



Science & Technology
Facilities Council

UK Research
and Innovation

DUNE Ethernet Readout 100G UDP IP CORE

Rob Halsall

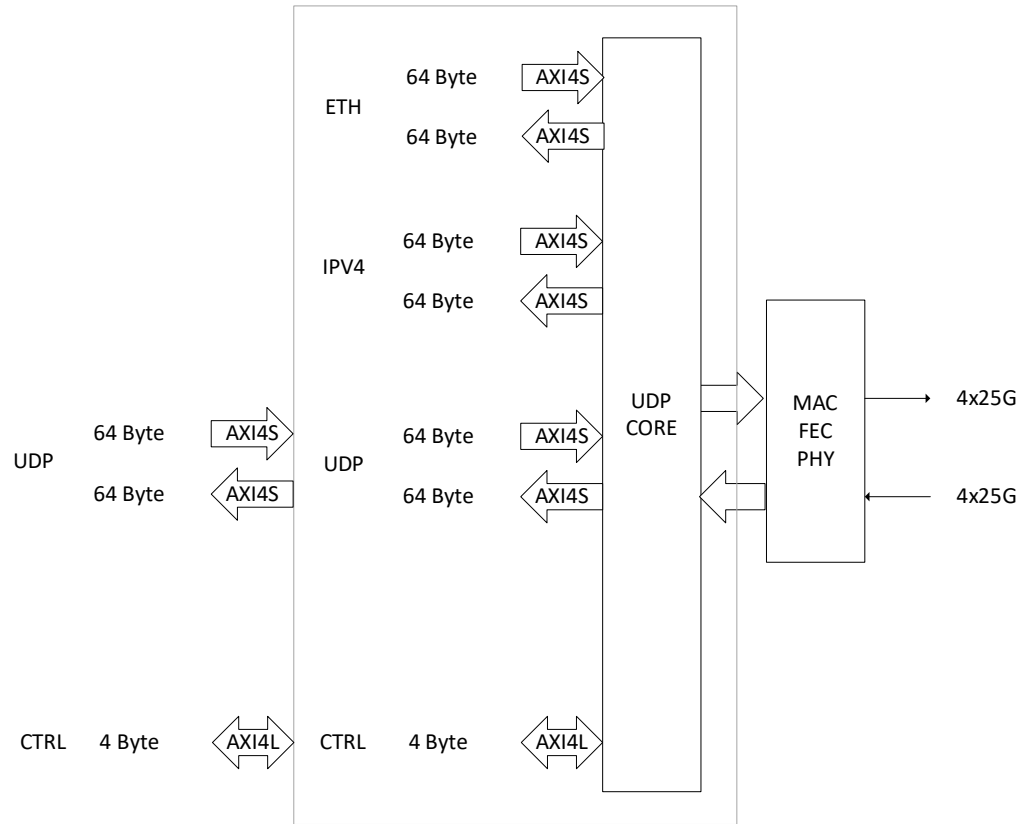
Matt Roberts

Chris Lyford

14-10-2021



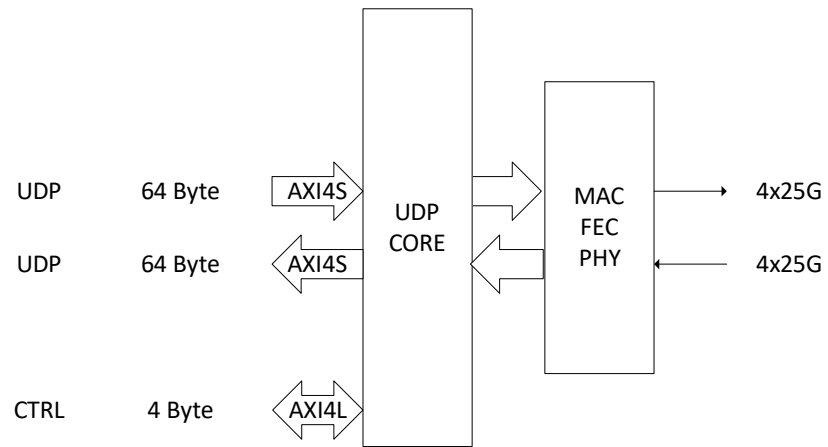
100G UDP Core



- Parameterised & Scalable
- UDP plus extendable – via ETH & IPV4 ports
- ARP & PING compile options
- Control Registers Mem Mapped via AXI4L
- UDP Tx & Rx via AXI4S
- Uses TKEEP to allow 'any' packet size
- Supports Jumbo frames
- Optional 'Farm Mode' on Tx Channel using TID
- Traffic Filter on Rx Channel
- Clocks - AXI4S ~195MHz & PHY ~322MHz
- Wrap to hide excess functionality



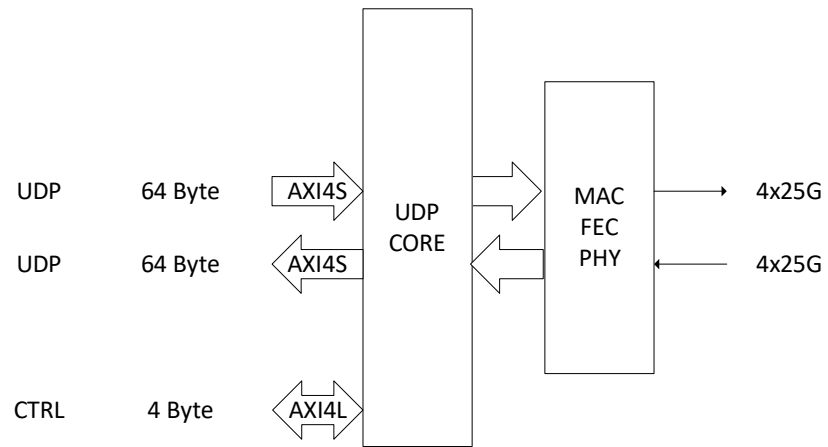
100G UDP Core



- AXI4S
- 64Byte @ 195MHz nominal
- AXI4S Tx data frame -> Outgoing UDP packet
- Incoming UDP packet -> AXI4S Rx data frame
- Loopback will return tx data frame on rx channel



100G UDP Core



- AXI4L
 - xml file – flat memory map
- Approximately 32x 32 bit Control Registers
 - Source & destination addresses - mac/ip/port
 - Rx Filter settings
 - Stats registers
 -
- Tx Channel Farm Mode RAM
 - 256 entry mac/ip/port table
 - Selected by Tx AXI4S 8 bit TID signal
 - mac entries can be arp populated



100G UDP Core – Entity/generic

```
G_FPGA_VENDOR      : string := "xilinx";      --! Selects the FPGA Vendor for Compilation
G_FPGA_FAMILY      : string := "all";          --! Selects the FPGA Family for Compilation
G_FIFO_IMPLEMENTATION : string := "auto";      --! Selects how the Fitter implements the FIFO Memory
G_FIFO_TYPE        : string := "inferred_mem"; --! Selects how to implement the Core's FIFOs
G_NUM_OF_ARP_POS    : natural := 32;          --! Number Of Positions In ARP Table
G_UDP_CLK_FIFOS     : boolean := false;       --! Generate Clk Crossing FIFOs At The I/O of UDP Payloads
G_EXT_CLK_FIFOS     : boolean := false;       --! Generate Clk Crossing FIFOs At The I/O of Unsupported Packet Types
G_TX_EXT_IP_FIFO_CAP : integer := 2048;       --! Capacity In Bytes Of Uns IPV4 Protocol Packets FIFOs in Tx Path
G_TX_EXT_ETH_FIFO_CAP : integer := 2048;      --! Capacity In Bytes Of Uns Ethernet Type Packets FIFOs in Tx Path
G_TX_OUT_FIFO_CAP   : integer := 2048;       --! Capacity Of FIFO at End Of Tx Path. Necessary For 100GbE, Less Important for 1/10GbE. Can Be 0.
G_RX_IN_FIFO_CAP    : integer := 2048;       --! Capacity Of FIFO at Start Of Rx Path. At least 16 Words Recommended.
G_RX_INPUT_PIPE_STAGES : integer := 1;       --! Pipeline Stages From External MAC/PHY To Rx Path
G_INC_PING          : boolean := true;        --! Generate Logic For Internal Ping Replies
G_INC_ARP           : boolean := true;        --! Generate Logic For Internal ARP Requests And Replies
G_CORE_FREQ_KHZ     : integer := 322262;     --! KHz Of Tx Path, Used For ARP Refresh Timers, Not Essential
G_INC_ETH           : boolean := false;       --! Generate Logic To Transmit Externally Provided Ethernet Payloads
G_INC_IPV4           : boolean := false;      --! Generate Logic To Transmit Externally Provided IPV4 Payloads
```

- Porting across Vendor/families requires wrapped/configured/licensed mac/fec/phy



100G UDP Core – Entity/Ports/UDP AXI4S

```
-- AXI4-Stream UDP Transmit Data:
udp_axis_s_clk      : in std_logic;                --! UDP Tx In Clk
udp_axis_s_reset    : in std_logic;                --! UDP Tx In Reset
udp_axis_s_tdata     : in std_logic_vector(511 downto 0); --! UDP Transmit Axi4s
udp_axis_s_tvalid    : in std_logic;                --! UDP Transmit Axi4s
udp_axis_s_tkeep     : in std_logic_vector(63 downto 0); --! UDP Transmit Axi4s
udp_axis_s_tlast     : in std_logic;                --! UDP Transmit Axi4s
udp_axis_s_tid       : in std_logic_vector(7 downto 0); --! UDP Transmit Axi4s, route packets via LUT entry using this signal & Farm Mode
udp_axis_s_tuser     : in std_logic_vector(31 downto 0); --! UDP Transmit Axi4s
udp_axis_s_tready    : out std_logic;               --! UDP Transmit Axi4s

-- AXI4-Stream UDP Receive Data:
udp_axis_m_clk      : in std_logic;                --! UDP Rx Out Clk
udp_axis_m_reset    : in std_logic;                --! UDP Rx Out Reset
udp_axis_m_tdata     : out std_logic_vector(511 downto 0); --! UDP Receive Axi4s
udp_axis_m_tvalid    : out std_logic;               --! UDP Receive Axi4s
udp_axis_m_tkeep     : out std_logic_vector(63 downto 0); --! UDP Receive Axi4s
udp_axis_m_tlast     : out std_logic;               --! UDP Receive Axi4s
udp_axis_m_tid       : out std_logic_vector(7 downto 0); --! UDP Receive Axi4s
udp_axis_m_tuser     : out std_logic_vector(31 downto 0); --! UDP Receive Axi4s
udp_axis_m_tready    : in std_logic;
```

- Non-record structure AXI4S Wrapper – required for IP Packaging/Vitis



100G UDP Core – Entity/Ports/PHY AXI4S

```
tx_core_clk      : in std_logic;           --! Tx Path Main Clock, Set To PHY Tx Clk or Set to Lower Clk and Buffer Full Packets Before PHY
rx_core_clk      : in std_logic;           --! Rx Path Main Clock, can set to PHY Rx Clock, or any f down to ~200MHz
-- AXI4-Stream Slave Side (UDP Core Tx Data):
rx_axis_s_clk     : in std_logic;           --! UDP Core Rx PHY Clock
rx_axis_s_rst     : in std_logic;           --! UDP Core Rx Active-High Reset
rx_axis_s_tdata    : in std_logic_vector(511 downto 0); --! Rx Axi4s Signals From PHY
rx_axis_s_tvalid   : in std_logic;           --! Rx Axi4s Signals From PHY
rx_axis_s_tkeep    : in std_logic_vector(63 downto 0); --! Rx Axi4s Signals From PHY
rx_axis_s_tlast    : in std_logic;           --! Rx Axi4s Signals From PHY
rx_axis_s_tid      : in std_logic_vector(7 downto 0);  --! Rx Axi4s Signals From PHY (unused)
rx_axis_s_tuser    : in std_logic_vector(31 downto 0); --! Rx Axi4s Signals From PHY (unused)
rx_axis_s_tready   : out std_logic;          --! Rx Path Backpressure, only used for debugging
-- AXI4 Stream Master Out (UDP Core Rx Data):
tx_axis_m_clk     : in std_logic;           --! UDP Core Tx PHY Clock
tx_axis_m_rst     : in std_logic;           --! UDP Core Tx Active-High Reset
tx_axis_m_tdata    : out std_logic_vector(511 downto 0); --! Tx Axi4s Signals To PHY
tx_axis_m_tvalid   : out std_logic;          --! Tx Axi4s Signals To PHY
tx_axis_m_tkeep    : out std_logic_vector(63 downto 0); --! Tx Axi4s Signals To PHY
tx_axis_m_tlast    : out std_logic;          --! Tx Axi4s Signals To PHY
tx_axis_m_tid      : out std_logic_vector(7 downto 0);  --! Tx Axi4s Signals To PHY (unused)
tx_axis_m_tuser    : out std_logic_vector(31 downto 0); --! Tx Axi4s Signals To PHY (unused)
tx_axis_m_tready   : in std_logic;          --! Tx PHY Backpressure, USED
```

- Interface to Wrapped MAC/FEC/PHY
- Non-record structure AXI4S Wrapper – required for IP Packaging/Vitis



100G UDP Core – Entity/Ports/AXI4L

```
-- AXI4-Lite Interface (defined in axi4lite_pkg):
axi4lite_aclk      : in std_logic;                --! AXI4-Lite Clock
axi4lite_aresetn   : in std_logic;                --! AXI4-Lite Active-Low Asynchronous Reset
axi4lite_araddr    : in std_logic_vector(31 downto 0); --! Axi4-Lite Signals
axi4lite_awaddr    : in std_logic_vector(31 downto 0); --! Axi4-Lite Signals
axi4lite_arvalid   : in std_logic;                --! Axi4-Lite Signals
axi4lite_awvalid   : in std_logic;                --! Axi4-Lite Signals
axi4lite_bready    : in std_logic;                --! Axi4-Lite Signals
axi4lite_rready    : in std_logic;                --! Axi4-Lite Signals
axi4lite_wdata     : in std_logic_vector(31 downto 0); --! Axi4-Lite Signals
axi4lite_wstrb     : in std_logic_vector(3 downto 0); --! Axi4-Lite Signals
axi4lite_wvalid    : in std_logic;                --! Axi4-Lite Signals
axi4lite_arready   : out std_logic;               --! Axi4-Lite Signals
axi4lite_awready   : out std_logic;               --! Axi4-Lite Signals
axi4lite_bresp     : out std_logic_vector(1 downto 0); --! Axi4-Lite Signals
axi4lite_bvalid    : out std_logic;               --! Axi4-Lite Signals
axi4lite_rdata     : out std_logic_vector(31 downto 0); --! Axi4-Lite Signals
axi4lite_rresp     : out std_logic_vector(1 downto 0); --! Axi4-Lite Signals
axi4lite_rvalid    : out std_logic;               --! Axi4-Lite Signals
axi4lite_wready    : out std_logic;               --! Axi4-Lite Signals
```

- Non-record structure AXI4L Wrapper – required for IP Packaging/Vitis



100G UDP Core

- Hosted on STFC Gitlab
- Automatic documentation
- Sphinx/Read the Docs/doxygen
- Mem Map Tables from XML
- Helper Scripts
- VUNIT Simulation
- BSD 3-Clause License
- packaged

UDP Core

Science & Technology
Facilities Council

1.0.1

Search docs

UDP Core

License

UDP Speeds

Instantiating the Core

Memory Map

Rx Path

Testbench

Tx Path

Core Settings

Changelog

Doxygen Index

Docs » UDP Core

View page source

Science & Technology
Facilities Council

Electronic System Design Group

docs passing

UDP Core

The [UDP Core](#) Converts between Axi4s Data Frames & 802.3 Ethernet Packets. Incoming UDP payloads are enclosed into 802.3 Ethernet Packets to be transmitted over an Ethernet Network by a PHY IP. Similarly, UDP Payloads are extracted from received Packets which pass user set filtering requirements.

The core can be generated with the capabilities to reply to Ping & ARP Requests, and send ARP requests to maintain its own ARP table. It also can be generated to receive and transmit other Ethernet Type or IPV4 Protocol Packets, allowing the user to add other packet capabilities.

The Core is designed to be scalable, and can operate at speeds of 1GbE, 10GbE, 40GbE and 100GbE. It is also Vendor agnostic and has been tested on both IntelFPGA and Xilinx devices.

The operation of the core is controlled using a series of memory mapped control registers accessed via an Axi4lite interface, described in [Core Settings](#) . The full memory map of the core including absolute addresses can be seen at [Memory Map](#).

The architecture of the Core is split up into two main functional blocks; the [Tx Path](#) and the [Rx Path](#). Information on instantiating the UDP Core can be found in [Instantiating the Core](#).

A [Testbench](#) and a script to drive it is also included in the project. It can be used as an example on how to instantiate the Core.

Control plane Axi4lite modules required by the Core need to be generated using XML2VHDL before use. The testbench script drives XML2VHDL to produce the necessary files. You will need to modify the testbench script to add the location off the main XML2VHDL script.

License

BSD 3-Clause License

Copyright(c) 2021 UNITED KINGDOM RESEARCH AND INNOVATION

Electronic System Design Group, Technology Department, Science and Technology Facilities Council

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.



100G UDP Core

UDP Core

Science & Technology
Facilities Council

1.0.1

Search docs

UDP Core

License

UDP Speeds

Instantiating the Core

Memory Map

Rx Path

Testbench

Tx Path

Core Settings

Changelog

Doxygen Index

Docs » Memory Map

[View page source](#)



docs **passing**

Memory Map

Auto Generated Register Tables

The following tables have been automatically generated using the XML file used to create the Memory-Mapped Register locations. Descriptions have been extracted from the XML file used to generate the Memory-Mapped Registers and their corresponding fields.

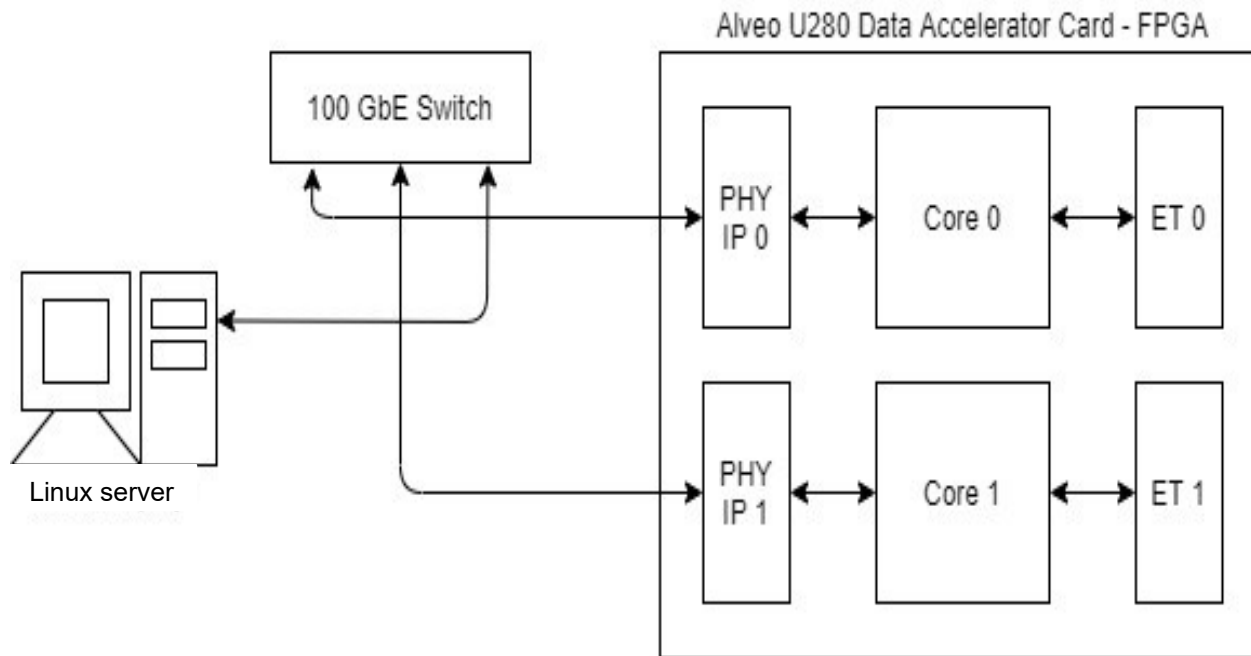
All Register

The Absolute Addresses are shown at this level of hierarchy. They may change if a connection to an upstream AXI4-Lite is made.

Register	Absolute Address
udp_core_top_udp_core_control.src_mac_addr_lower	0x00000000
udp_core_top_udp_core_control.src_mac_addr_upper	0x00000004
udp_core_top_udp_core_control.dst_mac_addr_lower	0x0000000C
udp_core_top_udp_core_control.dst_mac_addr_upper	0x00000010
udp_core_top_udp_core_control.ethertype	0x00000014
udp_core_top_udp_core_control.ipv4_header_0	0x00000018
udp_core_top_udp_core_control.ipv4_header_1	0x0000001C
udp_core_top_udp_core_control.ipv4_header_2	0x00000020
udp_core_top_udp_core_control.dst_ip_addr	0x00000024
udp_core_top_udp_core_control.src_ip_addr	0x00000028
udp_core_top_udp_core_control.udp_ports	0x0000002C
udp_core_top_udp_core_control.udp_length	0x00000030
udp_core_top_udp_core_control.filter_control	0x00000038
udp_core_top_udp_core_control.ifg	0x00000040
udp_core_top_udp_core_control.control	0x00000048
udp_core_top_udp_core_control.udp_count	0x0000004C
udp_core_top_udp_core_control.ping_count	0x00000050
udp_core_top_udp_core_control.arp_count	0x00000054
udp_core_top_udp_core_control.uns_etype_count	0x00000058
udp_core_top_udp_core_control.uns_prio_count	0x0000005C
udp_core_top_udp_core_control.dropped_mac_count	0x00000060
udp_core_top_udp_core_control.dropped_ip_count	0x00000064
udp_core_top_udp_core_control.dropped_port_count	0x00000068
udp_core_top_udp_core_control.ip_id	0x0000006C
udp_core_top_udp_core_control.udp_core_id	0x00000074
udp_core_top_arp_node_control.arp_control	0x00000080
udp_core_top_arp_node_control.positions_active_0	0x00000084
udp_core_top_arp_node_control.positions_active_1	0x00000088
udp_core_top_arp_node_control.positions_active_2	0x00000090

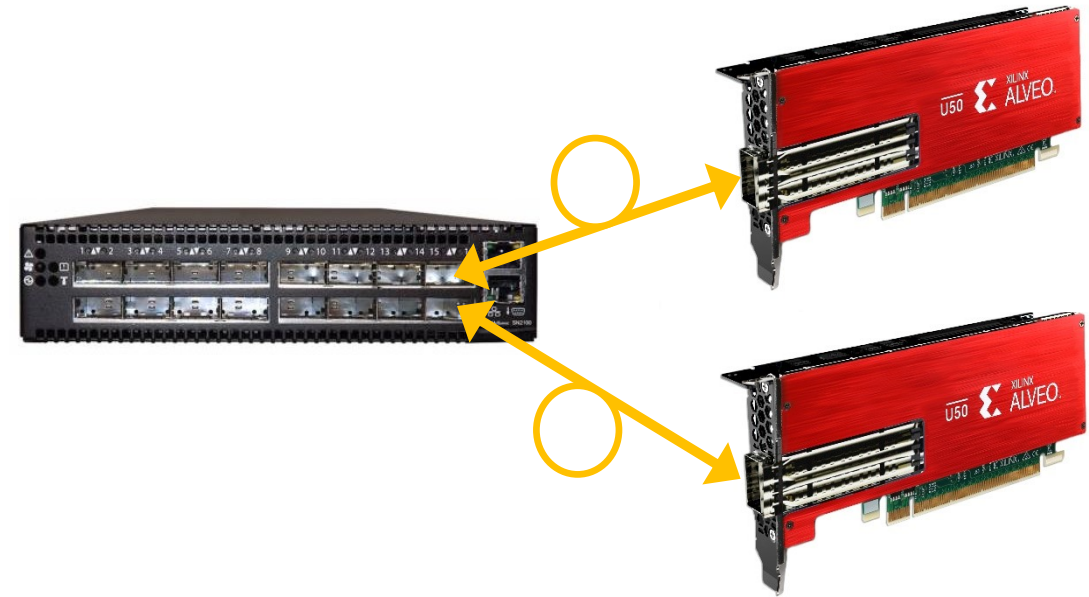
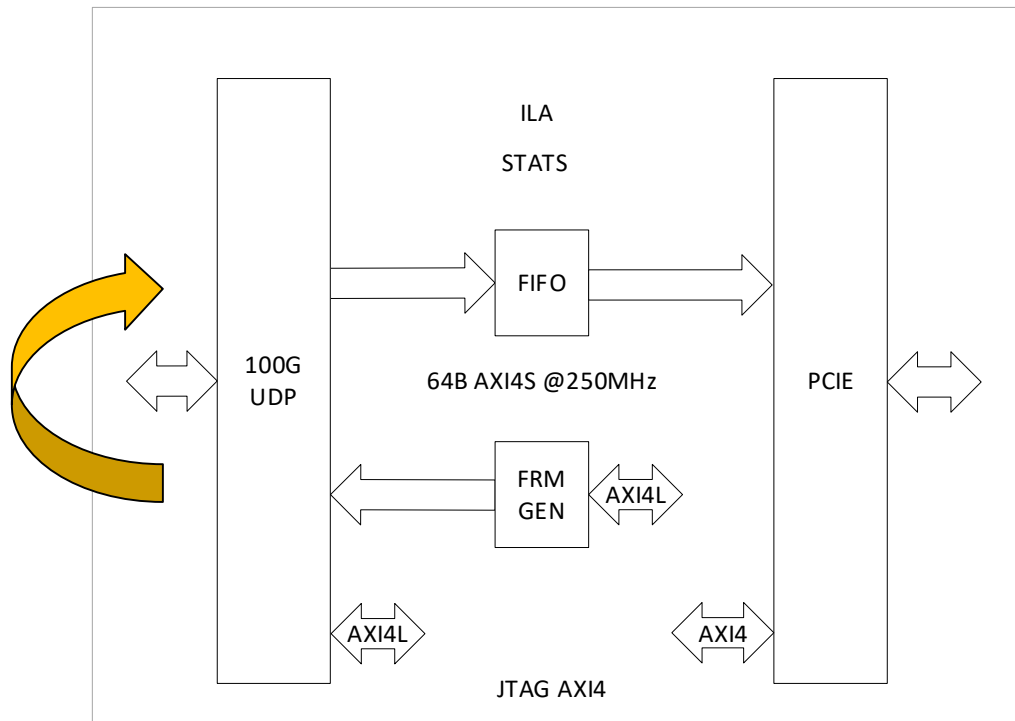
- Hosted on STFC Gitlab
- Automatic documentation
- Sphinx/Read the Docs/doxygen
- Mem Map Tables from XML
- Helper Scripts
- VUNIT Simulation
- BSD 3-Clause License
- packaged

100G UDP Core – H/W Test



- Alveo U280 & HTG RFSoc
- FPGA <-> FPGA via switch
- Ran Embedded Test overnight, no errors or dropped packets
- Script reads switch counters over time period to give a measured speed – 99.8 Gb/s
- JTAG AXI4L core & xtcl setup
- ILA

100G UDP-PCle Test Top Level



Full Processing FPGA Top Level

