OpenCAPI 3.1 Transaction Layer RX (TLXR) Workbook

Version 1.0

TLXR Reference Design for the OpenCAPI 3.1 Transaction Layer Specification from OpenCAPI Consortium:
https://opencapi.org/technical/specifications/

Used in Open Memory Interface (OMI) ASIC Devices

1 OpenCAPI 3.1 Transaction Layer RX (TLXR) Workbook

1.1 Functional Description

The TLX RX component handles decoding of OpenCAPI frames in accordance with the OpenCAPI transaction layer spec v3.1. The logic sits in between the Data Link Layer and the Memory Sequencer and write data flow. The TLX RX component handles parsing and decoding TL command and response packets from a command flit, and steering data into the write data flow.

1.2 VHDL Diagram

1.2.1 Logic Block Descriptions

cb_tlxr_mac – OpenCAPI Transaction Layer. Decodes downstream OpenCAPI command flits, processes fast activate commands and manages the data buffers that are contained within cb_wdf_mac logic.

cb_tlxr_buf_ctl – Maintains a list of data that has been promised for writes. This must arrive in the same order as the write commands which use it. Drives many of the control signals to wdf.

cb_tlxr_xlat – Provides controlled multiplexing of OpenCAPI address bits onto the Row, Column, bank, bank group , master, and slave DDR address bits.

cb_tlxr_pkg – Contains functions and procedures used in the other vhdl source files.

1.3 Interfaces

Signal Name	I/O	Size	Description	Clock Domain
		Input from top level		
syncr	I	1	Synchronous reset	gckn
		Input from DLX		
dlx_tlxr_flit_vld	I	1	Indicates this cycle has valid data and parity	gckn
dlx_tlxr_flit_error	I	1	Indicates that this control flit and all previous data flits are to be discarded. This can be asserted when dlx_tlxr_**flit_vld** is inactive	gckn
dlx_tlxr_flit_data	I	128	Partial Flit Data (16 bytes of the 64 bytes flit)	gckn

Signal Name	I/O	Size	Description	Clock Domain
dlx_tlxr_flit_pty	I	16	Parity protecting dlx_tlxr_**flit_data**, 1 bit per byte of flit data	gckn
dlx_tlxr_link_up	I	1	Indicates that the OpenCapi link is trained and ready to accept flits	gckn
dlx_tlxr_fast_act_info	I	35	Early address for slot 0 reads when dlx_tlxr_idle_transition is active. Used for fast activate only.	gckn
dlx_tlxr_idle_transition	I	1	Early Indication of beat 0 of a control flit. Used for fast_activate only.	gckn
			Interface with SRQ	
tlxr_srq_cmd_val	O	1	read or write cmd is valid when asserted	gckn
tlxr_srq_cmd	O	64	**FOR READS** 0:15 → OC Tag(15 : 0) OC15 bypass (overloaded on tag if P9) OC14 retry (overloaded on tag if P9) OC13:0 tag **FOR WRITES** 0:3 reserved 4:9 2nd write buffer pointer(5:0)` 10:15 1st write buffer pointer(5:0) **Mainline Commands (Request type=00XX)** 16:50 → Address 0:1 Bank Group 2:3 Bank 4:5 Master Rank 6:8 Slave Rank 9:26 Row 27:34 Col 51:52 → Data Length '00' → 32 Bytes; '01' → 64 Bytes; '10' → 128 Bytes; '11' → reserved; 53 → BG+Bank+Row even parity 54 → BG+Bank+Col even parity 55 → Mrank+Srank + Dimm even parity 56 → Tag+Length+Request type even parity 57 → Tag+Length+Request type even parity 58 → Dimm **MMIO/config Commands** (Request type=10XX or 01XX) 16:50 → PA(34:0) (16 is MSB) 51:53 → pL(2:0) 54 → T(config only) 55 → Even Parity on 0:63 excluding 55 56:58 → Reserved	gckn

Signal Name	I/O	Size	Description	Clock Domain
			Read Fail response (Request type=1110) 16:19 → Read fail response code 20 → '0' 51:52 → dL(1:0) 56 → even parity on 0:19,51:52,59-63 20-50,53-55,57-58 → Reserved 59 – '0' (used to identify TLXR) 60:63 → Request type '0000' → read; '0001' → write '0010' → partial write '0011' → pad_mem '0100' → MMIO read '0101' → MMIO write '1000' → config read '1001' → config write '1110' → read fail	
tlxr_srq_fast_act_val	O	1	Fast Activate valid	gckn
tlxr_srq_fast_addr	O	27	BG 2 bits Bank 2 bits Master 2 bits (CS/CID) Slave 3 bits Row 18 bits	gckn
tlxr_srq_fast_dimm	O	1	Dimm bit for fast activate	gckn
tlxr_srq_wrbuf_crcval_vec	O	64	Write buffer Data CRC valid bit vector. Each bit indicates valid data in its corresponding data buffer.	gckn
tlxr_srq_memctl_req	O	1	Pulse for mem_cntl command received	
tlxr_srq_memcntl_cmd_flag	O	4	Flag associated with mem_ctl	
srq_tlxr_wdone_val	I	1	SRQ is done with the data buffer	gckn
srq_tlxr_wdone_last	I	1	Write done is for last 64B of this line.	gckn

Signal Name	I/O	Size	Description	Clock Domain
srq_tlxr_wdone_tag	I	7	Write done buffer tag	gckn
srq_tlxr_wdone_p	I	1	Even parity on last,tag derr	gckn
srq_tlxr_epow	I	1	In early write done, for all writes already issued to SRQ, a write_resp will be queued and SRQ will complete the command. For late write done ops, tlxr continues to queue write_resps for any dones from SRQ or MMIO. Once seen, no new commands are issued to SRQ. (Credit returns continue as normal).	gckn
Interface with Write Data Flow				
tlxr_wdf_wrbuf_wr	O	1	Write data is valid when asserted	gckn
tlxr_wdf_wrbuf_wptr	O	6	Write data buffer pointer	gckn
tlxr_wdf_wrbuf_woffset	O	2	Offset for the 64B write data buffer	gckn
tlxr_wdf_wrbuf_wr_p	O	1	EVEN parity over wrbuf_wr & wrbuf_wptr & wrbuf_woffset	gckn
tlxr_wdf_wrbuf_dat	O	146	16B Write data + 2 bits meta + 16 bits ECC	gckn
tlxr_wdf_wrbuf_bad	O	65	Bits 0:63: OpenCAPI "Bad Data" bit associated with each Write Buffer. Remains valid from when write data delivered to WDF, until srq/mmio write done Bit number corresponds to buffer number, so bit 0 is for write buffer number 0. Bit 64: Even parity over bits 0:63. Valid and checked every clock cycle.	gckn
tlxr_wdf_be_wr	O	1	Writing BEs this cycle	gckn
tlxr_wdf_be_wptr		6	Write Buffer number BEs go with	gckn
tlxr_wdf_be_wr_p		1	EVEN parity over be_wr & be_wptr	gckn
tlxr_wdf_be		72	0:63 BE, 64:71 ecc (Note: RMW never 128 byte op)	gckn
Interface with TLX TX				
tlxt_tlxr_early_wdone_disable	I	2	These bits disable the early write done mechanism so that write responses are only sent once the write has completed at the dram. See 1.4.2	gckn
tlxt_tlxr_wr_resp_full	I	1	Indicates that tlxt cannot accept a response at this time. If it comes on in any cycle when tlxr is driving the valid then tlxr will stall with valid and the same response until this signal goes inactive ..	gckn
tlxt_tlxr_control	I	16	These register bits control the debug bus and chicken switches in tlxr. Bits 7 downto 0 control the debug bus -see 1.9.2 on page30.	
tlxt_tlxt_low_lat_mode	I	1	Signals that TLXR is in low latency mode and (if not in hal;f dimm mode) only 64 or 128 byte reads are	

Signal Name	**I/O**	**Size**	**Description**	**Clock Domain**
			allowed from DDR.	
tlxr_tlxt_write_resp	O	22	Entry from the tag FIFO, popped from the queue after sequencer indicates write buffer can be freed. This bus has 21:6 as the OpenCAPI tag, 5:2 as response code (0 is success) and 1:0 is the original DL value..	gckn
tlxr_tlxt_write_resp_p	O	3	Even Parity on : 21:14 top half of tag 13:6 bottom half of tag 5:0 & valid	gckn
tlxr_tlxt_write_resp_val	O	1	Indicates that the ***tlxr_tlxt_write_resp*** is valid and should be pushed into the write response queue.	gckn
tlxr_tlxt_consume_vc0	O	1	Number of vc0 credits consumed by the host. Only supported use of vc0 is interrupt response from the host.	gckn
tlxr_tlxt_consume_vc1	O	1	Number of host vc1 credits consumed . Alll currently supported commands use 1 vc1 credit.	
tlxr_tlxt_consume_dcp1	O	3	Number of DCP1 credits consumed . Consumed when 64B write data buffer is allocated (before data has arrived for that buffer). For 128 byte transfers max value is 10.	gckn
tlxr_tlxt_return_val	O	1	Indicates that credits have been returned from the TL. Add number of credits indicated by ***tlxr_tlxt_credits_vc**** and ***tlxr_tlxt_credits_dcp**** signals to respective counters.	gckn
tlxr_tlxt_return_vc0	O	4	Number of credits returned to virtual channel 0 on receipt of a ***return_tlx_credits*** command from the host.	gckn
tlxr_tlxt_return_vc3	O	4	Number of credits returned to virtual channel 3 on receipt of a ***return_tlx_credits*** command from the host.	gckn
tlxr_tlxt_return_dcp0	O	6	Number of credits returned to Data credit Pool 0 on receipt of a ***return_tlx_credits*** command from the host.	gckn
tlxr_tlxt_return_dcp3	O	6	Number of credits returned to Data credit Pool 3 on receipt of a ***return_tlx_credits*** command from the host.	gckn
tlxr_tlxt_vc1_release	O	1	Credit return for set_pad_pattern	gckn
tlxr_tlxt_vc0_release	O	2	Credit return for mem_cntl and intrp_resp	gckn
tlxr_tlxt_dcp1_release	O	3	DCP1 credit is returned each time a 64B buffer becomes free and is available for reuse. This will typically be signalled before the tag release	gckn
tlxr_tlxt_intrp_resp	O	8	1:0 is for tag 1 ……. 7:6 is for tag 4 the two bit fields are coded in this way: “01” - good completion “10” - retry “11” - fail “00” - inactive	gckn

Signal Name	I/O	Size	Description	Clock Domain
tlxr_tlxt_errors	O	64	Error pulses for tlxr internal errors.. see 1.5.4	
		Interface with MMIO		
mmio_tlxr_wr_buf_free	I	1	MMIO has finished with a buffer	gckn
mmio_tlxr_wr_buf_tag	I	6	This is the buffer that MMIO has finished with	gckn
mmio_tlxr_resp_code	I	4	Completion code for mmio write.	gckn
mmio_tlxr_wr_buf_par	I	1	E Parity on tag free and response code	gckn
cfg_f1_csh_memory_space	I	1	Enable memory commands to MMIO	gckn
cfg_f1_csh_mmio_bar0	I	29	BAR0. Used to define MMIO base	gckn
cfg_f1_csh_p	I	1	Even parity on bar0 and memory_space checked every clock	gckn
cfg_f1_octr100_metadata_enabled	I	1	This enables metadata. When disabled, metadata is set to all '0'.	gckn
		Interface with Trace		
tlxr_tp_fir_trace_err	O	1	Pulse to stop trace	gckn

1.3.1 Incoming Interfaces

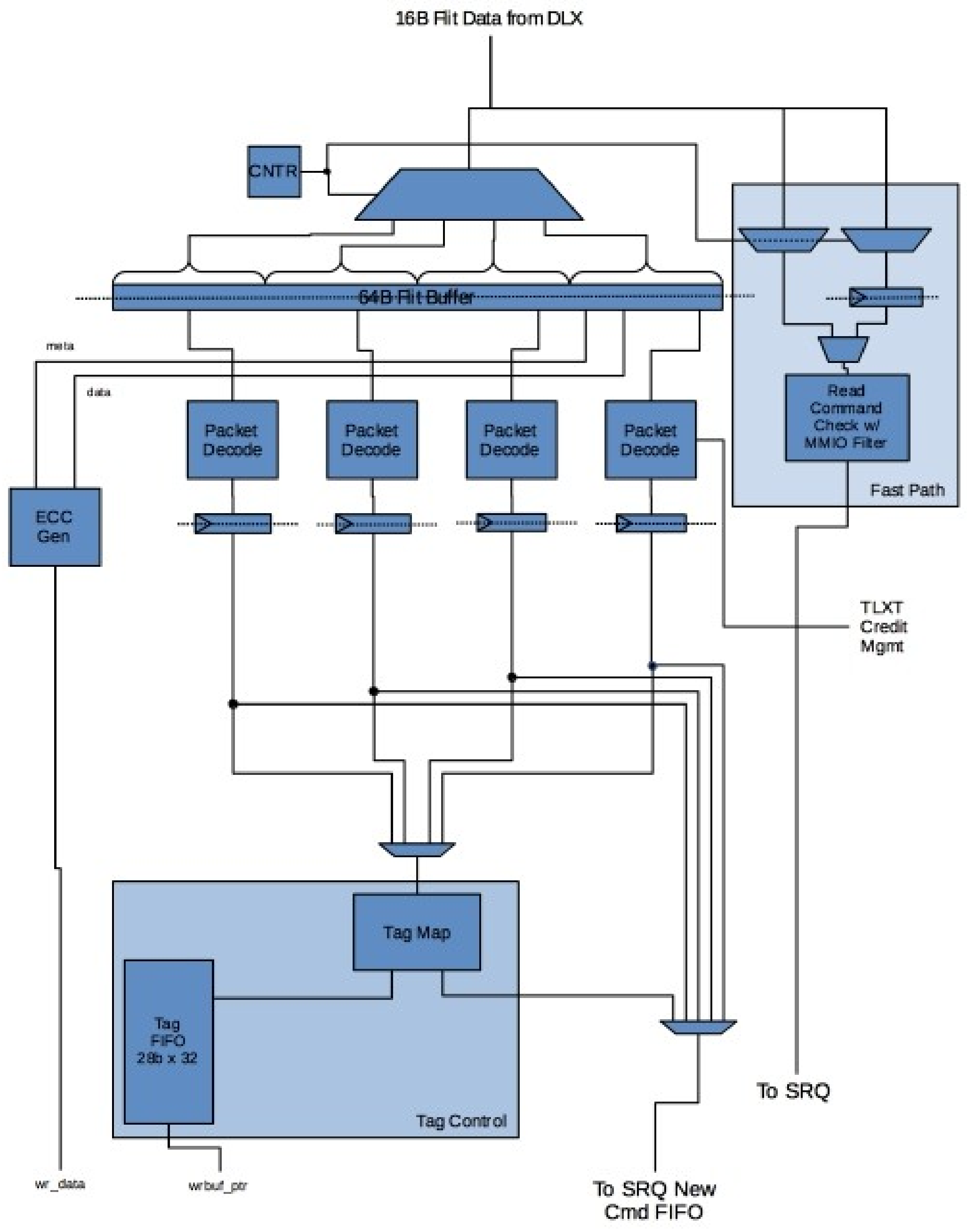
Interfaces incoming from the DLX are little endian bit order. Incoming interfaces coming in from the SRQ and WDF are big endian bit order.

1.3.2 Outgoing Interfaces

Outgoing interfaces to SRQ and WDF are big endian bit order. Outputs to DLX are little endian bit order.

1.4 Design Details

1.4.1 Unit Data Flow



1.4.1.1 General Command Flow

Each 16B flit partial is buffered up from the dlx_tlxr*_flit_data* interface into the flit buffer while awaiting the DL content in the fourth 16B. When the DL content is received each TL packet (max of 4) is pushed into one of the 4 packet decoders. Read and Writes are decoded much in the same manner, passing the OpenCAPI PA through an MMIO filter and then Address translation. Each is packed up in the format shown in the ***tlxr_srq_cmd*** signal.

1.4.1.2 Fast Activate

In order to provide the lowest possible latency for at least some memory reads, the OMI ASIC implements two fast activate mechanisms that allow the sequencer to activate the required row (which may also involve closing the current row) in order to save time.

The first of these two mechanisms mechanisms looks at the first 16 byte beat of a control flit arriving from the host and if it is a rd_mem, and if the logical to physical address translation is set up to be in a particular subset of the allowed range (see table in address translation description), then even before the entire control flit (and its CRC check) has been received , TLXR sends a translated address and strobe signal to give SRQ an early indication of a read command that will probably be arriving in a few clocks time.

Note that for the templates that the OMI ASIC supports, this mechanism will only work for a rd_mem command in slots 0-3 of a template 1 flit. The OMI ASIC does not know the template number until the last beat of a control fit and this means that this mechanism can potentially signal a false fast activate for template_7's and template_A's.

This path is unlatched in TLXR, it will provide the lowest read latency for a subset of mem_rd's.

The second fast activate mechanism looks for rd_mem or pr_rd_mem commands in all four beats of each control flit and applies a filter to mask out as many false activates in template_7's and template_A's as possible. It writes the address of any of these reads into a short (4 entry) fifo inside TLXR. Note that there is no concept of overflowing this fifo – the write side of this fifo will always write an entry if a good read candidate is detected. There is no performance advantage to stall writing the fifo for any reason.

The fifo read side is controlled by SRQ. It can read fifo entries or drain the fifo depending on the workload and the banks that are currently open. More details should be in the SRQ workbook.

This mechanism is slower than the unlatched scheme described above, but will work for more reads and potentially for a wider range of translation options. It is also better at rejecting false activates for the templates that contain data as well as control.

These mechanisms are not perfect. For example data in a template-7 flit can potentially be mistaken for a rd_mem, and spurious fast_activate will be generated. There is a significant performance penalty for opening a bank which is not used but the analysis has been that the benefit outweighs this downside, and spurious fast_activates will be infrequent.

1.4.1.3 Write Flow

When a write command is decoded by one of the packet decoder blocks it retains its OpenCAPI tag when pushed into the command staging latches. The muxes are set such that the write command will flow through the tag mapping block before being pushed into the new command fifo. The OpenCAPI tag is assigned two write buffer tags, one for the first 64B of data and one for the second. The OpenCAPI tag and both write buffer tags are pushed into the tag buffer awaiting data to arrive on the DLX interface. The OpenCAPI tag in the command sent to the SRQ is replaced with the write buffer tags and the command is pushed into the SRQ.

If the Tag FIFO is empty then the first tag is placed in a bypass path around the Tag fifo, to make sure that the tag can be pushed to the write data flow the same cycle as data which arrives the cycle after the control flit. If the tag FIFO is not empty then the tags flow in like normal. After receiving the control flit after the data, bad data bits are checked and used in conjunction with the CRC valid to create the CRC valid vector sent to the sequencer. When data crc is confirmed good the entries which correspond to those data flits are popped from the FIFO and pushed into the TLXT.

1.4.2 Early Write Done

In early write done mode the write_response for mainline memory writes (not config', not mmio, not partial) will be queued

once all the data for the transfer has arrived with good crc. A good completion is assumed at this time.
When not in early write done mode (or for config', mmio or partial writes) the write_response will be queued when the SRQ signals srq_tlxr_wdone or the MMIO signals mmio_tlxr_wr_buf_free. At this time any error will be sent in the response.

The release of dcp1 credit is always tied to the srq_tlxr_wdone / mmio_tlxr_wr_buf_free signals independent of early write done mode.

	write_mem (dram)	pr_wr_mem (to dram)	write_mem.be (to dram)	pr_wr_mem (mmio)	config_write	mem_cntl		pr_wr_mem to pad mem buffer
	Early/Late sel by ewd(0)	Early/Late sel by ewd(1)	Early/Late sel by ewd(1)	Late	Late	(not expected)		(not decoded) goes to dram instead

1.4.3 A5 / Column address 2 Masking

OpenCAPI address bit A5 is sometimes masked off by TLXR before the address is passed to SRQ. In some cases this is because it isn't supplied by the OC command, and in some cases it is a design choice made between TLXR and SRQ. This table shows what TLXR does.

The target of the operation matters too. Eg a pr_wr_mem to mmio will not mask A5 but the same command to a DRAM target will have A5 masked out (driven to '0'). This is shown in the table.

Note that OC A5 appears in the mainline memory format version of tlxr_srq_cmd as Column address 2 – bit 43.

Command (target)	OC supplies A5 ?	A5 passed to SRQ
wr_mem.be (ddr)	no	no
pr_wr_mem (pad pattern)	yes	no
pr_wr_mem (ddr)	yes	no
write_mem (ddr)	no	no
All others, (reads,config,mmio)	yes	yes

1.4.4

1.4.4.1

1.4.5 Address Translation (copied from w_addr_xlat document 10 jan 18)

The memory translation logic receives the address from the frame decoder. The address is translated to a physical memory address (rank/bank/row/col).

Depending on the memory type and size, certain physical address bits won't be valid. The following valid bits are needed to consider the different DIMM possibilities.

- Slot 0 valid
- Slot 1 valid
- Value of D bit for Slot 0 memory (set to 0 if lower region of memory; 1 if upper)
- Value of D bit for Slot 1 memory (set to 0 if lower region of memory; 1 if upper)

A separate slot 0 and slot 1 valid bit is needed for each of the following:

- Master Bit (M0)
- Master Bit (M1)
- Slave Bit 0 (S0)
- Slave Bit 1 (S1)
- Slave Bit 2 (S2)
- Row Bit 15 (R15)
- Row Bit 16 (R16)
- Row Bit 17 (R17)

Each of the physical address bits are mapped to a MC address bit through a configuration/mapping mux. Some bits (like the row bits) are always hard-coded to specific bits. The remaining bits can be assigned based on the configuration specified and hashing desired. For example, if the intent was to increase page hits or reduce the number of banks active at a time, more column bits might be assigned to lower order address bits.

For bits that exist on both DIMM slots, the physical bit must be mapped to the same CAPI address bit. For bits that exist only on one DIMM, the bits should be assigned as highest-order address bits. An exception might is if each DIMM has a unique bit. For example, if DIMM0 has a M1 bit and DIMM1 has a S2 bit, both of those bits could be mapped to the same bit and placed anywhere in the address map.

OMI ASIC Address Translate Address Assignment Options		
Physical Address (DRAM) Bit	CAPI address bit choices	Bit Choices that will enable fast_activate on reads in a template_1 slot 0
DIMM / side	Can be assigned to true or complement of CAPI_addr(39-31) or	CAPI addr(39-32) or CAPI(11)

bit	CAPI_addr(15-5).	
Master Rank M0,M1 (CSN)	If present on device, can be assigned to CAPI_addr(39-31) or CAPI_addr(15-5)	M0 CAPI(39 - 33)
		M1 CAPI(39-32) or CAPI(11)
Slave Rank S0,S1,S2 (CID)	If present on device, can be assigned to CAPI_addr(39-31) or CAPI_addr(15-5)	S0 CAPI(39-34)
		S1 CAPI(39-33)
		S2 CAPI(39-32) or CAPI(11)
Bank Group BG0,BG1	Can be assigned to CAPI_addr(39-31) or CAPI_addr(15-5)	BG0 CAPI((12,10,8 or 6)
		BG1 CAPI(11,9,7 or 5)
Bank B0,B1	Can be assigned to CAPI_addr(39-31) or CAPI_addr(15-5)	B0 CAPI(14,12,10 or 8)
		B1 CAPI(13,11,9 or 7)
Column C9-C4	Can be assigned to CAPI_addr(39-31) or CAPI_addr(15-5)	No requirement on column bit mapping
Column C3	Fixed to CAPI_addr(6)	As above
Column C2	Reads always driven by CAPI addr bit 5 by TLXR and masked out by SRQ as appropriate. Driven ‘0’ for writes.	As above (Not firmware programmable anyway)
Column C1	Fixed '0'	As above
Column C0	Fixed '0'	As above
Row R14-R0	Fixed to CAPI_addr(30-16)	Fixed. No requirement
Row R17-R15	If present on device, can be assigned to CAPI_addr(39-31)	R17 CAPI(39-34)
		R16 CAPI(39-33)
		R15 CAPI(39-32)

<u>NOTES:</u>

- All DDR4 accesses are burst-length-8, and column bits C0-C2 are used to indicate burst within the BL8 access.
- A configuration bit exists to enable critical-ow-first which assigns column C2 to CAPI_addr(5) for reads.
-
- The decodes shown in the above table are implemented using a zero field value for the rightmost address option and counting up with no breaks. EG a DIMM bit field of “00001” results in OpenCAPI address bit (6) being selected.
- All decoding control fields selecting OpenCAPI address bits are five bits wide and each value of these fields maps consistently to a particular OpenCAPI address bit. Thus for R17-R15 , the control value to select OpenCAPI address bit 32 would be “01100”. For R1X maps, values “00000” to “01010” are not used.
- The requirement for column bit 3 to be driven by OpenCAPI bit 6 Firmware is required to set up the column bit 3 selection field to “00001” using the normal mechanism.
- If only one DIMM is valid then the DIMM bit is forced to that DIMM.
- C2 is driven by TLXR from OpenCAPI bit 5 for mmio and config ops and for memory rd_mem and pr_rd_mem . For any memory write TLXR drives col2 <= '0' .
- The fast_activate mechanism (which provides reduced read latency) will be invoked if all the

conditions in the third column of the table are met, AND if the read arrives in slot 0 of a template 1 control flit. If these conditions are not met, the mechanism will not be activated.

1.4.6 Timing Diagram

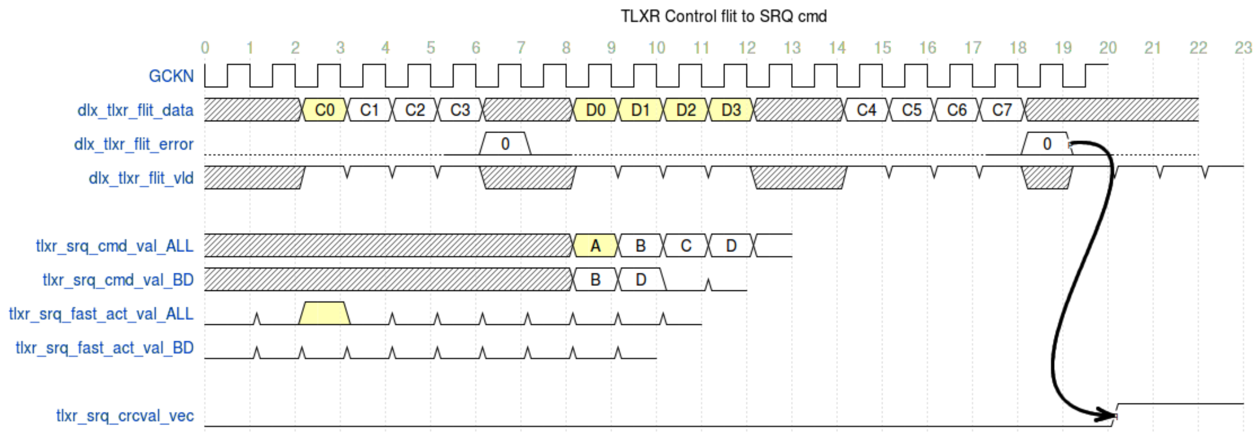
TLXR is a (very slightly flexible) pipeline from the DLX OpenCAPI input to SRQ. The essential signals are shown in the diagram below. The diagram shows a write sequence, and this description relates to memory reads and writes with DDR as the target. Please refer to the diagram.

OpenCAPI flits arrive in four beats, each 16 bytes , and the 64 byte control flits supported by the OMI ASIC can contain between zero and four command packets . The packets are 14 bytes each so the alignment with the beats is not 1:1. the CRC which covers a control flit and any preceeding data flits is not checked until late in the cycle after the last data beat – which constrains the TLXR command timing.

Consider that in the diagram packet A is a write_mem. It is wholly contained in control flit beat C0. TLXR will send the command to SRQ using the signal “tlxr_srq_cmd” six clocks after C0. TLXR packs the packets in the control flit into the four clocks shown as 9,10,11 and 12. As an example the diagram shows the situation if the control flit had contained only packets B and D (packets at A and C were NOP in this case).

The diagram shows two dead clocks after the first control flit – but that could be any number including zero – I added them just to emphasise the new flit. That new flit is 64 bytes of data for the write A . TLXR writes this data into a buffer in the WDF component. TLXR then has to wait until the CRC for that data has been checked. That happens at clock 19 – where the error signal must be '0'. (If it was not '0' then the data would have to be re-transmitted) . In the normal case the CRC is good and TLXR tells SRQ that the data for write_mem A is good in the buffer using the tlxr_srq_wrbuf_crcval_vec signal at clock 20.

The same diagram can also be used to describe the read timing. Imagine that the packet A is now a rd_mem. Subject to the template used, TLXR sends a fast_activate signal and address in clock 2, which allows SRQ to do a precharge and open the required DDR bank while it waits for the read itself. As described above, the read command will be sent to SRQ six clocks after C0. At that point, TLXR is finished with the read.



1.4.7 Reset Requirements

The TLXR requires a single synchronous reset.

1.5 Error Reporting and Handling

1.5.1 OpenCAPI Specified Errors

The OMI ASIC supports the following lists of fatal and non-fatal errors defined in the OpenCAPI 3.1 specification. Some of the required fatal errors are supported but unlikely to appear during normal operation given the commands and responses supported by the OMI.

Description	Trigger	Severity	Expected?	Action Taken
Bad Data Received	A data flit received in the previous group of data flits had bad data. Marked in the bad data field of DL content.	Non-Fatal	Yes	
Bad opcode and Template combination	Opcode indicates packet size not supported by template indicated.	Fatal	Yes	
Bad Response Received	AFUTag in response is not matched to the AFUTag in any commands send previously.	Fatal	No	
Bad Template x'00' Format	Template x'00' specified and slot 0 does not contain a nop or credit return.	Fatal	Yes	
Control Flit Overrun	Destination is unable to accept a subsequent control flit.	Fatal	N/A	
Illegal credit return Location	Credit Return command is not located in slots (11:10) for template x'07' or slots(1:0) for other templates.	Fatal	Yes	
PA outside of specified bounds	Specified PA is out of the configured bounds.	Fatal	Yes	
Reserved Opcode used	Opcode specified for specific command is a value reserved by the architecture.	Fatal	Yes	
Reserved field value used	On receipt of a comand or response packet it is determined that a field specification contains an architecturally reserved value.	Fatal	No	
Unexpected data flit	Calculated number of data flits from Run length is mismatched to number of data flits calculated from commands and responses.	Fatal	Yes	
Unsupported Template Format	Control flit received indicates a template format which is not supported by the OMI ASIC TLXR. Templates x'02', x'03',x'05', x'06', x'08', x'09'	Fatal	Yes	

1.5.2 ECC

ECC is generated on data stored in the two arrays within the TLXR. One stores OpenCAPI tags and the other stores a list of buffer numbers in order to keep write commands and their associated data in sync. ECC is checked when the array is read. If correction is required and possible then an informational "error" bit is set, and the operation proceeds normally. If correction is required but not possible, a fatal error bit is set and further downstream commands are ignored. Responses for operations that had already completed successfully will continue to be returned.

The ECC scheme will correct any single bit error, and detect all two bit errors.

1.5.3 Parity

Parity is checked on incoming signals from the DLX and is generated on all critical outputs to the sequencer. Parity on data is replaced by ecc (errors are maintained) before the data is written into the write buffers.

On receipt of a control function parity error internal to the TLXR, the commands associated are dropped, a FIR bit is set and typically a reset will be required, along with software cleanup, before operations can resume.

1.5.4 Hardware Error Checkers

All checkers inside TLXR assert bits of the tlxr_tlxt_errors (58 downto 0) bus which sets bits in a TLXT SCOM satellite register. Each error has an associated mask bit, and a pointer to the fir bit that an unmasked error should contribute to. The FIR bits are listed here, with the error bits that they are derived from below.

Note that the standard register macro used is big-ended whereas OpenCAPI is little ended. So if you are poking around in the registers themselves, the bit numbers will be 63 - the OpenCAPI number.

Fir bit pointer (inside regs logic)	Fir bit in OpenCAPI environment	Name	Description
16	47	TLXR_Shutdown	TLXR is in shutdown mode and most packets are being dropped (maybe not credit updates). I²C will be required to read the registers if this bit is on.
17	46	TLXR_BAR0_or_MMIO_non_fatal	Bar0 parity error or MMIO fail response sent (non fatal versions).
18	45	TLXR_OC_Malformed	An OpenCAPI control flit is malformed.
19	44	TLXR_OC_Protocol_Error	Ope Protocol violations all configurable as fatal/non-fatal using the same configuration bit.
20	43	TLXR_Addr_Xlat	Unworkable logical to physical memory address translation
21	42	TLXR_Metadata_unc_dparr	Data or metadata parity/ecc has failed inside the OMI ASIC and SUE has been written to memory
22	41	TLXR_OC_Unsupported	OpenCAPI has sent us something which the OMI ASIC does not support in its current operating mode.
23	40	TLXR_OC_Fatal	Fatal OC packet received.
24	39	TLXR_Control_error	Control logic error possibly caused by design bug , bit flip or invalid inputs to TLXR.
25	38	TLXR_Internal _error	Catastrophic internal TLXR logic failure. Should never happen by design. Cosmic ray or broken chip event.
26	37	TLXR_Informational	One of the ecc checkers has successfully corrected something, and other information-only event . No firmware recovery needed.
27	36	TLXR_Trace_Stop	Not an error. Information that memctl trace stop received

Here is a list of the error bits:

Name	Description	TLXR bitno	TLXT bitno	FIR Bit	Expected Action	Sim Injection
Reserved		58	5	16		
TLXR_Shutdown	TLXR has detected a fatal error and is dropping packets controlled by the shutdown control settings	57	6	16	In order to read this bit as a '1' I²C register access will be required since the OC interface is shut down.	Any fatal error
mmio_bad_wr_resp_nf	MMIO signalled error for write. Fatal/Non-Fatal controlled by "tlxcfg1_shutdown_on_mmio_bad".	56	7	17	Non-Fatal(default). TLXR passes the fail response code out in a mem_wr_fail response.	Clear "tlxcfg1_shutdown_on_mmio_bad" then do mmio write with a length of 16 – (one example of many possible errors). (Might save time to do bit 10 afterwards too).
bar0_perr_nf	Bar0 parity error Fatal/Non-Fatal controlled by "tlxcfg1_shutdown_on_bar0_bad" .	55	8	17	Non Fatal (default). The parity is checked on every cycle so it may be possible to rewrite the bar 0 base register and recover.	sticks required.
Bad_template_opcode_combo	The format off the control flit in error .Opcodes indicate packet sizes larger than the template allows. OC fatal error "Bad opcode and template combination"	54	9	18	Fatal . Reset required. Signature format 0	Eg write_mem.be in a template 1. Anything where the op is too big for the slot.
Reserved		53	10	18		
Bad_credit_return_slot	Return_tlx_credit not in slots 1:0. OC fatal error "illegal return credit command location"	52	11	18	Fatal. Reset required. Signature format 0	Send credit return in a disallowed slot – ie not 1:0 in any supported template..
bad_template_0_format	Slot 0 must contain a nop or a return_tlx_credits opcode. OC fatal error "Bad template x'00' format".	51	12	18	Fatal. Reset required. Signature format 0	Eg write_mem in template_0 slot 0.

Name	Description	TLXR bitno	TLXT bitno	FIR Bit	Expected Action	Sim Injection
Reserved_field_value	A Reserved field value has been used. OC fatal error "Reserved field value used"	50	13	18	Fatal. Reset required. Signature register contains detail. Signature format 1	pr_rd/wr_mem or config op with length field "11-" invalid intrp_resp/rdy codes
data_seq_err	Data carrier alignment problem. A 64B data flit is received when the current write address is on an odd 32B boundary.	49	14	18	Fatal . Reset required. A mem_wr_fail with response code "1110" is sent for the write and subsequent commands are dropped.	64B write followed by 32B data in T7 followed by a data flit.
rd_bdy_err	Invalid address / length combination on pr_rd_mem. Fatal/Non-Fatal controlled by "tlxcfg1_shutdown_on_opt_err".	48	15	19	Non-Fatal(default) mem_rd_fail with response code of "1011" is returned. Fatal. This is effectively regarded as a malformed packet. No response is sent and subsequent commands are dropped as for any other fatal error. Signature format 1	pr_rd_mem with length of 8 to address xxxx4. (for example)
wr_bdy_err	Invalid address/length combination on a write – either pr_wr_mem or write_mem. Fatal/Non-Fatal controlled by "tlxcfg1_shutdown_on_opt_err".	47	16	19	Non-Fatal(default) mem_wr_fail with response code of "1011" is returned. Fatal. This is effectively regarded as a malformed packet. No response is sent and subsequent commands are dropped as for any other fatal error. Signature format 1	pr_wr_mem with length of 8 to address xxxx4. (for example) or a 128B write_mem to odd 64B address (another example)
Reserved		46	17	19		
bad_intrp_resp_tag	Intrp_resp with tag /= [1234]. Fatal/Non-Fatal controlled by "tlxcfg1_shutdown_on_opt_err".	45	18	19	Non-Fatal (default). Packet ignored apart from VC0 credit release. Fatal.	OC Intrp_resp with tag /= [1-4]

Name	Description	TLXR bitno	TLXT bitno	FIR Bit	Expected Action	Sim Injection
	Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_opt_err”.				VC0 credit released and subsequent commands dropped. Signature format 1	
bad_memctl	Mem_cntl packet received with unknown flag value. Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_opt_err”.	44	19	19	Non-Fatal (default). Passed on as normal for SRQ while TLXR sends mem_cntl_done response of “1110” . Fatal. subsequent commands dropped. Signature format 1	Mem_cntl with unallocated flag. Flags 0-8 are defined.
Reserved		43	20	19		
Reserved		42	21	20		
Reserved		41	22	20		
address_dropped	A ‘1’ bit in the OpenCapi address for a pr_rd_mem or rd_mem is not mapped to any dram address/select pinto physical. Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_opt_err”. Effectively OC Fatal error “PA specified is out of bounds”	40	23	20	Non-Fatal. (default) mem_rd_fail or mem_wr_fail with response code of “1110” queued as appropriate. Fatal As above, plus subsequent commands dropped. Signature format 2	Send any type of DDR write with an address higher than the maximum size implied by the address translation controls.
addr_xlat_error_rd	Address translate error on a DDR read – both sides selected by address translate for a single address. Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_opt_err”.	39	24	20	Non-Fatal. (default) mem_rd_fail with response code of “1110” queued. Fatal As above, plus subsequent commands dropped.	Set up address translate with both slots enabled and overlapping selects for them. Read DDR .
addr_xlat_error_wr	Address translate error on a DDR	38	25	20	Non-Fatal.	Set up address translate with both slots

Name	Description	TLXR bitno	TLXT bitno	FIR Bit	Expected Action	Sim Injection
	write – both sides/dimms selected by address translate for a single address. Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_opt_err”.				mem_wr_fail with response code of “1110” queued. Fatal As above, plus subsequent commands dropped.	enabled and overlapping selects for them. Write DDR.
dataflow_perr_nf	Parity/ecc error in the internal TLXR dataflow. Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_dflow_err ”.	36	27	21	Non Fatal. (default) Treated by SRQ/MMIO as if OC had signalled BDI for this data. SUE is written if target is memory.	!!! sticks required.
metadata_unc	ECC uncorrectable corruption of stored write metadata. Note that this error can occur even for operations that do not use write metadata (eg config write) if a double bit-flip occurs within the relevant latched ecc word.	35	28	21	Non Fatal. SUE is written if target is memory.	!!! sticks required.
Unsupported pr_rd_ddr	Partial read of DDR when TLXT is in low latency mode	34	29	22	Non-Fatal Packet dropped and Unsupported length “1001” response code sent in mem_rd_fail.	pr_rd_mem with ddr as the target when tlxt is in low latency mode.
Unsupported_pad_mem	pad_mem	30	33	22	Non-Fatal Packet dropped and mem_wr_fail with response code of “1110” sent.	Use pad_mem

Name	Description	TLXR bitno	TLXT bitno	FIR Bit	Expected Action	Sim Injection
Unsupported_write_length	Unsupported length for write_mem packet. The only length that the OMI ASIC does not support is 256 bytes.	28	35	22	Non Fatal. WR fail Response “1001” queued. Data for this write dropped so that subsequent writes use correct data.	OC 256 byte write using write_mem.
Unsupported_read_length	Unsupported length for rd_mem packet. The only length that the OMI ASIC does not support is 256 bytes.	27	36	22	Non Fatal. RD fail Response “1001” queued using TLXR-SRQ interface.	OC 256 byte read using rd_mem.
Unsupported_opcode	An opcode that we do not decode has been presented . OC Fatal error “Reserved opcode used”	26	37	23	Fatal. Reset required. An opcode in a valid template has not been decoded. The credit is therefore unknown and we cannot recover. Signature format 0	Try an operation we don't support. AMO_RD for example.
Unsupported_template	Unsupported OC template – not 0147 or A. OC Fatal error “Unsupported template format”	25	38	23	Fatal . Reset required. Signature register format 0.	OC template other than 0147A
Buff_cntl_perr	One of the 64 internal TLXR buffer FSM's has suffered a a parity error. This is a catastrophic failure which requires a broken chip or bit flip to occur.	23	40	24	Fatal. Reset required.	!!! sticks required.
d_list_unc	Internal TLXR ECC uncorrectable buffer control array corruption.This is a catastrophic failure which requires a broken chip or bit flip to occur.	22	41	24	Fatal. Reset required.	!!! sticks required.
tagstore_unc	Internal TLXR ECC uncorrectable tag_store corruption. This is a catastrophic failure which requires a broken chip or bit flip to occur.	21	42	24	Fatal. Reset required.	!!!! sticks required.

Name	Description	TLXR bitno	TLXT bitno	FIR Bit	Expected Action	Sim Injection
b_mgmt_3	Internal TLXR buffer control logic parity error. This is a catastrophic failure which requires a broken chip or bit flip to occur.	20	43	24	Fatal. Reset required.	!!!sticks required.
Reserved		19	44	24		
flit_perr	Flit data parity error caused by DLX or TLXR internal error.	18	45	24	Fatal . Reset required.	Force parity error on flit data (has to be in valid cycle)
mmio_wd_perr	Detected parity error on MMIO wdone.	17	46	24	Fatal. Reset required.	!!!sticks required.
mmio_wd_inv	MMIO wdone for idle buffer	16	47	24	Fatal. Reset required.	!!!sticks required.
srq_wd_perr	SRQ_wdone parity error	15	48	24	Fatal – reset required.	!!!sticks required.
srq_wd_inv	SRQ wdone for idle buffer	14	49	24	Fatal – reset required.	!!! sticks required.
d_list_underflow	Data buffer control underflow. OC Fatal error “Unexpected data carrier”.	13	50	25	Fatal. Reset required.	Send more data flits (or data carrier in template_7/A) than required to satisfy the previous writes.
dataflow_perr_f	Parity/ecc error in the internal TLXR dataflow. Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_dflow_err”.	12	51	25	Fatal. (non-default) Treated by SRQ/MMIO as if OC had signalled BDI for this data. SUE is written if target is memory.	!!! sticks required.
d_list_overflow	Data buffer control overflow. The total amount of data required for the received write commands exceeds the OMI ASIC buffer capacity.	11	52	25	Fatal. Reset required.	!!!! sticks required.
mmio_bad_wr_resp_f	MMIO signalled error for write. Fatal/Non-Fatal controlled by “tlxcfg1_shutdown_on_mmio_bad”.	10	53	25	Fatal(non-default). TLXR passes the fail response code out in a mem_wr_fail response, and all subsequent packets dropped.	Set “tlxcfg1_shutdown_on_mmio_bad” then do mmio write with a length of 16 – (one example of many possible errors). (Might save time to do bit 56 afterwards too).

Name	Description	TLXR bitno	TLXT bitno	FIR Bit	Expected Action	Sim Injection
bar0_perr_f	Bar0 parity error Fatal/Non-Fatal controlled by "tlxcfg1_shutdown_on_bar0_bad" .	9	54	25	Fatal (non-default). The parity is checked on every cycle . An error will result in subsequent packets being dropped.	sticks required.
b_mgmt_1	A buffer is required for an incoming write, but all buffers are currently busy.	8	55	25	Fatal. Suspect a dcp1 credit problem.	!!!sticks probably required since vc1 credit is only 31.
b_mgmt_2	Two buffers are required for an incoming 128B write, but only one buffer is not busy.	7	56	25	Fatal	!!!sticks required.
addr_xlate_hole	In the address translate block, the OC Address map has a hole. TLXR expects DDR usage to be such that there are no address holes in the mapping.	6	57	26	Information only.	Set a translate configuration that does not us an OC address bit but does use a more significant bit.
bad_data_rxd	Bad data bits set for valid flit OC Non-fatal error "Bad data received"	5	58	26	Information only. Bad data has been signalled by OC for one or more data carriers.	Send BDI bits in control flit, or bad and valid data bits in T7 or TA..
metadata_corr	ECC error corrected in metadata	4	59	26	Information only.	!!! sticks required.
tag_buffer_corr	ECC error corrected in tagstore	3	60	26	Information only.	sticks required.
d_list_corr	ECC error corrected in data list.	2	61	26	Information only.	sticks required.
EPOW_signalled	TLXR received an emergency power down signal from srq.	1	62	26	All operations are ignored except for credit updates and write data. A reset is required to re-establish normal operation.	Part of a wider system so there must be top level system input I think.
Trace_stop	Memcntl trace stop flag =0001	0	63	27	Not an error but included here as a dedicated FIR is required.	Send mem_cntl with flag of "0001"

1.5.5 Mapping between OpenCAPI required errors and TLXR errors

Note that in the case of multiple errors detected at the same time, the signature register is loaded with the lowest active error in this table.

OpenCAPI Error	TLXR error reporting bit number	FIR bit number /name (in mmio number is 63-n)	Signature information (for error not FIR)
Bad data received	5 (Non-Fatal)	26 - TLXR_Informational	None (optional)
Bad opcode and template combination	54 (Fatal)	18 - TLXR_OC_Malformed	31:26 Template 23:16 opcode 15:12 slot 3:0 “1000”
Bad response received (variant 0)	45 (OptionallyFatal)	19 - TLXR_OC_Protocol_Error	31:16 Tag 3:0 “1001”
Bad template x‘00’ format	51(Fatal)	18 - TLXR_OC_Malformed	31:4 Slot 0 3:0 “1010”
Illegal return credit comm-and location	52(Fatal)	18 - TLXR_OC_Malformed	31:26 Template 15:12 slot 3:0 “1011”
PA specified is out of bounds	40 (Optionally Fatal)	20 - TLXR_Addr_Xlat	63:13 OC 63:13 12 OR of OC(12:5) 11:4 Opcode 3:0 “1100”
Reserved field value used	50(Fatal)	18 - TLXR_OC_Malformed	23:16 Opcode 15:8 bit offset of field 3:0 “1101”
Reserved opcode used	26(Fatal)	23 - TLXR_OC_Unsupported_GP1	23:16 Opcode 3:0 “1110”
Unexpected data carrier	13(Fatal)	25 - TLXR_Control_HW_Error	None
Unsupported template format	25(Fatal)	23 - TLXR_OC_Unsupported_GP1	Template number 31:26 Template 3:0 “1111”
Bad template usage	These are OpenCAPI required errors which cannot occur in the OMI ASIC.		
Bad response received (variant 1)			
Control flit overrun			

1.5.6 Simulation Sticks for some errors

event	comments
<e2,chip,unit,><e02>	Do some writes and while they are going on flip any 1 bit of buff_ctl.array_dout_raw when buff_ctl.array_read is '1'. see bit e22 as well
<e1,chip,unit,><e03>	Tag store correctable ecc – wait for resp_cnt(2)='1' and corrupt any oneo bit of tag_store_out(23 downto 0) for that cycle.

<e1,chip,unit,> <e04>	Flip any one bit of meta_store_q(47 downto 0) in cb_tlxr_mac at any time (not during reset).
<e1,chip,unit,> <e07>	Stick buffer_busy(63 downto 20) all '1' and buffer busy(18 downto 0) all '1' as well. Then do a 128 byte write.
<e1,chip,unit,><e08>	Try to use a buffer when all buffers are in use stick buffer_busy(63 downto 4) to x"FFFFFFFF_FFFFFFF" and run some writes (9 x 128 byte writes outstanding should make it fail) (- see HW446412 which looks like it tests this error bit indadvertently ! (error bit looks Ok)).
<e2,chip,unit,><e09>	Use stick to flip any bit in CFG_F1_CSH_MMIO_BAR0 or CFG_F1_CSH_MEMORY_SPACE when control bit "tlxcfg1_shutdown_on_bar0_bad" is set (ie Fatal). Also see error bit 55 (with the control bit clear).
<e1,chip,unit,><e11> ***	need to poke in enough writes to promise more than 64 buffers worth of data here but it's hard to do that without setting off error 8 (impossible to allocate buffer). I recommend that you stick MB_SIM.TLXR.BUFF_CTL.FULL to '1' and then do a write to create this error.
<e1,chip,unit,><e14> ***	Start from a reset. Send less than 8 writes of any type and after the last write, stick srq_tlxr_wdone_tag(0 to 5) to any value > "001000" and < "110111" when srq_tlxr_wdone_val = '1'
<e1,chip,unit,><e15> ***	Do some writes to ddr. Wait for srq_tlxr_wdone_val = '1' and flip SRQ_TLXR_WDONE_P with a stick.
<e1,chip,unit,><e16> ***	start from a reset. Send less than 8 MMIO writes and after the last write, stick any two bits out of mmio_tlxr_wr_buf_tag(5 downto 3) to '1' and '1' when mmio_tlxr_wr_buf_free = '1'.
<e1,chip,unit,><e17> ***	Do some writes to mmio/config wait for mmio_tlxr_wr_buf_free = '1' and flip mmio_tlxr_wr_buf_par with a stick.
<e1,chip,unit,><e20>	Flip any bit of pop(43 downto 0) at any time.
<e1,chip,unit,><e21>	Tag store uncorrectable ecc – wait for resp_cnt(2)='1' and corrupt any two bits of tag_store_out(23 downto 0) for that cycle.
<e1,chip,unit,><e22>	Do some writes and while they are going on flip any 2 bits of buff_ctl.array_dout_raw when buff_ctl.array_read is '1'. (see bit e02).
<e1,chip,unit,><e23>	Flip any bit of buf_state_d(511 downto 0) at any time.
<e1,chip,unit,><e35>	Flip any two bits of meta_store_q(47 downto 0) in cb_tlxr_mac at any time (not during reset).
<e1,chip,unit,><e36>	Flip any bit except (65 or 0) of tlxr/ecc_data_in(129 downto 0). control bit tlxcfg1_shutdown_on_dflow_err must be clear for this error (non-fatal). If it's set you should get the fatal version – error 12.
<e1,chip,unit,><e43>	(I saw this work in a JIRA for something else). Deliver the first 32 bytes of data for a 64 or 128 byte write using a template-7. Then before a second template-7 delivers 32 more bytes of data, send 64 bytes of data in a data flit.
<e1,chip,unit,><e55>	Use stick to flip any bit in CFG_F1_CSH_MMIO_BAR0 or CFG_F1_CSH_MEMORY_SPACE when control bit "tlxcfg1_shutdown_on_bar0_bad" is clear (ie Non- Fatal). Also see error bit 9 (with the control bit set).

1.5.7 Error Signature register

A 64 bit register is provided which provides an error signature for some of the errors. The format of this register's content is shown in this table (not all format fields apply to all opcodes):

ERROR	Figtree bit and name	Format of Error signature[63:0]
(54) *	9 Bad_template_opcode_combo	63:56 slot 12 opcode 55:48 slot 10 opcode 47:40 slot 8 opcode 39:32 slot 4 opcode 31:24 slot 0 opcode 23:22 “00” 21 template_A data valid bit 20 template_A data bad bit 19 template_7 data valid bit 18 template_7 data bad bit 17:12 template number 11:0 x”001”
(52)*	11 Bad_credit_return_slot	As above
(51)*	12 Bad_template_0_format	As above
(50)	13 Reserved_field_value	63:56 opcode 55: '0' 54:52 length 51:44 PA(7:0) (not all opcodes have all bits) 43:40 Intrp_code or PA[27:24] 39 '0' 38:23 Intrp_tag 22:3 zeros 2:0 slotX*
(48)	15 RD_bdy_error	As above
(47)	16 WR_bdy_error	As above
(45)	18 Bad_intrp_resp_tag	As above
(44)	19 Bad_memctl	As above
(40)	23 Address_dropped	63:11 PA[63:11] 10:3 opcode 2:0 “010”
(28)*	37 Unsupported_opcode	As format for “Bad_template_opcode_combo”
(25)*	38 Unsupported_template	As format for “Bad_template_opcode_combo”

* slotX : “100” = slot0 , “101” = slot4, “110” = slot8 (slot 10 for T7) , “111” = slot 12

1.5.8 Error Injection Capabilities

The TLXR macro contains no error injection capabilities.

1.6 Shutdown

After some errors, or when an EPOW (emergency power down imminent) is received, TLXR will go into a mode in which some or all commands are dropped. This is known as “shutdown” mode – and there are four submodes described briefly in paragraph 1.8.

If a shutdown is caused by EPOW, TLXR will drop any incoming commands except for credit updates . This allows reads which are already underway to complete.

If a shutdown is caused by errors that make it unsafe to continue, the idea is to prevent corruption of commands that are already underway. Control bits determine:

- Some control over which errors are used to enter shutdown mode. A core group always causes shutdown (unless the "debug-only" never shutdown is in operation. Then a subgroup of FIR19 and FIR 20 errors can be enabled to cause shutdown too. And 3 individual errors can also be used or not to go into shutdown.
- If a shutdown happens because of errors, control bits can control whether or not credit updates are dropped. If the shutdown happened because of EPOW, TLXR will not drop credit updates.
- Control bits can prevent shutdown mode either for errors, or for errors and EPOW. It is not expected that this would be useful in normal operation. It may be convenient in some debug situations though.
- Shutdown modes only apply to control flits. Data for writes that arrived before the shutdown will be processed normally.

1.7

1.8 Control bits

There are sixteen chicken switch control bits that affect TLXR operation. They are contained int the TLXCFG1_tlxr_ctrl register and listed in the following table :

(all bits default to '0', and a "fatal" error means that TLXR will enter its shutdown mode, ignoring new operations.)

0 (15)	Slow_clock	TLXR needs to know the clock rate to use for its long back-off timers associated with retrying interrupts. When this bit is '0' a frequency of 1600MHz is assumed and when '1' a frequency of 1333MHz is assumed.
1:2 (14:13)	Shutdown mode	"00" - Always process credit updates when shutdown. "01" - Error shutdown does not process credit updates . EPOW shutdown does. "10" - Shutdown only for EPOW (and process credit updates) "11" - Never shutdown (debug use only – note that this does not control SRQ. SRQ will respond to EPOW by processing commands while it is not idle. When it runs out of commands, it will then ignore subsequent commands. Note also that this mode does not inhibit the informational FIR bit to indicate that EPOW has been received.
3 (12)	Shutdown_on_opt_err	When set, the errors feeding FIR bits 19 and 20 will be treated as fatal. Otherwise not.
4 (11)	Shutdown_on_mmio_bad	When clear, a bad write response from mmio will be non-fatal within tlxr (informational only). When set, it's fatal and shutdown occurs.

5 (10)	Shutdown_on_dflow_ err	When clear, an internal TLXR dataflow parity error will be non-fatal (informational only). When set, it's fatal and shutdown occurs.
6 (9)	Shutdown_on_bar0_b ad	When clear, a parity error on BAR0 will be non-fatal within tlxr (informational only). When set, it's fatal and shutdown occurs.
8:11 (7:4)	Debug_sel_ls	Select for ms half of tlxr debug_bus.
12:15 (3:0)	Debug_sel_ms	Select for ls half of tlxr debug bus.

1.9 Debug Capabilities

1.9.1 FEDC bits (first pass 11 nov17)

partial_flit_count_q(1downto 0)
data_flits_owed_q
tlxr_srq_cmd_vld
tlxr_srq_cmd(59 to 63)
tlxr_wdf_wr
translating
flit_q (463 downto 460)
tlxr_tlxt_write_resp_val
tlxr_tlxt_dcp1_release
tlxr_tlxt_consume_dcp1
tlxr_srq_memcntl_sync

1.9.2 Debug Bus

The 88 bit bus is split into two halves, bits (87 downto 44) and bits (43 downto 0). Each half can select the following groups of signals based on tlxt_tlxr_control(7 downt 4) for the top half and tlxt_tlxr_control(3 downto 0) for the bottom half.

Control 4:7 = “0000”	Control 3:0 = “0000”	Signal traced
87:86	43:42	partial_flit_count_0_ne
85	41	coded_tplate
84	40	dlx_tlxr_flit_vld
83:76	39:32	flit_q(7 downto 0)
75::68	31:24	flit_q(119 downto 112)
67:60	23:16	flit_q(231 downto 224)
59:52	15:8	flit_q(287 downto 280)
51:44	7:0	flit_q(343 downto 336)

Control 4:7 = “0001”	Control 3:0 = “0001	Signal traced
87:86	43:42	partial_flit_count_q(1 downto 0)
85	41	tlxr_srq_cmd_val_i
84:73	40:29	b_nos_4_srq(4 to 15)
72:70	28:26	length_format_2(2 downto 0)
69:67	25:23	length_format_3(2 downto 0)
66	22	mask_a5
65	21	write_mem
64	20	data_xfer
63:52	19:8	flit_q(459 downto 448) – template+ bad bits+ run length
51	7	t7_bad_q
50	6	tA_bad_q
49:46	5:2	memcntl_tag(3 downto 0)
45:44	1:0	tlxr_tlxt_consume_dcp1_q(1 downto 0)

Control 4:7 = “0010”	Control 3:0 = “0010”	Signal traced
87:48	43:4	oc_addr(39 downto 0)
47	3	translating
46	2	tlxr_srq_fast_act_val_i
45	1	tlxr_srq_cmd_val_i
44	0	idle_flit ;

Control 4:7 = “0011”	Control 3:0 = “0011”	Signal traced
87	43	TLXT_TLXR_WR_RESP_FULL
86	42	dcp1_rls_notag
85	41	dcp1_rls_earlies
84	40	dcp1_rls_lates
83	39	or_reduce(tag_rls_earlies)
82	38	or_reduce(tag_rls_lates)
81	37	vc1_rls_patt
80	36	vc0_rls_memctl_q
79	35	vc0_rls_intrp
78	34	tlxr_tlxt_return_val_i
77:74	33:30	flit_q(11 downto 8)
73:70	29:26	flit_q(23 downto 20)

69:64	25:20	flit_q(37 downto 32)
63:58	19:14	flit_q(55 downto 50)
57:56	13:12	tlxr_tlxt_dcp1_release_i(1 downto 0)
55:54	11:10	tlxr_tlxt_vc0_release_i(1 downto 0)
53	9	tlxr_tlxt_consume_vc0_i
52	8	vc0_rls_intrp
51	7	mmio_tlxr_wr_buf_free
50:45	6:1	mmio_tlxr_wr_buf_tag(5 downto 0)
44	0	partial_flit_count_3

Control 4:7 = “0100”	Control 3:0 = “0100”	Signal traced
87:83	43:39	resp_cnt_in(4 downto 0)
82:59	38:15	oc_tag_dl_ecc(23 downto 0)
58:52	14:8	resp_cnt_out(6 downto 0)
51:44	7:0	tlxr_tlxt_intrp_resp_q(7 downto 0)

Control 4:7 = “0101”	Control 3:0 = “0101”	Signal traced
87:86	43:42	partial_flit_count_q(1 downto 0)
85:80	41_36	first_tag_q(5 downto 0)
79:44	35:0	op_held_q(35 downto 0)

1.10 Errata

1.10.1 Mem_rd_fail DL field wrong for two specific errors

This bug applies only to pr_rd_mem commands which target DDR. It does not apply to config reads or pr_rd_mem commands which target mmio.

If either one of two errors is detected, the mem_rd_fail packet returned by the OMI ASIC will contain "00" in its dL field but the OpenCAPI architecture defines this field as "01" . The rest of the fail response packet is as it should be.

The errors are :

Bit 24 "Address translate setup error on a read". Results in Fir 20. This bit is set when a read of DDR is attempted when the address translation setup and the read address cause zero or two side (aka
dimm) selects to be active.

and

Bit 23 - "A non-zero OC address bit has been dropped by translation." also routed to Fir 20. This bit is set when there is a non-zero address bit specified in the read packet, but the address translation is set up so that this bit is not mapped to any DDR address bit.

1.10.2 EPOW during template-A/7

If there is no outstanding write data for previous writes outstanding, and EPOW goes active in a one clock window before a template_A or template_7 with a write command and data, then the command is passed to SRQ but the data is dropped.

When more data arrives it will be written to the write buffer where the dropped data should have been, and once all data for that final write command has arrived, tlxr_srq_wrbuf_crcval_vec can indicate that the write may proceed. Incorrect data can then be written to memory.