BitSim

Data Sheet

Bit-10G-UDP Ethernet

10 Gb/s full HW stack UDP/IP Transmitter/Receiver IP block for high performance data transfers from embedded Ethernet based systems

1. Document Information

1.1. Definitions and abbreviations

Term	Explanation
ARP	Address Resolution Protocol
AXI	Advanced eXtensible Interface
CRC	Cyclic Redundancy Check
DDR	Double Data Rate
DIC	Deficit Idle Count
EA	Ethernet Address
GMII	Gigabit Media Independent Interface
IP	Internet Protocol
MAC	Media Access Control
MII	Media Independent Interface
PCIe	Peripheral Component Interconnect express
PCS	Physical Coding Sublayer
PHY	Physical layer
PMA	Physical Medium Attachment
UDP	User Datagram Protocol
XGMII	10 Gigabit MII

2. Introduction

Bit-10G-UDP is a configurable design for FPGA or ASIC* and it provides means to encapsulate data in UDP packets. Any design can be connected to a regular 10Gbps Ethernet network and used to send and receive data. The design can be configured to be completely autonomous with communication to an unknown host, or it can be configured to communicate with one or more peers using different IP addresses in the same or different subnets.
Bit-10G-UDP implements the XGMII SDR interface. The design can be connected to a PCS/PMA component operating at 10Gbit/s SDR. Optionally, DDR flipflops can be used to convert from SDR to DDR if connecting to an XGMII capable PHY.

Deliveries

- HDL source code (readable or encrypted) and testbench with stimuli and checkers
- PC SW test code (Windows, Linux)
- VHDL Simulation Test Benches with BitSim’s Simulation Environment (BSE)
- Expert technical support and maintenance
- User guide

Licensing

- For more information or license purchase, contact sales@bitsim.com

*targeting ASIC requires adaption for RAM components

3. System Description

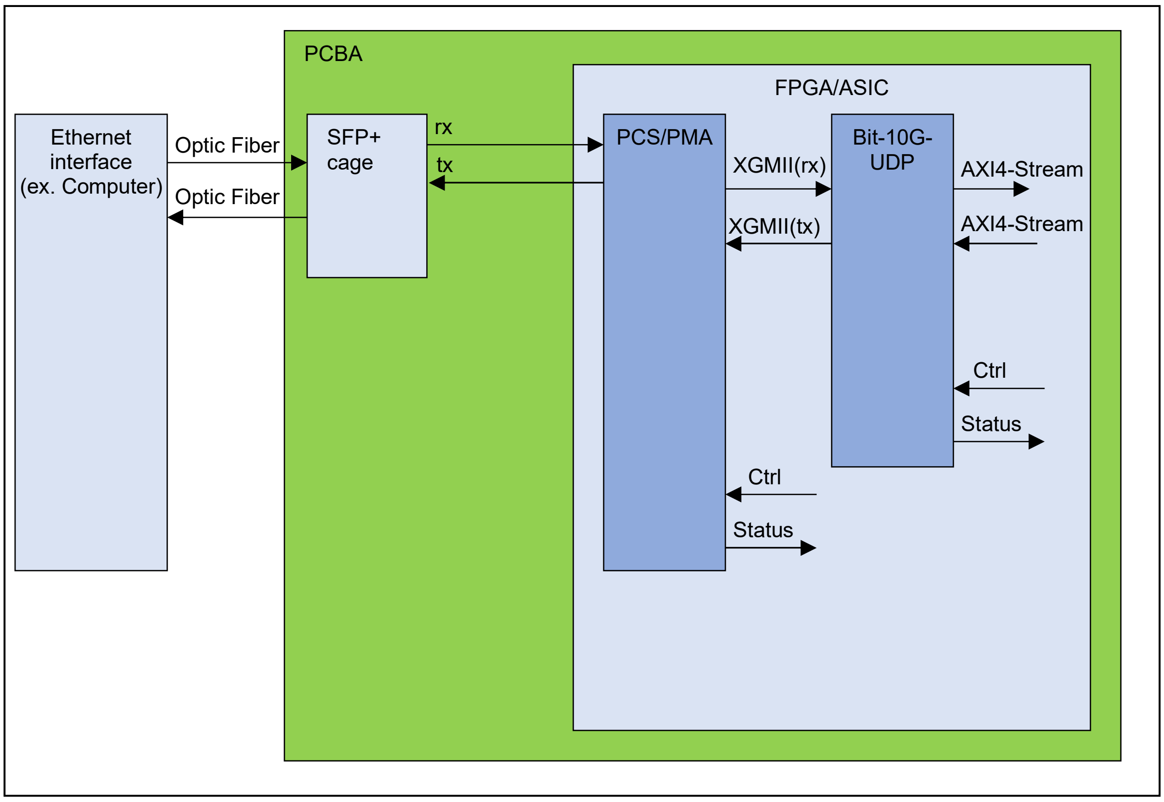

Figure 1 Bit-10G-UDP in a system

For this example, the ethernet traffic travels on an optical cable. The Ethernet interface of, in our example a computer, is connected to one end of the cable, the SFP+ cage with a fibre transceiver mounted is connected on the other end. The fibre transceiver gives the data directly to the FPGA.

A vendor specific PCS/PMA block is used to receive those signals and then output the Ethernet traffic on the XGMII interface of the Bit-10G-UDP block. The Bit-10G-UDP block checks the Ethernet packets and confirms they have the right address and port number. If no errors have been found during the decoding of the packets the data is output on the AXI4-Stream interface where your design can further process the data. When your design is sending data Bit-10G-UDP receives the data on another AXI4-Stream. Checks if it knows the MAC address of the recipient and if not, it uses ARP to resolve that address. Then it sends the packaged payload over the XGMII interface to the vendor-specific PCS/PMA block, which in turn sends it to the fibre transceiver which will send it out to the computer.

4. Features

- Maximum Tx rate (in 8 steps from 1G up to 10G) can be configured statically at synthesis-time
- Maximum Rx rate is 1G
- AXI4-Stream data input and output
- IP addresses and port numbers can be configured dynamically at run-time
- Checksum checking for packet payload data
- A vendor specific integrated 10 Gb/s PCS/PMA PHY enables a direct connection to SFP+ cages
- Loopback at both AXI-Stream and XGMII interfaces are possible for debugging purposes
- Written in VHDL, prepared for instantiation in Verilog or SystemVerilog design
- Statistics counters and protocol filters

4.1. Block Description

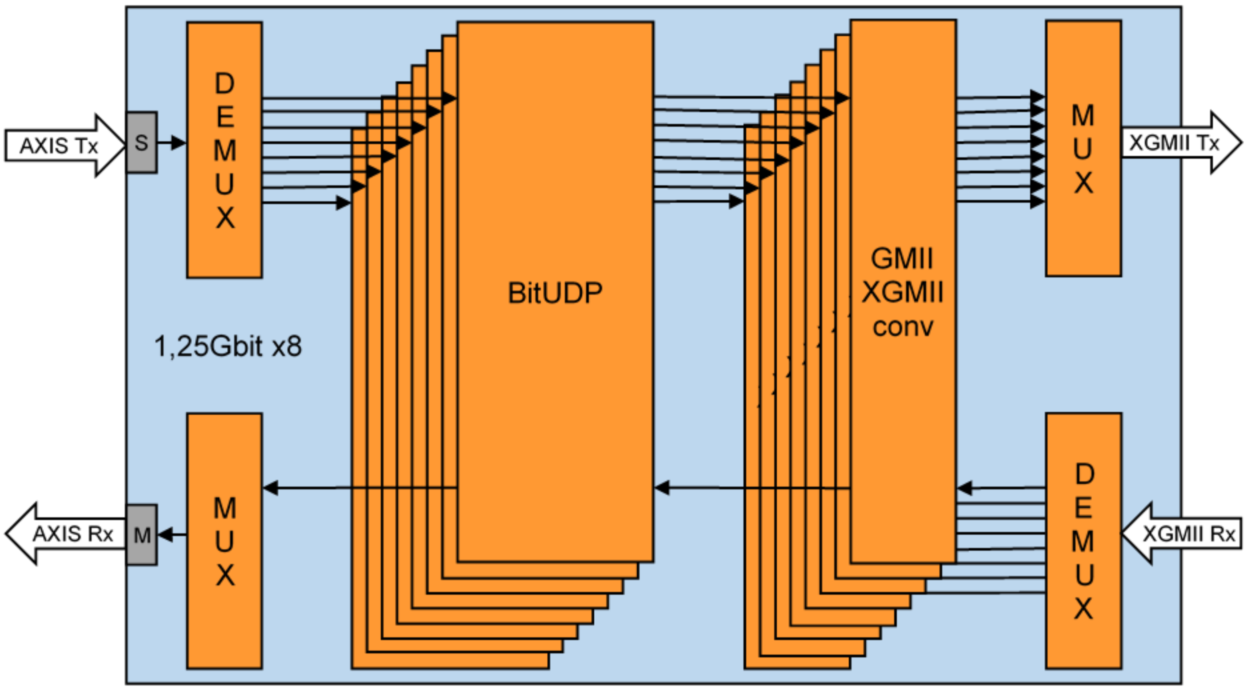

Figure 2 Bitcsi2rx block diagram

shows the Bit-10G-UDP block diagram at full tx speed.

For embedded applications with extremely high demands for fast data transfers, BitSim has launched its IP core, Bit-10G-UDP Ethernet. This core is a real-time offload engine where the communication is accelerated in the FPGA, and uses the 10 Gb/s Ethernet protocol to the host. Standard functions for transmit, receive, ARP-handling, AXI-stream and loop back are included.

Demux(tx)

Forwards the incoming payload data to be sent to one of the Bit-1G-UDP instances for further processing.

Demux(rx)

Forwards the incoming Ethernet data to be received to one of the GMII-XGMII converter instances for further processing.

BitUDP

An instance of Bit-1G-UDP.

GMII-XGMII converter

Bit-1G-UDP speaks GMII to an Ethernet PHY, but 10G PHYs speak XGMII. This block handles the conversion between the protocols.

Mux(tx)

This mux takes Ethernet data from one of the converter instances that is ready to send and outputs it on the XGMII interface.

Mux(rx)

This mux takes payload data from the Bit-1G-UDP instance that has data available and outputs it on the AXI-stream interface.

Subblocks*:

Clocks and reset
The AXI stream interface is asynchronous with the internal logic, which is driven by the Ethernet transmit clock except for the Ethernet MAC.

SDR clocking
The byte and bit alignment and ordering for XGMII is described here.

Interpacket gap, deficit idle count
There must be a minimum number of idle cycles between packets.

ARP Handling
There are several methods that can be used for ARP handling. ARP requests can be triggered either when there is AXI stream Tx data, or it can be triggered in advance.

AXI Stream
Implements a subset of AXI stream.

5. Ports

This port list is not complete, it is only intended for an overview*.

Signal	Range	Dir	Description
			System
rst	-	in	Active high asynchronous reset. The reset must be active during 5 cycles of the slowest of clk and xgmii_tx_clk clocks.
rst_core	-	in	Active high asynchronous reset.
clk	-	in	System clock. Used only with AXI stream. Independent of Ethernet transmit and receive clocks. This clock is the source to AXI master and slave ports. Some configuration and status signals are semi-static and not sourced by this clock, even though usage is related to it.

Cfg local (xgmii_tx_clk)			
cfg_local_ea	-	in	Local Ethernet/MAC address
cfg_local_ip	-	in	Local IP address
cfg_local_udp_port	-	in	Local UDP listen port
Cfg peer (xgmii_tx_clk)			
cfg_peer_ip	-	in	Peer IP address
cfg_peer_udp_port	-	in	Peer UDP listen port
cfg_peer_en	-	in	‘1’ = ARP table entry, or index, is enabled (peer communication with peer <index> is enabled) ‘0’ = Communication with peer <index> is disabled. Transmissions using peer<index> will be discarded internally and the data will be lost.
Cfg network (xgmii_tx_clk)			
cfg_gw_ip	-	in	Gateway IP address. Used when destination/peer IP is in a different subnet.
cfg_submask	-	in	Mask for local subnet. Used to determine if a peer IP address belongs to the local subnet.
Cfg ARP bypass (xgmii_tx_clk)			
cfg_peer_ea	-	in	Used when fixed EA/MAC address is enabled. If changing this signal, and the corresponding entry is not configured to use fixed EA, the ARP table is not automatically flushed and updated.
cfg_peer_ea_fixed_en	-	in	‘1’ = Enable fixed EA/MAC address for peer <index> ‘0’ = Use ARP to resolve EA/MAC address for peer <index>
cfg_peer_gw_ea	-	in	Used when fixed gateway EA/MAC address is enabled
cfg_peer_gw_ea_fixed_en	-	in	‘1’ = Enable fixed EA/MAC address for the gateway ‘0’ = Use ARP to resolve EA/MAC address for the gateway
Cfg legacy (xgmii_tx_clk)			
cfg_arp_rx_sniff_en	-	in	‘1’ = Use the IP address from the most recent incoming UDP/ARP request during Tx and allow any IP during Rx. This is useful in applications where the peer IP address is unknown. ‘0’ = Use ARP table entry <index>.
cfg_arp_table_use_req_en	-	in	‘1’ = ARP table is also updated by incoming requests ‘0’ = ARP table is only updated by incoming replies

Cfg AXI stream (clk)			
cfg_pcie_fmt_en	-	in	‘1’ = PCIe header format enabled for Tx. Instead of the size field, each Tx packet is expected to be prepended with a PCIe formatted header. This can be used to tunnel PCIe traffic. For Rx, a PCIe header is simply considered part of the UDP payload. ‘0’ = Tx uses the size field.
cfg_tx_prefill_level	-	in	2^n AXI words for each packet are buffered before UDP Tx is started internally in BitUDP1G. This provides some elasticity to prevent underrun situations. The AXI stream Tx FIFO depth must be configured to be large enough to hold the desired prefill level. Valid values for n: $0 \leq n \leq 3$
cfg_tx_prefill_packet	-	in	‘1’ = Complete packet buffered, overrides level prefill. If the AXI stream cannot be supplied with a complete packet at the required rate in a slow system, this enables buffering of the complete packet before initiating transmission. The AXI stream Tx FIFO must be configured to be large enough for a complete packet. ‘0’ = Initiate transmission towards IP_UDP_CORE as soon as there is AXI stream data available.
Cfg ctrl (gmii_tx_clk)			
cfg_validate_rx_packet_en	-	in	‘1’ = Complete Rx packet is validated (IP header, CRC) before UDP Rx data is driven ‘0’ = UDP Rx data is driven immediately upon reception and could prove to be erroneous by Rx error interrupt/status. Intended to be used when low Rx latency is preferred to internal packet validation.
cfg_rx_size_hdr_en	-	in	‘1’ = Prepend size header to UDP Rx data. This makes the Rx format identical to Tx. Suitable when size of the incoming UDP packet is of interest. It also enables looping Rx data back on Tx without further processing. If using IP_UDP_CORE standalone, a FIFO between Rx and Tx is

			required for loopback. '0' = Only drive UDP Rx data
cfg_tx_ttl	-	in	Time to live. The recommended value is 64 to 128. Can be of importance when transmitting to a different subnet with many hops.
Ctrl (xgmii_tx_clk)			
trig_arp_request	-	in	'1 '= Populate ARP table entry for peer <index>. Sends ARP requests and retrieves the EA/MAC addresses for all enabled peers. All bits can be tied off to '1' if desired. '0' = Populate ARP entry for <index> when Tx wants to send data. Mainly useful for applications with communication with a single peer. This is mostly a legacy feature.
flush_arp_table	-	in	'1' = Scrap any stored EAs, soft reset (new peer HW, timeout for lost ARP reply, etc...) '0' = Normal operation Note: Ignored when busy, do not assert during transmission and expect a flush.
clear_packet_cnt	-	in	'1' = Clear statistics counters '0' = Normal operation
Interrupts and status (clk or xgmii_tx_clk)			
irq_sts_arp_done	-	out	Continuous status. May be used to derive interrupt signalling.
irq_sts_arp_sniff_done	-	out	Continuous status. May be used to derive interrupt signalling.
irq_sts_pause	-	out	Continuous status. May be used to derive interrupt signalling.
irq_tx_notvld_done	-	out	Single-cycle interrupt. Indicates completed transmission of a good (or bad, future use) packet of any type except valid UDP packet.
irq_tx_done	-	out	Single-cycle interrupt. UDP packet transmitted ok.
irq_tx_length_err	-	out	Single-cycle interrupt. Packet size mismatch, single cycle. The number of payload bytes are different from the advertised size. Ethernet CRC fault is forced.
irq_tx_size_err	-	out	Single-cycle interrupt. Triggered when illegal packet size 0 or larger than 1472 is detected. No packet is transmitted.
irq_tx_mismatch	-	out	Single-cycle interrupt. This interrupt is

		out	triggered if the AXI stream converter detects an out of range or disabled peer. The associated data with that packet is discarded.
irq_tx_uflow	-	out	Single-cycle interrupt. This interrupt is triggered if an AXI stream underflow situation occurs. An underflow is recognized by AXI stream data that cannot meet the Tx GMII bandwidth when UDP data for a packet has not been prefilled.
irq_rx_notvld_done	-	out	Single-cycle interrupt. Indicates completed reception of a good or bad packet of any type except valid UDP packet.
irq_rx_done	-	out	Single-cycle interrupt. UDP packet received ok.
irq_rx_packet_err	-	out	Single-cycle interrupt. Broken packet received, bad CRC.
irq_rx_chksum_err	-	out	Single-cycle interrupt. Bad IP header checksum detected.
irq_rx_match_err	-	out	Single-cycle interrupt. Good packet detected, but no match.
irq_rx_oflow	-	out	Single-cycle interrupt. This interrupt is triggered if an AXI stream overflow situation occurs. An overflow is recognized by AXI stream data not being read at a rate that meets the Rx bandwidth.
cnt_sts_tx_notvld_done		out	The statistics counters increment for each occurrence of the corresponding interrupt.
cnt_sts_tx_done		out	
cnt_sts_tx_length_err		out	
cnt_sts_rx_notvld_done		out	
cnt_sts_rx_done		out	
cnt_sts_rx_chksum_err		out	
cnt_sts_rx_packet_err		out	
cnt_sts_rx_match_err		out	
link_init_done	-	out	Continuous status. 1 = Initialization complete after reset 0 = Initializing, internal reset
link_local_fault	-	out	Continuous status. 1 = Fault detected on RX, link down 0 = Link up (not valid during internal reset)
link_remote_fault	-	out	Continuous status. 1 = Fault detected by peer, link down 0 = Link up (not valid during internal reset)
link_recovering	-	out	Continuous status. 1 = Internal recovery after link fault 0 = Normal operation

Status (xgmii_tx_clk)			
peer_ea_status	-	out	Reflects the current EA/MAC addresses in the ARP table. Not synchronized to clk.
s_axis_tready	-	out	
s_axis_tdata	-	in	
s_axis_tlast	-	in	
s_axis_tkeep	-	in	One bit for each byte in s_axis_tdata.
s_axis_tvalid	-	in	
s_axis_tuser			
AXI stream Rx (clk)			
m_axis_tready	-	in	
m_axis_tdata	-	out	
m_axis_tlast	-	out	
m_axis_tkeep	-	out	One bit for each byte in m_axis_tdata.
m_axis_tvalid	-	out	
m_axis_tuser			.
XGMII Tx			
xgmii_tx_clk	-	in	
xgmii_txc	-	out	SDR: col = [3:0], col = [7:4])
xgmii_txd	-	out	SDR: col = [31:0], col = [63:32])
XGMII Rx			
xgmii_rx_clk	-	in	
xgmii_rxc	-	in	SDR: col = [3:0], col = [7:4])
xgmii_rxd	-	in	SDR: col = [31:0], col = [63:32])

Parameters
There are several compile time parameters, like Number of parallel BitUDP1G instances used, Slave and master port FIFO depth, Cnt_width, Data_width, ARP_table_size, Respect interpacket gap or not, etc*.

6. Performance

Performance of the IP is dependent of the technology used for implementation and the user configuration of the IP.

The table below* is from a typical utilization report for a 10G implementation in a series 7 FPGA from Xilinx, with a Master and Slave fifo depth of 4k.

```
+-----------+------------+------------+---------+------+-------+-----------+-------------+
| Module    | Total LUTs | Logic LUTs | LUTRAMs | SRLs | FFs   | Block RAM | DSP48 blocks |
+-----------+------------+------------+---------+------+-------+-----------+-------------+
| bit10gudp |   33020    |   32654    |  366    |  0   | 23779 |   40,5    |    0        |
+-----------+------------+------------+---------+------+-------+-----------+-------------+
```

*Please contact BitSim for more details.

Document Revision History

Release revision A.