UDP10G-IP Core

March 8, 2019 | Product Specification | Rev1.1

Design Gateway Co.,Ltd

54 BB Building 14th Fl., Room No.1402 Sukhumvit
21 Rd. (Asoke), Klongtoey-Nua, Wattana,
Bangkok 10110
Phone: 66(0)2-261-2277
Fax: 66(0)2-261-2290
E-mail: ip-sales@design-gateway.com
URL: www.design-gateway.com

Core Facts

Provided with Core	
Documentation	User Guide, Design Guide
Design File Formats	Encrypted HDL
Constraints Files	User constraint file
Verification	Simulation model
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on KCU105, ZCU102
Simulation Tool Used	
Vivado Simulator	
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- UDP/IP stack implementation
- Support IPv4 protocol
- Support one session per one UDP10G-IP (Multisession can be implemented by using multiple UDP10G-IP)
- Transmit data bus size is 64-bit, so transmit packet size must be aligned 64-bit
- Received data bus size is 64-bit, so total received size must be aligned 64-bit
- Transmit/Receive buffer size, adjustable for optimized resource and performance
- Simple data interface by standard FIFO interface
- Simple control interface by standard register interface
- 64-bit AXI4 stream to interface with 10-Gbps Ethernet MAC from Xilinx
- One clock domain interface by fixed 156.25 MHz clock frequency
- Reference designs available on KCU105, ZCU102 evaluation board
- Support IP fragmentation

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	Slice Regs	Slice LUTs	Slices[1]	IOB[2]	BRAMTile [1]	Design Tools
Kintex-7	XC7K325TFFG900-2	156.25	1883	2170	782	-	36	Vivado2017.4
Virtex-7	XC7VX485TFFG1761-2	156.25	1883	2170	796	-	36	Vivado2017.4
Zynq-7000	XC7Z045FFG900-2	156.25	1883	2177	798	-	36	Vivado2017.4

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Block memory resources are based on 64k Tx data buffer size, 16k Tx packet buffer size, and 64k Rx data buffer size.

Table 2: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB	IOB[2]	BRAMTile [1]	Design Tools
Kintex-Ultrascale	XCKU040FFVA1156-2E	156.25	1871	2223	433	-	34.5	Vivado2017.4
Zynq-Ultrascale+	XCZU9EG-FFVB1156-2	156.25	1871	2220	433	-	34.5	Vivado2017.4

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Block memory resources are based on 64k Tx data buffer size, 16k Tx packet buffer size, and 64k Rx data buffer size.

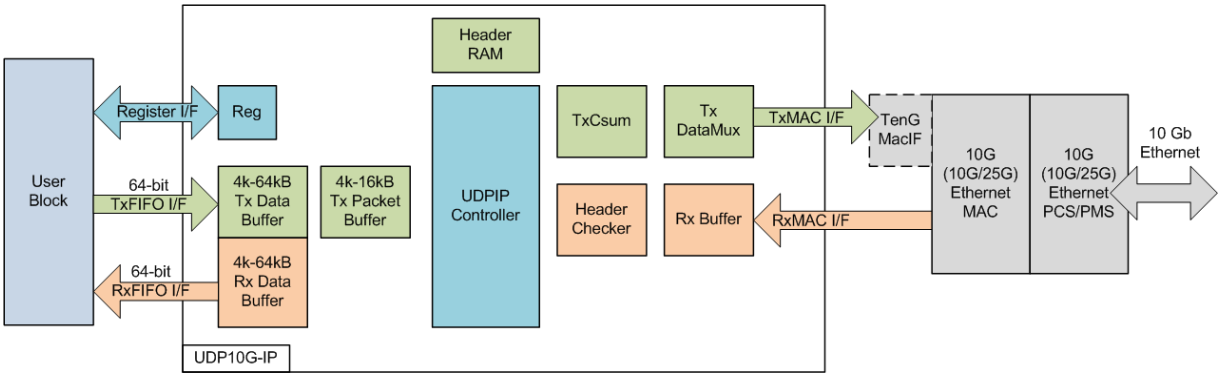

Figure 1: UDP10G-IP Block Diagram

Applications

UDP10G-IP is designed for network application such as video data streaming by using UDP/IP protocol to transfer high speed data through 10 Gb Ethernet. By using this IP, user can easily transfer data with some devices through UDP/IP protocol without CPU and external meomory in the system.

General Description

UDP10G-IP core with Xilinx 10G (10G/25G) EMAC IP and 10G (10G/25G) Ethernet PCS/PMA is implemented as UDP/IP stack, Transport layer, Internet layer, Link layer, and Physical layer for network data transmission. User sends and receives 10 Gb Ethernet data with some network devices through UDP/IP protocol by using this system.

There are three types of user interface, i.e. control signal by register access, transmit and received data signal by FIFO access. To initialize system, user needs to set up system parameters such as MAC address, port number, IP address through register interface. After finishing initialization, UDP10G-IP is ready to receive command from user.

After user sends command, data is transferred from user logic through TxFIFO interface. Transmit data from user is stored within Tx Data buffer before forwarding to Ethernet MAC. For the receive direction, if the header of UDP packet is valid, UDP10G-IP will extract only UDP data to store in Rx Data buffer. User reads received data through RxFIFO interface.

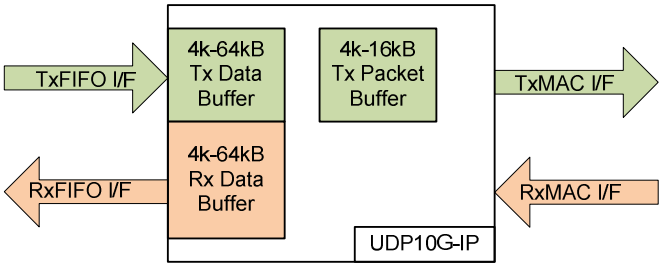

Figure 2: Adjustable Tx/Rx Buffer Size

User sets the size of three buffers in UDP1G-IP (Tx Data buffer, Tx Packet buffer, and Rx Data buffer) by defining as constant value to UDP10G-IP HDL code. The different size is provided to optimize resource utilization for user application. The recommended value of Tx Data buffer size and Tx Packet buffer size depends on Tx packet size (PKL register from user). Tx Packet Buffer must be more than the Tx packet size while Tx Data buffer size should be at least two times of the Tx packet size.

For Rx data buffer, the size should be at least two times of received packet size for storing new receive packet and sending out data to user at the same time.

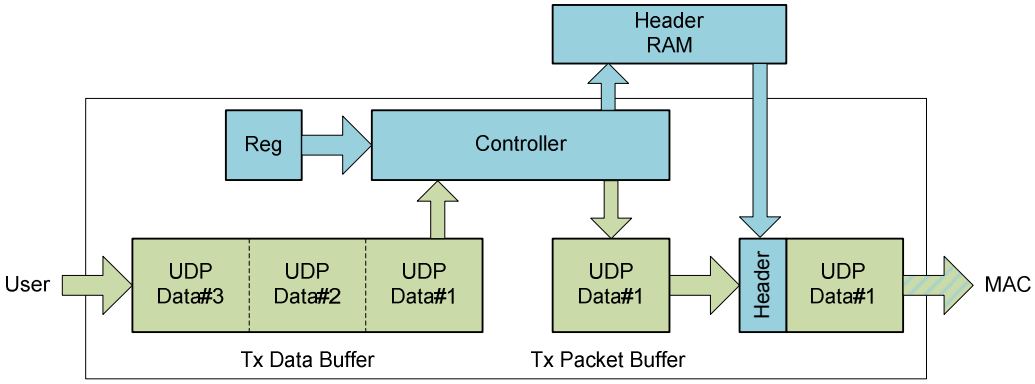

Figure 3: Transmit Data Flow

To transmit data, data from Tx Data buffer is split into packet size to store in Tx Packet buffer. Data output from Tx Packet buffer is combined with header data in Header RAM before sending out to EMAC. UDP and IP checksum are auto calculated within UDP10G-IP. Busy flag of UDP10G-IP is de-asserted to '0' after sending all data. User sets total size of transmit data and packet size through register interface.

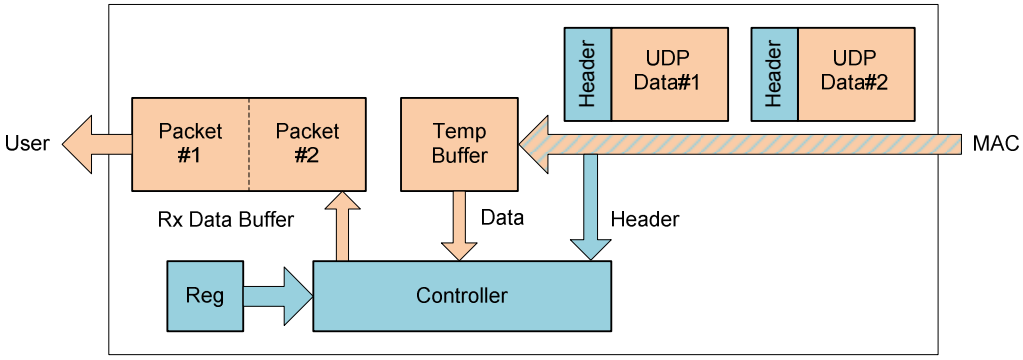

Figure 4: Received Data Flow

When new packet is received from EMAC, received packet is stored to temp buffer firstly. Header and checksum of Rx packet are verified. If network parameters in the header do not match to set value from user or checksum of received packet is error, Rx packet will be ignored. Data of ignored packet from Temp Buffer does not store to Rx Data buffer. Only data of valid packet is extracted and stored to Rx Data buffer.

Functional Description

UDP10G-IP core can be divided into three parts, i.e. control block, transmit block, and received block.

Control Block

- **Reg**

User can set parameters for UDP/IP operation by using register interface. Register address of this interface is equal to 4-bit. The description of each register address is defined as shown in Table 3. After system reset is released, all internal parameters are updated by new value.

- **UDPIP Controller**

After IP reset is changed from '1' to '0', UDP10G-IP starts initialization process which can be selected in two modes (set by user through register interface), i.e. server mode and client mode. In client mode, UDP10G-IP sends ARP request to extract Target MAC address from ARP reply. In server mode, UDP10G-IP waits ARP request from the target to get MAC address, and then returns ARP reply to complete intialization process.

After complete initialization process, UDP10G-IP is in Idle status to wait new command (Send data command) from user and ready to extract data from received packet (Receive direction).

Table 3: Register map Definition

RegAddr [3:0]	Reg Name	Dir	Bit	Description
0000b	RST	Wr /Rd	[0]	Reset IP. '0': Release reset, '1': Reset. Default value is '1'. **After all parameters are assigned, user sets '0' to this register to load parameters and start system initialization. Reset needs to be set/clear again to reload parameter if user changes the value of SML, SMH, DIP, SIP, DPN, or SPN register.**
0001b	CMD	Wr	[0]	Set '1' to start data sending out. **Before setting this register, user needs to check system busy flag to confirm that IP does not run any operations.**
		Rd	[0]	IP busy flag. '0': Idle, '1': IP is in initialization or data transmission. Similar to Busy output signal.
0010b	SML	Wr /Rd	[31:0]	Define 32-bit lower MAC address (bit [31:0]) for this IP. **User needs to set this register before clearing RST register.**
0011b	SMH	Wr /Rd	[15:0]	Define 16-bit upper MAC address (bit [47:32]) for this IP. **User needs to set this register before clearing RST register.**
0100b	DIP	Wr /Rd	[31:0]	Define 32-bit target IP address. **User needs to set this register before clearing RST register.**
0101b	SIP	Wr /Rd	[31:0]	Define 32-bit IP address for this IP. **User needs to set this register before clearing RST register.**
0110b	DPN	Wr /Rd	[31:0]	[15:0]-Define 16-bit target port number for IP sending data. [31:16]-Define 16-bit target port number for IP receiving data. **User needs to set this register before clearing RST register.**
0111b	SPN	Wr /Rd	[15:0]	Define 16-bit port number for this IP. **User needs to set this register before clearing RST register.**

RegAddr [3:0]	Reg Name	Dir	Bit	Description
1000b	TDL	Wr	[31:0]	Total Tx data length transfer in byte unit, but the size must be aligned to 64-bit or 8-byte. Valid from 8-0xFFFFFFF8 (bit[2:0] is ignored). **User needs to set this register before setting CMD register='1'. This value are loaded when CMD register is set. User can prepare the new value for next transmit after UDP10G-IP starts sending operation.** If the next data transmission uses the same length, this register will not need to set again. UDP10G-IP uses the lastest value for the next data transmission.
		Rd		Remaining data transfer length in byte unit which still not transmit.
1001b	TMO	Wr	[31:0]	Define timeout value for waiting ARP reply packet after sending ARP request. The counter runs by using 156.25 MHz, so timer unit is about 6.4 ns. This value should be more than 0x6000.
		Rd		[0]-Timeout from not receiving ARP reply packet After timeout, IP resends ARP request until ARP reply is received. [8]-Rx packet ignored because of Rx data buffer full [9]-Rx packet ignored because of checksum failed [10]-Rx packet ignored because of MacRxUser error
1010b	PKL	Wr /Rd	[15:0]	Data size of Tx packet in byte unit, but packet length must be aligned to 64-bit. Valid from 8-16000. Default value is 1472 byte (Maximum size for non-jumbo frame). Bit[2:0] of this register is ignored to align 64-bit unit. **This value must not be changed when data transmission still not complete (Busy='1'). If the next data transmission uses same packet size, user will not need to set this register. The IP loads the previous value from latch register.**
1110b	SRV	Wr/ Rd	[0]	'0': Client mode (default mode). After IP reset is released to '0', the IP sends ARP request to get Target MAC address from IP address. IP busy is deasserted to '0' after IP receives ARP reply. '1': Server mode. After IP reset is released to '0', the IP waits ARP request to get Target MAC address from IP address. IP busy is deasserted to '0' after IP returns ARP reply. **Note: In Server mode, after UDP10G-IP is reset, the Target needs to resend ARP request to UDP10G-IP to complete IP initialization.**

Table 4 TxBuf/TxPac/RxBufBitWidth Parameter description

Value of BitWidth	Buffer Size	TxBufBitWidth	TxPacBitWidth	RxBufBitWidth
9	4kByte	Valid	Valid	Valid
10	8kByte	Valid	Valid	Valid
11	16kByte	Valid	Valid	Valid
12	32kByte	Valid	No	Valid
13	64kByte	Valid	No	Valid

Transmit Block

- **Tx Data Buffer**

This buffer size is set by "TxBufBitWidth" parameter of the IP. The valid value is 9-13 which is equal to the address size of 64-bit buffer, as shown in Table 4.

The buffer size should be at least two times of Tx Packet Size in PKL register. During sending operation, one packet data is forwarded from this buffer to Tx Packet buffer. At the same time, the next packet is received from user logic. Data in Tx Data buffer is flushed after the data is forwarded to EMAC. If Tx Data buffer size is much enough (more than two times of Tx Packet size), current packet will be transmitted to EMAC continuously. This buffer is also used to be data buffer between user logic and UDP10G-IP for sending data operation.

- **Tx Packet Buffer**

The size is set by "TxPacBitWidth" parameter of the IP. The valid value is 9-11 and the description of the parameter is shown in Table 4. This buffer size must be more than Tx Packet size + 32. For example, when TxPacBitWidth=9, maximum value for PKL is 4064 (4096-32). Tx Packet buffer could store data at most two packets. There is no advantage to use Tx Packet buffer size which is more than two times of maximum value to set to PKL register.

- **Header RAM**

This RAM is applied to store header part of Transmit packet. The header part consists of network parameters which are set by user through register interface and packet checksum which is calculated by TxCsum block.

- **TxCsum**

This module is designed to calculate checksum of Tx packet before sending out. After complete to calculate checksim, the checksum output is loaded to Header RAM to be the header of Tx packet.

- **TxDataMux**

This module is designed to merge header from Header RAM to data from Tx Packet Buffer. The ethernet packet including header and data is forwarded to EMAC.

Received Block

- **Rx Buffer**

This is temporary buffer to store Rx packets from EMAC. The objective of this buffer is to wait Header Checker to validate the received packet before forwarding to Rx Data buffer. Only UDP data of valid packet is extracted and forwarded to Rx Data buffer.

- **Header Checker**

Header in Rx packet consists of network parameters and checksum value. If some network parameters is not matched to the set value in register or checksum is not correct, Rx packet will be rejected. UDP checksum is not verified when received packet is IP fragmented packet.

- **Rx Data Buffer**

This buffer size is set by “RxBufBitWidth” parameter of the IP. The valid value is 9-13 which more details are shown in Table 4. This is the data buffer between user logic and UDP1G-IP for receiving data operation. If buffer is full, new received packet will be ignored. It is recommended to set Rx data buffer size to be more than or equal to two times of received packet size.

User Block

This block is user module for setting command and monitoring status signal through register interface, writing data to Tx FIFO, and reading data from Rx FIFO. This module can be designed by simple hardware logic.

TenGMacIF

This block is designed to connect tx interface between UDP10G-IP and Xilinx 10G (10G/25G) Ethernet MAC. tx_axis_tready of Xilinx Ethernet MAC can be de-asserted to ‘0’ during packet transmission. But tx interface of UDP10G-IP needs to send data of each packet continuously. So, this block is included in the reference design as HDL code to store data output from UDP10G-IP during 10G (10G/25G) Ethernet MAC not ready to receive the data.

10 Gb Ethernet MAC and 10 Gb Ethernet PCS/PMA

Both blocks are softIPcore provided by Xilinx. Please read more details from following website.

10G Ethernet MACN and PCS/PMA
https://www.xilinx.com/products/intellectual-property/do-di-10gemac.html
https://www.xilinx.com/products/intellectual-property/10gbase-r.html

10G/25G Ethernet Subsystem
https://www.xilinx.com/products/intellectual-property/ef-di-25gemac.html

Core I/O Signals

Descriptions of all parameters and signal I/O are provided in Table 5 and Table 6. MAC Interface of the IP is 64-bit AXI4 stream interface.

Table 5: Core Parameters

Name	Value	Description
TxBufBitWidth	9-13	Setting Tx Data buffer size. The value is referred to address bus size of this buffer.
TxPacBitWidth	9-11	Setting Tx Packet buffer size. The value is referred to address bus size of this buffer.
RxBufBitWidth	9-13	Setting Rx Data buffer size. The value is referred to address bus size of this buffer.

Table 6: Core I/O Signals

Signal	Dir	Description
Common Interface Signal		
RstB	In	Reset IP core. Active Low.
Clk	In	156.25 MHz fixed clock frequency input from PHY layer of Xilinx block.
User Interface		
RegAddr[3:0]	In	Register address bus
RegWrData[31:0]	In	Register Write data bus. Synchronous to RegAddr signal for write process.
RegWrEn	In	Register Write enable pulse. Assert with valid value of RegAddr and RegWrData signals.
RegRdData[31:0]	Out	Register Read data bus. Available after asserting RegAddr with 1 Clk period latency
Busy	Out	IP busy status (‘0’-Idle, ‘1’-IP Intialization or IP sending data)
IntOut	Out	Assert to high for 1 Clock cycle when timeout is detected or Rx packet is ignored. More details of Interrupt status could be checked from TMO register.
Tx Data Buffer Interface		
UDPTxFfFull	Out	Transmit buffer full flag. User needs to stop writing data within 4 clock period after this flag is asserted to high.
UDPTxFfWrEn	In	Transmit buffer write enable. Assert to ‘1’ to write data to Transmit buffer.
UDPTxFfWrData[63:0]	In	Transmit buffer write data bus. Synchronous with UDPTxFfWrEn.
Rx Data Buffer Interface		
UDPRxFfRdCnt[12:0]	Out	Received buffer data counter to show total received data (in 64-bit unit) in buffer.
UDPRxFfLastRdCnt[2:0]	Out	Received byte of the last data in received buffer. Valid from 0-7. If total data is not aligned to 8, this signal will not equal to ‘0’.
UDPRxFfRdEmpty	Out	Received buffer empty flag. User needs to stop reading data immediately.
UDPRxFfRdEn	In	Received buffer read enable. Assert to ‘1’ to read data from Received buffer.
UDPRxFfRdData[63:0]	Out	Received buffer read data bus. Valid in the next clock after UDPRxFfRdEn asserts to ‘1’.

Signal	Dir	Description
MAC Interface		
rx_axis_tdata[63:0]	In	Received data bus.
rx_axis_tvalid	In	Received data valid signal. Synchronous with rx_axis_tdata. This signal must be asserted to '1' continuously during start and end of received packet.
rx_axis_tlast	In	Control signal to indicate the final word in the frame.
rx_axis_tuser	In	Control signal asserted at the end of received frame to indicate that the frame has an error. '1': normal packet, '0': error packet.
rx_axis_tready	Out	Asserted to '0' after receive the last data in the frame (rx_axis_tlast='1'). This signal is de-asserted to '1' for 1 clock cycle to pause received data of the next packet.
tx_axis_tdata[63:0]	Out	Transmitted data.
tx_axis_tkeep[7:0]	Out	Tranmistted data byte enable. Synchronous with tx_axis_tdata.
tx_axis_tvalid	Out	Transmitted data valid signal to EMAC. Synchronous with tx_axis_tdata.
tx_axis_tlast	Out	Control signal to indicate the final word in the frame.
tx_axis_tuser	Out	Control signal to indicate an error condition. This signal is always '0'.
tx_axis_tready	In	Handshaking signal. Asserted when tx_axis_tdata has been accepted. This signal must be asserted to '1' continuously during start and end of transmitted packet. Since Xilinx EMAC deasserts this signal to '0' during packet transferring, the adapter logic with small buffer must be applied to connect between UDP10G-IP and Xilinx EMAC. The adapter logic is provided as HDL code in UDP10G-IP reference design.

Timing Diagram

IP Initialization

For initialization process after user releases RST register, UDP10G-IP can operate in two modes depending on SRV register setting, i.e. Client mode and Server mode.

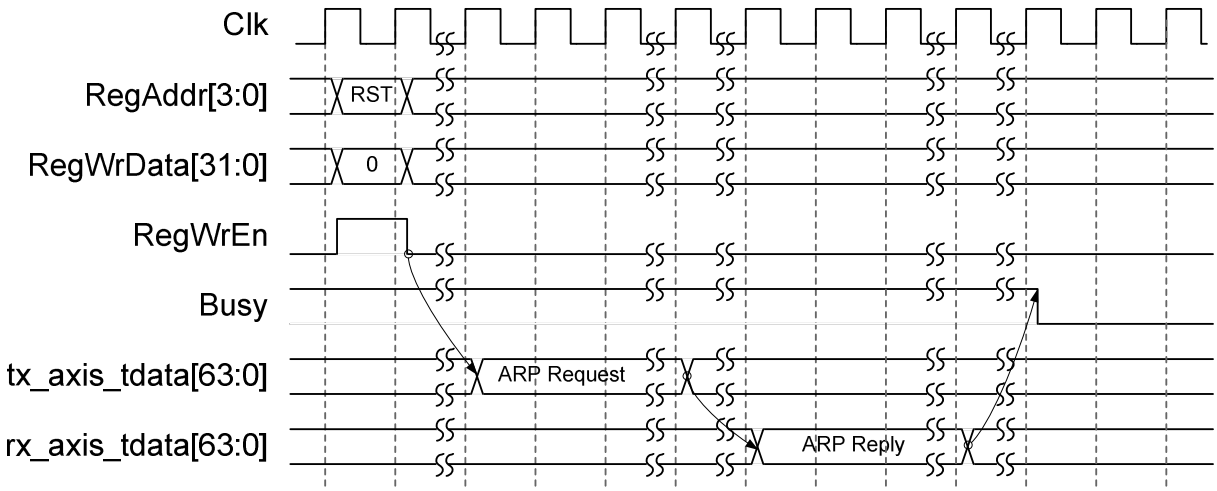

Figure 5: IP Initialization in Client mode

In Client mode, UDP10G-IP sends ARP request and waits ARP reply from the target. Target MAC address is extracted from ARP reply packet. After that, Busy signal is de-asserted to '0'.

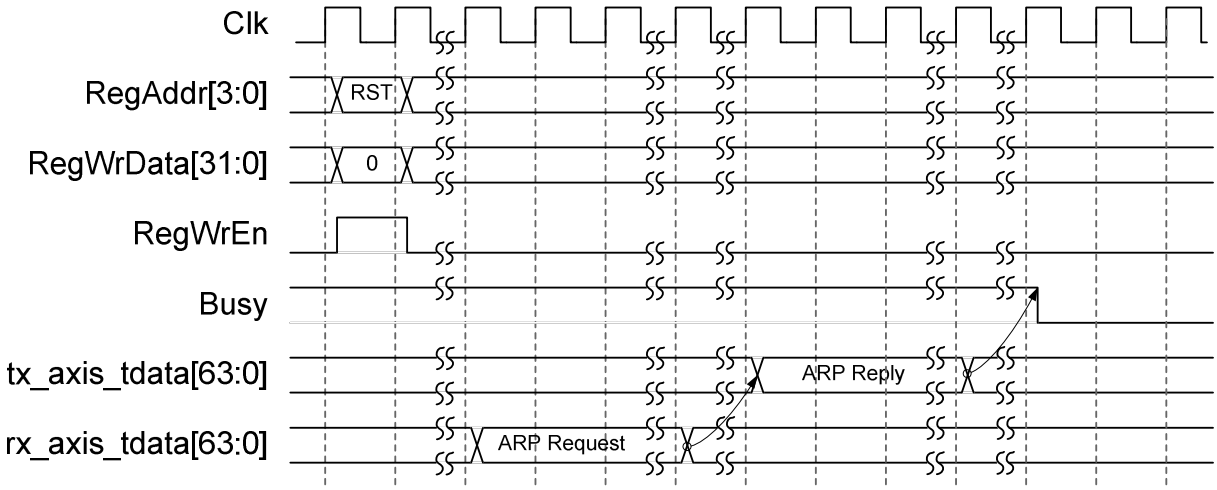

Figure 6: IP Initialization in Server mode

In Server mode, after UDP10G-IP reset is released to '0', UDP10G -IP waits ARP request from the target. After receiving ARP request which the header is matched to set value, UDP10G-IP returns ARP reply to the target. Target MAC address is extracted from ARP request packet. Final stage, Busy signal is de-asserted to '0'.

Register Interface

User writes or reads control signals of UDP10G-IP trhoug Register interface. Timing diagram of register interface is shown in Figure 7. Register map address is designed as shown in Table 3. To write register, user sets RegWrEn='1' with valid value of RegAddr and RegWrData. To read register, user sets RegAddr value and then RegRdData is valid in the next clock.

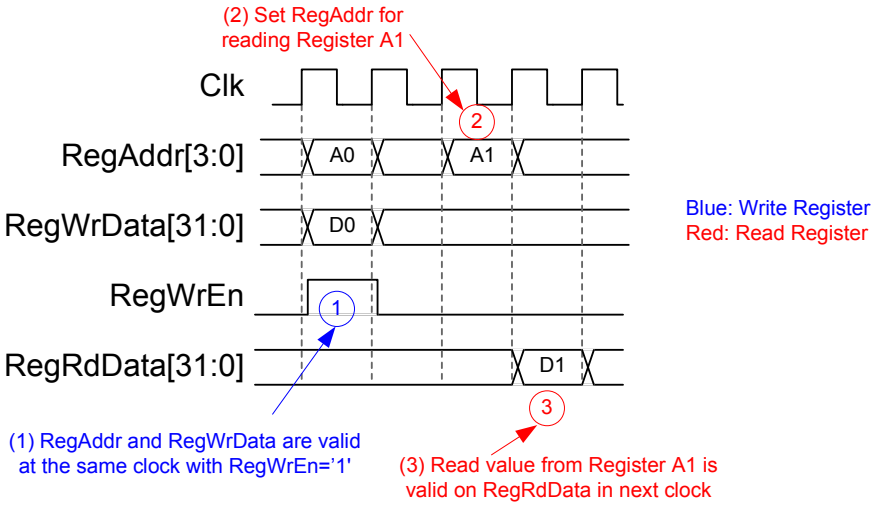

Figure 7: Register Interface Timing Diagram

Before starting sending data operation, busy flag must be monitored that it is equal to '0' (IP is in Idle status). After CMD register is set, busy flag is asserted to '1', as shown in Figure 8. Busy is de-asserted to '0' when send data command is completed.

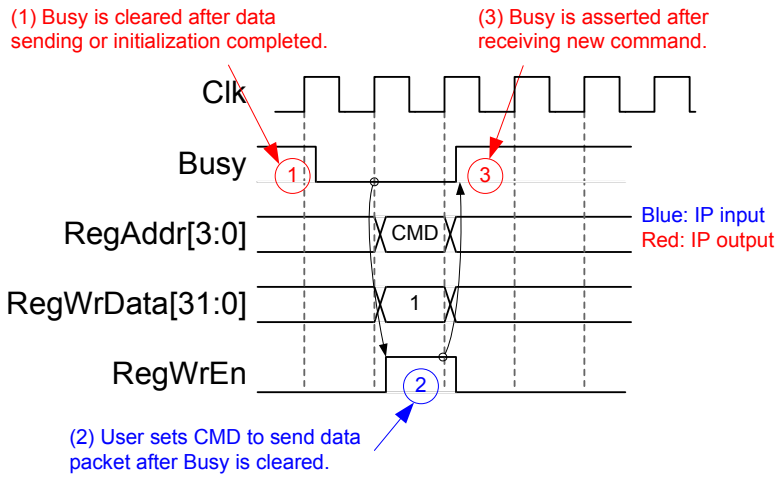

Figure 8: Set CMD register when Busy is de-asserted

Tx FIFO Interface

User sends data to UDP10G-IP by using FIFO interface, as shown in Figure 9. Before sending data, user needs to check full flag (UDPTxFfFull) that is not asserted to '1'. Then, set UDPTxFfWrEn='1' with valid value of UDPTxFfWrData. UDPTxFfWrEn must be de-asserted to '0 within 4 clock cycles to stop data sending after UDPTxFfFull is asserted to '1'. During IP is in reset condition, UDPTxFfFull is asserted to '1' and all data in FIFO are flushed.

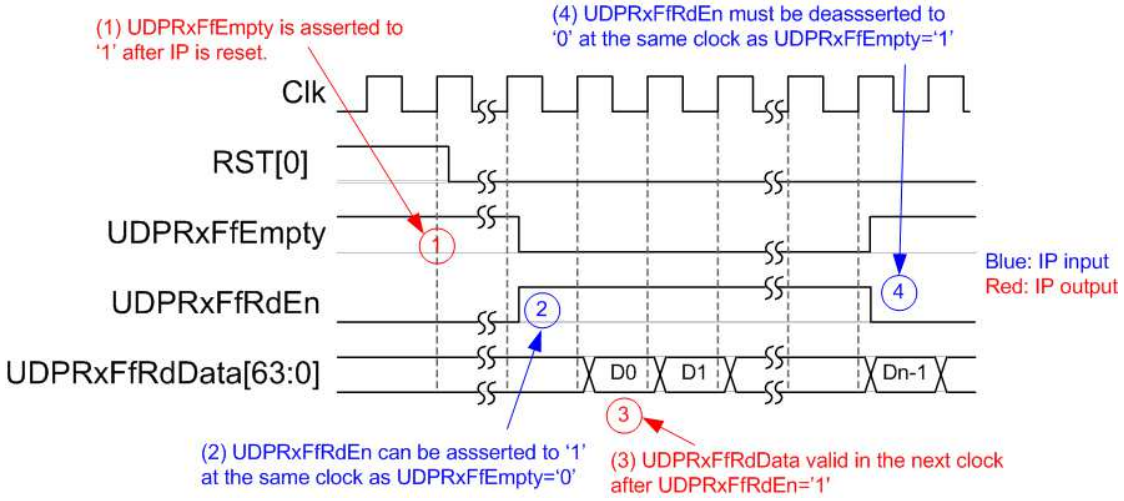

Figure 9: Tx Data Buffer Interface Timing Diagram

Rx FIFO Interface

After the received data extracted from valid received packet is stored in Rx Data buffer, UDPRxFfEmpty is de-asserted to '0'. When user logic detects that new data is received by monitoring UDPRxFfEmpty signal, UDPRxFfRdEn could be asserted to '1' to read data through RxFIFO interface, as shown in Figure 10. UDPRxFfRdData is valid in the next clock. When UDPRxFfEmpty = '1', UDPRxFfRdEn must be de-asserted to '0' at the same clock to stop data reading process. All data in Rx data buffer are flushed when IP is reset. UDPRxFfEmpty is asserted to '1' during reset condition.

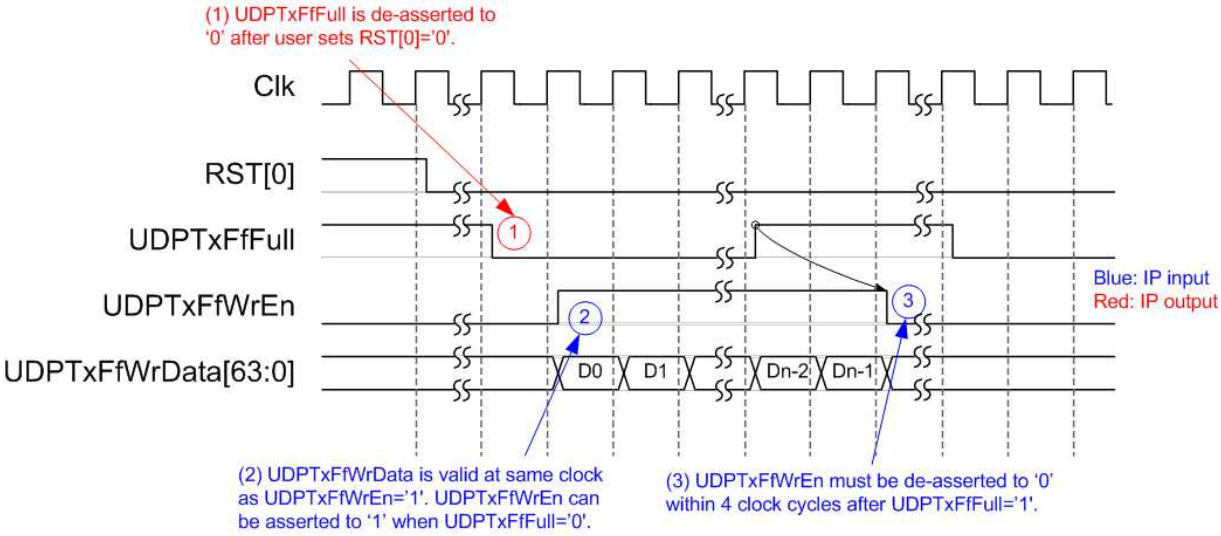

Figure 10: Rx Data Buffer Interface by Empty flag Timing Diagram

In addition, Rx data buffer status can be monitored by using UDPRxFfRdCnt. RdCnt is used when the logic to read data is designed as burst transfer. RdCnt shows total data in Rx data buffer. So, user asserts UDPRxFfRdEn='1' for many clocks to read more than one data from the buffer, as shown in Figure 11. If UDPRxFfLastRdCnt is not equal to 0, it means that current total size is not aligned to 64-bit. The unaligned received size is shown in UDPRxFfLastRdCnt signal.

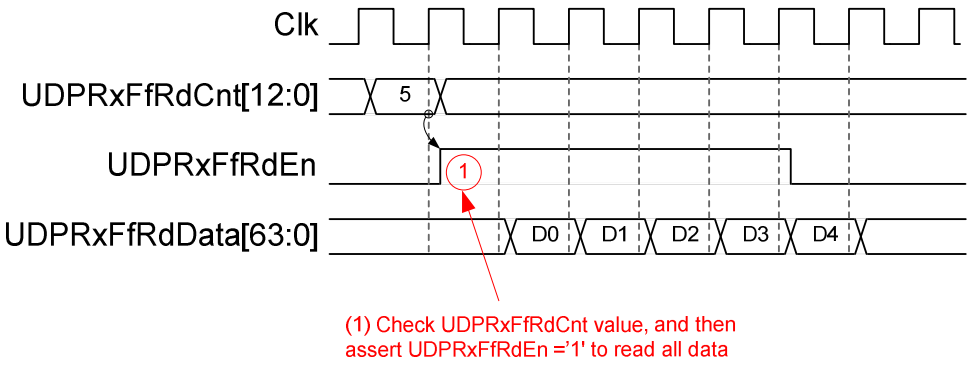

Figure 11: Rx Data Buffer Interface by Read counter Timing Diagram

EMAC Interface

To transmit packet, the IP asserts tx_axis_tvalid with the first data of the packet on tx_axis_tdata. The signals is latched until tx_axis_tready is asserted to '1' to acknowledge data transmit request. After that, tx_axis_tready must be asserted to '1' until the packet is end of transmission. tx_axis_tlast and tx_axis_tvalid are asserted to '1' with the last transmitted data to show end-of-packet status.

Xilinx EMAC de-asserts tx_axis_tready signal when the packet still not complete (between the 1st data and the last data). So, it needs to design the adapter logic with small buffer to connect between UDP10G-IP and Xilinx EMAC to store data from UDP10G-IP when EMAC is busy. This adapter logic has provided in the reference design.

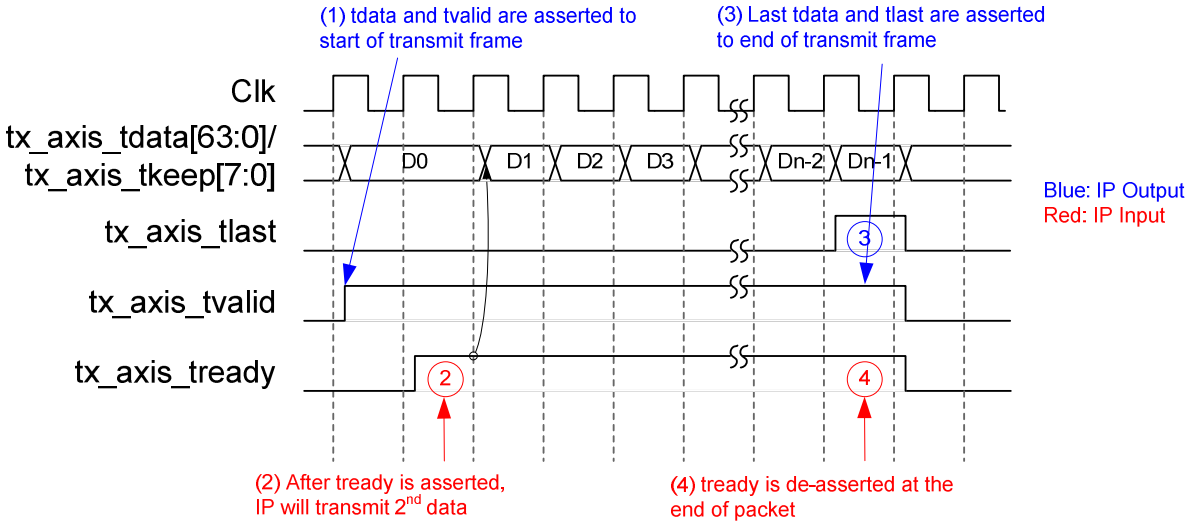

Figure 12: Transmit EMAC Interface

Figure 13 shows timing diagram of Receive side. The IP monitors start of received frame from rx_axis_tvalid which changes from '0' to '1'. rx_axis_tdata is transferred continuously until rx_axis_tlast is asserted to '1' at the end of packet with valid value of rx_axis_tuser. rx_axis_tvalid must be asserted to '1' between the 1st tvalid and tlast. rx_axis_tready is de-asserted to '0' to pause data transmission for 1 clock cycle after end of each packet.

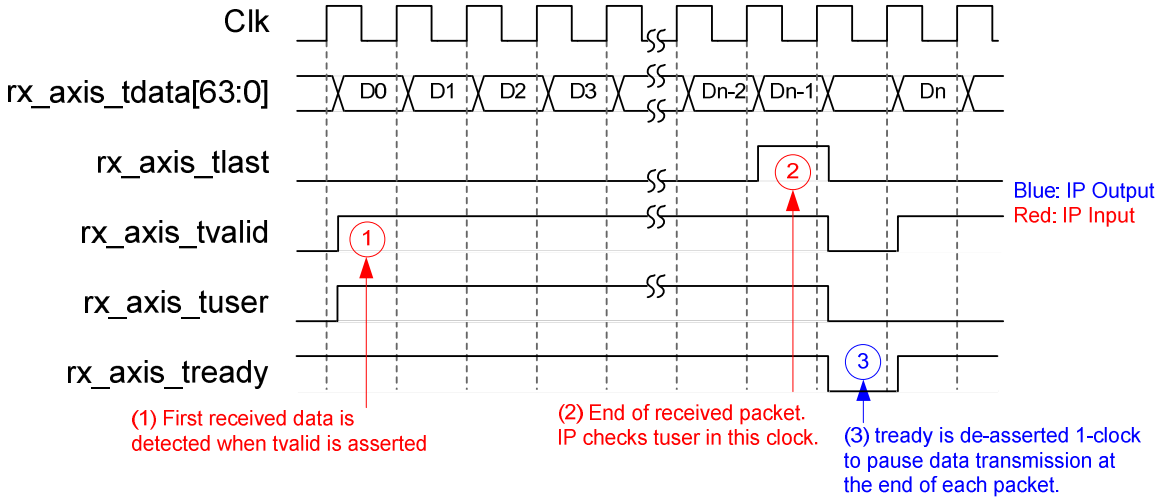

Figure 13: Receive EMAC Interface

Example usage

Client mode (SRV[0]='0')

The example of register setting sequence for data transmission and reception in Client mode is shown as follows.

1) Set RST register='1' to reset the IP

2) Set SML/SMH for MAC address, DIP/SIP for IP address, and DPN/SPN for port number.

3) Set RST register='0' to release the reset and then IP starts initialization by sending ARP request to get Target MAC address from ARP reply. After end of intialization, Busy signal will be cleared to '0'.

4) a. For data transmission, set TDL for total transfer length and PKL for packet size. Next, set CMD register to start data transmission. User sends data to TxFIFO and monitors busy flag until it is equal to 0'. After complete the command, user can change TDL/PKL value for new data transmission without IP reset.

b. For data reception, user monitors RxFIFO status and read data out when RxFIFO is not empty.

Server mode (SRV[0]='1')

The different point between server mode and client mode is the initialization process to get MAC address of the target. In client mode MAC address is extracted from ARP reply after UDP10G-IP sends ARP request. In server mode, MAC address is extracted from ARP request which has matched Target IP address. The process to send and receive data is same as client mode. The example sequence of server mode is shown as follows.

1) Set RST register='1' to reset the IP

2) Set SML/SMH for MAC address, DIP/SIP for IP address, and DPN/SPN for port number.

3) Set RST register='0' to clear the reset. IP starts initialization by waiting ARP request to get Target MAC address. Next, IP creates ARP reply to the Target. After end of intialization, busy signal is cleared to '0'.

4) Data process is same as client mode

Verification Methods

The UDP10G-IP Core functionality was verified by simulation and also proved on real board design by using KCU105/ZCU102 evaluation board.

Recommended Design Experience

User must be familiar with HDL design methodology to integrate this IP into the design.

Ordering Information

This product is available directly from Design Gateway Co., Ltd. Please contact Design Gatway Co., Ltd. for pricing and additional information about this product using the contact information on the front page of this datasheet.

Revision History

Revision	Date	Description
0.1	Aug-15-2017	Draft release
1.0	Sep-15-2017	Update IP specification
1.1	Mar-8-2019	Support ZCU102