# 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 exact list of registers implemented for a given design can be found in the noc\_reference\_manual.html under the <project> folder after the RTL is generated by NocStudio.

Registers are divided into the following categories:

* Router registers
* NSIP (Streaming) Bridge registers
* Regbus registers

A set of performance counter registers can be automatically cleared upon software read by setting “mesh\_prop register\_clear\_on\_read\_enabled yes” in the configuration file.

* Rtr\_ivc\_event\_counter (formerly known as REC)
* Rtr\_ovc\_event\_counter (formerly known as ROEC)
* Noc\_rx\_event\_counter0 (formerly known as R\_2)
* Noc\_rx\_event\_counter1 (formerly known as BRHST\_CNTR1)
* Noc\_tx\_event\_counter0 (formerly known as T\_2)

## 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.

## NSIP Registers

### NOC\_RX\_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 Streaming Tx bridge. Coarse clock gating for selective Steaming Rx Bridges can be overridden by locally setting this register, if the user does not want to 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 NOC\_RX\_CG\_CTRL register.

### NOC\_RX\_CG\_HYST\_COUNT

This register is used by coarse grained clock gating logic. The register holds the count value for 'number of cycles' the Streaming RX Bridge has to wait before de-asserting its 'Busy' signal for all output NoC layers. Since the Streaming RX Bridge does not send flits to any NoC element, this counter is not used currently. This is a read write register and its value does not have any effect on the operation of the Streaming Rx Bridge.

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 NOC\_RX\_CG\_HYST\_COUNT register.

### 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\_COUNTER0

The event counter control registers can be used to count performance or debug events in the receive section of the streaming bridge. This is the portion of the bridge that accepts packets from the NoC and sends it to the host. There are 4 register in the event counter control register set.

Register 3 is currently unused.

Register 2 is the event counter itself. It is 64 bits wide. Whenever a selected event occurs in hardware, the event counter will increment. This register defaults to zero. It can be read at any time, and can be written to any value. When the counter reaches its peak value, it will roll over to zero and continue counting. The rollover condition can be set up to trigger an interrupt.

Register 1 is a mask register. It defaults to zero, meaning no events are counted by default. To enable counting, the user can write 1s to the appropriate bits.

The mask (Register 1) is used in combination with Register 0 to select the event(s) to count. Register 0 can be programmed with comparison fields. Register 1 mask will determine which bits to compare. For instance, to count on SOPs seen, Register 0 and 1 should both be set to 0x1.

The event is determined by a bit-wise comparison. The more mask bits used, the less likely a comparison will match and an event will be counted. If it is just looking for SOP, it will count the number of packets. If it looks for SOP and EOP, it will count the number of single-flit packet.

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 NOC\_RX\_EVENT\_COUNTER0 register.

### NOC\_RX\_EVENT\_COUNTER0\_MASK

The event counter control registers can be used to count performance or debug events in the receive section of the streaming bridge. This is the portion of the bridge that accepts packets from the NoC and sends it to the host. There are 4 register in the event counter control register set.

Register 3 is currently unused.

Register 2 is the event counter itself. It is 64 bits wide. Whenever a selected event occurs in hardware, the event counter will increment. This register defaults to zero. It can be read at any time, and can be written to any value. When the counter reaches its peak value, it will roll over to zero and continue counting. The rollover condition can be set up to trigger an interrupt.

Register 1 is a mask register. It defaults to zero, meaning no events are counted by default. To enable counting, the user can write 1s to the appropriate bits.

The mask (Register 1) is used in combination with Register 0 to select the event(s) to count. Register 0 can be programmed with comparison fields. Register 1 mask will determine which bits to compare. For instance, to count on SOPs seen, Register 0 and 1 should both be set to 0x1.

The event is determined by a bit-wise comparison. The more mask bits used, the less likely a comparison will match and an event will be counted. If it is just looking for SOP, it will count the number of packets. If it looks for SOP and EOP, it will count the number of single-flit packet.

Attribute: RW

Security: Non-secure

Bit field description:

* **MASK**[31:0] - 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table NOC\_RX\_EVENT\_COUNTER0\_MASK register.

### NOC\_RX\_EVENT\_COUNTER1

The event counter control registers can be used to count performance or debug events in the receive section of the streaming bridge. This is the portion of the bridge that accepts packets from the NoC and sends it to the host. There are 4 register in the event counter control register set.

Register 3 is currently unused.

Register 2 is the event counter itself. It is 64 bits wide. Whenever a selected event occurs in hardware, the event counter will increment. This register defaults to zero. It can be read at any time, and can be written to any value. When the counter reaches its peak value, it will roll over to zero and continue counting. The rollover condition can be set up to trigger an interrupt.

Register 1 is a mask register. It defaults to zero, meaning no events are counted by default. To enable counting, the user can write 1s to the appropriate bits.

The mask (Register 1) is used in combination with Register 0 to select the event(s) to count. Register 0 can be programmed with comparison fields. Register 1 mask will determine which bits to compare. For instance, to count on SOPs seen, Register 0 and 1 should both be set to 0x1.

The event is determined by a bit-wise comparison. The more mask bits used, the less likely a comparison will match and an event will be counted. If it is just looking for SOP, it will count the number of packets. If it looks for SOP and EOP, it will count the number of single-flit packet.

Attribute: RW

Security: Non-secure

Bit field description:

* **CNTR**[31:0] - Bridge receive host counter-1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 NOC\_RX\_EVENT\_COUNTER1 register.

### NOC\_RX\_EVENT\_COUNTER1\_MASK

The event counter control registers can be used to count performance or debug events in the receive section of the streaming bridge. This is the portion of the bridge that accepts packets from the NoC and sends it to the host. There are 4 register in the event counter control register set.

Register 3 is currently unused.

Register 2 is the event counter itself. It is 64 bits wide. Whenever a selected event occurs in hardware, the event counter will increment. This register defaults to zero. It can be read at any time, and can be written to any value. When the counter reaches its peak value, it will roll over to zero and continue counting. The rollover condition can be set up to trigger an interrupt.

Register 1 is a mask register. It defaults to zero, meaning no events are counted by default. To enable counting, the user can write 1s to the appropriate bits.

The mask (Register 1) is used in combination with Register 0 to select the event(s) to count. Register 0 can be programmed with comparison fields. Register 1 mask will determine which bits to compare. For instance, to count on SOPs seen, Register 0 and 1 should both be set to 0x1.

The event is determined by a bit-wise comparison. The more mask bits used, the less likely a comparison will match and an event will be counted. If it is just looking for SOP, it will count the number of packets. If it looks for SOP and EOP, it will count the number of single-flit packet.

Attribute: RW

Security: Non-secure

Bit field description:

* **MASK**[31:0] - Bridge receive host counter-1 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table NOC\_RX\_EVENT\_COUNTER1\_MASK register.

### NOC\_RX\_EVENT\_COUNTER\_CTRL

The event counter control registers can be used to count performance or debug events in the receive section of the streaming bridge. This is the portion of the bridge that accepts packets from the NoC and sends it to the host. There are 4 register in the event counter control register set.

Register 3 is currently unused.

Register 2 is the event counter itself. It is 64 bits wide. Whenever a selected event occurs in hardware, the event counter will increment. This register defaults to zero. It can be read at any time, and can be written to any value. When the counter reaches its peak value, it will roll over to zero and continue counting. The rollover condition can be set up to trigger an interrupt.

Register 1 is a mask register. It defaults to zero, meaning no events are counted by default. To enable counting, the user can write 1s to the appropriate bits.

The mask (Register 1) is used in combination with Register 0 to select the event(s) to count. Register 0 can be programmed with comparison fields. Register 1 mask will determine which bits to compare. For instance, to count on SOPs seen, Register 0 and 1 should both be set to 0x1.

The event is determined by a bit-wise comparison. The more mask bits used, the less likely a comparison will match and an event will be counted. If it is just looking for SOP, it will count the number of packets. If it looks for SOP and EOP, it will count the number of single-flit packet.

Attribute: RW

Security: Non-secure

Bit field description:

* **NOC\_VALID**[8] - Valid flit from NoC to host interface
* **NO\_CREDIT**[7] - No credit from host
* **INTF\_VALID**[6] - Valid flit on host interface
* **IF\_ID**[5:2] - Bit map selecting host Interface
* **EOP**[1] - End-of-packet
* **SOP**[0] - Start-of-packet

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| r | | | | | | | | | | | | | | | | | | | | | | | NOC\_VALID | NO\_CREDIT | INTF\_VALID | IF\_ID | | | | EOP | SOP |

Table NOC\_RX\_EVENT\_COUNTER\_CTRL 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\_ID

This register holds a unique 8-bit identifier for the receiving bridge. 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:

* **ZEROES**[15:8] - Forced to zero
* **ID**[7:0] - A unique 8-bit identifier assigned to the bridge to uniquely identify it on the NoC. It is equal to the corresponding RXID 8-bit identifier on the Tx side of the bridge.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 NOC\_RX\_ID 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\_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 Streaming Tx bridge. Coarse clock gating for selective Steaming Tx Bridges 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 NOC\_TX\_CG\_CTRL register.

### NOC\_TX\_CG\_HYST\_COUNT

This register is used by coarse grained clock gating logic. The register holds the count value for 'number of cycles' the Streaming TX Bridge has to wait before de-asserting its 'Busy' signal for all output NoC layers.

This is to let the NoC elements (connected to it) on all output layer know that the Streaming Tx Bridge is in quiescent state and there are no pending transactions inside the Steaming Tx Bridge. This register is one per Streaming Tx Bridge and is used by all output layers to de-assert their 'Busy' signal. This is a read-write register.

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 NOC\_TX\_CG\_HYST\_COUNT register.

### NOC\_TX\_DEST\_DECODE\_ERROR

This register logs the route lookup information for the first occurrence of a lookup failure (illegal destination) for each host interface. Note the cfg\_bridge\_id is common to all lookups and is not logged here (but is reported in the noc\_tx\_id register). Subsequent error keys for a given host interface will not be logged until the corresponding valid bit is cleared.

Attribute: RW

Security: Non-secure

Bit field description:

* **VALID\_A**[15] -   
  1'b1: Host interface A illegal destintation error has been logged, subsequent errorneous lookup keys will not be logged until this bit is cleared.  
  1'b0: Host interface A illegal error log is old/invalid, next occurring error will be logged.
* **IFID\_A**[13:12] - Host interface A destintation interface ID (ifid) value of failing lookup.
* **QOS\_A**[11:8] - Host interface A QOS value of failing lookup.
* **HPID\_A**[7:0] - Host interface A destination host id (hpid) value of failing lookup.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | | | | u | | | | | | | | u | 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 | | | | u | | | | | | | | VALID\_A | u | IFID\_A | | QOS\_A | | | | HPID\_A | | | | | | | |

Table NOC\_TX\_DEST\_DECODE\_ERROR 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\_COUNTER0

The event counter control registers can be used to count performance or debug events in the transmit portion of the streaming bridge. This is the portion of the bridge that accepts packets from the host and sends it to the NoC. There are 4 register in the event counter control register set.

Register 3 is currently unused.

Register 2 is the event counter itself. It is 32 bits wide. Whenever a selected event occurs in hardware, the event counter will increment. This register defaults to zero. It can be read at any time, and can be written to any value. When the counter reaches its peak value, it will roll over to zero and continue counting. The rollover condition can be set up to trigger an interrupt.

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 NOC\_TX\_EVENT\_COUNTER0 register.

### NOC\_TX\_EVENT\_COUNTER0\_MASK

The event counter control registers can be used to count performance or debug events in the transmit portion of the streaming bridge. This is the portion of the bridge that accepts packets from the host and sends it to the NoC. There are 4 register in the event counter control register set.

Register 1 is a mask register. It defaults to zero, meaning no events are counted by default. To enable counting, the user can write 1s to the appropriate bits.

The mask is used in combination with Register 0 to select the event(s) to count. Register 0 can be programmed with comparison fields. Register 1 mask will determine which bits to compare. For instance, to count on SOPs seen, Register 0 and 1 should both be set to 0x1.

The event is determined by a bit-wise comparison. The more mask bits used, the less likely a comparison will match and an event will be counted. If it is just looking for SOP, it will count the number of packets. If it looks for SOP and EOP, it will count the number of single-flit packet.

Partial fields can be compared by setting only some of the mask bits for a field. Packets with QoS values of 8-15 could be set by marking only one mask bit corresponding to the QoS bit 3.

Attribute: RW

Security: Non-secure

Bit field description:

* **MASK**[31:0] - 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table NOC\_TX\_EVENT\_COUNTER0\_MASK register.

### NOC\_TX\_EVENT\_COUNTER\_CTRL

The event counter control registers can be used to count performance or debug events in the transmit portion of the streaming bridge. This is the portion of the bridge that accepts packets from the host and sends it to the NoC. There are 4 register in the event counter control register set.

The mask (Register 1) is used in combination with Register 0 to select the event(s) to count. Register 0 can be programmed with comparison fields. Register 1 mask will determine which bits to compare. For instance, to count on SOPs seen, Register 0 and 1 should both be set to 0x1.

The event is determined by a bit-wise comparison. The more mask bits used, the less likely a comparison will match and an event will be counted. If it is just looking for SOP, it will count the number of packets. If it looks for SOP and EOP, it will count the number of single-flit packet.

Partial fields can be compared by setting only some of the mask bits for a field. Packets with QoS values of 8-15 could be set by marking only one mask bit corresponding to the QoS bit 3.

Attribute: RW

Security: Non-secure

Bit field description:

* **DEST\_PORT\_ID**[23:16] - Destination port ID
* **QOS**[14:11] - QoS
* **SRC\_IF\_ID**[9:6] - Source interface ID
* **DEST\_IF\_ID**[3:2] - Destination interface ID
* **EOP**[1] -   
  1'b1: End of packet
* **SOP**[0] -   
  1'b1: Start of packet

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| r | | | | | | | | DEST\_PORT\_ID | | | | | | | | r | QOS | | | | r | SRC\_IF\_ID | | | | r | | DEST\_IF\_ID | | EOP | SOP |

Table NOC\_TX\_EVENT\_COUNTER\_CTRL 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\_FIFO\_STATUS

This register tracks the status of the bridge transmit FIFOs. There are up to 4 FIFOs, with one per interface in the streaming bridge. This is a read-only register.

Attribute: R

Security: Non-secure

Bit field description:

* **INTF\_D\_F**[14] -   
  1'b1: Buffer full for interface D
* **INTF\_D\_S**[13] -   
  1'b1: Head flip SOP for interface D
* **INTF\_D\_V**[12] -   
  1'b1: Head flit valid (buffer ready) for interface D
* **INTF\_C\_F**[10] -   
  1'b1: Buffer full for interface C
* **INTF\_C\_S**[9] -   
  1'b1: Head flip SOP for interface C
* **INTF\_C\_V**[8] -   
  1'b1: Head flit valid (buffer ready) for interface C
* **INTF\_B\_F**[6] -   
  1'b1: Buffer full for interface B
* **INTF\_B\_S**[5] -   
  1'b1: Head flip SOP for interface B
* **INTF\_B\_V**[4] -   
  1'b1: Head flit valid (buffer ready) for interface B
* **INTF\_A\_F**[2] -   
  1'b1: Buffer full for interface A
* **INTF\_A\_S**[1] -   
  1'b1: Head flip SOP for interface A
* **INTF\_A\_V**[0] -   
  1'b1: Head flit valid (buffer ready) 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 | | | | | | | | | | | | | | | | u | INTF\_D\_F | INTF\_D\_S | INTF\_D\_V | u | INTF\_C\_F | INTF\_C\_S | INTF\_C\_V | u | INTF\_B\_F | INTF\_B\_S | INTF\_B\_V | u | INTF\_A\_F | INTF\_A\_S | INTF\_A\_V |

Table NOC\_TX\_FIFO\_STATUS register.

### NOC\_TX\_ID

This register holds a unique 8-bit identifier for the transmitting bridge. 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:

* **ZEROES**[15:8] - Forced to zero
* **ID**[7:0] - A unique 8-bit identifier assigned to the bridge to uniquely identify it on the NoC. It is equal to the corresponding RXID 8-bit identifier on the Rx side of the bridge.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 NOC\_TX\_ID 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.

## Regbus Master/Slave Registers

### 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 be 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\_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\_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] -   
  011: W (for captured event only )  
  010: R (for captured event only )  
  001: AW (for captured event or response latency)  
  000: AR (for 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\_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\_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

### 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\_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.