OpenCAPI 3.0
TLX Reference Design

Version 1.0
20 August 2019

Contents

Contents 3
List of Tables 4
List of Figures 4
1 Overview 5
1.1 The Parser and the Framer 6
1.2 Limitations 8
1.3 Physical Implementation 8
1.4 Reference Diagrams 8
2 TLX Parser 10
2.1 Control and Data Flow 10
2.2 Template Support 11
2.3 TLX Parser Interfaces 12
2.3.1 DLX to TLX Flit Interface 12
2.3.2 TLX to AFU CAPP Command Interface (VC1) 12
2.3.3 TLX to AFU CAPP Configuration Command Interface (VC1) 14
2.3.4 TLX to AFU CAPP Response Interface (VC0) 15
2.3.5 TLX to AFU Data Interfaces (DCP0, DCP1) 17
2.3.6 TLX Parser Miscellaneous Ports 19
3 TLX Framer 20
3.1 Overview 20
3.2 Framer Data Flow Rules 20
3.3 Control and Data Flow and Pipeline Latency 21
3.4 TLX Framer Interfaces 24
3.4.1 TLX Framer Miscellaneous Ports 24
3.4.2 TLX Parser to Framer Credit Interfaces 24
3.4.3 AFU to TLX AP Command Interface (VC3, DCP3) 25
3.4.4 AFU to TLX AP Response Interface (VC0, DCP0) 27
3.4.5 AFU to TLX AP Configuration Response Interface (VC0, DCP0) 29
3.4.6 TLX to DLX Flit Interface 31
3.4.7 TLX Framer Configuration ports 31
3.4.8 TLX Framer Debug Ports 32
3.5 TLX Framer: Internal State Machines 33
4 Credit Management 35
5 Configuration 40
6 Error Detection and Handling 42

List of Tables

Table 1: TLX Parser – DLX to TLX Flit Interface 12
Table 2: TLX Parser – TLX to AFU CAPP Command Interface (VC1) 12
Table 3: TLX Parser – TLX to AFU CAPP Configuration Command Interface (VC1) 14
Table 4: TLX Parser – TLX to AFU CAPP Response Interface (VC0) 15
Table 5: TLX Parser – TLX to AFU CAPP Response Data Interface (DCP0) 18
Table 6: TLX Parser – TLX to AFU CAPP Command Data Interface (DCP1) 19
Table 7: TLX Parser – Miscellaneous Ports 19
Table 8: TLX Framer – Miscellaneous Ports 24
Table 9: TLX Framer – TLX Parser to Framer TLX Credit Interface 24
Table 10: TLX Framer – TLX Parser to Framer TL Credit Interface 24
Table 11: TLX Framer – AFU to TLX AP Command Interface (VC3, DCP3) 25
Table 12: TLX Framer – AFU to TLX AP Response Interface (VCO, DCP0) 27
Table 13: TLX Framer – AFU to TLX AP Configuration Response Interface (VCO, DCP0) 29
Table 14: TLX Framer – TLX to DLX Flit Interface 31
Table 15: TLX Framer – Configuration Ports 31
Table 16: TLX Framer – Debug Ports 32
Table 17: TLX Credit Management – TLX Credit Interface 35
Table 18: TLX Credit Management – TL Credit Interface 35
Table 19: TLX Credit Management – Credits for CAPP Commands 36
Table 20: TLX Credit Management – Credits for CAPP Responses 36
Table 21: TLX Credit Management – Credits for AP Commands and Responses 36
Table 22: TLX Credit Management – TLX/DLX Interface 37
Table 23: TLX Framer Template Configuration Ports 40
Table 24: TLX Parser Template Configuration Ports 41
Table 25: TLX Version Hardware Version Number 42
Table 26: TLX/DLX Error Information Interface 42
Table 27: Parser to Framer Error Information Interface 43

List of Figures

Figure 1: OpenCAPI Stack 5
Figure 2: TLX AFU Interface Overview 6
Figure 3: TLX Architecture Diagram 7
Figure 4: TL Control Flit and Packing Templates 9
Figure 5: TLX Parser FIFOs 10
Figure 6: TLX Parser Flow 11
Figure 7: Command Latency 13
Figure 8: Response Latency 16
Figure 9: Data Read Request 17
Figure 10: Data Latency 17
Figure 11: Multiple Data Request 18
Figure 12: Multiple Read Requests 18
Figure 13: TLX Framer Block Diagram 22
Figure 14: TLX Framer Latency Diagram 23
Figure 15: AFU Command with data 27
Figure 16: AFU Response with data followed by more responses 29
Figure 17: TLX Framer - Template Packer State Diagram 33
Figure 18: TLX Framer - Flit Framer State Diagram 34
Figure 19: TLX Flow Control Mechanisms 39

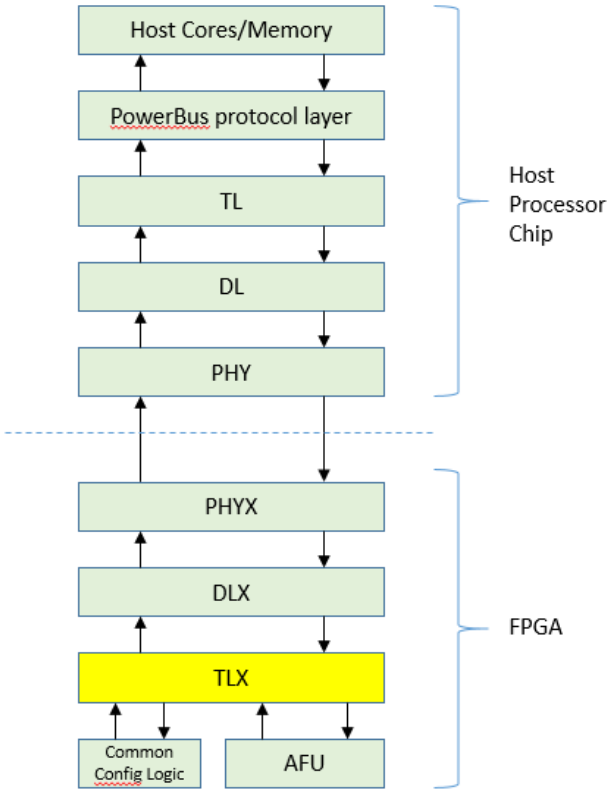
1 Overview

This document describes a logical implementation of the OpenCAPI™ external transaction layer. This logic is designed to support the operations described in the "OpenCAPI™ 3.0 Transaction Layer Specification". The transaction layer specification terms and architecture will be referenced multiple times throughout this document.

This reference design can be implemented in an FPGA or can be used as the basis for additional study and/or development.

The external transaction layer, described in this document, is abbreviated as the TLX (Transaction Layer – eXternal). The diagram below shows where the TLX fits in the OpenCAPI™ stack.

Figure 1: OpenCAPI Stack



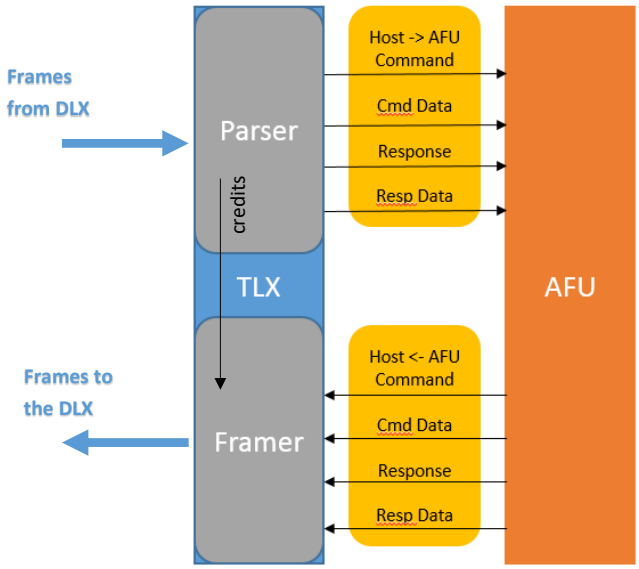
1.1 The Parser and the Framer

The TLX consists of two main sections: the “Parser” and the “Framer”.

The **Parser** handles the downstream traffic coming from the host. It receives DL control and data flits and parses those flits into CAPP commands, CAPP responses, and credit packets. The CAPP commands and CAPP responses are passed to the AFU. The credits are passed to the Framer.

The **Framer** handles the upstream traffic going from the AFU to the host. It receives AP commands and AP responses from the AFU, and credits from the Parser. The AP commands and responses, along with TL credits, are packed into control flits according to various packing templates. The control flits and data flits are put into frame sequence and sent to the DLX.

Figure 2: TLX AFU Interface Overview



The diagram on the following page shows more detail of the flow through the TLX:

Figure 3: TLX Architecture Diagram

1.2 Limitations

Here are some know limitations of this reference design:

- This design implements version 3.0 of the "OpenCAPI™ Transaction Layer Specification". There are many features detailed in later versions of the specification which are **not** supported in this reference design.
- This design does not support using Template x02 for packing upstream control flits. Only templates x00, x01, and x03 are supported for upstream traffic.
- The Framer does not support packing both AP commands and AP responses into the same control flit.

1.3 Physical Implementation

The logical description of this reference design is written in Verilog.

It has been implemented in a Xilinx FPGA along with DLX and AFU reference designs.

Clock speed:

LUT usage:

p-block area:

Block ram usage:

Other IP required?

It has been simulated with various simulators.

1.4 Reference Diagrams

Included below are some diagrams which may be helpful to those using this reference design.

Figure 4: TL Control Flit and Packing Templates

2 TLX Parser

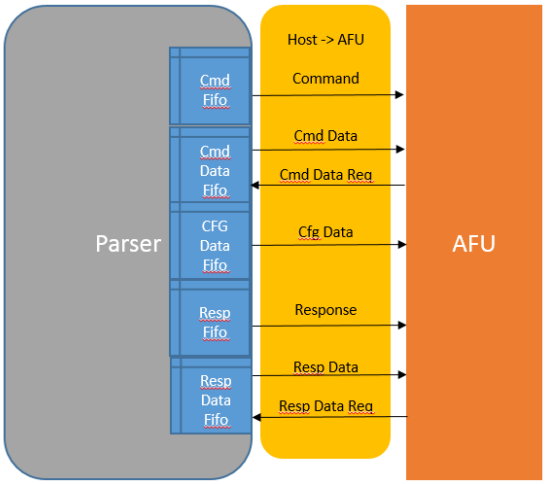
2.1 Control and Data Flow

This section contains a figure that illustrates the general flow for all commands, responses and data. The TLX determines if an operation is a command or response by which virtual channel the respective operation is assigned to. **NOTE: While the operations; xlate_done, intrp_rdy may be defined as commands, since they are assigned to virtual channel 0 the operations are treated as responses and will be sent to the AFU on the response interface.** As the flow chart in Figure 6 illustrates, as flits are received the first decision made is if the flit is a control flit or data flit. If it is a control flit it is parsed based on the template type and the parts are stored in the respective command and response FIFOs. If AFU credits are available and the data for the command or response has been stored, then the operation is passed to the AFU. If a flit is found to be a data flit, the flit is passed to the data pipeline and once the data flit's virtual channel is assigned the flit is then store in the respective data FIFO. The data is held in the FIFOs until the AFU requests data from the FIFO.

To support AFU configuration requirements, the AFU configuration commands and data are separated from the other VC1 and DCP1 commands and data. The configuration commands and data are sent to the AFU via a separate interface.

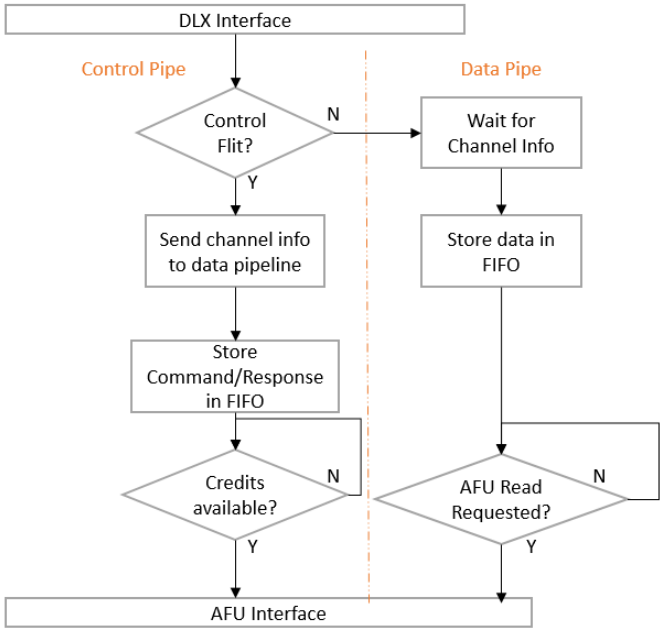
There are five FIFOs in the parser: commands, responses, config command data, other command data, and response data, as illustrated in the figure below. Each is parameterized and the depth can be altered to best suit the AFU's needs. The FIFO depths are set by the width of the respective address, so $fifo\ depth = 2^{addr\ width}$. The FIFO depth values are directly used to communicate to the host how many credits and how much buffer space is initially available. On power up there is an initial credit return sequence that is performed during link training. During this sequence the parsing logic passes the initial credits for all four FIFOs to the DLX framing logic, then to the host.

Figure 5: TLX Parser FIFOs



There are also credits between the AFU and TLX to throttle how many commands or responses can be sent from the TLX to the AFU. Data for corresponding commands and responses are not covered by credits as data is requested as desired by the AFU. If a command or response has data associated with it, then the command or response is held within the TLX until all the data for the command or response has been received and the data has been validated with good CRC. This ensures that once the AFU receives a command or response that the data is available. At that time the command or response is sent to the AFU, and the AFU must request the associated data. The TLX returns credits to the host once the command, response, or data leaves the FIFOs.

Figure 6: TLX Parser Flow



2.2 Template Support

The TLX Parser supports receiving the following templates: x”00”, x”01”, x”02”, x”03”. Template formats can be found in Chapter 6 of the OpenCAPI 3.0 TL Specification. Due to the host running at a higher clock frequency the TLX has rate limits for each of the supported templates. The rate limit is the number of TLX clock cycles needed between valid control flits to allow for parsing. For example, if the parser receives a control flit and the control flit is using template 1, template 1 can contain 4 TLX packets, so

TLX takes 4 cycles to parse the control flit. Therefore a minimum of 3 cycles would need to be inserted before the next control flit could be processed. The minimum rate limit for the supported templates follows this equation: $template\ rate\ limit = (\#of\ TLX\ packets\ in\ the\ template - 1)$.

During the rate limited cycles the only flits the TLX will accept are data flits, null credit return flits, and nop flits, all other flits will be dropped.

2.3 TLX Parser Interfaces

2.3.1 DLX to TLX Flit Interface

The DLX interface sends commands, responses and data to the TLX through the 512 bit wide flit signal. The link up signal from the DLX is passed straight through to the AFU to inform the AFU when the link is available. The crc error signal is used by the DLX to inform the TLX that at least one data flit in the current frame has bad crc. Once a crc error has been seen by the TLX, all pipelines are flushed, all necessary FIFO pointers are reset, and the TLX waits for the frame of data to be replayed.

Table 1: TLX Parser – DLX to TLX Flit Interface

Signal Name	Bits	Source	Description
dlx_tlx_flit_valid	1	DLX	Indicates valid flit is available. Validates dlx_tlx_flit.
dlx_tlx_flit	512	DLX	Data link layer packet.
dlx_tlx_link_up	1	DLX	Indicates OpenCAPI link has been trained.
dlx_tlx_flit_crc_err	1	DLX	Indicates a CRC error has been detected and the data is a frame should be dropped.

2.3.2 TLX to AFU CAPP Command Interface (VC1)

Note: This interface is used for all VC1 CAPP commands except configuration reads and writes (opcodes x11100000 and x11100001) There is a separate interface defined below for the configuration commands.

Table 2: TLX Parser – TLX to AFU CAPP Command Interface (VC1)

Signal Name	Bits	Source	Description
afu_tlx_cmd_initial_credit	7	AFU	Specifies initial setting of credits for sending CAPP commands to the AFU.
afu_tlx_cmd_credit	1	AFU	CAPP command credit return. Indicates that the AFU can now accept another command from the TLX.
tlx_afu_cmd_valid	1	TLX	Command Valid. The remaining signals in this table are valid coincident with the assertion of tlx_afu_cmd_valid.

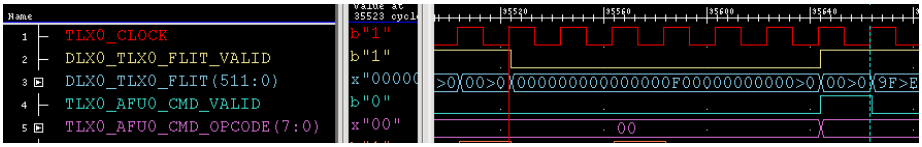
tlx_afu_cmd_opcode	8	TLX	Command Opcode. Note: Please see OpenCAPI 3.0 TL Specification for valid opcodes
tlx_afu_cmd_capptag	16	TLX	Unique handle specifying the host CAPP and command instance. Provided by the CAPP requesting command services of the TL.
tlx_afu_cmd_dl	2	TLX	Command Data Length Encodings Size 2b'00 Reserved 2b'01 64 Bytes 2b'10 128 Bytes 2b'11 256 Bytes
tlx_afu_cmd_pl	3	TLX	Partial Length Encodings Size 3b'000 1 Byte 3b'001 2 Bytes 3b'010 4 Bytes 3b'011 8 Bytes 3b'100 16 Bytes 3b'101 32 Bytes 3b'110-111 Reserved
tlx_afu_cmd_be	64	TLX	Byte Enable
tlx_afu_cmd_end	1	TLX	Operand Endianness. '0' Operands are little endian. '1' Operands are big endian.
tlx_afu_cmd_pa	64	TLX	Physical Address
tlx_afu_cmd_flag	4	TLX	Specifies atomic memory operation
tlx_afu_cmd_os	1	TLX	Ordered segment. When set to '1' ordering is guaranteed. **Reserved for OpenCAPI 4.0**

CAUTION! The TLX presents individual signals for fields used across the entire set of TL commands. While it makes it easier for the AFU to capture specific fields, it can also imply that fields are valid when they are not. The AFU designer must refer to the OpenCAPI TL spec to know which fields are valid for a particular opcode. For instance, 'write_mem' provides a dL field containing the number of data FLITs, while 'pr_wr_mem' infers a dL value since it is always 1 FLIT. Be sure to only use the fields that are valid for a particular command. Fields that are not used will have unpredictable, and possibly erroneous, values on them.

Note that tlx_afu_cmd_valid and tlx_cfg_valid cannot both be asserted at the same time because they both use virtual channel 1 in the TLX logic.

The figure below illustrates the delay of a flit containing a command with no data through TLX. The flit is a template x"00" and contains a read memory command. As seen in the figure there are seven cycles of latency between when the flit is received from the DLX and when the command is presented on the AFU interface.

Figure 7: Command Latency



The data associated with CAPP commands, if any, is passed to the AFU using the command data interface described below.

2.3.3 TLX to AFU CAPP Configuration Command Interface (VC1)

In order to support AFU configuration functions, TLX separates the configuration commands from other CAPP commands. The configuration commands are sent to the AFU Configuration Logic using a separate interface which is described in this section.

This interface is used only for CAPP commands with opcodes of x11100000 (config_reads) and x11100001 (config_writes).

Table 3: TLX Parser – TLX to AFU CAPP Configuration Command Interface (VC1)

Signal Name	Bits	Source	Description
cfg_tlx_initial_credit	4	AFU config logic	Initial setting of credits to be used to send configuration commands to the AFU
cfg_tlx_credit_return	1	AFU config logic	Return of credit: Indicates that the AFU is ready and able to accept an additional configuration command.
tlx_cfg_valid	1	TLX	Command Valid. The following signals in this table are valid coincident with the assertion of tlx_cfg_valid.
tlx_cfg_opcode	8	TLX	Command Opcode. Note: Please see OpenCAPI 3.0 TL Specification for valid opcodes
tlx_cfg_capptag	16	TLX	Unique handle specifying the host CAPP and command instance. Provided by the CAPP requesting command services of the TL.
tlx_cfg_pa	64	TLX	Physical Address
tlx_cfg_t	1	TLX	Configuration read or write command type '0': Indicates a type 0 configuration read or write command '1': Indicates a type 1 configuration read or write command.
tlx_cfg_pl	3	TLX	Partial Length Encodings Size 3b'000 1 Byte 3b'001 2 Bytes 3b'010 4 Bytes

			3b'011 8 Bytes 3b'100 16 Bytes 3b'101 32 Bytes 3b'110-111 Reserved
tlx_cfg_data_bus	32	TLX	Data for config_write commands. (Comes one cycle after the valid)
tlx_cfg_data_bdi	1	TLX	Bad Data Indicator. If asserted indicates the data received during the same cycle has an error and cannot be trusted. (Comes one cycle after the valid)

CAUTION! The TLX presents individual signals for fields used across the entire set of TL commands. While it makes it easier for the AFU to capture specific fields, it can also imply that fields are valid when they are not. The AFU designer must refer to the OpenCAPI TL spec to know which fields are valid for a particular opcode. Be sure to only use the fields that are valid for a particular command. Fields that are not used will have unpredictable, and possibly erroneous, values on them.

The timing of configuration commands is the same as for other CAPP commands, except that the data associated with configuration commands, if any, is passed to the AFU one cycle later than the command. In other words, the AFU does not need to request the data for config_write commands, it immediately follows the command itself.

Note that tlx_afu_cmd_valid and tlx_cfg_valid cannot both be asserted at the same time because they both use virtual channel 1 in the TLX logic.

Also note that for partial config writes, the CFG logic expects the data to remain in its byte lanes as if it were a full write. See config logic documentation for details.

2.3.4 TLX to AFU CAPP Response Interface (VC0)

Table 4: TLX Parser – TLX to AFU CAPP Response Interface (VC0)

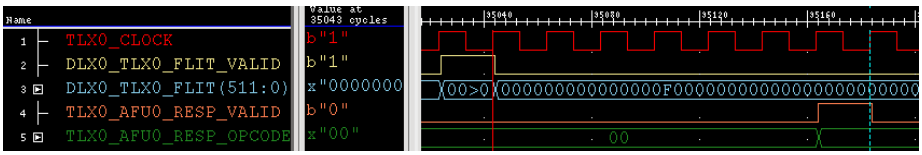
Signal Name	Bits	Source	Description
afu_tlx_resp_initial_credit	7	AFU	Specifies initial setting of credits for sending CAPP responses to the AFU.
afu_tlx_resp_credit	1	AFU	CAPP response credit return. Indicates that the AFU can now accept another response from the TLX.
tlx_afu_resp_valid	1	TLX	Indicates valid response from the TLX. The remaining signals in this table are valid coincident with the assertion of tlx_afu_resp_valid.
tlx_afu_resp_opcode	8	TLX	Response Opcode. Note: Please see OpenCAPI 3.0 TL Specification for valid opcodes
tlx_afu_resp_afutag	16	TLX	Response Tag.
tlx_afu_resp_code	4	TLX	Reports the reason for a failed transaction.

tlx_afu_resp_pg_size	6	TLX	Page Size.
tlx_afu_resp_dl	2	TLX	Data Length. Encodings Size 2b'00 Reserved 2b'01 64 Bytes 2b'10 128 Bytes 2b'11 256 Bytes
tlx_afu_resp_dp	2	TLX	Data part. Indicates the data content of the current response packet
tlx_afu_resp_host_tag	24	TLX	Tag associated with data held in AFU L1. **Reserved for OpenCAPI 4.0**
tlx_afu_resp_addr_tag (obsolete – no longer used)	18	TLX	Address Tag. The AFU has obtained an address translation tag and shall use it with an address offset for TLX commands that have a dot-t format. The address translation tag is associated with a specific page in memory and the size of the page is known. (Note: This signal is obsolete – no longer used)
tlx_afu_resp_cache_state	4	TLX	Specifies the cache state the cache line has obtained. **Reserved for OpenCAPI 4.0**

CAUTION! The TLX presents individual signals for fields used across the entire set of TL responses. While it makes it easier for the AFU to capture specific fields, it can also imply that fields are valid when they are not. The AFU designer must refer to the OpenCAPI TL spec to know which fields are valid for a particular opcode. Be sure to only use the fields that are valid for a particular response. Fields that are not used will have unpredictable, and possibly erroneous, values on them.

The figure below illustrates the latency through TLX of a response with no associated data. As seen in the figure there is seven cycle delay between when the flit is received from the DLX until the response is presented on the AFU interface.

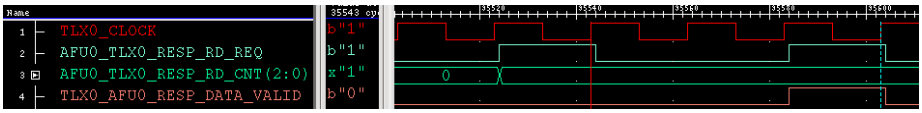
Figure 8: Response Latency



2.3.5 TLX to AFU Data Interfaces (DCP0, DCP1)

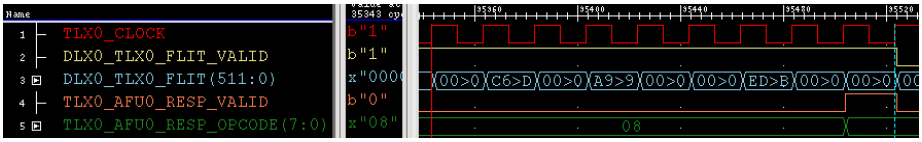
The data interfaces for commands and responses to the AFU consists of the 512 bit wide data bus with a corresponding valid bit, a bad data indicator, and a read request interface driven by the AFU. All data is sent in the order is was received respective to the virtual channel. There is a minimum three cycle delay between seeing a valid request and seeing the requested data on the bus. The number of outstanding beats of data requested cannot exceed the depth of the data FIFOs. The default size of the data FIFOs is 512x128, the depth can be edited to the AFU designer's need. Other than this restriction the AFU may request the data as it sees fit. All data is ordered with respect to virtual channel. When the bad data indicator is asserted the data sent during the same cycle has an uncorrectable error. The figure below shows the timing of a read request.

Figure 9: Data Read Request



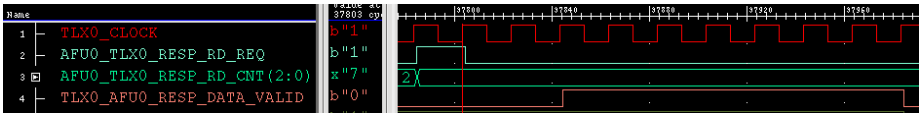
The figure below shows the latency for a command or response through the TLX. The TLX will not send a command or response with data until the bookend flit has been received, which verifies the data has good crc. As seen in the figure there is nine cycles of latency between the bookend flit and the response being sent to the AFU.

Figure 10: Data Latency



The figure below is an example of a multiple data flit read request by the AFU. The example requests seven data flits and after the three cycle delay all seven data flit are presented on the AFU data interface back to back.

Figure 11: Multiple Data Request



The figure below is another example of how to the TLX handles consecutive read requests. The first request of two data flit has the three second penalty. For the second request the three cycle penalty is over the same period of the first request. Once the first request has finished the data for the second request starts being sent immediately. This is repeated for the subsequent read requests in the figure.

Figure 12: Multiple Read Requests

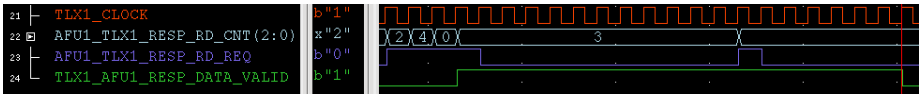


Table 5: TLX Parser – TLX to AFU CAPP Response Data Interface (DCP0)

Signal Name	Bits	Source	Description
tlx_afu_resp_data_valid	1	TLX	Response Valid. Valid data is present.
tlx_afu_resp_data_bus	512	TLX	Response Data. Data for a read request.
tlx_afu_resp_data_bdi	1	TLX	Bad Data Indicator. If asserted indicates the data received during the same cycle has an error and cannot be trusted.
afu_tlx_resp_rd_req	1	AFU	Response Read Request.
afu_tlx_resp_rd_cnt	3	AFU	Response Read Count. Encodings Size 3b'000 512 Bytes 3b'001 64 Bytes 3b'010 128 Bytes 3b'011 256 Bytes 3b'100 192 Bytes 3b'101 320 Bytes 3b'110 384 Bytes 3b'111 448 Bytes **Note: '001', '010', and '011' were set to match the data length encoding.**

Table 6: TLX Parser – TLX to AFU CAPP Command Data Interface (DCP1)

Signal Name	Bits	Source	Description
tlx_afu_cmd_data_valid	1	TLX	Command Data Valid. Valid data is present.
tlx_afu_cmd_data_bus	512	TLX	Command Data Bus.
tlx_afu_cmd_data_bdi	1	TLX	Bad Data Indicator. If asserted indicates the data received during the same cycle has an error and cannot be trusted.
afu_tlx_cmd_rd_req	1	AFU	Command Read Request.
afu_tlx_cmd_rd_cnt	3	AFU	Command Read Count. Encodings Size 3b'000 512 Bytes 3b'001 64 Bytes 3b'010 128 Bytes 3b'011 256 Bytes 3b'100 192 Bytes 3b'101 320 Bytes 3b'110 384 Bytes 3b'111 448 Bytes Note: '001', '010', and '011' were set to match the data length encoding.

2.3.6 TLX Parser Miscellaneous Ports

Table 7: TLX Parser – Miscellaneous Ports

Signal Name	Bits	Source	Description
clock	1	FPGA	Positive-active clock
reset_n	1	FPGA	Negative-active reset. Resets the framer when this signal goes low
tlx_afu_ready	1	TLX	Tells the AFU that the link is up and the TLX is ready for traffic.
rcv_xmt_debug_valid	1	TLX Parser	Indicates that the two debug ports listed below contain valid information
rcv_xmt_debug_fata	1	TLX Parser	Flag to indicate that TLX Parser has detected a fatal error (if valid is asserted.)
rcv_xmt_debug_info	32	TLX Parser	32 bits of information about errors detected in the TLX Parser

The TLX Parser has several other interfaces which are not described in this section because they are documented elsewhere in this document:

- There are signals internal to the TLX which go from the TLX Parser to the TLX Framer for handling the TL and TLX credits. These signals are described in both the TLX section and the Credit Management sections below.
- There are some signals from the TLX Parser which tell the AFU how the Parser is configured (template support and rate limits). These signals are described in the Configuration section below.

3 TLX Framer

3.1 Overview

The TLX framer accepts AP commands and AP responses from the AFU, and TL credits from the parser, and packs the commands, responses, and TL credits into control flits. Immediate data is put into data flits. The control flits and data flits are placed into TL frame sequence and sent to the DLX.

The Framer packs AP command, response, and TL credit packets into control flits as specified by the supported packing templates. The framer always supports template x00, and optionally supports templates x01 and x03. Template x02 is not needed for anticipated workloads, and is not supported in this reference design.

AP responses are sent to the host using virtual channel 0 (VC0). AP commands are sent to the host using virtual channel 3 (VC3). Responses are ordered in VC0. Commands are ordered in VC3. However, there is no implied ordering between VC0 and VC3 (so commands could pass responses and vice versa). TL credit packets are also sent to the host, but they are not assigned to any virtual channel.

The TLX Framer has separate interfaces with the AFU for AP responses (VC0) and for AP commands (VC3).

For timing purposes, this Framer design cannot pack both AP responses and AP commands into the same control flit. The architecture allows for commands and responses to be mixed in the same control flit, but this implementation does not support this. This implementation uses completely separate pipelines for each virtual channel, and content from each pipeline is packed into separate control flits.

Immediate data for AP responses goes through channel DCP0. Order of data packets in the channel is maintained. Immediate data for AP commands goes through channel DCP3. Data packets are ordered in DCP3 also. These is no implied ordering *between* DCP0 and DCP3.

To maintain ordering in each virtual channel, and to support smooth flow of commands and data, each virtual channel has its own FIFO. The VC0 and VC3 FIFOs are only 4 entries deep, just enough to maintain smooth traffic flow from the AFU. The DCP0 and DCP3 FIFOs are 16 entries deep.

3.2 Framer Data Flow Rules

1. If there are no data flits waiting to be sent, and the DLX is able to receive a flit, then TLX will send a control flit containing whatever commands, responses, and/or credits are ready to go. (TLX will not delay the control flit in order to pack more commands or responses into it.)
2. TLX may generate control flits that have more than 8 flits of associated immediate data. In this case, TLX will send the first control flit, followed by the first 8 data flits. Then TLX will insert another control flit before sending the remainder of the data. (A TL frame can only contain a maximum of 8 data flits.) All that data flits will be sent in order, even if they have control flits inserted between them.
3. If a control flit has been sent to DLX, then TLX will send up to 8 flits of associated immediate data, in order.
    - There is no need to check DCP credits again, because the DCP credits are required before the cmd/resp flit can be sent.
    - While data flits are being sent, the TLX logic can work on forming the next control flit.

4. Control flits can be sent back-to-back. Up to 8 data flits can be sent back to back.
5. After 1-8 data flits have been sent, TLX will send another control flit containing CRC for the data, regardless of whether or not there are new commands and or responses to be sent.
6. A command/response can be packed into a control flit for transmission to DLX when the following conditions are met:
    - At least one command or response credit is available for the appropriate VC
    - There are enough DCP credits available, in the associated DCPs, to send all of the immediate data flits.
    - All the previous commands or responses in this same VC have either been sent already, or they are already placed in the currently forming control flit.
7. A flit can be sent to DLX based on DLX control flow, subject to DL credits. The DLX can apply back pressure due to x4 mode or replays, etc.
8. A new control flit cannot be sent to the DLX if the flit rate counter > 0. In this case Null Flits must be sent to the DLX instead of control flits. These Null flits are required to preserve spacing in the DLX replay buffer.
9. (*) Commands and responses will be into separate control flits. This is a simplification for timing. Commands and responses can be packed in parallel.
10. (*) Responses have priority over new commands. Data flits will be sent in the order in which their associated commands or responses were put into the control flit. Ping-pong arbitration is be used for selecting between response and command flits if both are waiting to be sent.
11. (*) Any commands or responses from the AFU which have immediate data must send that data immediately (coincident with the command or response and continuing, brick-walled, until all the data beats have arrived.) The TLX Framer will consider it an error if it gets a command or response which should have immediate data, but no data arrives.
12. (*) AFU will not send any data to the TLX before sending the associated command or response. The TLX Framer will consider it an error if it gets data for which it has no corresponding command or response.
13. (*) The AFU can send new commands or responses which have no immediate data to the TLX while the AFU is still sending data beats for a previous command or response.
14. (*) The TLX will go ahead and pack any no-op commands from the AFU into control flits for the DLX. This will allow no-op commands to be packed into random places in the control flit for verification/validation.

(*) These rules are not required by the architecture, but they are implemented by this .70/.71/.72 TLX Reference Design

3.3 Control and Data Flow and Pipeline Latency

The figures below shows the flow of commands, responses, and data through the framing and transmitting logic of the TLX. The minimum latency through the framer is seven cycles. There are four FIFOs that the framer logic uses to buffer incoming data, responses, and commands from the AFU. As seen in the figure the framer interface between the TLX and AFU includes all bit fields for valid opcodes for OpenCAPI 3.0.

Figure 13: TLX Framer Block Diagram

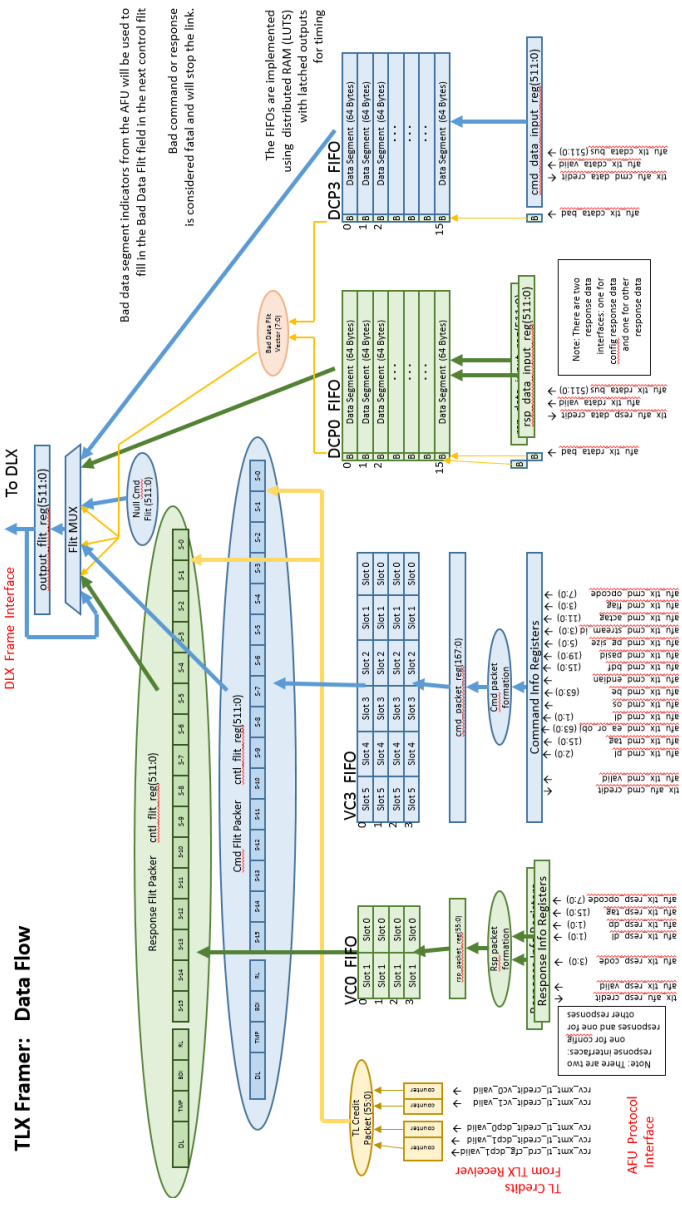
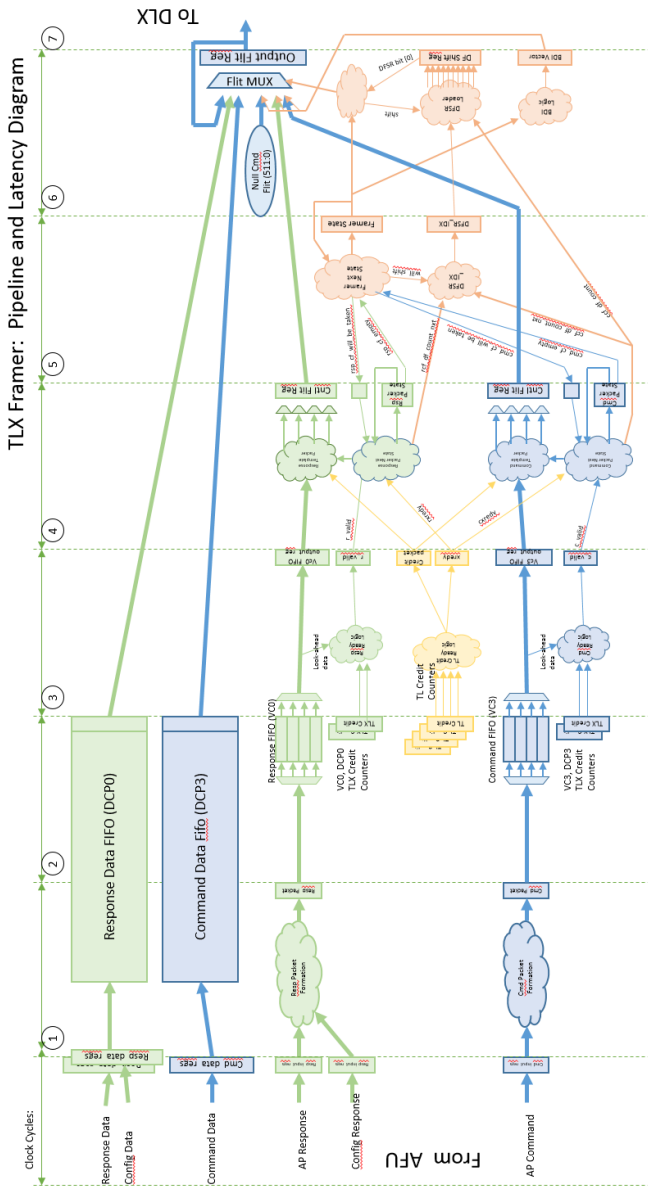


Figure 14: TLX Framer Latency Diagram



3.4 TLX Framer Interfaces

3.4.1 TLX Framer Miscellaneous Ports

Table 8: TLX Framer – Miscellaneous Ports

Signal Name	Bits	Source	Description
clock	1	FPGA	Positive-active clock
reset_n	1	FPGA	Negative-active reset. Resets the framer when this signal goes low

3.4.2 TLX Parser to Framer Credit Interfaces

These interfaces are used to send TLX and TL credits from the Parser to the Framer. These signals are discussed in greater detail in the "Credit Management" chapter of this document (below).

Table 9: TLX Framer – TLX Parser to Framer TLX Credit Interface

Signal Name	Bits	Source	Description
rcv_xmt_tlx_credit_vc0	4	TLX RCV	Response credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_dcp0	6	TLX RCV	Response data credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_vc3	4	TLX RCV	Command credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_dcp3	6	TLX RCV	Command data credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_valid	1	TLX RCV	Indicates valid credits to be returned to the TLX from the TL.

Table 10: TLX Framer – TLX Parser to Framer TL Credit Interface

Signal Name	Bits	Source	Description
rcv_xmt_tl_credit_vc0_valid	1	TLX RCV	Indicates a response has been passed to the AFU, and increments the VC0 TL credit counter in the TLX Framer
rcv_xmt_tl_credit_dcp0_valid	1	TLX RCV	Indicates response data has been taken by the AFU and increments the DCP0 TL credit counter in the TLX Framer.

rcv_xmt_tl_credit_vc1_valid	1	TLX RCV	Indicates a command has been passed to the AFU, and increments the VC1 TL credit counter in the TLX Framer
rcv_xmt_tl_credit_dcp1_valid	1	TLX RCV	Indicates command data has been taken by the AFU and increments the DCP1 TL credit counter in the TLX Framer.
rcv_xmt_tl_crd_cfg_dcp1_valid	1	TLX RCV	Indicates configuration data has been taken by the AFU and increments the DCP1 TL credit counter in the TLX Framer.

3.4.3 AFU to TLX AP Command Interface (VC3, DCP3)

The AFU uses this interface to send AP commands to the TLX for transmission to the host. The AFU to TLX command interface is broken out into all the possible fields for opcodes that are assigned to virtual channel 3 in the OpenCAPI 3.0 specification.

Table 11: TLX Framer – AFU to TLX AP Command Interface (VC3, DCP3)

Signal Name	Bits	Source	Description
tlx_afu_cmd_initial_credit	4	TLX Framer	Set to a constant by the TLX Framer and sent to the AFU to tell the AFU how many AP commands can be sent by the AFU to the TLX before a credit return is required. This value is hardcoded to 8 in the current reference design implementation.
tlx_afu_cmd_credit	1	TLX Framer	When this signal is asserted, the AFU can send another AP command to the TLX. The AFU needs to maintain a counter of credits for sending AP commands to the TLX. The counter should be decremented whenever a command is sent to the TLX. The counter should be incremented whenever tlx_afu_cmd_credit is asserted. The counter should be initialized using the value from tlx_afu_cmd_resp_initial_credit. The AFU should not send any AP commands to the TLX when the credit count is zero.
afu_tlx_cmd_valid	1	AFU	Indicates that a valid AP command has arrived from the AFU to the TLX. Any command field that pertains to the arriving opcode should contain valid information at this time. Other command fields are undefined and may contain garbage.
afu_tlx_cmd_opcode	8	AFU	AP Command Opcode. (see TL Specification)
afu_tlx_cmd_actag	12	AFU	Address Context tag (see TL Specification)
afu_tlx_cmd_stream_id	4	AFU	Stream ID (see TL Specification)
afu_tlx_cmd_ea_or_obj	68	AFU	Effective Address/Object Handle. (see TL Specification)
afu_tlx_cmd_afutag	16	AFU	AFU Tag. (see TL Specification)
afu_tlx_cmd_dl	2	AFU	Data Length (see TL Specification) If the AP command has more than 1 packet of immediate data, the DL field will be used to tell the TLX how many data packets belong to this command. Encoding Size 2b'00 Reserved 2b'01 64 Bytes (1 data packet) 2b'10 128 Bytes (2 data packets) 2b'11 256 Bytes (4 data packets)

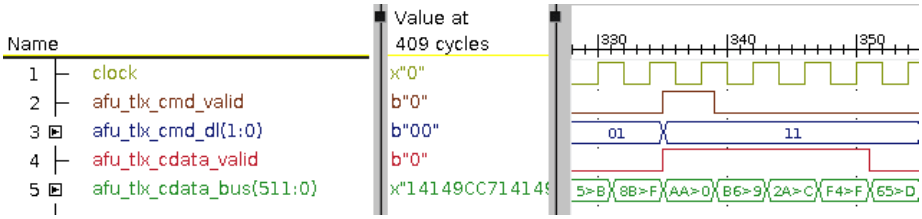
afu_tlx_cmd_pl	3	AFU	Partial Length (see TL Specification) Encodings Size 3b'000 1 Byte 3b'001 2 Bytes 3b'010 4 Bytes 3b'011 8 Bytes 3b'100 16 Bytes 3b'101 32 Bytes 3b'110-111 Reserved
afu_tlx_cmd_os	1	AFU	Ordered segment. (see TL Specification) **Reserved for OpenCAPI 4.0**
afu_tlx_cmd_be	64	AFU	Byte enable. (see TL Specification)
afu_tlx_cmd_flag	4	AFU	Command flag, used in atomic operations (see TL Specification)
afu_tlx_cmd_endian	1	AFU	Operand Endianness. (see TL Specification) Used for mem_atomics.* and atomics.* class commands to specify the endianness of the operands. For bitwise logical operations, the endianness of the operands does not change the result. The field is specified as follows. 0 Operands are little endian. 1 Operands are big endian.
afu_tlx_cmd_bdf	16	AFU	Bus Device Function (see TL Specification)
afu_tlx_cmd_pasid	20	AFU	User process ID (see TL Specification)
afu_tlx_cmd_pg_size	6	AFU	Page Size. (see TL Specification)
tlx_afu_cmd_data_initial_credit	6	TLX Framer	Set to a constant by the TLX Framer and sent to the AFU to tell the AFU how many command data packets can be sent by the AFU to the TLX before a data credit return is required. This value is hardcoded to 32 in the current reference design implementation.
tlx_afu_cmd_data_credit	1	TLX Framer	When this signal is asserted, the AFU can send another AP command data packet to the TLX. The AFU needs to maintain a counter of credits for sending command data packets to the TLX. The counter should be initialized using the value from tlx_afu_data_initial_credit. The counter should be incremented whenever tlx_afu_cmd_data_credit is asserted. The counter should be decremented whenever an AP command with immediate data is sent to the TLX. (The counter should be decremented by the number of immediate data packets that are associated with the command.) The AFU should not send any AP commands with immediate data to the TLX unless there are enough data credits in this counter to support the number of immediate data packets associated with the command.
afu_tlx_cdata_valid	1	AFU	AP Command Data Valid. Indicates that a valid packet of command immediate data has arrived from the TLX. The data bus and the bdi bit contain valid information.
afu_tlx_cdata_bus	512	AFU	AP Command Data Bus.
afu_tlx_cdata_bdi	1	AFU	Bad Data Indicator. Indicates that the AP command data packet is bad.

For AP commands with immediate data, the first data packet must be valid coincident with the command itself and the data must be brick walled until all immediate data pertaining to this command

has been sent to the TLX. A new data-less command could be sent while the data from a previous command is still being sent.

The figure below shows a legal transaction of the AFU sending the TLX a command with 256 Bytes of associated immediate data (four data packets).

Figure 15: AFU Command with data



3.4.4 AFU to TLX AP Response Interface (VC0, DCP0)

The AFU uses this interface to send AP responses to the TLX for transmission to the host. The AFU to TLX response interface is broken out into all the possible fields for opcodes that are assigned to virtual channel 0 in the OpenCAPI 3.0 specification. Reponses to AFU configuration commands are sent to the TLX using a separate interface shown below.

Table 12: TLX Framer – AFU to TLX AP Response Interface (VCO, DCP0)

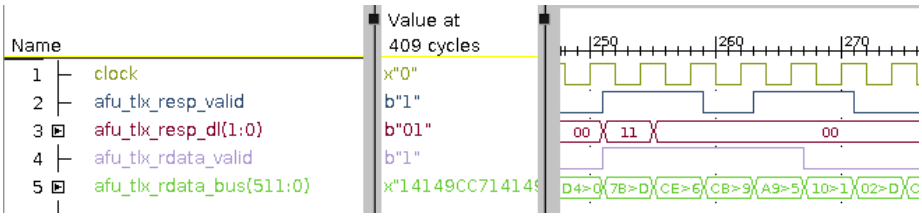
Signal Name	Bits	Source	Description
tlx_afu_resp_initial_credit	4	TLX Framer	Set to a constant by the TLX Framer and sent to the AFU to tell the AFU how many AP responses can be sent by the AFU to the TLX before a credit return is required. This value is hardcoded to 7 in the current reference design implementation.
tlx_afu_resp_credit	1	TLX Framer	When this signal is asserted, the AFU can send another AP response to the TLX. The AFU needs to maintain a counter of credits for sending AP responses to the TLX. The counter should be decremented whenever a response is sent to the TLX. The counter should be incremented whenever tlx_afu_resp_credit is asserted. The counter should be initialized using the value from tlx_afu_cmd_resp_initial_credit. The AFU should not send any AP responses to the TLX when the credit count is zero.
afu_tlx_resp_valid	1	AFU	Indicates that a valid AP response has arrived from the AFU to the TLX. Any response field that pertains to the arriving opcode should contain valid information at this time. Other response fields are undefined and may contain garbage.
afu_tlx_resp_opcode	8	AFU	AP Response Opcode (see TL Specification)
afu_tlx_resp_dl	2	AFU	Response Data Length (see TL Specification)

			If the AP response has more than 1 packet of immediate data, the DL field will be used to tell the TLX how many data packets belong to this response. Encoding Size 2b'00 Reserved 2b'01 64 Bytes (1 data packet) 2b'10 128 Bytes (2 data packets) 2b'11 256 Bytes (4 data packets)
afu_tlx_resp_capptag	16	AFU	Response Tag (see TL Specification)
afu_tlx_resp_dp	2	AFU	Data part. (see TL Specification) Indicates the data content of the current response packet.
afu_tlx_resp_code	4	AFU	Response Code (see TL Specification) Describes the reason for a failed transaction.
tlx_afu_resp_data_initial_credit	6	TLX Framer	Set to a constant by the TLX Framer and sent to the AFU to tell the AFU how many response data packets can be sent by the AFU to the TLX before a data credit return is required. This value is hardcoded to 32 in the current reference design implementation.
tlx_afu_resp_data_credit	1	TLX Framer	When this signal is asserted, the AFU can send another AP response data packet to the TLX. The AFU needs to maintain a counter of credits for sending response data packets to the TLX. The counter should be initialized using the value from tlx_afu_data_initial_credit. The counter should be incremented whenever tlx_afu_resp_data_credit is asserted. The counter should be decremented whenever an AP response with immediate data is sent to the TLX. (The counter should be decremented by the number of immediate data packets that are associated with the response.) The AFU should not send any AP responses with immediate data to the TLX unless there are enough data credits in this counter to support the number of immediate data packets associated with the response.
afu_tlx_rdata_valid	1	AFU	AP Response Data Valid. Indicates that a valid packet of response immediate data has arrived from the TLX. The data bus and the bdi bit contain valid information.
afu_tlx_rdata_bus	512	AFU	AP Response Data Bus
afu_tlx_rdata_bdi	1	AFU	Bad Data Indicator. Indicates that the AP response data packet is bad.

For AP responses with immediate data, the first data packet must be coincident with the response itself and the data must be brick walled until all data packets pertaining to this response have been sent to the TLX. A new data-less response could be sent to the TLX while the data from a previous response is still being sent.

The figure below shows a legal transaction of the AFU sending the TLX an AP response with 4 packets of immediate data, followed by two more data-less AP responses. The first data-less response overlaps with the data transmission of the first response.

Figure 16: AFU Response with data followed by more responses



3.4.5 AFU to TLX AP Configuration Response Interface (VC0, DCP0)

The AFU Configuration Logic uses this interface to send AP responses to the TLX for transmission to the host. This is low-bandwidth interface than handles only one configuration response at a time. Flow control is maintained using a valid-ack protocol instead of using credits. The AFU to TLX response interface is broken out into all the possible fields for opcodes that are assigned to virtual channel 0 in the OpenCAPI 3.0 specification.

Table 13: TLX Framer – AFU to TLX AP Configuration Response Interface (VC0, DCP0)

Signal Name	Bits	Source	Description
cfg_tlx_resp_valid	1	AFU config logic	Indicates that a valid AP response has arrived from the AFU to the TLX. Any response field that pertains to the arriving opcode should contain valid information at this time. Other response fields are undefined and may contain garbage. **Note – This signal must remain asserted until the TLX has taken the response as indicated by the tlx_cfg_resp_ack signal. At that time this valid signal must go to zero for at least one cycle before another config response can be sent.**
cfg_tlx_resp_opcode	8	AFU config logic	AP Response Opcode (see TL Specification)

~~cfg_tlx_resp_dl~~	~~2~~	~~AFU config logic~~	~~Response Data Length (see TL Specification)~~ ~~If the AP response has more than 1 packet of immediate data, the DL field will be used to tell the TLX how many data packets belong to this response.~~ ~~Encoding Size~~ ~~2b'00 Reserved~~ ~~2b'01 64 Bytes (1 data packet)~~ ~~2b'10 128 Bytes (2 data packets) (not used for config responses)~~ ~~2b'11 256 Bytes (4 data packets) (not used for config responses)~~ Hardwired internally to 2'b01 (1 data packet)
cfg_tlx_resp_capptag	16	AFU config logic	Response Tag (see TL Specification)
~~cfg_tlx_resp_dp~~	~~2~~	~~AFU config logic~~	~~Data part. (see TL Specification)~~ ~~Indicates the data content of the current response packet.~~ Hardwired internally to 2'b00;
cfg_tlx_resp_code	4	AFU config logic	Response Code (see TL Specification) Describes the reason for a failed transaction.
tlx_cfg_resp_ack	1	TLX Framer	TLX pulses this signal when it takes the config response. The AFU configuration logic must then drop the cfg_tlx_resp_valid signal for at least one cycle before sending another config response.
cfg_tlx_rdata_offset	4	AFU config logic	This signal specifies how the 32-bit configuration response data should be aligned into a 512-bit data flit. Encode Alignment 0000 cfg_tlx_rdata_bus is inserted into data_flit[31: 0] 0001 cfg_tlx_rdata_bus is inserted into data_flit[63: 32] 0010 cfg_tlx_rdata_bus is inserted into data_flit[95: 64] 0011 cfg_tlx_rdata_bus is inserted into data_flit[127: 96] 0100 cfg_tlx_rdata_bus is inserted into data_flit[159: 128] 0101 cfg_tlx_rdata_bus is inserted into data_flit[191: 160] 0110 cfg_tlx_rdata_bus is inserted into data_flit[223: 192] 0111 cfg_tlx_rdata_bus is inserted into data_flit[255: 224] 1000 cfg_tlx_rdata_bus is inserted into data_flit[287: 256] 1001 cfg_tlx_rdata_bus is inserted into data_flit[319: 288] 1010 cfg_tlx_rdata_bus is inserted into data_flit[351: 320] 1011 cfg_tlx_rdata_bus is inserted into data_flit[383: 352] 1100 cfg_tlx_rdata_bus is inserted into data_flit[415: 384] 1101 cfg_tlx_rdata_bus is inserted into data_flit[447: 416] 1110 cfg_tlx_rdata_bus is inserted into data_flit[479: 448] 1111 cfg_tlx_rdata_bus is inserted into data_flit[511: 480]
cfg_tlx_rdata_bus	32	AFU config logic	32-bit configuration response data. This data is inserted into a 512-bit data flit according to the offset specified by cfg_tlx_rdata_offset. **The AFU is responsible for aligning the configuration data in this 32-bit vector.**
cfg_tlx_rdata_bdi	1	AFU config logic	Bad Data Indicator. Indicates that the AP response data packet is bad.

For AP configuration responses with immediate data, the single data packet must be coincident with the response.

Also note that for partial config reads, the CFG logic returns the data in its byte lanes as if it were a full write. See config logic documentation for details.

3.4.6 TLX to DLX Flit Interface

The interface between the TLX framer and DLX consists of flit signals driven by the TLX, and credit and configuration signals driven by the DLX. The table below further describes this interface.

Table 14: TLX Framer – TLX to DLX Flit Interface

Signal Name	Bits	Source	Description
dlx_tlx_link_up	1	DLX	Link up. DLX indicating that the link between the host and the FPGA has been trained.
dlx_tlx_init_flit_depth	3	DLX	Initial DLX flit buffer depth. DLX informs the TLX the initial number of flits the DLX can handle before a credit must be returned. This signal should be set to a constant by the DLX.
dlx_tlx_flit_credit	1	DLX	DLX flit credit return. DLX informs the TLX that it has finished processing a flit, so another flit can be sent from the TLX to the DLX
tlx_dlx_flit_valid	1	Framer	Flit Valid. Indicates to the DLX that tlx_dlx_flit is valid.
tlx_dlx_flit	512	Framer	Control or Data Flit. This 512 bit wide bus that contains the commands, responses, data, and TL credits to be sent across the link.

3.4.7 TLX Framer Configuration ports

This interface is used to set some configurable features of the TLX Framer. The signals come from the configuration registers in the AFU.

Table 15: TLX Framer – Configuration Ports

Signal Name	Bits	Source	Description
afu_cfg_xmit_tmpl_config_0	1	AFU	Indicates that the Framer should support packing template x00 (always true)

afu_cfg_xmit_tmpl_config_1	1	AFU	Indicates that the Framer should support packing template x01
afu_cfg_xmit_tmpl_config_2	1	AFU	Indicates that the Framer should support packing template x02 (ignored) This reference design does not support transmitting using template x02
afu_cfg_xmit_tmpl_config_3	1	AFU	Indicates that the Framer should support packing template x03
afu_cfg_xmit_rate_config_0	4	AFU	Used to set the flit-rate counter for template x00
afu_cfg_xmit_rate_config_1	4	AFU	Used to set the flit-rate counter for template x01
afu_cfg_xmit_rate_config_2	4	AFU	Used to set the flit-rate counter for template x02 (ignored)
afu_cfg_xmit_rate_config_3	4	AFU	Used to set the flit-rate counter for template x03

3.4.8 TLX Framer Debug Ports

Table 16: TLX Framer – Debug Ports

Signal Name	Bits	Source	Description
rcv_xmt_debug_valid	1	Parser	Indicates that debug info from the Parser to the Framer is valid (when asserted)
rcv_xmt_debug_info	32	Parser	Debug bus from the Parser to the Framer
rcv_xmt_debug_fatal	1	Parser	Fatal error indicator from the Parser to the Framer
dlx_tlx_dlx_config_info	32	DLX	Miscellaneous configuration bits from DLX Bit (31) is used to reset the sticky error capture registers in TLX
tlx_dlx_debug_encode	4	Framer	Debug encode information from TLX to DLX: Bit(0) – indicates a fatal error was detected Bit(1) – indicates that the debug info is coming from the AFU Bit(2) – indicates that the debug info is coming from the TLX Parser Bit(3) – indicates that the debug info is coming from the TLX Framer
tlx_dlx_debug_info	32	Framer	32 bits of information about errors detected in the hardware, from TLX to DLX

3.5 TLX Framer: Internal State Machines

Figure 17: TLX Framer - Template Packer State Diagram

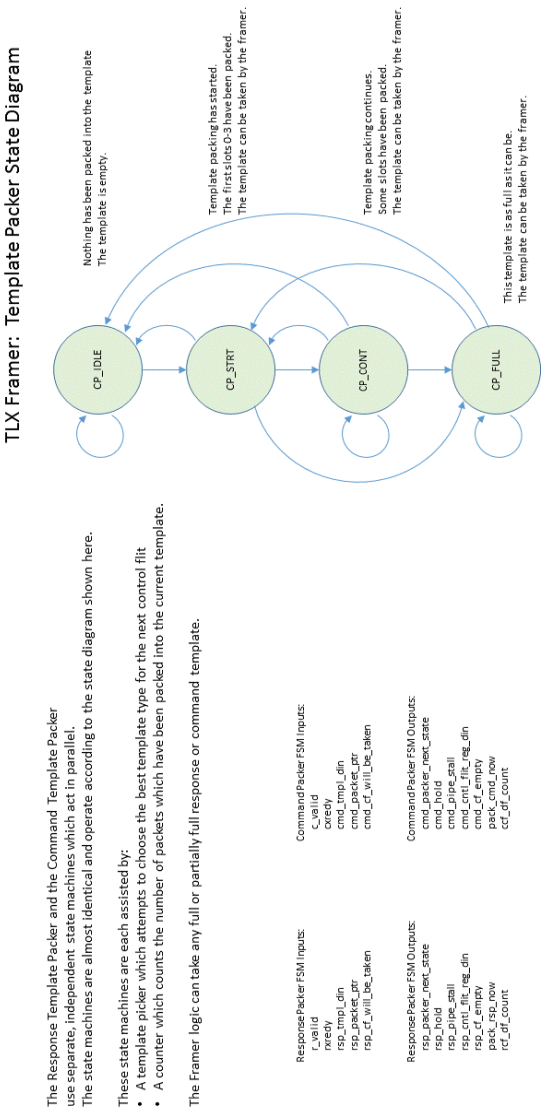
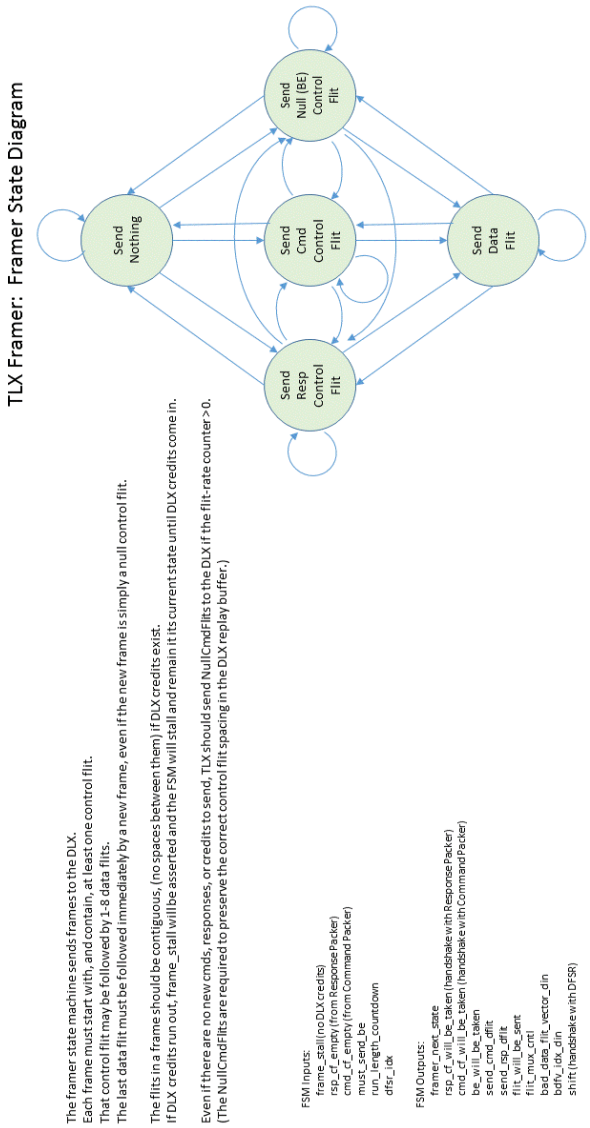


Figure 18: TLX Framer - Flit Framer State Diagram



4 Credit Management

The TLX reference design has several credit systems between the different units: AFU, DLX, TLX and TL.

TL and TLX credits are the end-to-end credits utilized between the TLX and TL are used to prevent either transaction layer from overrunning the other with an excess of commands or responses. TLX credits are issued by the TL and consumed by the TLX. TL credits are issued by the TLX and consumed by the TL. There are separate TL and TLX credits for each VC and DCP channel. The number of TL credits presented to the TL by the TLX is dependent on the size of each of the four FIFOs located in the TLX Parser.

The TL and TLX credits are embedded in the normal TL frame traffic. They are passed between the TL and TLX in slots 0 and 1 of any control flit.

The AFU does not manage TL or TLX credits. The TL to TLX credit interface is kept internal to the TLX. Using this interface credits are passed from the parser to the framer. At power up time the parser checks the depths of each FIFO to determine the number of credits and asserts TL credit valid signals.

Table 17: TLX Credit Management – TLX Credit Interface

Signal Name	Bits	Source	Description
rcv_xmt_tlx_credit_vc0	4	TLX RCV	Response credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_dcp0	6	TLX RCV	Response data credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_vc3	4	TLX RCV	Command credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_dcp3	6	TLX RCV	Command data credits returned to the TLX from the TL. Valid with rcv_xmt_credit_tlx_valid
rcv_xmt_tlx_credit_valid	1	TLX RCV	Indicates valid credits to be returned to the TLX from the TL.

Table 18: TLX Credit Management – TL Credit Interface

Signal Name	Bits	Source	Description
rcv_xmt_tl_credit_vc0_valid	1	TLX RCV	Indicates a response has been passed to the AFU, and increments the VC0 TL credit counter in the TLX Framer
rcv_xmt_tl_credit_dcp0_valid	1	TLX RCV	Indicates response data has been taken by the AFU and increments the DCP0 TL credit counter in the TLX Framer.
rcv_xmt_tl_credit_vc1_valid	1	TLX RCV	Indicates a command has been passed to the AFU, and increments the VC1 TL credit counter in the TLX Framer
rcv_xmt_tl_credit_dcp1_valid	1	TLX RCV	Indicates command data has been taken by the AFU and increments the DCP1 TL credit counter in the TLX Framer.
rcv_xmt_tl_crd_cfg_dcp1_valid	1	TLX RCV	Indicates configuration data has been taken by the AFU and increments the DCP1 TL credit counter in the TLX Framer.

Separate credit systems are used to manage the flow control between the AFU and the TLX.

CAPP command and response credits are used to throttle how many responses or commands that the TLX can push to the AFU. The signals in the two tables below further describe the functionality.

Table 19: TLX Credit Management – Credits for CAPP Commands

Signal Name	Bits	Source	Description
afu_tlx_cmd_credit	1	AFU	Returns a command credit to the TLX
afu_tlx_cmd_intial_credit	7	CNFG	Sets number of initial command credits available for the TLX to use in the AFU. Static value.

Table 20: TLX Credit Management – Credits for CAPP Responses

Signal Name	Bits	Source	Description
afu_tlx_resp_credit	1	AFU	Returns a response credit to the TLX
afu_tlx_resp_initial_credit	7	AFU	Indicates initial number of credits available to the TLX. Static value.

AP command and response credits are used between the TLX and AFU to throttle the number of commands or responses the AFU can send to the TLX. The table below outlines the signals that are used for this credit system. The command and response credits share an initial value signal for their respective credit pools, tlx_afu_cmd_resp_initial_credit.

Note: the signals in this interface we also listed in the Framer Interface section above.

Table 21: TLX Credit Management – Credits for AP Commands and Responses

Signal Name	Bits	Source	Description
tlx_afu_cmd_resp_initial_credit	3	TLX	Set to a constant by the TLX Framer and sent to the AFU to tell the AFU how many AP commands can be sent by the AFU to the TLX before a credit return is required. The same number is also used for AP responses. This value is hardcoded to 4 in the current reference design implementation.
tlx_afu_data_initial_credit	5	TLX	Set to a constant by the TLX Framer and sent to the AFU to tell the AFU how many command data packets can be sent by the AFU to the TLX before a data credit return is required. The same number is used for response data packets. This value is hardcoded to 16 in the current reference design implementation.

tlx_afu_resp_credit	1	TLX	TLX returns a response credit to the AFU when a response is taken from the FIFO. When this signal is asserted, the AFU can send another AP response to the TLX. The AFU needs to maintain a counter of credits for sending AP responses to the TLX. The counter should be decremented whenever a response is sent to the TLX. The counter should be incremented whenever tlx_afu_resp_credit is asserted. The counter should be initialized using the value from tlx_afu_cmd_resp_initial_credit. The AFU should not send any AP responses to the TLX when the credit count is zero.
tlx_afu_resp_data_credit	1	TLX	When this signal is asserted, the AFU can send another AP response data packet to the TLX. The AFU needs to maintain a counter of credits for sending response data packets to the TLX. The counter should be initialized using the value from tlx_afu_data_initial_credit. The counter should be incremented whenever tlx_afu_resp_data_credit is asserted. The counter should be decremented whenever an AP response with immediate data is sent to the TLX. (The counter should be decremented by the number of immediate data packets that are associated with the response.) The AFU should not send any AP responses with immediate data to the TLX unless there are enough data credits in this counter to support the number of immediate data packets associated with the response.
tlx_afu_cmd_credit	1	TLX	TLX returns a command credit to the AFU when a command is taken from the FIFO. When this signal is asserted, the AFU can send another AP command to the TLX. The AFU needs to maintain a counter of credits for sending AP commands to the TLX. The counter should be decremented whenever a command is sent to the TLX. The counter should be incremented whenever tlx_afu_cmd_credit is asserted. The counter should be initialized using the value from tlx_afu_cmd_resp_initial_credit. The AFU should not send any AP commands to the TLX when the credit count is zero.
tlx_afu_cmd_data_credit	1	TLX	When this signal is asserted, the AFU can send another AP command data packet to the TLX. The AFU needs to maintain a counter of credits for sending command data packets to the TLX. The counter should be initialized using the value from tlx_afu_data_initial_credit. The counter should be incremented whenever tlx_afu_cmd_data_credit is asserted. The counter should be decremented whenever an AP command with immediate data is sent to the TLX. (The counter should be decremented by the number of immediate data packets that are associated with the command.) The AFU should not send any AP commands with immediate data to the TLX unless there are enough data credits in this counter to support the number of immediate data packets associated with the command.

There is also a credit-based interface between the TLX and the DLX. The signals on this interface are used to manage the flow of flits from the TLX to the DLX.

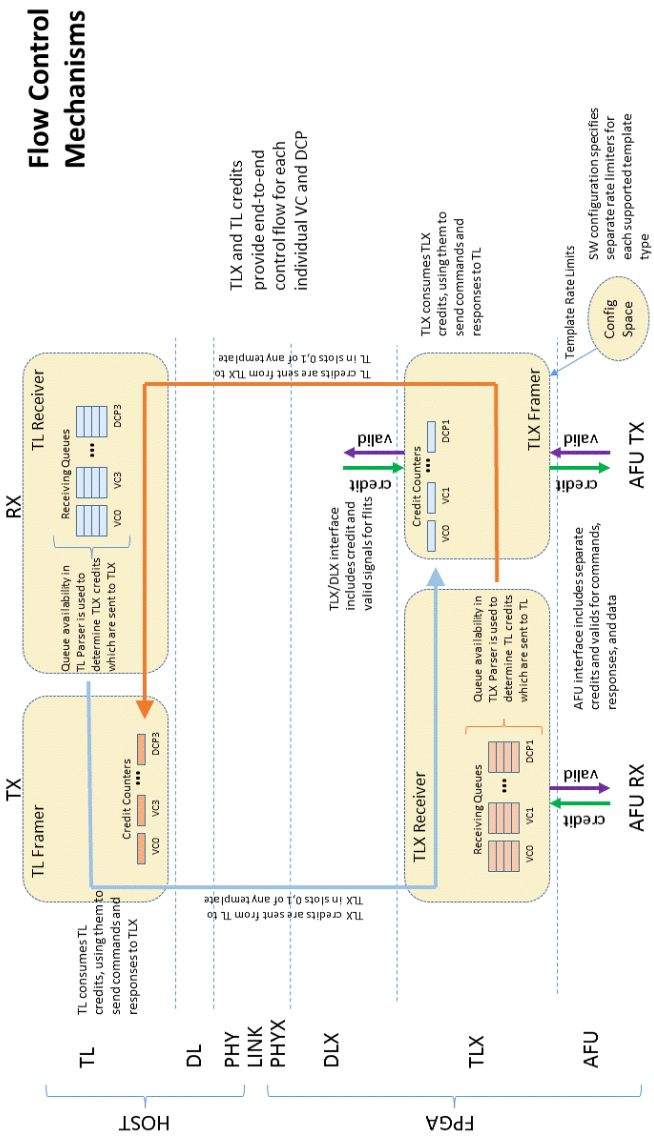
Note: These signals are also listed in the TLX Framer Interfaces section above.

Table 22: TLX Credit Management – TLX/DLX Interface

Signal Name	Bits	Source	Description

dlx_tlx_init_flit_depth	3	DLX	Initial DLX flit buffer depth. DLX informs the TLX the initial number of flits the DLX can handle before a credit must be returned. This signal should be set to a constant by the DLX.
dlx_tlx_flit_credit	1	DLX	DLX flit credit return. DLX informs the TLX that it has finished processing a flit, so another flit can be sent from the TLX to the DLX

Figure 19: TLX Flow Control Mechanisms



5 Configuration

Software running on the host can configure various aspects of the AFU and TLX behavior. The TL architecture defines CAPP config_read and config_write commands that can be used for that purpose. These configuration commands are passed through the TLX to the AFU, but unlike other CAPP commands, the configuration commands are passed to the AFU using a separate configuration interface. This configuration command interface is described in the TLX Parser chapter above. Similarly, configuration responses are passed back from the AFU to the TLX on a separate interface. This interface is described in the TLX Framer section above.

The TLX itself does not have any accessible configuration registers, but there are configuration registers and configuration logic in the AFU. There are some configuration signals that go between the AFU and the TLX to allow some TLX configuration using the configuration registers in the AFU.

Configuration registers in the AFU can be used to configure the TLX Framer to enable the use of specific templates. While template x"00" is enabled by default and must be supported to meet architecture requirements, templates x"01" and x"03" can be enabled or disabled to best suit the AFU designer's needs. Template x"02" is not supported for upstream traffic in this reference design. Configuration signals related to template x"02" are ignored.

Note that these TLX Framer configuration ports are also listed in the TLX Framer Interfaces (above).

Table 23: TLX Framer Template Configuration Ports

Signal Name	Bits	Source	Description
afu_cfg_xmit_tmpl_config_0	1	AFU	TLX Transmit Template 0 Enable. **OpenCAPI 3.0 Note:** Template 0 is enabled by default, as it is required by the link specification and cannot be disabled. This signal is ignored.
afu_cfg_xmit_tmpl_config_1	1	AFU	TLX Transmit Template 1 Enable.
afu_cfg_xmit_tmpl_config_2	1	AFU	TLX Transmit Template 2 Enable. **OpenCAPI 3.0 Note:** The TLX does not support the transmission of template 2. This signal is ignored.
afu_cfg_xmit_tmpl_config_3	1	AFU	TLX Transmit Template 3 Enable.
afu_cfg_xmit_rate_config_0	4	AFU	TLX Transmit Rate Template 0 Configuration.
afu_cfg_xmit_rate_config_1	4	AFU	TLX Transmit Rate Template 1 Configuration.
afu_cfg_xmit_rate_config_2	4	AFU	TLX Transmit Rate Template 2 Configuration. **OpenCAPI 3.0 Note:** The TLX does not support the transmission of template 2. This signal is ignored.
afu_cfg_xmit_rate_config_3	4	AFU	TLX Transmit Rate Template 3 Configuration.

There are also some configuration signals which are outbound from the TLX. These are static signals which specify the templates that can be received by the TLX Parser, and the template flit rates that the Parser can handle. These signals are set by the TLX Parser and are sent to the configuration registers in the AFU so they can be read by software and used to manage the TL. (They are sent to configuration registers in the AFU because the TLX does not have any configuration registers.)

Table 24: TLX Parser Template Configuration Ports

Signal Name	Bits	Source	Description
afu_cfg_in_rcv_tmpl_capability_0	1	TLX	TLX Receive Template 0 Enabled. Template 0 is always enabled, a requirement of the architecture. **OpenCAPI 3.0 Note**: TLX supports receiving all current templates
afu_cfg_in_rcv_tmpl_capability_1	1	TLX	TLX Receive Template 1 Enabled. **OpenCAPI 3.0 Note**: TLX supports receiving all current templates
afu_cfg_in_rcv_tmpl_capability_2	1	TLX	TLX Receive Template 2 Enabled. **OpenCAPI 3.0 Note:** TLX supports receiving all current templates
afu_cfg_in_rcv_tmpl_capability_3	1	TLX	TLX Receive Template 3 Enabled. **OpenCAPI 3.0 Note:** TLX supports receiving all current templates.
afu_cfg_in_rcv_rate_capability_0	4	TLX	TLX template 0 receive rate limit. **OpenCAPI 3.0 Note:** Minimum value supported = “0000”
afu_cfg_in_rcv_rate_capability_1	4	TLX	TLX template 1 receive rate limit. **OpenCAPI 3.0 Note:** Minimum value supported = “0011”
afu_cfg_in_rcv_rate_capability_2	4	TLX	TLX template 2 receive rate limit. **OpenCAPI 3.0 Note:** Minimum value supported = “0111”
afu_cfg_in_rcv_rate_capability_3	4	TLX	TLX template 3 receive rate limit. **OpenCAPI 3.0 Note:** Minimum value supported = “0010”

There is an additional vector that goes from the TLX to the AFU common Configuration logic which conveys the TLX version number, making the TLX version accessible to software.

Table 25: TLX Version Hardware Version Number

Signal Name	Bits	Source	Description
tlx_cfg_oc3_tlx_version	32	TLX	Send the TLX hardware version number over to the common Configuration space so that the TLX version can be read by software. This value is hardcoded in the TLX hardware. The format of the TLX version number is: h171016A0 where 17 = year (2017), 10 = month (Oct), 16 = day, and A,B,C,... = iterations on that day

6 Error Detection and Handling

The TLX logic is required to detect certain types of errors that may occur in the transmission of traffic between the host and the AFU. These errors are listed in the "OpenCAPI™ 3.0 Transaction Layer Specification".

The majority of the errors required by the TL specification for OpenCAPI 3.0 are detected in the TLX and reported to the host using the tlx_dlx_debug_encode and the tlx_tlx_debug_info interface to the DLX. (The TLX logic does not have any software accessible registers.)

The only exception is the Bad response received error. The AFUTag is not tracked in the TLX and can only be checked by the AFU. (Ed Note - This error is detected by the AFU. How do we get this error reported to the host? We may need to add an interface from the AFU to get this error.)

Commented [A1]: Due to this, do we need to add an error interface to the AFU?

The error detection registers in the TLX logic are "sticky" which means that if an error is detected, the error indicator bits stay asserted until they are cleared by software. Software should use bit [31] of the dlx_tlx_dlx_config_info signal to clear all the sticky error bits in TLX.

The TLX ports related to error handling are shown in the following table. Note that these ports are also listed in the TLX Framer interfaces (above).

Table 26: TLX/DLX Error Information Interface

Signal Name	Bits	Source	Description
dlx_tlx_dlx_config_info	32	DLX	Miscellaneous configuration bits from DLX Bit (31) is used to reset the sticky error capture registers in TLX
tlx_dlx_debug_encode	4	Framer	Debug encode information from TLX to DLX: Bit(0) – indicates a fatal error was detected Bit(1) – indicates that the debug info is coming from the AFU Bit(2) – indicates that the debug info is coming from the TLX Parser Bit(3) – indicates that the debug info is coming from the TLX Framer

tlx_dlx_debug_info	32	Framer	32 bits of information about errors detected in the hardware, from TLX to DLX

The following signals are internal to the TLX. They are used to send error information from the TLX Parser to the TLX Framer.

Table 27: Parser to Framer Error Information Interface

Signal Name	Bits	Source	Description
rcv_xmt_debug_valid	1	TLX Parser	Indicates that the other two signals on this interface contain valid information
rcv_xmt_debug_fatal	1	TLX Parser	Flag to indicate that TLX Parser has detected a fatal error (If valid is asserted.)
rcv_xmt_debug_info	32	TLX Parser	32 bits of information about errors detected in the TLX Parser