posted writes.  
   - 1'b0: Not forced to non-posted writes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WNP |

Table 85 AHBM\_CTL register

### AHBM\_IM

Interrupt Mask Register.

Attribute: RW

Bit field description:

* EWRP [2] - 1'b1: Mask interrupt due to Illegal WRAP.
* WER [1] - 1'b1: Mask interrupt due to Write error.
* RER [0] - 1'b1: Mask interrupt due to Read error.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | EWRP | WER | RER |

Table 86 AHBM\_IM register

### AHBM\_STS

Status Register.

Attribute: WZC

Bit field description:

* EWRP [2] - 1'b1: Illegal WRAP detected.
* WER [1] - 1'b1: Write error detected.
* RER [0] - 1'b1: Read error detected.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | EWRP | WER | RER |

Table 87 AHBM\_STS register

### AHBS\_CTL

Control Register.

Attribute: RW

Bit field description:

* REG [16] - 1'b1: Issue error response on unaligned read address.  
   - 1'b0: Issue OK response even if unaligned address was aligned and   
   issued.
* ZEROES [15:0] - Zeroes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | REG | ZEROES | | | | | | | | | | | | | | | |

Table 88 AHBS\_CTL register

### AHBS\_IM

Interrupt Mask Register

Attribute: RW

Bit field description:

* REM [17] - 1'b1: No interrupt raised on RE.  
   - 1'b0: Interrupt raised on RE.
* WEM [16] - 1'b1: No interrupt raised on WE.  
   - 1'b0: Interrupt raised on WE.
* ZEROES [15:0] - Zeroes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | REM | WEM | ZEROES | | | | | | | | | | | | | | | |

Table 89 AHBS\_IM register

### AHBS\_STS

Status Register.

Attribute: WZC

Bit field description:

* RE [17] - 1'b1: Unaligned read address was received.
* WE [16] - 1'b1: Unaligned write address (wrt asize)., or partial write strobes (wrt   
   asize) was received.
* ZEROES [15:0] - Zeroes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | RE | WE | ZEROES | | | | | | | | | | | | | | | |

Table 90 AHBS\_STS register

APB Bridge

### APBSLV\_BRIDGE\_ID

Bridge ID register.

Attribute: R

Bit field description:

* **ZEROES** [15:8] - Zeroes
* **ID** [7:0] - Unique bridge id.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | ZEROES | | | | | | | | ID | | | | | | | |

Table 91 APBSLV\_BRIDGE\_ID register

### APBSLV\_BRIDGE\_VERSION

Bridge version register.

Attribute: R

Bit field description:

* **VER** [3:0] - Bridge version.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | VER | | | |

Table 92 APBSLV\_BRIDGE\_VERSION register

### APBSLV\_SLVS\_SLEEP\_STATUS

Slave sleep status register.

Attribute: RW

Bit field description:

* **STS\_15** [15] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_14** [14] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_13** [13] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_12** [12] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_11** [11] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_10** [10] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_9** [9] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_8** [8] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_7** [7] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_6** [6] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_5** [5] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_4** [4] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_3** [3] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_2** [2] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_1** [1] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active
* **STS\_0** [0] - 1'b1: Slave is in sleep mode  
   - 1'b0: Slave is active

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | STS\_15 | STS\_14 | STS\_13 | STS\_12 | STS\_11 | STS\_10 | STS\_9 | STS\_8 | STS\_7 | STS\_6 | STS\_5 | STS\_4 | STS\_3 | STS\_2 | STS\_1 | STS\_0 |

Table 93 APBSLV\_SLVS\_SLEEP\_STATUS register

## DVM Host Registers

### DVM\_ACTIVE\_VECTOR

ACTIVE\_VECTOR\_0/1/2/3: Register indicating active DVM agents in the system.

The DVM IP module is parameterized for a maximum number of agents supporting DVM in the system. This parameterization is taken care of by NocStudio. When reset is de-asserted, the DVM IP module expects that all specified DVM agents in the system are active. This is reflected in this readable, writable ACTIVE\_VECTOR register. If, due to low power mode, or other reasons, a DVM agent in the system is shut down, the DVM IP module needs to be made aware of this. To do this, the bit position of the DVM agent in this ACTIVE\_VECTOR register should be set to 0.

On seeing a 0 for an agent in the ACTIVE\_VECTOR, the DVM IP module ensures that no snoops are sent to it, and does not wait for snoop responses or completions from this agent.

The maximum number of DVM agents supported is 256. The entire vector can hence take from one to four 64-bit registers. Unused bits within the 64-bit registers are tied to 0.

Each agent, or host, in the system is assigned a corresponding bridge ID, based on the bridge it is connected to. The active vector register is based on this bridge ID.

* ACTIVE\_VECTOR\_0 contains the vector for agents with bridge ID 0 to 63.
* ACTIVE\_VECTOR\_1 contains the vector for agents with bridge ID 64 to 127.
* ACTIVE\_VECTOR\_2 contains the vector for agents with bridge ID 128 to 191.
* ACTIVE\_VECTOR\_3 contains the vector for agents with bridge ID 192 to 255.

For example, if the host at bridge ID=2 is being shut down, bit 2 of ACTIVE\_VECTOR\_0 should be set to 0. If the host at bridge ID=68 is being shut down, bit 4 of ACTIVE\_VECTOR\_1 should be set to 0.

Note that bits in this register should only be changed when there is no outstanding DVM traffic to the DVM IP module, or from the DVM IP module.

Attribute: R

Security: Non-secure

Bit field description:

* **ACTIVE\_VECTOR**[63:0] - Vector for agents with bridge ID 0 to 63.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| ACTIVE\_VECTOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACTIVE\_VECTOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table DVM\_ACTIVE\_VECTOR register.

### DVM\_AGENT\_DISABLE\_STATUS

This status register is used to indicate that a change to the active agent vector has taken effect. When an agent is removed from the active agent vector, snoops targeting that master may still be queued or inflight. While snoops are outstanding, the master must continue to operate to satisfy snoop requests.

This register indicates that all snoops to the disabled agents have completed and no new snoops will be issued to those agents.

The status indicates whether any snoops outstanding at the time of the last modification of the DVM active vector register(s) have all completed or not. A value of 1 indicates that the snoops have all completed and it is safe to power off or otherwise disable the masters that were removed from the active vector. A value of 0 indicates that snoops are still outstanding and the master must continue to operate and accept snoops.

Attribute: R

Security: Non-secure

Bit field description:

* **stat**[0] -   
  1'b1: A value of 1 indicates snoops have completed for disabled agents  
  1'b0: A value of 0 indicates snoops are still oustanding

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | stat |

Table DVM\_AGENT\_DISABLE\_STATUS register.

### DVM\_FAULT\_LOG

FAULT\_LOG\_0/1/2/3: Register logging DVM agents returning snoop responses.According to the AMBA spec on Distributed Virtual Memory transations, when an agent in the system receives a DVM transaction, it must respond in one of two ways.

1. If it can perform the requested action, it must respond with CRRESP = 0b00000.
2. If it is unable to perform the requested action, it must respond with CRRESP = 0b00010.

The FAULT\_LOG register logs which of the agents responded with CRRESP=0b00010. It is a sticky register, so once set, it remains set until cleared by writing 0 to that bit.The maximum number of DVM agents supported is 256. The entire fault log can hence take from one to four 64-bit registers. Unused bits within the 64-bit registers are tied to 0.Each agent, or host, in the system is assigned a corresponding bridge ID, based on the bridge it is connected to. The fault log register is based on this bridge ID.

* FAULT\_LOG\_0 contains the fault log for agents with bridge ID 0 to 63.
* FAULT\_LOG\_1 contains the fault log for agents with bridge ID 64 to 127.
* FAULT\_LOG\_2 contains the fault log for agents with bridge ID 128 to 191.
* FAULT\_LOG\_3 contains the fault log for agents with bridge ID 192 to 255.

For example, if the host at bridge ID=2 returns CRRESP=0b00010 for a transaction, bit 2 of FAULT\_LOG\_0 will be set to 1. If the host at bridge ID=68 returns CRRESP=0b00010 for a transaction, bit 4 of FAULT\_LOG\_1 will be set to 1.

Attribute: WZC

Security: Non-secure

Bit field description:

* **FAULT\_LOG**[63:0] - Fault log for agents with bridge ID 0 to 63.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| FAULT\_LOG | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FAULT\_LOG | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table DVM\_FAULT\_LOG register.

### DVM\_STS

This register indicates the "busy" or non-idle state of the internal logic within the DVM IP module. When all traffic has been serviced, and no new transactions are received, the bitwise-AND of these bits will be 0, indicating a state of quiescence.

Attribute: R

Security: Non-secure

Bit field description:

* **A**[12] - Snoop arbiter not idle
* **O**[11] - Sync tracker returning sync response to originating agent
* **C**[10] - Sync tracker waiting for completions
* **S**[9] - Sync tracker sending snoops
* **D7**[8] - DRT entry 7 not empty
* **D6**[7] - DRT entry 6 not empty
* **D5**[6] - DRT entry 5 not empty
* **D4**[5] - DRT entry 4 not empty
* **D3**[4] - DRT entry 3 not empty
* **D2**[3] - DRT entry 2 not empty
* **D1**[2] - DRT entry 1 not empty
* **D0**[1] - DRT entry 0 not empty
* **F**[0] - Input FIFO not empty

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | A | O | C | S | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | F |

Table DVM\_STS register.

## CCC Registers

### CCC\_ACTIVE\_VECTOR\_0

This register specifies which of the ACE masters are currently active and capable of receiving snoop requests. This register contains a bit vector where each bit corresponds to an ACE master bridge. The Bridge ID determines which bit corresponds to each bridge.

This vector can be used to disable snoop-only ACE agents, or to power off an ACE agent that may still have stale directory content. Note that ACE CPUs will often leave stale directory content.

A value of 1 indicates that the ACE agent is enabled and capable of accepting snoops. A value of 0 indicates that the ACE agent is disabled and snoops should not be sent. The default for this register is all ACE agents are active.

Attribute: R

Security: Non-secure

Bit field description:

* **ACTIVE\_AGENT\_VECTOR**[63:0] - Active agent vector

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| ACTIVE\_AGENT\_VECTOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACTIVE\_AGENT\_VECTOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_ACTIVE\_VECTOR\_0 register.

### CCC\_AGENT\_DISABLE\_STATUS

This status register is used to indicate that a change to the active agent vector has taken effect. When an agent is removed from the active agent vector, snoops targeting that master may still be queued or inflight. While snoops are outstanding, the master must continue to operate to satisfy snoop requests.

This register indicates that all snoops to the disabled agents have completed and no new snoops will be issued to those agents. The status indicates whether any snoops outstanding at the time of the last modification of the ccc\_active\_vector register(s) have all completed or not. A value of 1 indicates that the snoops have all completed and it is safe to power off or otherwise disable the masters that were removed from the active vector. A value of 0 indicates that snoops are still outstanding and the master must continue to operate and accept snoops.

Attribute: R

Security: Non-secure

Bit field description:

* **stat**[0] -   
  1'b1: Snoops have completed for disabled agents.  
  1'b0: Snoops are still outstanding.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | stat |

Table CCC\_AGENT\_DISABLE\_STATUS register.

### CCC\_CRT\_STATUS

This register tracks the current state of the Coherency Request Trackers. These are the state-machines that track coherent request from the time they are started to the time they are completed. This can be used to determine if there are requests that have made it to the CCC but are not done being processed.

A bit value of 1 indicates a request is still in flight. There are 32 CRTs in a CCC. The reset value for this register is 0, and that is the value under idle conditions where no request are outstanding.

Attribute: R

Security: Non-secure

Bit field description:

* **CRT\_STATUS\_63\_0**[63:0] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| CRT\_STATUS\_63\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CRT\_STATUS\_63\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_CRT\_STATUS register.

### CCC\_DIRECTORY\_INV

This is a control register that can be used to invalidate the entire directory. Setting this register will kick off a hardware engine that will block normal coherent traffic and invalidate all entries of the directory.

The register also acts as a status register. When the control bit is set, it triggers the hardware state machine. The value of that register will stay high until the state machine completes. At that point, it will automatically transition to 0. Since coherent traffic will be blocked until the invalidation sequence has completed, it is not always necessary to check the status of this register.

Writing a value of 1 will trigger the invalidation engine, and it will transition to 0 when completed. All other bits are unused, and the default value is 0.

This register requires secure access, since it invalidates directory content, causing incoherency for secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **VLD**[0] -   
  1'b1: Writing a value of 1 will trigger the invalidation engine, and it will transition to 0 when completed.  
  1'b0: (default)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | VLD |

Table CCC\_DIRECTORY\_INV register.

### CCC\_ECC\_CONTROL

This control register can be used to control ECC correction and detection, if the IP is configured to have ECC. If no ECC is present, the control register doesn't do anything. A value of 1 in bit 0 disables ECC.

A value of 0 in bit 0 indicates ECC is enabled when available. Default value is enabled. Bits 1 and 2 convert data check bits to poison, for CCCM and CCCS respectively. Bits 3 and 4 disable data checking for CCCM andCCCS respectively.

Attribute: RW

Security: Non-secure

Bit field description:

* **DIS**[0] -   
  1'b1: Disable ECC.  
  1'b0: ECC is enabled (default).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | u | u | u | u | DIS |

Table CCC\_ECC\_CONTROL register.

### CCC\_ECC\_INFO

This register monitors ECC errors and saves information for potential debug support. The register holds two kinds of information. It keeps a count of all ECC errors that have been detected in a 16 bit ECC counter.

It also tracks at least one RAM location where an ECC event has occurred. The first ECC even to occur will update the SB or DB bits and the index. The SB stands for single-bit, and connotes a correctable error. DB is double-bit, and applies to all uncorrectable error cases, which include double-bit corruption or more. The register also tracks which of the two RAMs the error was detected in.

While the first error of either kind will be added to the register, and the first uncorrectable error will overwrite any status that was caused by an earlier single-bit event. This is because the uncorrectable error is typically fatal and the status is more important.

This register is readable and writeable, with default value of 0.

Attribute: RW

Security: Non-secure

Bit field description:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_ECC\_INFO register.

### CCC\_EVENT\_COUNTER\_MASK

This register is used to program the event counter. Each bit of this register enables the performance counter to increment if the event occurs. A value of 1 for a bit indicates that this event should be counted.

If multiple bits are set to 1, the logic will only count if all of the corresponding events occur on the same cycle. This allows for combinations of events, such as a directory miss for a Shared line request. The events that can be counted are all related to directory accesses and occur in the same cycle of the pipeline, so they can be combined easily.

When an event satisfies all of the requirements, the ccc\_event\_counter\_value register will be updated.

A value of 1 indicates that the event is selected for counting. A value of 0 the event is not selected. By default, this register will be set to 0 for all bits, indicating no event counting should occur.

Attribute: RW

Security: Non-secure

Bit field description:

* **E13**[13] - Cache maintenance request being processed in LUP2
* **E12**[12] - Directory evict request being processed in LUP2
* **E11**[11] - Directory hit and may need snoops in LUP2
* **E10**[10] - Request missed in directory in LUP2 cycle
* **E9**[9] - Request caused eviction in LUP2 cycle
* **E8**[8] - Request for copyback in LUP2 cycle
* **E7**[7] - Request for unique line in LUP2 cycle
* **E6**[6] - Request for shared line in LUP2 cycle
* **E5**[5] - Able to insert evicted entry into directory in LUP2 cycle
* **E4**[4] - Locked CleanUnique gets OKAY in LUP2 cycle
* **E3**[3] - Locked CleanUnique gets EXOKAY in LUP2 cycle
* **E2**[2] - Non-canceled directory lookup in LUP2 cycle
* **E1**[1] - Canceled directory lookup in LUP2 cycle
* **E0**[0] - Valid directory lookup in LUP2 cycle

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |

Table CCC\_EVENT\_COUNTER\_MASK register.

### CCC\_EVENT\_COUNTER\_VALUE

This register holds the current event count. As selected events occur, the count will increase. When this register rolls over, it can generate an interrupt if the interrupt mask is set correctly.

The register can be read or written through register access. By writing the register, a counter can be clear. It can also be set to a value to force an overflow earlier, to create an interrupt when desired.

Attribute: RW

Security: Non-secure

Bit field description:

* **VALUE**[31:0] - Event counter count value

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VALUE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_EVENT\_COUNTER\_VALUE register.

### CCC\_FLOP\_STRUCT\_PARITY\_ERROR\_INJECTION

This register enables the injection of flop structure parity errors into every structure which can detect them. This register is written by sw and should be cleared by software. This register requires secure access, since it can force unrecoverable errors.

Attribute: RW

Security: Secure access only

Bit field description:

* **INJ23**[23] -   
  1'b1: Directory control evict buffer parity error injection
* **INJ22**[22] -   
  1'b1: Directory control update buffer parity error injection
* **INJ21**[21] -   
  1'b1: Read queue bypass fifo parity error injection
* **INJ20**[20] -   
  1'b1: Write unique queue crtid fifo parity error injection
* **INJ19**[19] -   
  1'b1: Write unique queue fifo parity error injection
* **INJ18**[18] -   
  1'b1: Writeback queue data info fifo parity error injection
* **INJ17**[17] -   
  1'b1: Writeback queue input queue fifo parity error injection
* **INJ16**[16] -   
  1'b1: Snoop response srfifo parity error injection
* **INJ15**[15] -   
  1'b1: W ch output queue prefetch fifo parity error injection
* **INJ14**[14] -   
  1'b1: W ch output snoop response queue fifo parity error injection
* **INJ13**[13] -   
  1'b1: W ch output queue writeback fifo parity error injection
* **INJ12**[12] -   
  1'b1: Snoop response queue fifo parity error injection
* **INJ11**[11] -   
  1'b1: R ch output queue spec fifo parity error injection
* **INJ10**[10] -   
  1'b1: R ch output queue nonspec fifo parity error injection
* **INJ9**[9] -   
  1'b1: B ch output queue fifo parity error injection
* **INJ8**[8] -   
  1'b1: Datapath prefetch byen storage parity error injection
* **INJ7**[7] -   
  1'b1: Datapath prefetch ctl entry parity error injection
* **INJ6**[6] -   
  1'b1: Datapath idfs rdptr fifo parity error injection
* **INJ5**[5] -   
  1'b1: Datapath idfs prf\_ctrl fifo parity error injection
* **INJ4**[4] -   
  1'b1: Datapath idfs snoop resp fifo parity error injection
* **INJ3**[3] -   
  1'b1: Datapath idfs snoop resp entry parity error injection
* **INJ2**[2] -   
  1'b1: Datapath idfs wu fifo parity error injection
* **INJ1**[1] -   
  1'b1: Datapath idfs wb fifo parity error injection
* **INJ0**[0] -   
  1'b1: Datapath idfs wb entry parity error injection

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | INJ23 | INJ22 | INJ21 | INJ20 | INJ19 | INJ18 | INJ17 | INJ16 | INJ15 | INJ14 | INJ13 | INJ12 | INJ11 | INJ10 | INJ9 | INJ8 | INJ7 | INJ6 | INJ5 | INJ4 | INJ3 | INJ2 | INJ1 | INJ0 |

Table CCC\_FLOP\_STRUCT\_PARITY\_ERROR\_INJECTION register.

### CCC\_FLOP\_STRUCT\_PARITY\_FAULT\_ISOLATION

This register indicates which flop structure detected a parity error. It only has meaning if the flop structure parity err bit (bit 7) is set in ccc\_interrupt\_err register. This register is written by hw and should be cleared by software.

Attribute: WZC

Security: Non-secure

Bit field description:

* **E23**[23] -   
  1'b1: Directory control evict buffer parity error
* **E22**[22] -   
  1'b1: Directory control update buffer parity error
* **E21**[21] -   
  1'b1: Read queue bypass fifo parity error
* **E20**[20] -   
  1'b1: Write unique queue crtid fifo parity error
* **E19**[19] -   
  1'b1: Write unique queue fifo parity error
* **E18**[18] -   
  1'b1: Writeback queue data info fifo parity error
* **E17**[17] -   
  1'b1: Writeback queue input queue fifo parity error
* **E16**[16] -   
  1'b1: Snoop response srfifo parity error
* **E15**[15] -   
  1'b1: W ch output queue prefetch fifo parity error
* **E14**[14] -   
  1'b1: W ch output snoop response queue fifo parity error
* **E13**[13] -   
  1'b1: W ch output queue writeback fifo parity error
* **E12**[12] -   
  1'b1: Snoop response queue fifo parity error
* **E11**[11] -   
  1'b1: R ch output queue spec fifo parity error
* **E10**[10] -   
  1'b1: R ch output queue nonspec fifo parity error
* **E9**[9] -   
  1'b1: B ch output queue fifo parity error
* **E8**[8] -   
  1'b1: Datapath prefetch byen storage parity error
* **E7**[7] -   
  1'b1: Datapath prefetch ctl entry parity error
* **E6**[6] -   
  1'b1: Datapath idfs rdptr fifo parity error
* **E5**[5] -   
  1'b1: Datapath idfs prf\_ctrl fifo parity error
* **E4**[4] -   
  1'b1: Datapath idfs snoop resp fifo parity error
* **E3**[3] -   
  1'b1: Datapath idfs snoop resp entry parity error
* **E2**[2] -   
  1'b1: Datapath idfs wu fifo parity error
* **E1**[1] -   
  1'b1: Datapath idfs wb fifo parity error
* **E0**[0] -   
  1'b1: Datapath idfs wb entry parity error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 | E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |

Table CCC\_FLOP\_STRUCT\_PARITY\_FAULT\_ISOLATION register.

### CCC\_HASH\_BYPASS

This control register allows the directory to avoid index hashing. The directory is designed with two different indexing mechanisms. For way 0, the index bits specified in the configuration are used directly from the address. For way 1, additional address bits XORed with the index bits. This hashing is used to reduce the probability of conflicts within the directory.

The hash bypass register will skip the hashing that is performed for way 1. This can allow a more straightforward debug. It can also be used with the indirect access control registers to enable a more intuitive access of the RAM array.

A change of this bypass will change the lookup mechanism of the RAM. Since it changes the location where addresses can reside, this bypass should only be used when the directory is invalid or during debug operations. If changed during normal operation, coherency will likely be violated.

A value of 0 indicates hashing should be used, which is also the default. A value of 1 indicates hashing should be ignored during debug operations.

This register requires secure access, since it disables the hash function, which can cause incoherency for secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **DIS**[0] -   
  1'b1: Hashing should be ignored.  
  1'b0: Hashing should be used (default).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | DIS |

Table CCC\_HASH\_BYPASS register

### CCC\_HASH\_SELECT

This control register specifies which hashing function the directory should use if hashing is enabled in the ccc\_hash\_bypass register. The directory is designed with two different hashing mechanisms. If bit 0 is set to 0, the original hashing function will be used: the tag bits are kept in their original order before being used in the hash. If bit 0 is set to 1, the tag bits will be reversed before being used in the hashing function. Having two separate hashing functions reduces the probability of conflicts within the directory.

This register requires secure access, since it controls the hash function, which can cause incoherency for secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **DIS**[0] -   
  1'b1: The reversed tag bits should be used in the hash.  
  1'b0: The unmodified tag bits should be used in the hash (default).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | DIS |

Table CCC\_HASH\_SELECT register.

### CCC\_INDIRECT\_ACCESS\_TRIG

This registers is the indirect access trigger. Indirect access is a mechanism that allows register-based access to the directory RAM. This can be used for testing RAM bits or reading content on an error condition.

The indirect access is based on a content+trigger mechanism. For writes, the content register is written first to accumulate the data that should be written. Once the content is ready, the trigger register is used to kick off the hardware write mechanism. For reads, the trigger register kicks off a read, and provides data by placing the result into the content registers where it can be accessed.

Each of the indirect access commands can be issued during normal operation, but the Write commands can have side-effects that break coherency functionality. The Read Raw is not disruptive, and the Read-Modify-Write can be performed atomically so single-bit errors can be introduced while maintaining functionality.

The indirect access trigger registers is readable and writeable. To trigger the RAM access, this register must be written. Reads will not have side-effects and will only return the current value of the trigger register.

The trigger register has a number of fields that must be set correctly. The CMD field indicates which kind of indirect access to perform. The WAY field indicates whether RAM 0 or 1 should be accessed. The RAM index indicates the entry to access within the RAM. The RAM index is a pure index, which will avoid any hashing function for WAY 1.

This register requires secure access, since it can modify directory content, causing incoherency for secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **INDEX**[13:3] -
* **WAY**[2] -
* **CMD**[1:0] -   
  00: Read Raw data. When triggered, a read to the directory RAM will be performed and the resulting data, without ECC correction, will be copied into the content register.  
  01: Write Raw Data. When triggered, the content register values will be written into the directory RAM. This will include the ECC bits if present.  
  10: Write Data with Generated ECC. When triggered, this will write to the RAM entry. The content register will be used to specify the data to be written. However, if ECC hardware is present, the ECC bits will be generated based on the data instead of coming from the content register. This allows a directory entry to be written with correct ECC value without needing to calculate it first.   
  11: Read-Modify-Write. This command will perform a specific kind of read-modify-write operation on a directory entry. It will read the content of the directory, XOR that content with the indirect content register, and write the combined value into the same directory entry. This can be used to introduce single or double bit errors into the directory to test error detection and handling. The content register will not be modified during this operation, so it can be used to introduce errors into multiple lines.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | INDEX | | | | | | | | | | | WAY | CMD | |

Table CCC\_INDIRECT\_ACCESS\_TRIG register.

### CCC\_INDIRECT\_RAM\_CONT

This is the indirect access RAM content register. Its use is conjunction with the indirect access trigger register. On an indirect read, data is written to this register. On an indirect write, content from this register is written into the RAM. On a read-modify-write, content from this register is used for the XOR function.

Since the RAM data width may be larger than 64 bits, multiple registers are used to hold the data. Any bits beyond the data width are unused.

This register requires secure access, since it can modify directory content, causing incoherency for secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **RAM\_CONTENTS\_63\_0**[63:0] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| RAM\_CONTENTS\_63\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RAM\_CONTENTS\_63\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_INDIRECT\_RAM\_CONT register.

### CCC\_INTERRUPT\_ERR

This interrupt is a status register that tracks the interrupt generating events. This includes multi-bit ECC error, single-bit ECC error, and event counter overflow. When these events occur, this register is updated and will hold the bit value until cleared. It can be cleared by writing to the register. To allow per-bit clearing control, the write value should use a value of 1 when it doesn't want to make a change, or a value of 0 when it wants to clear.

Attribute: WZC

Security: Non-secure

Bit field description:

* **E7**[7] -   
  1'b1: Flop Structure parity error interrupt
* **E6**[6] -   
  1'b1: Prefetch buffer data parity error interrupt
* **E5**[5] -   
  1'b1: Data parity error interrupt
* **E4**[4] -   
  1'b1: Register parity error interrupt
* **E3**[3] -   
  1'b1: Snoop Response interrupt status
* **E2**[2] -   
  1'b1: Event counter overflow interrupt status
* **E1**[1] -   
  1'b1: Single-bit error interrupt status
* **E0**[0] -   
  1'b1: Multi-bit error interrupt status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |

Table CCC\_INTERRUPT\_ERR register.

### CCC\_INTERRUPT\_FATAL\_MASK

This register determines which errors in ccc\_interrupt\_err should be considered fatal. 1 is fatal and 0 is non-fatal.

Attribute: RW

Security: Non-secure

Bit field description:

* **E7**[7] -   
  1'b1: Flop structure parity error interrupt
* **E6**[6] -   
  1'b1: Prefetch buffer data parity error interrupt
* **E5**[5] -   
  1'b1: Data parity error interrupt
* **E4**[4] -   
  1'b1: Register parity error interrupt
* **E3**[3] -   
  1'b1: Snoop Response interrupt status
* **E2**[2] -   
  1'b1: Event counter overflow interrupt status
* **E1**[1] -   
  1'b1: Single-bit error interrupt status
* **E0**[0] -   
  1'b1: Multi-bit error interrupt status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |

Table CCC\_INTERRUPT\_FATAL\_MASK register.

### CCC\_INTERRUPT\_MASK

This register is used for determining what kind of events can trigger an interrupt from the CCC.

A bit value of 1 indicates that the event will not send an interrupt. A bit value of 0 means the event will cause an interrupt. The default value is 3'b110, which means only the Multi-bit error case will send an interrupt, as it is a fatal error.

Attribute: RW

Security: Non-secure

Bit field description:

* **M7**[7] -   
  1'b1: Flop Structure parity error interrupt enable
* **M6**[6] -   
  1'b1: Prefetch buffer data parity error interrupt enable
* **M5**[5] -   
  1'b1: Data parity error interrupt enable
* **M4**[4] -   
  1'b1: Register parity error interrupt enable
* **M3**[3] -   
  1'b1: Snoop Response Error interrupt enable
* **M2**[2] -   
  1'b1: Event counter overflow interrupt enable
* **M1**[1] -   
  1'b1: Single-bit error interrupt enable
* **M0**[0] -   
  1'b1: Multi-bit error interrupt enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | M7 | M6 | M5 | M4 | M3 | M2 | M1 | M0 |

Table CCC\_INTERRUPT\_MASK register.

### CCC\_LLC\_CONTROL

When the CCC is connected to a Last Level Cache, it has some unique programmable features that allows the two to interact more closely. This register allows programmable control of these features.

The CM bit is the enable for the Cache Maintenance Operation propagation. A Cache Maintenance Operation flushes or invalidates a cache line from the coherency domain. This can allow interaction with non-coherent devices that access the slave directly. Since a Last Level Cache can hold additional data, it may be necessary to flush or invalidate lines from the LLC to memory. When this register bit is set, the CCC will propagate any Cache Maintenance Operations to the LLC, where the cache can take the specified action. When this bit is set to zero, the Cache Maintenance Operations will not be propagated.

The WE bit is the enable for Write Evict propagation. In ACE protocol rev E, the WriteEvict command was created in order to write clean data to a downstream cache, such as the Last Level Cache. This allows the cache to only allocate a line when it is dropped by one of the ACE masters, allowing a better utilization of RAM storage. When the WE bit is set to 1, the WriteEvict will be propagated from the CCC to the LLC. If set to 0, the WriteEvict will drop the data and only update the CCC directory to indicate that the master has given up its copy of the line.

This register requires secure access, since it can change the behavior of cache maintenance operations, causing incoherency for secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **CM**[1] -   
  1'b1: Cache maintenance operations will be propagated from CCC.  
  1'b0: Cache maintenance operations will not propagate past CCC.
* **WE**[0] -   
  1'b1: WriteEvict will be forwarded to LLC.  
  1'b0: WriteEvict will not be forwarded.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CM | WE |

Table CCC\_LLC\_CONTROL register.

### CCC\_QOS\_OVERRIDES

This control register allows the QoS field of requests sent from the CCC to be overridden. By default, read requests sent from the CCC use the same QoS value as the read request sent to the CCC that triggered the read. For writes, the QoS value is dependent on the cause of the writes. If the write on the master port is triggered by a WriteUnique, WriteBack, WriteClean, or WriteEvict, the QoS field is determined by the originating write. Writes are also generated by the CCC when a snoop response with dirty data is returned for a read request. The default value of the QoS is the QoS value of the request that triggered the snoop. And finally, if the directory needs an eviction due to storage conflict, the directory eviction needs a QoS value.

The control register controls each of these four QoS values.For AR requests, Write requests, and Snoop responses, there is an override field and an enable to determine whether to use the default value or the override value. For Directory Evict, the DirEvt QoS value in this register always determine the QoS value of the generated write request, with a default value of 0.

This QoS override register is unrelated to the master port bridge QoS override, which if enabled can override any of these values.

Attribute: RW

Security: Non-secure

Bit field description:

* **DE\_QOS**[18:15] - QoS field for writes triggered by directory evictions.
* **EN2**[14] - Enable to override the QoS for snoop responses.
* **RN\_QOS**[13:10] - Override value for QoS for writes from snoop responses.
* **EN1**[9] - Enable to override qos for writes sent to CCC.
* **WR\_QOS**[8:5] - Override value for all writes sent to CCC.
* **EN0**[4] - Enable to override AR QoS field.
* **RD\_QOS**[3:0] - Override value for any Read request sent from CCC.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | DE\_QOS | | | | EN2 | RN\_QOS | | | | EN1 | WR\_QOS | | | | EN0 | RD\_QOS | | | |

Table CCC\_QOS\_OVERRIDES register.

### CCC\_SHDW\_CRT\_INFO

This register reports the info stored at the CRT entry number specified by the ccc\_shdw\_crt\_prf\_select register.

Attribute: R

Security: Secure access only

Bit field description:

* **VLD**[63] -   
  1'b1: The CRT entry is valid - currently in use.  
  1'b0: The CRT entry is invalid - currently not in use.
* **SEC**[62] - The secure value stored in the selected CRT entry.
* **SNOOP**[61:56] - The snoop value stored in the selected CRT entry.
* **ORIGID**[55:48] - The origid value stored in the selected CRT entry.
* **ADDR**[47:0] - Bits 52:5 of the address value stored in the selected CRT entry.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| VLD | SEC | SNOOP | | | | | | ORIGID | | | | | | | | ADDR | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_SHDW\_CRT\_INFO register.

### CCC\_SHDW\_CRT\_PRF\_SELECT

This register selects which CRT entry and prefetch buffer entry is selected to be read upon access to the ccc\_shdw\_crt\_info register and ccc\_shdw\_prf\_info register respectively.

Attribute: RW

Security: Secure access only

Bit field description:

* **PI**[16] -   
  1'b1: Allows reading of contents of invalid entries in prefetch buffer, and it must be set to read aid\_ptr field regardless of valid state.  
  1'b0: Disallows reading of contents of invalid entries in prefetch buffer - returned data is zeroed out when entry is invalid, and aid\_ptr is zeroed out even if entry is valid.
* **INDEX**[8:0] - This specfies the index into the CRT and prefetch buffer arrays.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | PI | u | | | | | | | INDEX | | | | | | | | |

Table CCC\_SHDW\_CRT\_PRF\_SELECT register.

### CCC\_SHDW\_PIPE\_STATUS

This register reports incoming request activity to the coherency pipeline and resource availability for fulfilling those requests. Insight into pipeline stalls may be gleaned by polling this register and looking for chronically active request sources that do not have resource availability.

Attribute: R

Security: Secure access only

Bit field description:

* **QREQS**[40:32] - Vector showing qualified request status to the CCC from various sources. These are active requests quailified by resource availability. Only when all required resources for a particular request source are 1 will the filtered request status be 1 (if the unfiltered request status is 1).

[8] rply  
[7] snprply  
[6] evt  
[5] extsnp  
[4] chis  
[3] deq  
[2] wuq  
[1] wbq  
[0] rdq

* **REQS**[24:16] - Vector showing request status to the CCC from various sources.

[8] rply  
[7] snprply  
[6] evt  
[5] extsnp  
[4] chis  
[3] deq  
[2] wuq  
[1] wbq  
[0] rdq

* **RSRCS**[11:0] - Vector showing the availability status of the various resources of the CCC.

[11] woq\_wb  
[10] roq\_spec  
  [9] snpq  
  [8] evt\_for\_rpl  
  [7] evt  
  [6] updt  
  [5] crt\_snp  
  [4] crt\_evt  
  [3] deq\_crt  
  [2] wb\_crt  
  [1] crt  
  [0] dctl\_ram NOT busy

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | QREQS | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | REQS | | | | | | | | | u | | | | RSRCS | | | | | | | | | | | |

Table CCC\_SHDW\_PIPE\_STATUS register.

### CCC\_SHDW\_PRF\_INFO

This register reports the info stored at the prefetch buffer entry number specified by the ccc\_shdw\_crt\_prf\_select register.

Attribute: R

Security: Secure access only

Bit field description:

* **DATA**[63:0] - Data from the selected PRF entry. The vld bit is always returned, but the rest of the bits will be zeroed out unless either vld == 1 or the PI bit of the ccc\_shdw\_crt\_prf\_select register is 1.

[63] vld - entry is valid  
[62:39] req\_aid (lowest 24 bits)  
[38] request\_half0  
[37] request\_nonstr\_atomic  
[36] request\_nodata  
[35] request\_onedata  
[34] passdirty  
[33:32] need\_rdrsp  
[31] chi\_readspec  
[30:28] vud  
[27] req\_chi\_snptype  
[26] req\_chi\_type  
[25:24] req\_resp  
[23:21] req\_chi\_resp  
[20] req\_fasttap  
[19] req\_tracetag  
[18] req\_addr4  
[17] req\_addr5  
[16:8] aid\_ptr - returns 0 unless PI bit set regardless of vld state  
[7:0] req\_mstrid

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| DATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_SHDW\_PRF\_INFO register.

### CCC\_SPARE

This register contains writable spare bits. Software may read and write these bits. They retain any value written to them (between reset events), but otherwise they have no functional effect.

Attribute: RW

Security: Non-secure

Bit field description:

* **BITS**[31:0] - Spare Bits

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BITS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_SPARE register.

### CCC\_SPEC\_FETCH

This register controls the speculative fetch behavior of the Cache Coherency Controller. Speculative fetch means that the CCC will issue reads to memory or next level of cache before the directory lookup is performed or the snoops are sent. This allows a lower latency for these reads in the case of a directory miss. If snoops need to be sent for this request, data may be retrieved from a caching agent. This means the speculative fetch may be increasing memory bandwidth in case where it wouldn't have needed to send a read.

This set of registers specifies a bit vector where each bit corresponds to a master agent on the NoC. When requests from this agent arrive at the CCC, a comparison with the bit vector will determine whether speculative fetch is currently enabled for that agent.

The bit vector is sized based on the number of masters in the system. The bit position of each agent is determined by the agent's bridge ID. Unused bits are not included in the register and cannot be written.

A value of 1 means speculative fetch is enabled for that agent. A value of 0 means speculative fetch is disabled for that agent. The initial values are determined by the bridge property cc\_axi4m\_speculative\_fetch.

Attribute: RW

Security: Non-secure

Bit field description:

* **SPEC\_FETCH\_VECTOR**[63:0] - Speculative fetch vector

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| SPEC\_FETCH\_VECTOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPEC\_FETCH\_VECTOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CCC\_SPEC\_FETCH register.

## LLC Host Registers

### LLC\_ALLOC\_ARCACHE\_EN

This register holds one bit for each of the LLC Allocation Classes. If the bit is marked as one, this indicates that read allocation for that LLC Allocation Class is controlled by the ARCACHE bits. If marked as zero, it means the allocation will be controlled by the llc\_alloc\_rd\_en register. The default of this register is configured within NocStudio.

Attribute: RW

Security: Non-secure

Bit field description:

* **ARCACHE\_Allocation\_Enable**[7:0] - Read allocation for that LLC Allocation Class is controlled by the ARCACHE bits/llc\_alloc\_rd\_en register.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | ARCACHE\_Allocation\_Enable | | | | | | | |

Table LLC\_ALLOC\_ARCACHE\_EN register.

### LLC\_ALLOC\_AWCACHE\_EN

This register holds one bit for each of the LLC Allocation Classes. If the bit is marked as one, this indicates that write allocation for that LLC Allocation Class is controlled by the AWCACHE bits. If marked as zero, it means the allocation will be controlled by the llc\_alloc\_wr\_en register. The default of this register is configured within NocStudio.

Attribute: RW

Security: Non-secure

Bit field description:

* **AWCACHE\_Allocation\_Enable**[7:0] - Write allocation for that LLC Allocation Class is controlled by the AWCACHE bits/llc\_alloc\_wr\_en register.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | AWCACHE\_Allocation\_Enable | | | | | | | |

Table LLC\_ALLOC\_AWCACHE\_EN register.

### LLC\_ALLOC\_RD\_EN

This register holds one bit for each of the LLC Allocation Classes. The use of this register is controlled by the llc\_alloc\_arcache\_en register, which indicates whether ARCACHE bits should be used for allocation, or whether this register should decide on allocation. If this register is used for an LLC Allocation Class, a value of one will indicate that reads should allocate into the LLC. A value of zero indicates that reads should not allocate.

Attribute: RW

Security: Non-secure

Bit field description:

* **Read\_Allocation\_Enable**[7:0] - Reads should/should not allocate into the LLC

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | Read\_Allocation\_Enable | | | | | | | |

Table LLC\_ALLOC\_RD\_EN register.

### LLC\_ALLOC\_WR\_EN

This register holds one bit for each of the LLC Allocation Classes. The use of this register is controlled by the llc\_alloc\_awcache\_en register, which indicates whether AWCACHE bits should be used for allocation, or whether this register should decide on allocation. If this register is used for an LLC Allocation Class, a value of one will indicate that writes should allocate into the LLC. A value of zero indicates that writes should not allocate.

Attribute: RW

Security: Non-secure

Bit field description:

* **Write\_Allocation\_Enable**[7:0] - Writes should/should not allocate into the LLC.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | Write\_Allocation\_Enable | | | | | | | |

Table LLC\_ALLOC\_WR\_EN register.

### LLC\_CACHE\_WAY\_ENABLE

This register indicates whether a way is enabled for cache access. If enabled, a cache lookup will read the associated Tag values and perform an address comparison. If disabled, the Tags won't be accessed and the contents of the Tags won't be compared.

This allows the Tags to be powered down or the lines to be used for RAM access. The register has one bit per way, allowing each way to be individually enabled or disabled. A value of 1 indicates that the way is enabled, while a value of 0 indicates it is disabled. All ways are enabled by default. Before disabling a cache way, the way must be disabled in the llc\_global\_alloc register, and the contents should be flushed.

This register requires secure access, since it disables ways of the cache, potentially modifying the contents of secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Cache\_Way\_Enable**[31:0] - Cache Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Cache\_Way\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_CACHE\_WAY\_ENABLE register.

### LLC\_CLASS\_ALLOC

The class allocation control registers are used to specify which associative ways can be written to. Each master in the system belongs to an LLC class, and each class allocation control register indicates which ways that class of agents can allocate into.

These registers can be used to provide dedicated associativity for different agents or groups of agents. The default value of these registers indicates that all ways are accessible by all agents, with a value of one indicating allocation is allowed. Setting the value to zero will disable allocation for an agent.

It is permissible to turn off allocation for all ways, which will prevent any accesses from that class from allocating into the cache.

Note that the llc\_global\_alloc register can override these values. If global allocation is disabled for a way, none of the agents can allocate into those ways regardless of what the llc\_class\_allocate registers indicate.

Attribute: RW

Security: Non-secure

Bit field description:

* **WGE\_0**[7:0] - Class 0 Allocation Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | WGE\_0 | | | | | | | |

Table LLC\_CLASS\_ALLOC register.

### LLC\_DATA\_INV\_CTL

This register controls a state machine that can invalidate the contents of data array banks. Each way has a corresponding bit in this register. When the register is written, the invalidation engine will kick off and invalidate the data for each of the specified ways. Writing a value of 1 to a bit indicates that the corresponding way should be invalidated. Writing a value of 0 to a bit indicates that the content of that way shouldn't be invalidated.

The register can be read to determine the current status of the data bank invalidation sequence. When hardware has completed the invalidation sequence for a way, it will change the value of that register bit from 1 to 0. If the entire register has a value of 0, then the invalidation engine has completed. The reset value of this register is zero.

Invalidation of the data array is needed when switching between cache mode and scratchpad RAM mode, since the RAM mode allows direct access to the data. Any secure data that was stored in the cache may be visible to RAM mode accesses unless it is invalidated first. Similarly, if security permissions are removed for the Scratchpad RAM, the prior contents should be invalidated before removing the security check.

An invalidation sequence should be completed before a second sequence is requested.

This register requires secure access, since it invalidates data array, potentially modifying the contents of secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Data\_Bank\_Invalidation\_Enable**[31:0] - Data Bank Invalidation Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data\_Bank\_Invalidation\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_DATA\_INV\_CTL register.

### LLC\_ECC\_DATA\_ADDR

This is a status register that tracks the address of ECC errors that occur in the Data ram. If only one of either the SB bit or the DB bit is set in the ecc\_data\_info register, this address contains the address for that error. If both the SB and the DB ecc error bits are set in the ecc\_data\_info register, this register contains the address of the DB error.

Attribute: RW

Security: Non-secure

Bit field description:

* **Address**[59:0] - Address of first detected error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | Address | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_ECC\_DATA\_ADDR register.

### LLC\_ECC\_DATA\_INFO

This is a status register that tracks ECC errors that occur in the Data array. The register will track the number of ECC errors, as well as whether single-bit or double-bit errors have been detected. If the SB bit is set, at least one single bit error has been detected. If the DB bit is set, at least one double-bit error has been detected.

Additionally, the register tracks information about the first error detected. It stores the index of the tag array that had the error, as well as the way group. It also tracks which half of the cache line failed, which is needed to identify the sub-bank that failed. If a double-bit error occurs after a single-bit error has already been recorded, the double-bit error will overwrite the content of the register. This is because double-bit errors are fatal, and the information about how a fatal error is more important that the information about a non-fatal error.

The register can be read for status, but can also be written. If the SB and DB bit are written with zeros, the sampling of the first detected error will happen as described above.

Attribute: RW

Security: Non-secure

Bit field description:

* **Index**[45:32] - index of first detected error
* **ECC\_Count**[31:16] - number of ECC errors found
* **hlf**[9] - Which half of cache line reported error
* **way**[8:2] - Way group of first detected error
* **db**[1] - Detected double or multi bit error
* **sb**[0] - Detected single bit error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | Index | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC\_Count | | | | | | | | | | | | | | | | u | | | | | | hlf | way | | | | | | | db | sb |

Table LLC\_ECC\_DATA\_INFO register.

### LLC\_ECC\_DISABLE

This register allows ECC to be disabled for either the Data arrays or the Tag arrays. These are independently controlled. A bit value of 1 indicates that ECC is disabled. A bit value of 0 indicates ECC is enabled, if present. The register value resets to value 0, meaning ECC is enabled.

Attribute: RW

Security: Non-secure

Bit field description:

* **D**[1] - Disable Data ECC Check/Correct
* **T**[0] - Disable Tag ECC Check/Correct

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | D | T |

Table LLC\_ECC\_DISABLE register.

### LLC\_ECC\_TAG\_ADDR

This is a status register that tracks the address of ECC errors that occur in the Tag ram. If only one of either the SB bit or the DB bit is set in the ecc\_tag\_info register, this address contains the address for that error. If both the SB and the DB ecc error bits are set in the ecc\_tag\_info register, this register contains the address of the DB error.

Attribute: RW

Security: Non-secure

Bit field description:

* **Address**[59:0] - Address of first detected error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | Address | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_ECC\_TAG\_ADDR register.

### LLC\_ECC\_TAG\_INFO

This is a status register that tracks ECC errors that occur in the Tag array. The register will track the number of ECC errors, as well as whether single-bit or double-bit errors have been detected. If the SB bit is set, at least one single bit error has been detected. If the DB bit is set, at least one double-bit error has been detected.

Additionally, the register tracks information about the first error detected. It stores the index of the tag array that had the error, as well as the way group. If a double-bit error occurs after a single-bit error has already been recorded, the double-bit error will overwrite the content of the register. This is because double-bit errors are fatal, and the information about how a fatal error is more important that the information about a non-fatal error.

The register can be read for status, but can also be written. If the SB and DB bit are written with zeros, the sampling of the first detected error will happen as described above.

Attribute: RW

Security: Non-secure

Bit field description:

* **Index**[45:32] - Index of first detected error
* **ECC\_Count**[31:16] - Number of ECC errors found
* **way**[6:2] - Way group of first detected error
* **db**[1] - Detected double or multi bit error
* **sb**[0] - Detected single bit error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | Index | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC\_Count | | | | | | | | | | | | | | | | u | | | | | | | | | way | | | | | db | sb |

Table LLC\_ECC\_TAG\_INFO register.

### LLC\_EVENT\_COUNTER\_0

This register is the first of two event counters. When its event counter mask control enables certain events to be counted, they will increment this counter. When the counter overflows, in can produce an interrupt if the interrupt mask is enabled. This can be used to trap to software after a number of specified events has occurred.

The counter can be read or written. Writing the value can initialize the counter to a larger value which can speed up the point at which counter will overflow and the interrupt will be triggered.

Attribute: RW

Security: Non-secure

Bit field description:

* **Event\_Counter\_Value\_0**[31:0] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Event\_Counter\_Value\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_EVENT\_COUNTER\_0 register.

### LLC\_EVENT\_COUNTER\_1

This register is the first of two event counters. When its event counter mask control enables certain events to be counted, they will increment this counter. When the counter overflows, in can produce an interrupt if the interrupt mask is enabled. This can be used to trap to software after a number of specified events has occurred.

The counter can be read or written. Writing the value can initialize the counter to a larger value which can speed up the point at which counter will overflow and the interrupt will be triggered.

Attribute: RW

Security: Non-secure

Bit field description:

* **Event\_Counter\_Value\_1**[31:0] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Event\_Counter\_Value\_1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_EVENT\_COUNTER\_1 register.

### LLC\_EVENT\_COUNTER\_MASK\_0

This register is used to program its corresponding event counter. Each bit of this register enables the performance counter to increment if the event occurs. A value of 1 for a bit indicates that this event should be counted.

If multiple bits are set to 1, the logic will only count if all of the corresponding events occur on the same cycle. This allows for combinations of events, such as a cache miss that causes an eviction. The events that can be counted are all related to cache accesses and occur in the same cycle of the pipeline, so they can be combined easily.

When an event satisfies all of the requirements, the llc\_event\_counter\_0 register will be updated. A value of 1 indicates that the event is selected for counting. A value of 0 the event is not selected. By default, this register will be set to 0 for all bits, indicating no event counting should occur.

Attribute: RW

Security: Non-secure

Bit field description:

* **e27**[27] - Allocate conflict
* **e26**[26] - Indx after indx tmp conflict
* **e25**[25] - Wr after alloc conflict
* **e24**[24] - Wr after evict conflict
* **e23**[23] - Rd after rd alloc conflict
* **e22**[22] - Rd alloc after wr conflict
* **e21**[21] - Rd after evict
* **e20**[20] - Tag update
* **e19**[19] - Request CleanUnique
* **e18**[18] - External snoop
* **e17**[17] - Excl wr fails store conditional
* **e16**[16] - Exclusive write
* **e15**[15] - Atomic op
* **e14**[14] - Write through
* **e13**[13] - Rd dealloc on dirty
* **e12**[12] - Rd dealloc
* **e11**[11] - Partial line wr
* **e10**[10] - Fetch due to partial wr
* **e9**[9] - Retry access
* **e8**[8] - Retry needed
* **e7**[7] - Eviction
* **e6**[6] - Cache maint op
* **e5**[5] - Partial write
* **e4**[4] - Cache miss
* **e3**[3] - Cache hit
* **e2**[2] - Scratchpad access
* **e1**[1] - Cache write
* **e0**[0] - Cache read

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | e27 | e26 | e25 | e24 | e23 | e22 | e21 | e20 | e19 | e18 | e17 | e16 | e15 | e14 | e13 | e12 | e11 | e10 | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table LLC\_EVENT\_COUNTER\_MASK\_0 register.

### LLC\_EVENT\_COUNTER\_MASK\_1

This register is used to program its corresponding event counter. Each bit of this register enables the performance counter to increment if the event occurs. A value of 1 for a bit indicates that this event should be counted.

If multiple bits are set to 1, the logic will only count if all of the corresponding events occur on the same cycle. This allows for combinations of events, such as a cache miss that causes an eviction. The events that can be counted are all related to cache accesses and occur in the same cycle of the pipeline, so they can be combined easily.

When an event satisfies all of the requirements, the llc\_event\_counter\_1 register will be updated. A value of 1 indicates that the event is selected for counting. A value of 0 the event is not selected. By default, this register will be set to 0 for all bits, indicating no event counting should occur.

Attribute: RW

Security: Non-secure

Bit field description:

* **e27**[27] - Allocate conflict
* **e26**[26] - Indx after indx tmp conflict
* **e25**[25] - Wr after alloc conflict
* **e24**[24] - Wr after evict conflict
* **e23**[23] - Rd after rd alloc conflict
* **e22**[22] - Rd alloc after wr conflict
* **e21**[21] - Rd after evict
* **e20**[20] - Tag update
* **e19**[19] - Request CleanUnique
* **e18**[18] - External snoop
* **e17**[17] - Excl wr fails store conditional
* **e16**[16] - Exclusive write
* **e15**[15] - Atomic op
* **e14**[14] - Write through
* **e13**[13] - Rd dealloc on dirty
* **e12**[12] - Rd dealloc
* **e11**[11] - Partial line wr
* **e10**[10] - Fetch due to partial wr
* **e9**[9] - Retry access
* **e8**[8] - Retry needed
* **e7**[7] - Eviction
* **e6**[6] - Cache maint op
* **e5**[5] - Partial write
* **e4**[4] - Cache miss
* **e3**[3] - Cache hit
* **e2**[2] - Scratchpad access
* **e1**[1] - Cache write
* **e0**[0] - Cache read

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | e27 | e26 | e25 | e24 | e23 | e22 | e21 | e20 | e19 | e18 | e17 | e16 | e15 | e14 | e13 | e12 | e11 | e10 | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table LLC\_EVENT\_COUNTER\_MASK\_1 register.

### LLC\_FLUSH\_CNTRL

This register controls the automated flush capability of the LLC. Two types of automated flush are available. Bit 0 can be set by SW to request a WBINV style flush. When this flush is complete, the way allocation registers will be returned to their original values following the flush, and the ways will be left enabled as well. Bit 1 can be set to request a power-down style flush. When this flush completes it will leave all ways de-allocated and disabled. Both bits will stay set until their respective flush completes, when the respective bit will be cleared by hardware. Only 1 bit should ever be set at a time.

Attribute: RW

Security: Secure access only

Bit field description:

* **pwrdn\_flsh\_en**[1] -   
  1'b1: When set to 1 by SW, HW should perform an automated power-down-style flush.   
  1'b0: When set to 0 by HW, the automated power-down-style flush is complete.
* **wbinv\_flsh\_en**[0] -   
  1'b1: When set to 1 by SW, HW should perform an automated writeback-invalidate flush.   
  1'b0: When set to 0 by HW, the automated wbinv flush is complete.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | pwrdn\_flsh\_en | wbinv\_flsh\_en |

Table LLC\_FLUSH\_CNTRL register.

### LLC\_FORCE\_PARTIAL\_WR\_ALLOC

This register holds one bit for each of the LLC Allocation Classes. When set to 0, all partial write misses will follow normal write allocation rules (llc\_alloc\_wr\_en or awcache). When set to 1, all partial write misses will be forced to allocate (fetch first followed by merge). All partial write hits will be merged into the cache regardless of the configuration.

Attribute: RW

Security: Non-secure

Bit field description:

* **Force\_Partial\_Write\_Allocation**[7:0] - Force\_Partial\_Write\_Allocation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | Force\_Partial\_Write\_Allocation | | | | | | | |

Table LLC\_FORCE\_PARTIAL\_WR\_ALLOC register.

### LLC\_GLOBAL\_ALLOC

This register controls whether lines can be allocated into a way by any agent.

If a way is disabled from allocation in this register, no agents can allocate even if the llc\_class\_allocate registers are set. This register is used as part of a sequence to remove ways from use by the cache for either Scratchpad RAM usage, or for power gating. By removing allocation ability, a flush engine can remove the existing contents of the line without fear that new entries will be added during or after the flush.

The default value of this register enables allocation for all ways of the cache, and so each bit corresponding to a way is set to 1. To disable allocation, the bits should be set to 0.

Attribute: RW

Security: Non-secure

Bit field description:

* **Global\_Allocation\_Way\_Enable**[31:0] - Global Allocation Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Global\_Allocation\_Way\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_GLOBAL\_ALLOC register.

### LLC\_INDIRECT\_RAM\_CONT

This is the indirect access RAM content register. It is used in conjunction with the indirect access trigger register. On an indirect read, data is written to this register. On an indirect write, content from this register is written into the RAM. On a read-modify-write, content from this register is used for the XOR function.

Since the RAM data width may be larger than 64 bits, multiple registers are used to hold the data. Any bits beyond the data width are unused.

This register requires secure access, since it can be used to modify or observe the contents of data.

Attribute: RW

Security: Secure access only

Bit field description:

* **RAM\_content**[63:0] –

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| RAM\_content | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RAM\_content | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_INDIRECT\_RAM\_CONT register.

### LLC\_INDIRECT\_TRIGGER

This register is the indirect access trigger. Indirect access is a mechanism that allows register-based access to the RAM arrays. This can be used for testing RAM bits or reading content on an error condition.

The indirect access is based on a content+trigger mechanism. For writes, the content register is written first to accumulate the data that should be written. Once the content is ready, the trigger register is used to kick off the hardware write mechanism. For reads, the trigger register kicks off a read, and provides data by placing the result into the content registers where it can be accessed.

The indirect access supports 4 sub-commands.

1. Read Raw data. When triggered, a read to the RAM array will be performed and the resulting data, without ECC correction, will be copied into the content register.
2. Write Raw Data. When triggered, the content register values will be written into the RAM. This will include the ECC bits if present.
3. Write Data with Generated ECC. When triggered, this will write to the RAM entry. The content register will be used to specify the data to be written. However, if ECC hardware is present, the ECC bits will be generated based on the data instead of coming from the content register. This allows the RAM entry to be written with correct ECC value without needing to calculate it first.
4. Read-Modify-Write. This command will perform a specific kind of read-modify-write operation on a RAM entry. It will read the content of the RAM, XOR that content with the indirect content register, and write the combined value into the same RAM entry. This can be used to introduce single or double bit errors into the directory to test error detection and handling. The content register will not be modified during this operation, so it can be used to introduce errors into multiple lines.

Each of the indirect access commands can be issued during normal operation, but the Write commands can have side-effects that break coherency functionality. The Read Raw is not disruptive, and the Read-Modify-Write can be performed atomically so single-bit errors can be introduced while maintaining functionality.The indirect access trigger registers is readable and writeable. To trigger the RAM access, this register must be written. Reads will not have side-effects and will only return the current value of the trigger register.

The trigger register has a number of fields that must be set correctly. The CMD field indicates which kind of indirect access to perform. The WAY field indicates which way group to access. The TYP field indicates whether the Data array or Tag array should be accessed. If the Data array is accessed, the hlf bit indicates which sub-bank is accessed. The RAM index indicates the entry to access within the RAM.

This register requires secure access, since it can be used to modify or observe the contents of data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Index**[31:11] - Index of RAM to access
* **way**[10:9] - Way position in the waygroup
* **hlf**[8] - Which half of cache line reported error
* **waygroup**[7:3] - Way group of first detected error
* **typ**[2] -   
  0: Tag array access  
  1: Data array access
* **cmd**[1:0] -   
  00: Read raw array content including any ECC bits and copy to RAM Content register  
  01: Write RAM content register directly into array  
  10: Write RAM content minus ECC bits to array, use ECC generation logic to set ECC bits in array  
  11: Read-modify-write. Read array content, XOR with RAM content register, and write modified data into array

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Index | | | | | | | | | | | | | | | | | | | | | way | | hlf | waygroup | | | | | typ | cmd | |

Table LLC\_INDIRECT\_TRIGGER register.

### LLC\_INTERRUPT\_ERR

This is a status register that tracks the interrupt generating events. This includes multi-bit ECC error, single-bit ECC error, RAM mode disallowed accesses, and event counter overflow. When these events occur, this register is updated and will hold the bit value until cleared. It can be cleared by writing to the register. To allow per-bit clearing control, the write value should use a value of 1 when it doesn't want to make a change, or a bit value of 0 when it wants to clear.

Attribute: WZC

Security: Non-secure

Bit field description:

* **e9**[9] - Flush Control Register Programming Error
* **e8**[8] - Data Parity Error
* **e7**[7] - Register Parity Error
* **e6**[6] - Event Counter Overflow
* **e5**[5] - Scratchpad Security Failure status
* **e4**[4] - Scratchpad RAM Disabled status
* **e3**[3] - Data ECC Double Bit Error status
* **e2**[2] - Data ECC Single Bit Error status
* **e1**[1] - Tag ECC Double Bit Error status
* **e0**[0] - Tag ECC Single Bit Error status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table LLC\_INTERRUPT\_ERR register.

### LLC\_INTERRUPT\_FATAL\_MASK

This register determines which errors in llc\_interrupt\_err should be considered fatal. 1 is fatal and 0 is non-fatal.

Attribute: RW

Security: Non-secure

Bit field description:

* **e9**[9] - Flush Control Register Programming Error
* **e8**[8] - Data Parity Error
* **e7**[7] - Register Parity Error
* **e6**[6] - Event Counter Overflow
* **e5**[5] - Scratchpad Security Failure status
* **e4**[4] - Scratchpad RAM Disabled status
* **e3**[3] - Data ECC Double Bit Error status
* **e2**[2] - Data ECC Single Bit Error status
* **e1**[1] - Tag ECC Double Bit Error status
* **e0**[0] - Tag ECC Single Bit Error status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table LLC\_INTERRUPT\_FATAL\_MASK register.

### LLC\_INTERRUPT\_MASK

This register is used for determining what kind of events can trigger an interrupt from the LLC.

A bit value of 1 indicates that the event will not send an interrupt. A bit value of 0 means the event will cause an interrupt. The default values are listed in the bit field description.

Attribute: RW

Security: Non-secure

Bit field description:

* **m9**[9] -   
  1'b1: Flush control reg programming error interrupt disabled.  
  1'b0: Flush control reg programming error interrupt enabled (default).
* **m8**[8] -   
  1'b1: Data parity error interrupt disabled.  
  1'b0: Data parity error interrupt enabled (default).
* **m7**[7] -   
  1'b1: Register parity error interrupt disabled.  
  1'b0: Register parity error interrupt enabled (default).
* **m6**[6] -   
  1'b1: Event Counter Overflow interrupt disabled (default).  
  1'b0: Event Counter Overflow interrupt enabled.
* **m5**[5] -   
  1'b1: Scratchpad Security Check Failure interrupt disabled.  
  1'b0: Scratchpad Security Check Failure interrupt enabled (default).
* **m4**[4] -   
  1'b1: Scratchpad RAM Access while Disabled interrupt disabled.  
  1'b0: Scratchpad RAM Access while Disabled interrupt enabled (default).
* **m3**[3] -   
  1'b1: Data ECC Double Bit Error interrupt disabled.  
  1'b0: Data ECC Double Bit Error interrupt enabled (default).
* **m2**[2] -   
  1'b1: Data ECC Single Bit Error interrupt disabled (default).  
  1'b0: Data ECC Single Bit Error interrupt enabled.
* **m1**[1] -   
  1'b1: Tag ECC Double Bit Error interrupt disabled.  
  1'b0: Tag ECC Double Bit Error interrupt enabled (default).
* **m0**[0] -   
  1'b1: Tag ECC Single Bit Error interrupt disabled (default).  
  1'b0: Tag ECC Single Bit Error interrupt enabled.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | m9 | m8 | m7 | m6 | m5 | m4 | m3 | m2 | m1 | m0 |

Table LLC\_INTERRUPT\_MASK register.

### LLC\_OPERATION\_CONTROL

This register is used to change the behavior of the LLC in various ways, as listed in the description for each bit. The default values are listed in the bit field description.

Attribute: RW

Security: Non-secure

Bit field description:

* **FAS**[1] -   
  1'b1: Force address serialization--only 1 operation can issue to memory for an address.   
  1'b0: Do not force address serialization. Default value.
* **LRU**[0] -   
  1'b1: Enable LRU updates on a cache hit.  
  1'b0: Disable LRU updates on a cache hit.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FAS | LRU |

Table LLC\_OPERATION\_CONTROL register.

### LLC\_RAM\_ADDRESS\_BASE

This register indicates the system address offset of the RAM mode. The address range should always be allocated as the full size of the LLC capacity rounded up to a power of 2, and the address offset must be programmed to a naturally aligned address for that size. The default value of this register is the address range base specified during NoC construction.

This register requires secure access, since it controls the address range for RAM storage, implicitly changing the contents of that range.

Attribute: RW

Security: Secure access only

Bit field description:

* **Address**[63:0] - Address

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 148 LLC\_RAM\_ADDRESS\_BASE register.

### LLC\_RAM\_WAY\_ENABLE

This register is used to enable cache ways to be used as a Scratchpad RAM instead of a cache. The register indicates which of the ways should be used as a RAM instead of a cache. It is possible to use some of the LLC as a cache, and some as a RAM, by selecting which ways are used by each. By default, this register is set to 0 so that all ways are used as cache. To set this register, the lines must be removed from cache usage by the llc\_cache\_way\_enable register. Any cache contents should be flushed before enabling the RAM mode.

This register requires secure access, since it enables RAM storage, allowing previous content of the cache to be visible to scratchpad reads.

Attribute: RW

Security: Secure access only

Bit field description:

* **Scratchpad\_Ram\_Way\_Group\_Enable**[31:0] - Scratchpad Ram Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Scratchpad\_Ram\_Way\_Group\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 149 LLC\_RAM\_WAY\_ENABLE register.

### LLC\_RAM\_WAY\_SECURE

This register allows the Scratchpad RAM to have trust-zone security checking. Each way can be individually controlled. If the security bit is set, only secure accesses (those with AxPROT[1] set to secure) can access that address. Non-secure accesses will be responded to with an error, and an interrupt will be triggered if the interrupt is enabled.

A value of 1 indicates secure accesses are required, while a value of 0 indicates no security check is needed (secure or non-secure accesses are enabled). By default, this register is 0, so no security checking occurs.

This register requires secure access, since it controls the security check for scratchpad RAM storage.

Attribute: RW

Security: Secure access only

Bit field description:

* **Scratchpad\_Ram\_Way\_Group\_Security**[31:0] - Scratchpad Ram Way Security

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Scratchpad\_Ram\_Way\_Group\_Security | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 150 LLC\_RAM\_WAY\_SECURE register.

### LLC\_READ\_DEALLOCATE

This register holds one bit for each of the LLC Allocation Classes. When set to 1, the read hits for the LLC class will result in the cacheline being invalidated. Dirty cachelines will either be discarded or flushed depending on the setting in the LLC\_READ\_DISCARD\_DIRTY register. When set to 0, normal cache read behavior will apply.

Attribute: RW

Security: Non-secure

Bit field description:

* **LLC\_READ\_DEALLOCATE**[7:0] - Per Allocation Class Read-And-Invalidate Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | LLC\_READ\_DEALLOCATE | | | | | | | |

Table LLC\_READ\_DEALLOCATE register.

### LLC\_READ\_DISCARD\_DIRTY

This register holds one bit for each of the LLC Allocation Classes. When set to 1, and the corresponding bit (class) in LLC\_READ\_DEALLOCATE is also set to 1, read hits on a dirty line will be discarded without write back.

Attribute: RW

Security: Non-secure

Bit field description:

* **LLC\_READ\_DISCARD\_DIRTY**[7:0] - Per Allocation Class Control For Dirty Line When Read-And-Invalidate (Read Deallocate)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | LLC\_READ\_DISCARD\_DIRTY | | | | | | | |

Table 152 LLC\_READ\_DISCARD\_DIRTY register.

### LLC\_SHDW\_PIPE\_STATUS

This register reports incoming request activity and resource availability for fulfilling those requests. Insight into pipeline stalls may be gleaned by polling this register and looking for chronically active request sources that do not have resource availability.

Attribute: R

Security: Non-secure

Bit field description:

* **QREQS**[41:32] - Vector showing qualified request status to the LLC from various sources. These are active requests qualified by resource availability. Only when all required resources for a particular request source are 1 will the filtered request status be 1 (if the unfiltered request status is 1).

[9] evt  
[8] extsnp  
[7] tagupdate  
[6] flush\_req  
[5] rd\_sprt1  
[4] rd\_sprt0  
[3] retry\_rd  
[2] wr\_sprt1  
[1] wr\_sprt0  
[0] retry\_wr

* **REQS**[25:16] - Vector showing request status to the LLC from various sources.

[9] evt  
[8] extsnp  
[7] tagupdate  
[6] flush\_req  
[5] rd\_sprt1  
[4] rd\_sprt0  
[3] retry\_rd  
[2] wr\_sprt1  
[1] wr\_sprt0  
[0] retry\_wr

* **PS**[14] - TAC pipeline stall signal status. All incoming requests to lookup arbiter are held off while this signal is active, regardless of resource availability.
* **RSRCS**[13:0] - Vector showing the availability status of the various resources of the LLC.

[13] acem\_ar  
[12] chim\_ar  
[11] chim\_aww  
[10] mprt\_aww  
  [9] mprt\_ar  
  [8] missq\_wr  
  [7] missq\_rd  
  [6] dac\_cmd  
  [5] dac\_wctl  
  [4] tac  
  [3] retry  
  [2] wrsp  
  [1] evict  
  [0] atomics

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | QREQS | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | REQS | | | | | | | | | | u | PS | RSRCS | | | | | | | | | | | | | |

Table LLC\_SHDW\_PIPE\_STATUS register.

### LLC\_SPARE

This register contains writable spare bits. Software may read and write these bits. They retain any value written to them (between reset events), but otherwise they have no functional effect.

Attribute: RW

Security: Non-secure

Bit field description:

* **BITS**[31:0] - Spare Bits

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BITS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_SPARE register.

### LLC\_TAG\_INV\_CTL

The Tag Invalidation Control register triggers a state machine that will invalidate the contents of one or more way groups of the cache. The register has one bit per way group, and the bit vector written into this register will invalidate the corresponding way groups. A value of 1 will indicate that the corresponding way group should be invalidated. A value of 0 will indicate that the way group should not be invalidated. This per-way group control allows portions of the cache to be powered down and restarted later, with the ability to reset just the way groups that were powered down and powered back on.

A write to the register will kick off the invalidation engine, invalidating the specified way groups. A read of the register will indicate whether the invalidation is in progress. When the invalidation engine has completed, the bit vector will transition to a value of zero. So a read value of zero will indicate that the state machine has completed. The reset value of this register is zero. An invalidation sequence should be completed before a second sequence is requested.

This register requires secure access, since it invalidates tags, potentially modifying the contents of secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Tag\_Way\_Group\_Invalidation\_Enable**[31:0] - Tag Way Group Invalidation Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Tag\_Way\_Group\_Invalidation\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_TAG\_INV\_CTL register.

### LLC\_WAY\_FLUSH

This register controls a state machine that flushes specified ways of the cache. The intent of this engine is to remove all content from the specified ways, pushing any dirty data that may exist to memory. It also invalidate clean lines. The Way Flush engine should be run while the llc\_cache\_way\_enable is still on for those ways so the contents are still accessible, but the llc\_global\_alloc register should have disabled the way for allocation. This ensure that as lines are removed from the cache, they won't unintentionally get added again. Clean lines are invalidated to ensure that dirty line writes do not write into the ways being flushed.

Writing the register will kick off the flush engine. If the write value specifies a bit value of 1, then that way group will be flushed. If the bit value written is zero, that way will not be flushed. When the sequence is completed, hardware will transition the bit values to zero. A register value of 0 indicates the state machine has complete. The default value for this register is zero.

A flush sequence should be completed before a second sequence is requested.

This register requires secure access, since it flushes cache lines.

Attribute: RW

Security: Secure access only

Bit field description:

* **Way\_Flush\_Control**[31:0] - Way Flush Control

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Way\_Flush\_Control | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table LLC\_WAY\_FLUSH register.

### LLC\_WRITETHROUGH\_EN

This register holds one bit for each of the LLC Allocation Classes. When set to 1, the write accesses that allocate will be write-through. When set to 0, write accesses will be write-back.

Attribute: RW

Security: Non-secure

Bit field description:

* **LLC\_WRITETHROUGH\_EN**[7:0] - Write should be write-through

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | LLC\_WRITETHROUGH\_EN | | | | | | | |

Table LLC\_WRITETHROUGH\_EN register.

## OCP Master Bridge Registers

### OCPM\_ADDR\_BASE

These registers specify the base addresses and masks of different slave ranges accessible from this master. One base, mask, and reloc register set per address range assigned to the master. These registers can be individually designated as read-only or read-write based on NocStudio property assigned to address ranges.

Even if the register is read-only, the range can be disabled using the appropriate bits described below which are always programmable.A slave address range is specified using the above base address and mask pair. An address on the AR or AW channel has a match against a range if it satisfies the equation

AxADDRS & OCPM\_ADDR\_MASK[i] == OCPM\_ADDR\_BASE[i]

Note that programmed 'base' must already factor the 'mask'. The base should not have a 1'b1 bit where the corresponding mask bit is 1'b0. What this means is that the programmed base should already have performed a bit-wise AND operation with the 'mask'. An address which doesn't match any range results in a decode error response. Note that programming of these registers must ensure that an address matches only against one range. Match against multiple ranges is a fatal error and will raise an interrupt.

Address ranges are specified at 64B cache line boundary. Lower six bits if OCPM\_ADDR\_BASE and OCPM\_ADDR\_MASK are used for specifying access permissions on an address range.

OCPM\_ADDR\_BASE[5:0]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **5** | **4** | **3** | **2** | **1** | **0** |
| X | DI | R/Wn | I | NS | P |

OCPM\_ADDR\_MASK[5:0]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **5** | **4** | **3** | **2** | **1** | **0** |
| X | X | Valid | I | NS | P |

Bits [2:0] act as value and mask for checking against AxPROT of an incoming command. A command is allowed access to a range if

AxPROT & OCPM\_ADDR\_MASK[2:0] == OCPM\_ADDR\_BASE[2:0] & OCPM\_ADDR\_MASK[2:0]

If the above check fails, then the command is denied access to the range and decode error response is returned. The encoding is specified below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OCPM\_ADDR\_BASE[4] Disable** | **OCPM\_\_ADDR\_MASK[3] RD/WRn valid** | **OCPM\_ADDR\_BASE[3] RD/WRn** |  | **Interpretation** |
| 1 | X | X |  | range disabled |
| 0 | 1 | 1 |  | Read only |
| 0 | 1 | 0 |  | Write only |
| 0 | 0 | X |  | read/write |

Attribute: RW

Security: Secure access only

Bit field description:

* **BASE\_ADDRESS**[63:6] - Base address
* **LLC**[5] - LLC disable
* **DI**[4] -   
  1'b1: Address range disabled
* **R\_Wn**[3] -   
  1'b1: Read enabled to range  
  1'b0: Write enabled to range
* **I**[2] -   
  1'b1: Instruction
* **NS**[1] -   
  1'b1: Non-secure
* **P**[0] -   
  1'b1: Privileged

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| BASE\_ADDRESS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BASE\_ADDRESS | | | | | | | | | | | | | | | | | | | | | | | | | | LLC | DI | R\_Wn | I | NS | P |

Table OCPM\_ADDR\_BASE register

### OCPM\_ADDR\_MASK

See OCPM\_ADDR\_BASE.

Attribute: RW

Security: Secure access only

Bit field description:

* **MASK**[63:6] - Mask
* **KO**[5] - Keep out/reject read and/or write accesses
* **TM**[4] -   
  1'b1: Enable Trusted Master behavior for secure transactions
* **VAL**[3] -   
  1'b1: R\_Wn field is valid
* **I**[2] -   
  1'b1: Instruction field is valid
* **NS**[1] -   
  1'b1: Non-secure field is valid
* **P**[0] -   
  1'b1: Privileged field is valid

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| MASK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MASK | | | | | | | | | | | | | | | | | | | | | | | | | | KO | TM | VAL | I | NS | P |

Table OCPM\_ADDR\_MASK register

### OCPM\_AUTOWAKE\_POWER\_DOMAIN

Configures the master bridge's support for autowake of power domains.

When set, master bridge halts a request and issues wakeup requests for power domains that need to power up to complete the transaction. The power domains should support auto wake. When reset, master bridge issues DECERR for any transaction which has dependent power domains in sleep state.

Attribute: RW

Security: Non-secure

Bit field description:

* **AW**[0] -   
  1'b1: Autowake enabled  
  1'b0: Autowake disabled

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | AW |

Table OCPM\_AUTOWAKE\_POWER\_DOMAIN register

### OCPM\_BRIDGE\_ID

Unique identifier assigned to the master bridge.

Attribute: R

Security: Non-secure

Bit field description:

* **ID**[15:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | ID | | | | | | | | | | | | | | | |

Table OCPM\_BRIDGE\_ID register

### OCPM\_CHECK\_OUTSTANDING\_REQ\_TO\_SLAVEID

This register is used to check if there are any outstanding read/write commands to a slave specified by field slvid. NocStudio provides a table of slvids corresponding to the slave ports accessible from a master bridge. Outstanding status is reflected in OCPM\_EVENT\_STATUS.

Attribute: RW

Security: Non-secure

Bit field description:

* **SLVID**[15:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | SLVID | | | | | | | | | | | | | | | |

Table OCPM\_CHECK\_OUTSTANDING\_REQ\_TO\_SLAVEID register

### OCPM\_CLK\_GATING\_HYSTERESIS\_COUNT

Programmable interval used by coarse clock gating logic in master bridge.This interval is used to generate heart beat pulses using noc\_clk on that bridge. These heart beat pulses are broadcast to each local clock gating domain within the bridge where they are synchronized to the CG domain's clock. Four consecutive heart beat pulses in the CG domain is used as the inactivity/idle interval to initiate coarse clock gating of the CG domain.

Attribute: RW

Security: Non-secure

Bit field description:

* **HYSTERESIS\_COUNTER**[31:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HYSTERESIS\_COUNTER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table OCPM\_CLK\_GATING\_HYSTERESIS\_COUNT register

### OCPM\_CLK\_GATING\_OVERRIDE

Fast path override, when set to 1'b1 will cause the clock gating logic to be disabled. 1'b1 will allow activity based clock gating to be performed on the master bridge.

Attribute: RW

Security: Non-secure

Bit field description:

* **FPO**[0] -   
  1'b1: Clock gating enabled  
  1'b0: Clock gating is disabled

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FPO |

Table OCPM\_CLK\_GATING\_OVERRIDE register

### OCPM\_ERROR\_INTERRUPT\_MASK

Interrupt mask register. Individual bit positions match the error bit positions in OCPM\_ERROR\_INTERRUPT\_STATUS. When an INTM bit is set, occurrence of the corresponding error event will not cause an interrupt to be raised. When 1'b0, error event will cause interrupt to be raised.

Attribute: RW

Security: Non-secure

Bit field description:

* **E36**[36] -
* **E35**[35] -
* **E34**[34] -
* **E31**[31] -
* **E30**[30] -
* **E29**[29] -
* **E28**[28] -
* **E25**[25] -
* **E24**[24] -
* **E23**[23] -
* **E22**[22] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | E36 | E35 | E34 | u | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E31 | E30 | E29 | E28 | u | | E25 | E24 | E23 | E22 | u | | | | | | | | | | | | | | | | | | | | | |

Table OCPM\_ERROR\_INTERRUPT\_MASK register

### OCPM\_ERROR\_INTERRUPT\_STATUS

These error status bits record the first error event and have to be cleared by writing a 1'b0 before new errors are recorded. **For errors that are aggregated, such as flop structure parity error, one must clear the ocpm\_log\_flopparity\_error error bits before clearing ocpm\_error\_interrupt\_status**

Attribute: WZC

Security: Non-secure

Bit field description:

* **E36**[36] -   
  1'b1: [FATAL] Flop Structure Parity Err. This error bit is the aggregate of error bits in ocpm\_log\_flopparity\_error, as such, those errors must be cleared before clearning this bit.
* **E35**[35] -   
  1'b1: [FATAL] Parity error has been detected in CSR registers
* **E34**[34] -   
  1'b1: [FATAL] Traffic sent to a noc layer which is power gated
* **E31**[31] -   
  1'b1: [FATAL] Thread 3 unsupported cmd interrupt
* **E30**[30] -   
  1'b1: [FATAL] Thread 2 unsupported cmd interrupt
* **E29**[29] -   
  1'b1: [FATAL] Thread 1 unsupported cmd interrupt
* **E28**[28] -   
  1'b1: [FATAL] Thread 0 unsupported cmd interrupt
* **E25**[25] -   
  1'b1: Thread 3 respone timeout
* **E24**[24] -   
  1'b1: Thread 2 respone timeout
* **E23**[23] -   
  1'b1: Thread 1 respone timeout
* **E22**[22] -   
  1'b1: Thread 0 respone timeout

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | E36 | E35 | E34 | u | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E31 | E30 | E29 | E28 | u | | E25 | E24 | E23 | E22 | u | | | | | | | | | | | | | | | | | | | | | |

Table OCPM\_ERROR\_INTERRUPT\_STATUS register

### OCPM\_EVENT\_STATUS

Attribute: R

Security: Non-secure

Bit field description:

* **T3\_OS**[19] - 1'b1: Thread 3 commands are outstanding to the slave specified in OSSLV register
* **T2\_OS**[18] - 1'b1: Thread 2 commands are outstanding to the slave specified in OSSLV register
* **T1\_OS**[17] - 1'b1: Thread 1 commands are outstanding to the slave specified in OSSLV register
* **T0\_OS**[16] - 1'b1: Thread 0 commands are outstanding to the slave specified in OSSLV register
* **T3\_OE**[11] -   
  1'b1: Thread 3 (if exists): No outstanding transactions in the tag table  
  1'b0: Master bridge tag table has no outstanding transactions for thread 3
* **T2\_OE**[10] -   
  1'b1: Thread 2 (if exists): No outstanding transactions in the tag table  
  1'b0: Master bridge tag table has no outstanding transactions for thread 2
* **T1\_OE**[9] -   
  1'b1: Thread 1 (if exists): No outstanding transactions in the tag table  
  1'b0: Master bridge tag table has no outstanding transactions for thread 1
* **T0\_OE**[8] -   
  1'b1: Thread 0: No outstanding transactions in the tag table  
  1'b0: Master bridge tag table has no outstanding transactions for thread 0
* **T3\_OF**[3] -   
  1'b1: Thread 3 (if exists): Maximum supported number of commands are outstanding waiting for response and no more requests can be accepted  
  1'b0: Master bridge can accept more read requests on thread 3
* **T2\_OF**[2] -   
  1'b1: Thread 2 (if exists): Maximum supported number of commands are outstanding waiting for response and no more requests can be accepted  
  1'b0: Master bridge can accept more read requests on thread 2
* **T1\_OF**[1] -   
  1'b1: Thread 1 (if exists): Maximum supported number of commands are outstanding waiting for response and no more requests can be accepted  
  1'b0: Master bridge can accept more read requests on thread 1
* **T0\_OF**[0] -   
  1'b1: Thread 0: Maximum supported number of commands are outstanding waiting for response and no more requests can be accepted  
  1'b0: Master bridge can accept more read requests on thread 0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | T3\_OS | T2\_OS | T1\_OS | T0\_OS | u | | | | T3\_OE | T2\_OE | T1\_OE | T0\_OE | u | | | | T3\_OF | T2\_OF | T1\_OF | T0\_OF |

Table OCPM\_EVENT\_STATUS register

### OCPM\_LOG\_FLOPPARITY\_ERROR

Error logging register for flop structure parity errors. It has meaning only if the flop structure parity error bit in ocpm\_error\_interrupt\_status is set. Note that some fields of this register (such as RXFIFO\_PARITY\_ERR\_VC) will only reflect the last error that happened. These error bits should be cleared before ocpm\_error\_interrupt\_status flop structure parity error bit.

Attribute: WZC

Security: Non-secure

Bit field description:

* **T3\_TAGTBL\_PARITY\_ERR**[43] - Thread 3 Tag table parity error
* **T2\_TAGTBL\_PARITY\_ERR**[42] - Thread 2 Tag table parity error
* **T1\_TAGTBL\_PARITY\_ERR**[41] - Thread 1 Tag table parity error
* **T0\_TAGTBL\_PARITY\_ERR**[40] - Thread 0 Tag table parity error
* **T3\_PEG\_REQ\_PL\_CMD\_PERR**[31] - Thread 3 Pre-errgen req cmd pipeline parity error
* **T2\_PEG\_REQ\_PL\_CMD\_PERR**[30] - Thread 2 Pre-errgen req cmd pipeline parity error
* **T1\_PEG\_REQ\_PL\_CMD\_PERR**[29] - Thread 1 Pre-errgen req cmd pipeline parity error
* **T0\_PEG\_REQ\_PL\_CMD\_PERR**[28] - Thread 0 Pre-errgen req cmd pipeline parity error
* **T3\_PF\_REQ\_PL\_DATA\_PERR**[27] - Thread 3 Pre-framing req pipeline data parity error
* **T2\_PF\_REQ\_PL\_DATA\_PERR**[26] - Thread 2 Pre-framing req pipeline data parity error
* **T1\_PF\_REQ\_PL\_DATA\_PERR**[25] - Thread 1 Pre-framing req pipeline data parity error
* **T0\_PF\_REQ\_PL\_DATA\_PERR**[24] - Thread 0 Pre-framing req pipeline data parity error
* **T3\_PF\_REQ\_PL\_CMD\_PERR**[23] - Thread 3 Pre-framing req pipeline cmd parity error
* **T2\_PF\_REQ\_PL\_CMD\_PERR**[22] - Thread 2 Pre-framing req pipeline cmd parity error
* **T1\_PF\_REQ\_PL\_CMD\_PERR**[21] - Thread 1 Pre-framing req pipeline cmd parity error
* **T0\_PF\_REQ\_PL\_CMD\_PERR**[20] - Thread 0 Pre-framing req pipeline cmd parity error
* **RXFIFO\_PARITY\_ERR**[16] - Rx Switch Fifo Parity Err
* **RXFIFO\_PARITY\_ERR\_LAYER**[5:2] - Rx Switch Fifo Parity Error Layer
* **RXFIFO\_PARITY\_ERR\_VC**[1:0] - Rx Switch Fifo Parity Error Virtual Channel

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | T3\_TAGTBL\_PARITY\_ERR | T2\_TAGTBL\_PARITY\_ERR | T1\_TAGTBL\_PARITY\_ERR | T0\_TAGTBL\_PARITY\_ERR | u | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T3\_PEG\_REQ\_PL\_CMD\_PERR | T2\_PEG\_REQ\_PL\_CMD\_PERR | T1\_PEG\_REQ\_PL\_CMD\_PERR | T0\_PEG\_REQ\_PL\_CMD\_PERR | T3\_PF\_REQ\_PL\_DATA\_PERR | T2\_PF\_REQ\_PL\_DATA\_PERR | T1\_PF\_REQ\_PL\_DATA\_PERR | T0\_PF\_REQ\_PL\_DATA\_PERR | T3\_PF\_REQ\_PL\_CMD\_PERR | T2\_PF\_REQ\_PL\_CMD\_PERR | T1\_PF\_REQ\_PL\_CMD\_PERR | T0\_PF\_REQ\_PL\_CMD\_PERR | u | | | RXFIFO\_PARITY\_ERR | u | | | | | | | | | | RXFIFO\_PARITY\_ERR\_LAYER | | | | RXFIFO\_PARITY\_ERR\_VC | |

Table OCPM\_LOG\_FLOPPARITY\_ERROR register

### OCPM\_RESPONSE\_TIMEOUT\_CONTROL

This register is used to configure response timeouts.

OCPM\_RESPONSE\_TIMEOUT\_CONTROL[8] (En) needs to be set for timeout tracking to be enabled. When this bit is 1'b0, no timestamps are recorded to generate timeout interrupts. A 64-bit free running counter is used to time the response interval.

OCPM\_RESPONSE\_TIMEOUT\_CONTROL[5:0] (TI) specifies the lower bit index into this counter, from where 2-bits are picked up and recorded as the arrival time stamp of every incoming command. If response for a command does not return before the current time stamp rolls to arrival time stamp minus 1, the response is assumed to have timedout and an interrupt is raised along with the slave ID to which the timed out request was sent.

When changing the TI field, first write to the register with the En field cleared, then write a second time with the TI field to its new value, then a 3rd write to restore the En field to Enabled. During this update while the En field is cleared, existing timers will cancelled, and new timer starts will be inhibited.

Attribute: RW

Security: Non-secure

Bit field description:

* **EN**[8] -   
  1'b1: Enabled timeout tracking, a 64-bit free running counter is used to time the response interval.  
  1'b0: No timestamps are recorded to generate timeout interrupts
* **TI**[5:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | EN | u | | TI | | | | | |

Table OCPM\_RESPONSE\_TIMEOUT\_CONTROL register

### OCPM\_RESPONSE\_TIMEOUT\_SLAVEID

Indicates slave IDs to which a response timeout was detected.

Note that slvid encoding is not same as the bridge ID of the slave. NocStudio provides a table mapping the slvids to the actual slave ports accessible from the master bridge.

Attribute: R

Security: Non-secure

Bit field description:

* **T3\_SLVID**[63:48] - Spare bit(s).
* **T2\_SLVID**[47:32] - Spare bit(s).
* **T1\_SLVID**[31:16] - Spare bit(s).
* **T0\_SLVID**[15:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| T3\_SLVID | | | | | | | | | | | | | | | | T2\_SLVID | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T1\_SLVID | | | | | | | | | | | | | | | | T0\_SLVID | | | | | | | | | | | | | | | |

Table OCPM\_RESPONSE\_TIMEOUT\_SLAVEID register

### OCPM\_ERROR\_INTERRUPT\_SVRTY

Interrupt severity register. Individual bit positions match the error bit positions in OCPM\_ERROR\_INTERRUPT\_STATUS. When an INTS bit is set, occurrence of the corresponding error event will cause fatal interrupt to be raised. When 1'b0, error event will cause a non-fatal interrupt to be raised.

Attribute: RW

Security: Non-secure

Bit field description:

* **E36**[36] -   
  1'b1: Flop Structure Parity Error is fatal..  
  1'b0: Flop Structure Parity Error does factor into interrupt signals.
* **E35**[35] -   
  1'b1: A Parity Error in the CSR regsiters is fatal..  
  1'b0: A Parity Error in the CSR regsiters is non-fatal.
* **E34**[34] -   
  1'b1: Traffic sent to a noc layer which is power gated is fatal..  
  1'b0: Traffic sent to a noc layer which is power gated is non-fatal.
* **E31**[31] -   
  1'b1: Thread 3 unsupported cmd is fatal..  
  1'b0: Thread 3 unsupported cmd is non-fatal.
* **E30**[30] -   
  1'b1: Thread 2 unsupported cmd is fatal..  
  1'b0: Thread 2 unsupported cmd is non-fatal.
* **E29**[29] -   
  1'b1: Thread 1 unsupported cmd is fatal..  
  1'b0: Thread 1 unsupported cmd is non-fatal.
* **E28**[28] -   
  1'b1: Thread 0 unsupported cmd is fatal..  
  1'b0: Thread 0 unsupported cmd is non-fatal.
* **E25**[25] -   
  1'b1: Thread 3 response timeout is fatal..  
  1'b0: Thread 3 response timeout is non-fatal.
* **E24**[24] -   
  1'b1: Thread 2 response timeout is fatal..  
  1'b0: Thread 2 response timeout is non-fatal.
* **E23**[23] -   
  1'b1: Thread 1 response timeout is fatal..  
  1'b0: Thread 1 response timeout is non-fatal.
* **E22**[22] -   
  1'b1: Thread 0 response timeout is fatal..  
  1'b0: Thread 0 response timeout is non-fatal.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | E36 | E35 | E34 | u | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E31 | E30 | E29 | E28 | u | | E25 | E24 | E23 | E22 | u | | | | | | | | | | | | | | | | | | | | | |

Table OCPM\_ERROR\_INTERRUPT\_SVRTY register.

## OCP Slave Bridge Registers

### OCPS\_BRIDGE\_ID

Unique identifier assigned to the slave bridge.

Attribute: R

Security: Non-secure

Bit field description:

* **ID**[15:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| U | | | | | | | | | | | | | | | | ID | | | | | | | | | | | | | | | |

Table OCPS\_BRIDGE\_ID register

### OCPS\_CLK\_GATING\_HYSTERESIS\_COUNT

Programmable intervals used by coarse clock gating logic. This interval is used to generate heart beat pulses using noc\_clk on that bridge. These heart beat pulses are broadcast to each local clock gating domain within the bridge where they are synchronized to the CG domain's clock. Four consecutive heart beat pulses in the CG domain is used as the inactivity/idle interval to initiate coarse clock gating of the CG domain.

Attribute: RW

Security: Non-secure

Bit field description:

* **HYSTERESIS\_COUNTER**[31:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HYSTERESIS\_COUNTER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table OCPS\_CLK\_GATING\_HYSTERESIS\_COUNT register

### OCPS\_CLK\_GATING\_OVERRIDE

Fast path override, when set to 1'b1 will cause the clock gating logic to be disabled. 1'b1 will allow activity based clock gating to be performed on the slave bridge.

Attribute: RW

Security: Non-secure

Bit field description:

* **FPO**[0] -   
  1'b1: Clock gating enabled  
  1'b0: Clock gating is disabled

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FPO |

Table OCPS\_CLK\_GATING\_OVERRIDE register

### OCPS\_ERROR\_INTERRUPT\_MASK

Interrupt mask register.Individual bit positions match the error bit positions in OCPS\_ERROR\_INTERRUPT\_STATUS. When an INTM bit is set, occurrence of the corresponding error event will not cause an interrupt to be raised. When 1'b0, error event will cause interrupt to be raised.

Attribute: RW

Security: Non-secure

Bit field description:

* **E34**[34] -
* **E33**[33] -
* **E32**[32] -
* **E18**[18] –

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | E34 | E33 | E32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | E18 | u | | | | | | | | | | | | | | | | | |

Table OCPS\_ERROR\_INTERRUPT\_MASK register

### OCPS\_ERROR\_INTERRUPT\_STATUS

These error status bits record the first error event and have to be cleared by writing a 1'b0 before new errors are recorded. **For errors that are aggregated, such as flop structure parity error, one must clear the ocps\_log\_flopparity\_error error bits before clearing ocps\_error\_interrupt\_status**

Attribute: WZC

Security: Non-secure

Bit field description:

* **E34**[34] -   
  1'b1: [FATAL] Flop Structure Parity Err. This error bit is the aggregate of error bits in ocps\_log\_flopparity\_error, as such, those errors must be cleared before clearning this bit.
* **E33**[33] - 1'b1: [FATAL] Parity error has been detected in CSR registers
* **E32**[32] - 1'b1: [FATAL] Response traffic sent to a noc layer which is power gated
* **E18**[18] - 1'b1: [FATAL] Unknown response destination: resp routing table lookup produces a unknown destination

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | E34 | E33 | E32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| U | | | | | | | | | | | | | E18 | u | | | | | | | | | | | | | | | | | |

Table OCPS\_ERROR\_INTERRUPT\_STATUS register

### OCPS\_EVENT\_STATUS

Slave bridge status bits.

Attribute: R

Security: Non-secure

Bit field description:

* **ROE**[3] - 1'b1: There are no read commands outstanding to the attached slave device
* **WOE**[2] - 1'b1: There are no write commands outstanding to the attached slave device
* **ROF**[1] -   
  1'b1: Maximum number of supported read commands are outstanding to the attached slave device awaiting response, no more read commands will be issued to slave till responses are received.  
  1'b0: Slave bridge can accept more read commands from the NoC
* **WOF**[0] -   
  1'b1: Maximum number of supported write commands are outstanding to the attached slave device awaiting response, no more write commands will be issued to slave till responses are received.  
  1'b0: Slave device can expect more write commands from NoC

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | ROE | WOE | ROF | WOF |

Table OCPS\_EVENT\_STATUS register

### OCPS\_LOG\_FLOPPARITY\_ERROR

Error logging register for flop structure parity errors. It has meaning only if the flop structure parity error bit in ocps\_error\_interrupt\_status is set. Note that some fields of this register (such as RXFIFO\_PARITY\_ERR\_VC) will only reflect the last error that happened. These error bits should be cleared before ocps\_error\_interrupt\_status flop structure parity error bit.

Attribute: WZC

Security: Non-secure

Bit field description:

* **T3\_TAGTBL\_PARITY\_ERR**[23] - Thread 3 Tag table parity error
* **T2\_TAGTBL\_PARITY\_ERR**[22] - Thread 2 Tag table parity error
* **T1\_TAGTBL\_PARITY\_ERR**[21] - Thread 1 Tag table parity error
* **T0\_TAGTBL\_PARITY\_ERR**[20] - Thread 0 Tag table parity error
* **RXFIFO\_PARITY\_ERR**[16] - Rx Switch Fifo Parity Err
* **RXFIFO\_PARITY\_ERR\_LAYER**[5:2] - Rx Switch Fifo Parity Error Layer
* **RXFIFO\_PARITY\_ERR\_VC**[1:0] - Rx Switch Fifo Parity Error Virtual Channel

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | T3\_TAGTBL\_PARITY\_ERR | T2\_TAGTBL\_PARITY\_ERR | T1\_TAGTBL\_PARITY\_ERR | T0\_TAGTBL\_PARITY\_ERR | u | | | RXFIFO\_PARITY\_ERR | U | | | | | | | | | | RXFIFO\_PARITY\_ERR\_LAYER | | | | RXFIFO\_PARITY\_ERR\_VC | |

Table OCPS\_LOG\_FLOPPARITY\_ERROR register

### OCPS\_ERROR\_INTERRUPT\_SVRTY

Interrupt severity register. Individual bit positions match the error bit positions in OCPS\_ERROR\_INTERRUPT\_STATUS. When an INTS bit is set, occurrence of the corresponding error event that is not masked will cause a fatal interrupt to be raised. When 1'b0, error event will cause a non-fatal interrupt to be raised.

Attribute: RW

Security: Non-secure

Bit field description:

* **E34**[34] -   
  1'b1: Flop Structure Parity Error will be factored into the fatal\_interrupt signal.  
  1'b0: Flop Structure Parity Error will be factored into the nonfatal\_interrupt signal.
* **E33**[33] -   
  1'b1: Parity error in CSR registers will be factored into the fatal\_interrupt signal.  
  1'b0: Parity error in CSR registers will be factored into the nonfatal\_interrupt signal.
* **E32**[32] -   
  1'b1: Error due to response traffic sent to a noc layer which is power gated will be factored into the fatal\_interrupt signal.  
  1'b0: Error due to response traffic sent to a noc layer which is power gated will be factored into the nonfatal\_interrupt signal.
* **E18**[18] -   
  1'b1: Unknown response destination will be factored into the fatal\_interrupt signal.  
  1'b0: Unknown response destination will be factored into the nonfatal\_interrupt signal.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | E34 | E33 | E32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | E18 | u | | | | | | | | | | | | | | | | | |

Table OCPS\_ERROR\_INTERRUPT\_SVRTY register.

## Configurable Slave Registers

### CSLV\_BG\_BASE

Background address base register 0. Bits 63:6 hold the base address value. Bit 1 holds the secure access bit. The secure bit that accompanies the request will be compared against the value in bit 1 for each range register, and a security violation will occur if the bits don't match. The access will not be allowed to proceed in this case.

Attribute: RW

Security: Secure access only

Bit field description:

* **Address**[63:0] - Background address base.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CSLV\_BG\_BASE register

### CSLV\_BG\_MASK

Background address mask register 0.

Attribute: R

Security: Secure access only

Bit field description:

* **Address**[63:0] - Background address mask. 1s indicate which bits to use in comparison.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CSLV\_BG\_MASK register

### CSLV\_CLK\_GATING\_OVERRIDE

Clock gating override, when set to 1'b1 will cause the clock gating logic to be disabled. 1'b0 will allow activity based clock gating to be performed on the configurable slave logic.

Attribute: RW

Security: Secure access only

Bit field description:

* **CGO**[0] -   
  1'b1: Clock gating override is enabled (clock gating logic is disabled).  
  1'b0: Clock gating override is disabled (clock gating logic is enabled).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CGO |

Table CSLV\_CLK\_GATING\_OVERRIDE register

### CSLV\_ECC\_INFO

This register monitors ECC errors and saves information for potential debug support. The register keeps a count of all single bit and double bit ECC errors that have been detected.

ECC\_dbit\_err\_cnt is the count of double bit errors (uncorrectable errors). ECC\_sbit\_err\_cnt is the count of single bit errors (correctable errors).

This register is readable and writeable, with default value of 0.

Attribute: RW

Security: Non-secure

Bit field description:

* **ECC\_dbit\_err\_cnt**[31:16] - ECC double bit error count.
* **ECC\_sbit\_err\_cnt**[15:0] - ECC single bit error count.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC\_dbit\_err\_cnt | | | | | | | | | | | | | | | | ECC\_sbit\_err\_cnt | | | | | | | | | | | | | | | |

Table CSLV\_ECC\_INFO register

### CSLV\_ERR\_ADDRS\_LOG

Logs the address corresponding to the error interrupt bit set in the cslv\_err\_status register. It will log the address for the first single bit error, even if the interrupt bits for single bit errors are masked. To enable the register to log the address of the next single bit error, write 0s to this register. Otherwise, it will hold the value of the first single bit error, until a more serious error occurs. If any unrecoverable error occurs, the register will log the address for the unrecoverable error and will not change the address logged until the corresponding interrupt bit is cleared.

Attribute: RW

Security: Non-secure

Bit field description:

* **ADDRS**[63:0] - Address corresponding to error event.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| ADDRS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADDRS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CSLV\_ERR\_ADDRS\_LOG register

### CSLV\_ERR\_INJ

Error injection: For those bits implemented, the bit positions match the error bit positions in ERR\_STATUS. When an ERR\_INJ bit is written to 1, the corrsponding ERR\_STATUS bit will be set, and an error message will be sent at the severity indicated by ERR\_SVRTY if the corresponding ERR\_MASK bit is clear. All bits of this register are self-clearing (return immediately to 0).

Attribute: RW

Security: Secure access only

Bit field description:

* **I11**[11] - Injects CSR parity error.
* **I7**[7] - Injects decode error with no background range match.
* **I6**[6] - Injects decode error with foreground range match.
* **I5**[5] - Injects foreground memory range overlap error.
* **I4**[4] - Injects background memory range overlap error.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| U | | | | | | | | | | | | | | | | | | | u | I11 | u | u | u | I7 | I6 | I5 | I4 | u | u | u | u |

Table CSLV\_ERR\_INJ register

### CSLV\_ERR\_MASK

Error mask: bit positions match the error bit positions in ERR\_STATUS. When an ERR\_MASK bit is set to 1, occurrence of the corresponding error event will not cause an error message to be sent. When cleared to 0, error event will cause an error message to be sent.

Attribute: RW

Security: Non-secure

Bit field description:

* **M12**[12] -   
  1'b1: Disable error message for indirect storage invalid address errors.  
  1'b0: Enable error message for indirect storage invalid address errors.
* **M11**[11] -   
  1'b1: Disable error message for CSR parity errors.  
  1'b0: Enable error message for CSR parity errors.
* **M10**[10] -   
  1'b1: Disable error message for overflow of the event counters.   
  1'b0: Enable error message for overflow of the event counters.
* **M9**[9] -   
  1'b1: Disable error message for write decode errors with no background range match.   
  1'b0: Enable error message for write decode errors with no background range match.
* **M8**[8] -   
  1'b1: Disable error message for Write Security Error.  
  1'b0: Enable error message for Write Security Error.
* **M7**[7] -   
  1'b1: Disable error message for read decode errors with no background range match.   
  1'b0: Enable error message for read decode errors with no background range match.
* **M6**[6] -   
  1'b1: Disable error message for Read Security Error.  
  1'b0: Enable error message for Read Security Error.
* **M5**[5] -   
  1'b1: Disable error message for foreground memory range overlaps.   
  1'b0: Enable error message for foreground memory range overlaps.
* **M4**[4] -   
  1'b1: Disable error message for background memory range overlaps.   
  1'b0: Enable error message for background memory range overlaps.
* **M3**[3] -   
  1'b1: Disable error message for write ecc double bit error.   
  1'b0: Enable error message for write ecc double bit error.
* **M2**[2] -   
  1'b1: Disable error message for write ecc single bit error.   
  1'b0: Enable error message for write ecc single bit error.
* **M1**[1] -   
  1'b1: Disable error message for read ecc double bit error.   
  1'b0: Enable error message for read ecc double bit error.
* **M0**[0] -   
  1'b1: Disable error message for read ecc single bit error.   
  1'b0: Enable error message for read ecc single bit error.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | M12 | M11 | M10 | M9 | M8 | M7 | M6 | M5 | M4 | M3 | M2 | M1 | M0 |

Table CSLV\_ERR\_MASK register

### CSLV\_ERR\_STATUS

These error status bits record the first error event and have to be cleared by writing a 1'b0 before new errors are recorded.

Attribute: WZC

Security: Non-secure

Bit field description:

* **E12**[12] - 1'b1: Cslv Indirect Access Invalid Address Error
* **E11**[11] - 1'b1: Cslv CSR Parity Error
* **E10**[10] - 1'b1: Cslv Event Overflow Error
* **E9**[9] - 1'b1: Cslv Write Decode Error--No Background Range Match
* **E8**[8] - 1'b1: Cslv Write Security Error.
* **E7**[7] - 1'b1: Cslv Read Decode Error--No Background Range Match
* **E6**[6] - 1'b1: Cslv Read Security Error.
* **E5**[5] - 1'b1: Cslv Foreground Memory Range Overlap error
* **E4**[4] - 1'b1: Cslv Background Memory Range Overlap error
* **E3**[3] - 1'b1: Cslv Write Double Bit ECC error
* **E2**[2] - 1'b1: Cslv Write Single Bit ECC error
* **E1**[1] - 1'b1: Cslv Read Double Bit ECC error
* **E0**[0] - 1'b1: Cslv Read Single Bit ECC error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |

Table CSLV\_ERR\_STATUS register

### CSLV\_ERR\_SVRTY

Error severity: bit positions match the error bit positions in ERR\_STATUS, each determining the severity level of the associated error. When an ERR\_SVRTY bit is set to 1, the error is considered fatal. When cleared to 0, the error is considered non-fatal.

Attribute: RW

Security: Secure access only

Bit field description:

* **S12**[12] - Severity of indirect storage invalid address error.
* **S11**[11] - Severity of CSR parity error.
* **S10**[10] - Severity of event counter overflow.
* **S9**[9] - Severity of write decode error with no background range match.
* **S8**[8] - Severity of Write Security Error
* **S7**[7] - Severity of read decode error with no background range match.
* **S6**[6] - Severity of Read Security Error
* **S5**[5] - Severity of foreground memory range overlap error.
* **S4**[4] - Severity of background memory range overlap error.
* **S3**[3] - Severity of write ecc double bit error.
* **S2**[2] - Severity of write ecc single bit error.
* **S1**[1] - Severity of read ecc double bit error.
* **S0**[0] - Severity of read ecc single bit error.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | S12 | S11 | S10 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |

Table CSLV\_ERR\_SVRTY register

### CSLV\_EVENT\_COUNTER\_0

This register is an event counter. When the event counter mask control enables certain events to be counted, they will increment this counter. When the counter overflows, in can produce an interrupt if the interrupt mask is enabled. This can be used to trap to software after a number of specified events has occurred.

The counter can be read or written. Writing the value can initialize the counter to a larger value which can speed up the point at which counter will overflow and the interrupt will be triggered.

Attribute: RW

Security: Non-secure

Bit field description:

* **Event\_Counter\_Value\_0**[31:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Event\_Counter\_Value\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CSLV\_EVENT\_COUNTER\_0 register

### CSLV\_EVENT\_COUNTER\_1

This register is an event counter. When the event counter mask control enables certain events to be counted, they will increment this counter. When the counter overflows, in can produce an interrupt if the interrupt mask is enabled. This can be used to trap to software after a number of specified events has occurred.

The counter can be read or written. Writing the value can initialize the counter to a larger value which can speed up the point at which counter will overflow and the interrupt will be triggered.

Attribute: RW

Security: Non-secure

Bit field description:

* **Event\_Counter\_Value\_0**[31:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Event\_Counter\_Value\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CSLV\_EVENT\_COUNTER\_1 register

### CSLV\_EVENT\_COUNTER\_MASK\_0

This register is used to program the event counter. Each bit of this register enables the performance counter to increment if the event occurs. A value of 1 for a bit indicates that this event should be counted.

Multiple bits can be set in the event counter mask register.

When an event satisfies all of the requirements, the cslv\_event\_counter register will be incremented. A value of 1 indicates that the event is selected for counting. A value of 0 the event is not selected. By default, this register will be set to 0 for all bits, indicating no event counting should occur.

Attribute: RW

Security: Non-secure

Bit field description:

* **e14**[14] -   
  1'b1: Count the number of cycles where the target component is busy and a request is waiting.  
  1'b0: Don't count requests where the target component is busy.
* **e13**[13] -   
  1'b1: Count requests which have a security mismatch.   
  1'b0: Don't count requests which have a security mismatch.
* **e12**[12] -   
  1'b1: Count exclusive write requests with OKAY status.   
  1'b0: Don't count exclusive write requests with OKAY status.
* **e11**[11] -   
  1'b1: Count exclusive write requests with EXOKAY status.   
  1'b0: Don't count exclusive write requests with EXOKAY status.
* **e10**[10] -   
  1'b1: Count requests with exclusive flag set.  
  1'b0: Don't count requests with exclusive flag set.
* **e9**[9] -   
  1'b1: Count individual write packets.  
  1'b0: Don't count individual write packets.
* **e8**[8] -   
  1'b1: Count individual read packets.  
  1'b0: Don't count individual read packets.
* **e7**[7] -   
  1'b1: Count write ecc double bit errors.  
  1'b0: Don't count write ecc double bit errors.
* **e6**[6] -   
  1'b1: Count read ecc double bit errors.  
  1'b0: Don't count read ecc double bit errors.
* **e5**[5] -   
  1'b1: Count write ecc single bit errors.  
  1'b0: Don't count write ecc single bit errors.
* **e4**[4] -   
  1'b1: Count read ecc single bit errors.  
  1'b0: Don't count read ecc single bit errors.
* **e3**[3] -   
  1'b1: Count partial writes.  
  1'b0: Don't count partial writes.
* **e2**[2] -   
  1'b1: Count backdoor accesses .  
  1'b0: Don't count backdoor accesses.
* **e1**[1] -   
  1'b1: Count storage write commands.  
  1'b0: Don't count storage write commands.
* **e0**[0] -   
  1'b1: Count storage read commands.  
  1'b0: Don't count storage read commands.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| U | | | | | | | | | | | | | | | | | e14 | e13 | e12 | e11 | e10 | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table CSLV\_EVENT\_COUNTER\_MASK\_0 register

### CSLV\_EVENT\_COUNTER\_MASK\_1

This register is used to program the event counter. Each bit of this register enables the performance counter to increment if the event occurs. A value of 1 for a bit indicates that this event should be counted.

Multiple bits can be set in the event counter mask register.

When an event satisfies all of the requirements, the cslv\_event\_counter register will be incremented. A value of 1 indicates that the event is selected for counting. A value of 0 the event is not selected. By default, this register will be set to 0 for all bits, indicating no event counting should occur.

Attribute: RW

Security: Non-secure

Bit field description:

* **e14**[14] -   
  1'b1: Count the number of cycles where the target component is busy and a request is waiting.  
  1'b0: Don't count requests where the target component is busy.
* **e13**[13] -   
  1'b1: Count requests which have a security mismatch.   
  1'b0: Don't count requests which have a security mismatch.
* **e12**[12] -   
  1'b1: Count exclusive write requests with OKAY status.   
  1'b0: Don't count exclusive write requests with OKAY status.
* **e11**[11] -   
  1'b1: Count exclusive write requests with EXOKAY status.   
  1'b0: Don't count exclusive write requests with EXOKAY status.
* **e10**[10] -   
  1'b1: Count requests with exclusive flag set.  
  1'b0: Don't count requests with exclusive flag set.
* **e9**[9] -   
  1'b1: Count individual write packets.  
  1'b0: Don't count individual write packets.
* **e8**[8] -   
  1'b1: Count individual read packets.  
  1'b0: Don't count individual read packets.
* **e7**[7] -   
  1'b1: Count write ecc double bit errors.  
  1'b0: Don't count write ecc double bit errors.
* **e6**[6] -   
  1'b1: Count read ecc double bit errors.  
  1'b0: Don't count read ecc double bit errors.
* **e5**[5] -   
  1'b1: Count write ecc single bit errors.  
  1'b0: Don't count write ecc single bit errors.
* **e4**[4] -   
  1'b1: Count read ecc single bit errors.  
  1'b0: Don't count read ecc single bit errors.
* **e3**[3] -   
  1'b1: Count partial writes.  
  1'b0: Don't count partial writes.
* **e2**[2] -   
  1'b1: Count backdoor accesses .  
  1'b0: Don't count backdoor accesses.
* **e1**[1] -   
  1'b1: Count storage write commands.  
  1'b0: Don't count storage write commands.
* **e0**[0] -   
  1'b1: Count storage read commands.  
  1'b0: Don't count storage read commands.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | e14 | e13 | e12 | e11 | e10 | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table CSLV\_EVENT\_COUNTER\_MASK\_1 register

### CSLV\_INDIRECT\_ADDRESS

This register contains the system address where the indirect access will occur.

Attribute: RW

Security: Secure access only

Bit field description:

* **Address**[63:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CSLV\_INDIRECT\_ADDRESS register

### CSLV\_INDIRECT\_RAM\_CONT

This is the indirect access RAM content register. It is used in conjunction with the indirect access trigger register. On an indirect read, data is written to this register. On an indirect write, content from this register is written into the RAM. On a read-modify-write, content from this register is used for the XOR function.

Since the RAM data width may be larger than 64 bits, multiple registers are used to hold the data. Any bits beyond the data width are unused.

This register requires secure access, since it can be used to modify or observe the contents of data.

Attribute: RW

Security: Secure access only

Bit field description:

* **RAM\_content**[63:0] - Spare bit(s).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| RAM\_content | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RAM\_content | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table CSLV\_INDIRECT\_RAM\_CONT register

### CSLV\_INDIRECT\_TRIGGER

This register is the indirect access trigger. Indirect access is a mechanism that allows register-based access to the RAM arrays. This can be used for testing RAM bits or reading content on an error condition.

The indirect access is based on a content+trigger mechanism. For writes, the content register is written first to accumulate the data that should be written. Once the content is ready, the trigger register is used to kick off the hardware write mechanism. For reads, the trigger register kicks off a read, and provides data by placing the result into the content registers where it can be accessed.

The indirect access supports 4 sub-commands.

1. Read Raw data. When triggered, a read to the RAM array will be performed and the resulting data, without ECC correction, will be copied into the content register.
2. Write Raw Data. When triggered, the content register values will be written into the RAM. This will include the ECC bits if present.
3. Write Data with Generated ECC. When triggered, this will write to the RAM entry. The content register will be used to specify the data to be written. However, if ECC hardware is present, the ECC bits will be generated based on the data instead of coming from the content register. This allows the RAM entry to be written with correct ECC value without needing to calculate it first.
4. Read-Modify-Write. This command will perform a specific kind of read-modify-write operation on a RAM entry. It will read the content of the RAM, XOR that content with the indirect content register, and write the combined value into the same RAM entry. This can be used to introduce single or double bit errors into the directory to test error detection and handling. The content register will not be modified during this operation, so it can be used to introduce errors into multiple lines.

Each of the indirect access commands can be issued during normal operation, but the Write commands can have side-effects that break coherency functionality. The Read Raw is not disruptive, and the Read-Modify-Write can be performed atomically so single-bit errors can be introduced while maintaining functionality.The indirect access trigger registers is readable and writeable. To trigger the RAM access, this register must be written. Reads will not have side-effects and will only return the current value of the trigger register.

The trigger register must be set correctly. The CMD field indicates which kind of indirect access to perform. The RAM address indicates the entry to access within the RAM.

This register requires secure access, since it can be used to modify or observe the contents of data.

Attribute: RW

Security: Secure access only

Bit field description:

* **cmd**[1:0] -   
  11: Read-modify-write. Read array content, XOR with RAM content register, and write modified data into array  
  10: Write RAM content minus ECC bits to array, use ECC generation logic to set ECC bits in array  
  01: Write RAM content register directly into array  
  00: Read raw array content including any ECC bits and copy to RAM Content register

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | cmd | |

Table CSLV\_INDIRECT\_TRIGGER register

### CSLV\_SECURITY\_ERR\_MASK

Security Error mask: bit positions match the error bit positions in security\_error\_status. When an ERR\_MASK bit is set to 1, occurrence of the corresponding error event will not cause an error message to be sent. When cleared to 0, error event will cause an error message to be sent.

Attribute: RW

Security: Secure access only

Bit field description:

* **M8**[8] -   
  1'b1: Disable error message for Cslv write security error.   
  1'b0: Enable error message for Cslv write security error.
* **M6**[6] -   
  1'b1: Disable error message for Cslv read security error.   
  1'b0: Enable error message for Cslv read security error.
* **M3**[3] -   
  1'b1: Disable error message for Cslv Write Double Bit ECC error.   
  1'b0: Enable error message for Cslv Write Double Bit ECC error.
* **M1**[1] -   
  1'b1: Disable error message for Cslv Read Double Bit ECC error.   
  1'b0: Enable error message for Cslv Read Double Bit ECC error.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| U | | | | | | | | | | | | | | | | | | | | | | | M8 | u | M6 | u | | M3 | u | M1 | u |

Table CSLV\_SECURITY\_ERR\_MASK register

### CSLV\_SECURITY\_ERR\_STATUS

These error status bits record the first security error event and have to be cleared by writing a 1'b0 before new errors are recorded.

Attribute: WZC

Security: Secure access only

Bit field description:

* **E8**[8] - 1'b1: Cslv Write Security Error
* **E6**[6] - 1'b1: Cslv Read Security Error
* **E3**[3] - 1'b1: Cslv Write Double Bit ECC error
* **E1**[1] - 1'b1: Cslv Read Double Bit ECC error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | E8 | u | E6 | u | | E3 | u | E1 | u |

Table CSLV\_SECURITY\_ERR\_STATUS register

### CSLV\_STORAGE\_INIT\_STATUS

Provide a bit for SW to poll to see when the cslv power-on reset storage init sequence is completed.

Attribute: R

Security: Non-secure

Bit field description:

* **INIT\_STATUS**[0] -   
  1'b1: Cslv storage init sequence has completed.  
  1'b0: Cslv storage init sequence has not yet completed.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | INIT\_STATUS |

Table CSLV\_STORAGE\_INIT\_STATUS register