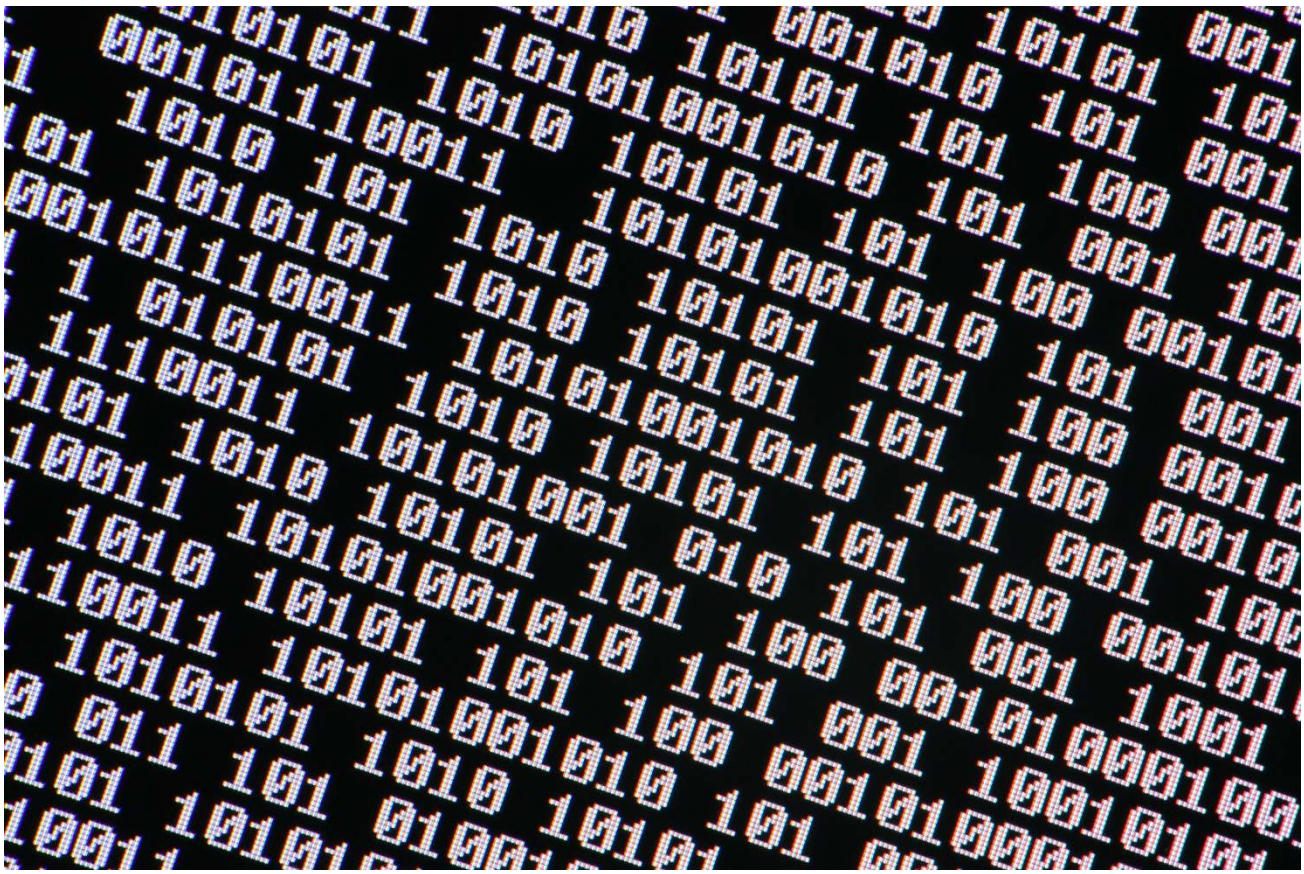
Data Sheet

Bit-1G-UDP Ethernet

10/100 or 1000 Mb/s full HW stack UDP/IP Transmitter/Receiver IP block for high performance data transfers for embedded Ethernet based systems



1.

1. Introduction

Bit-1G-UDP is a configurable design for implementation in FPGA or ASIC* devices providing means to encapsulate data in UDP packets. Any design can be connected to a regular 1Gbps 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-1G-UDP implements the (G)MII interface and can be connected directly to an Ethernet PHY or a PCS component.

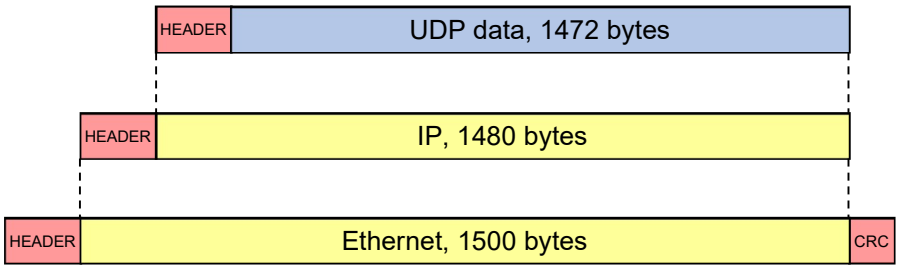


Figure 1, UDP packet encapsulation

Deliveries
- VHDL source code (readable or encrypted) and testbench with stimuli and checkers
- BitSim's Simulation Environment (BSE), a verification framework
- PC SW test code (Windows, Linux)
- 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

2. Features

- Receives and sends UDP-packets over Ethernet
- Supports 10, 100 and 1 000 Mb/s (1 Gb/s) transfer rates
- ARP-table, number of entries can be selected
- Programmable Source/Destination ports and IP/MAC/Default gateway address
- Works with standard Ethernet transceivers (PHYs) using (G)MII-interface (RGMII optional)
- Low latency (CRC on IP-header calculated on-the-fly)
- Payload data is transferred via AXI4-Stream or with a proprietary interface
- Loopback at both AXI-Stream and (G)MII interfaces are possible for debugging purposes
- PCIe header interpretation possible, for tunneling of PCIe
- Statistics counters and protocol filters
- Prepared for SyncE, including ESMC protocol
- Implemented as general VHDL code, independent of FPGA vendor
- Tested with Xilinx Zynq, Virtex-6, Virtex-5 and Marvell Ethernet PHYs 88E1512, M881111 EPHY

3. System Description

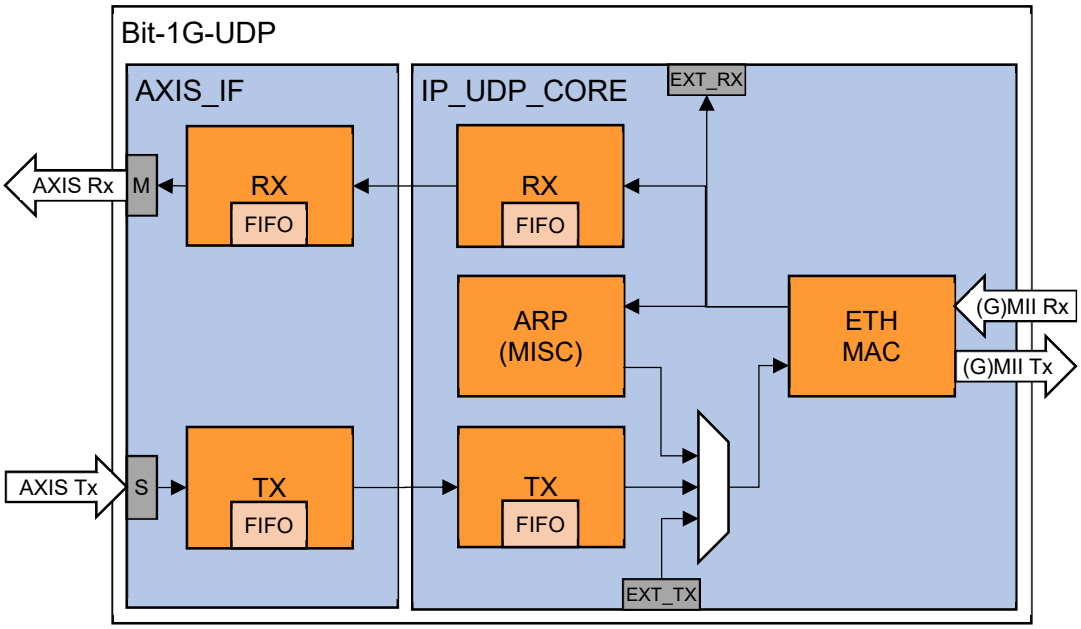


Figure 2 Bit-1G-UDP with AXI4-Stream interface

The design consists of IP_UDP_CORE and AXIS_IF (AXI stream interface). IP_UDP_CORE can be operated standalone without AXIS_IF as a minimal implementation. AXIS_IF also contains synchronization to a single system clock as well as interrupt generation and statistics counters. Data is prepended with a 16-bit field foretelling the payload size (can be enabled/disabled for Rx). This is predominantly aimed to provide low Tx latency. Once the payload size for Tx data is announced, IP_UDP_CORE starts to transmit the ethernet header and calculates the IP header checksum on the fly. The UDP header checksum is ignored for received packets and set to 0 (ignored by receiving end) for transmitted packets. If enabling the payload size field for Rx, AXI stream Rx can be directly connected to AXIS Tx to loop back any data for debug purposes. The (G)MII interface can also be connected in loopback assuming that the local and peer network addressing options are configured to be the same.

The Ethernet MAC core can operate at either 10, 100 or 1000 Mbit/s. For 10/100 Mbit, the upper bits of GMII data are unused, forming a 4-bit MII interface. Collision detection and retransmit (CSMA/CD) for 10/100 Mbit is not supported, it is presumed that a switch is used rather than a hub. IP_UDP accepts pause frames and stalls Tx data for the requested pause period.

There is also an option to transmit and receive raw Ethernet packets, EXT_TX and EXT_RX with the preamble, SFD and CRC being handled by the Ethernet MAC.

4.

4. IP Configuration

When the user integrates the **Bit-1G-UDP** it is possible to tune the IP for the actual needs. Some configuration is static and performed at compile time by setting VHDL generic values. Dynamic configuration is done by updating register values at run-time. The dynamic configuration is performed with signals described in chapter 5. The static configuration parameters are described in Table 1. Some parameters are only valid for the AXIS-IF block. See the user guide for details.

Parameter	Description
data_width_g #1	Payload data width in bits. Valid values are 32, 64, 128, … (2^n, n > 4)
fifo_data_pad_width_g #1	Use if inferred RAM for master or slave FIFO must be of a specific width, otherwise set to 0
s_fifo_addr_width_g #1	Slave port FIFO depth. Total number of bytes = (data_width_g / 8) · $2^{s\_fifo\_addr\_width\_g}$
m_fifo_addr_width_g #1	Master port FIFO depth. Total number of bytes = (data_width_g / 8) · $2^{m\_fifo\_addr\_width\_g}$
user_enable_g #1	true: Enable AXI stream TUSER bits false: Tie off, ignore
cnt_width_g #1	Number of bits used for each statistics counter
irq_sts_sync_to_clk_g #1	true: Synchronize interrupt and status signals to the system clock false: The interrupt and status signals are synchronous to the Ethernet transmit clock
mac_ipg_enable_g	true: Respect interpacket gap (IEEE 802.3). false: Ignore interpacket gap.
mac_ipg_reduction_enable_g	true: Apply reduced interpacket gap false: Apply full interpacket gap (IEEE 802.3)
mac_sync_to_tx_clk_enable_g	true: The Ethernet MAC synchronizes RX data to the TX domain false: Must only be selected if RX and TX clocks are truly synchronous
arp_table_size_g	Number of entries in the ARP table
arp_table_idx_width_g	Number of bits required to select arp_table_size_g indexes
arp_reply_timeout_us_g	ARP request retransmit rate for missing ARP replies. Must be set to 1000000 µs (1 second)
arp_reply_accept_all_en_g	If true, all incoming ARP replies are accepted. Otherwise, the ARP replies are only accepted if the IP address matches a sent ARP request.
Notes	
#1	This parameter only applies to the AXI stream component.

Table 1 Parameters for static configuration

5. Ports

The port list is valid for a configuration with the AXIS-IF included. See the user guide for a detailed description of the ports.

Signal	Dir	Width	Description
System			
reset_n	in	1	Global reset
reset_core_n	in	1	Core reset
clk	in	1	System clock. Used only in AXIS_IF
Ctrl (clk)			
cfg_pcie_fmt_en	in	1	'1' = PCIe header format enabled for Tx. '0' = Tx uses the size field
cfg_tx_prefill_level	in	2	AXI stream Tx FIFO size. 0 <= n <= 3. 2^n AXI words for each packet are buffered before UDP Tx is started towards IP_UDP_CORE
cfg_tx_prefill_packet	in	1	'1' = Complete packet buffered before initiating transmission towards IP_UDP_CORE '0' = Initiate transmission towards IP_UDP_CORE as soon as there is AXI stream data available
clear_packet_cnt	in	1	'1' = Clear statistics counters '0' = Normal operation
Interrupts and status (clk or gmii_tx_clk)			
irq_sts_arp_done	out	n	ARP done
irq_sts_arp_sniff_done	out	1	ARP address update
irq_sts_pause	out	1	Pause frame
irq_tx_done	out	1	Good UDP packet sent
irq_tx_notvld_done	out	1	Not valid UDP packet
irq_tx_size_err	out	1	Illegal packet size
irq_tx_length_err	out	1	Length error
irq_tx_mismatch	out	1	Out of range or disabled peer
irq_tx_uflow	out	1	AXI stream underflow
irq_rx_done	out	1	Good UDP packet received
irq_rx_notvld_done	out	1	Not valid UDP packet
irq_rx_packet_err	out	1	Broken packet, bad CRC
irq_rx_chksum_err	out	1	IP header checksum error
irq_rx_match_err	out	1	ARP error
irq_rx_oflow	out	1	AXI stream overflow
cnt_sts_tx_notvld_done	out	cnt_w	Statistics counters incremented for each occurrence of the corresponding interrupt
cnt_sts_tx_done	out	cnt_w	
cnt_sts_tx_length_err	out	cnt_w	
cnt_sts_rx_notvld_done	out	cnt_w	
cnt_sts_rx_done	out	cnt_w	
cnt_sts_rx_chksum_err	out	cnt_w	
cnt_sts_rx_packet_err	out	cnt_w	
cnt_sts_rx_match_err	out	cnt_w	
AXI stream Tx (clk)			
s_axis_tready	out	1	
s_axis_tdata	in		SeeTable 1, data_width_g
s_axis_tlast	in	1	

Signal	Dir	Width	Description
s_axis_tkeep	in		One bit for each byte in s_axis_tdata
s_axis_tvalid	in	1	
s_axis_tuser	in		SeeTable 1, arp_table_idx_width_g
AXI stream Rx (clk)			
m_axis_tready	in	1	
m_axis_tdata	out		SeeTable 1, data_width_g
m_axis_tlast	out	1	
m_axis_tkeep	out		One bit for each byte in m_axis_tdata
m_axis_tvalid	out	1	
m_axis_tuser	out		SeeTable 1, arp_table_idx_width_g
Cfg local (gmii_tx_clk)			
cfg_local_ea	in	48	Local Ethernet/MAC address
cfg_local_ip	in	32	Local IP address
cfg_local_udp_port	in	n*16	Local UDP listen port
Cfg peer (gmii_tx_clk)			
cfg_peer_ip	in	n*32	Peer IP address
cfg_peer_udp_port	in	n*16	Peer UDP listen port
cfg_peer_en	in	n	'1' = ARP table entry, or index, is enabled '0' = Communication with peer <index> is disabled
Cfg network (gmii_tx_clk)			
cfg_gw_ip	in	32	Gateway IP address
cfg_submask	in	32	Mask for local subnet
Cfg ARP bypass (gmii_tx_clk)			
cfg_peer_ea	in	n*48	Used when fixed EA/MAC address is enabled
cfg_peer_ea_fixed_en	in	n	'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	48	Used when fixed gateway EA/MAC address is enabled
cfg_peer_gw_ea_fixed_en	in	n	'1' = Enable fixed EA/MAC address for the gateway '0' = Use ARP to resolve EA/MAC address for the gateway
Cfg legacy (gmii_tx_clk)			
cfg_arp_rx_sniff_en	in	1	'1' = Use the IP address from the most recent incoming UDP/ARP request during Tx and allow any IP during Rx '0' = Use ARP table entry <index>
cfg_arp_table_use_req_en	in	1	'1' = ARP table is also updated by incoming requests '0' = ARP table is only updated by incoming replies
Cfg ctrl (gmii_tx_clk)			
cfg_link_speed_sel	in	2	'11' = Reserved '10' = GMII 1 Gbit/s (125 MHz) '01' = MII 100 Mbit/s (25 MHz) '00' = MII 100 Mbit/s (2,5 MHz)

Signal	Dir	Width	Description
cfg_validate_rx_packet_en	in	1	'1' = Complete Rx packet is validated before UDP Rx data is driven '0' = UDP Rx data is driven immediately upon reception
cfg_rx_size_hdr_en	in	1	'1' = Prepend size header to UDP Rx data. This makes the Rx format identical to Tx '0' = Only drive UDP Rx data
cfg_tx_ttl	in	8	Time to live.
Ctrl (gmii_tx_clk)			
trig_arp_request	in	n	'1 '= Populate ARP table entry for peer <index>. Sends ARP requests and retrieves the EA/MAC addresses for all enabled peers. '0' = Populate ARP entry for <index> when Tx wants to send data
flush_arp_table	in	1	'1' = Scrap any stored EAs, soft reset '0' = Normal operation
force_pause	in	1	'1' = Pause UDP transmission '0' = Normal operation
Status (gmii_tx_clk)			
peer_ea_status	out	n*48	Reflects the current EA/MAC addresses in the ARP table
External Tx arbitration (gmii_tx_clk)			
ext_tx_req	in	1	'1' = Request arbitration to transmit external Tx data '0' = Not requesting arbitration for external Tx data
ext_tx_gnt	out	1	'1' = Arbitration granted to transmit external Tx data '0' = Not granted
ext_tx_act	in	1	'1' = Active transmission '0' = Transmission ended
GMII Tx			
gmii_tx_clk	in	1	Ethernet transmit clock
gmii_txd	out	8	Data
gmii_tx_en	out	1	Enable
gmii_tx_er	out	1	Error
GMII Rx			
gmii_rx_clk	in	1	Ethernet receive clock
gmii_rxd	in	8	Data
gmii_rx_dv	in	1	Data valid
gmii_rx_er	in	1	Error
External Tx, raw Ethernet payload (gmii_tx_clk)			
ext_tx_data	in	8	This interface is used for transmission of raw Ethernet packets. See the user guide for details.
ext_tx_data_en	in	1	
ext_tx_data_er	in	1	
ext_tx_ready	out	1	
ext_tx_done	out	1	
External Rx, raw Ethernet payload (gmii_tx_clk)			
ext_rx_data	out	8	
ext_rx_data_en	out	1	

Signal	Dir	Width	Description
ext_rx_data_er	out	1	This interface is used for reception of raw Ethernet packets. See the user guide for details.
ext_rx_done	out	1	
ext_rx_sop	out	1	

Table 2 Bit-1G-UDP entity ports

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 1G implementation in a series 7 FPGA from Xilinx, with a Master and Slave FIFO depth of 4k. This example is using a 64-bit AXI-Stream interface.

Module	Total LUTs	Logic LUTs	LUTRAMs	SRLs	FFs	RAMB36	RAMB18	DSP48 blocks
ip_udp	2799	2791	8	0	1291	3	3	0

*Please contact BitSim for more details.

Document Revision History
Release revision A.