Design Gateway Co.,Ltd
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
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on KCU105, ZCU102
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- UDP/IP stack implementation
- Support IPv4 protocol
- Full-duplex transferring by using two port numbers for each transfer direction
- Support more sessions by using multiple UDP10G IPs
- Support Jumbo frame
- Packet size for transmit must be aligned to 64-bit because Transmit data bus size is 64-bit
- Received data bus size is 64-bit, so total receive size must be aligned to 64-bit
- Transmit/Receive buffer size, adjustable for optimizing resource and performance
- Simple data interface by standard FIFO interface
- Simple control interface by single port RAM interface
- 64-bit AXI4 stream to interface with 10G/25G Ethernet MAC
- One clock domain interface by fixed 156.25 MHz clock frequency
- Reference designs available on KCU105, ZCU102 evaluation board
- Support IP fragmentation
- Customized service for following features
  - Multicast IP
  - Unaligned 64-bit data transferring
  - Network parameter assignment by other methods

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

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.

Applications

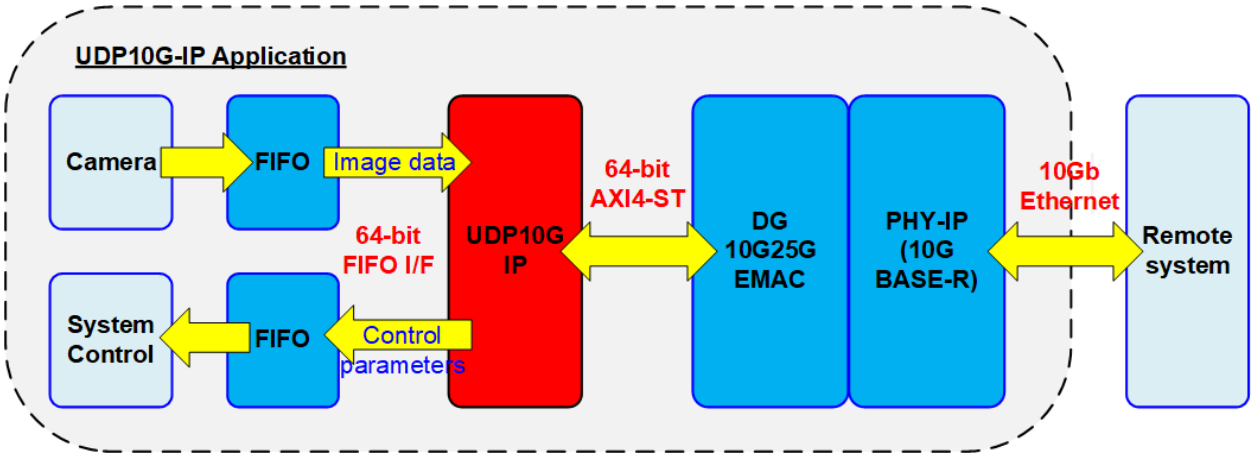

Figure 1: UDP10G IP Application

10Gb Ethernet is the communication channel which can transfer data at very high speed with remote controlling system. By using UDP/IP protocol for data streaming via 10 Gb Ethernet, the system can transfer very big data at very high speed rate. UDP10G IP is the IP which is integrated to the system for transfering data via 10 Gb Ethernet without using CPU and external memory. So, the IP can fit with the application which needs to send and receive data at high-speed rate based on FPGA solution such as video data streaming from camera and the monitoring system.

Figure 1 shows the example application of video camera system. The video data from camera is streaming to the FIFO and forwarded to remote system via 10 Gb Ethernet by UDP10G IP. UDP10G IP is desigend to support bi-directional transfer by using different port numbers at the same time, so Remote system can send the updated parameters for real-time controlling the system via 10 Gb Ethernet.

General Description

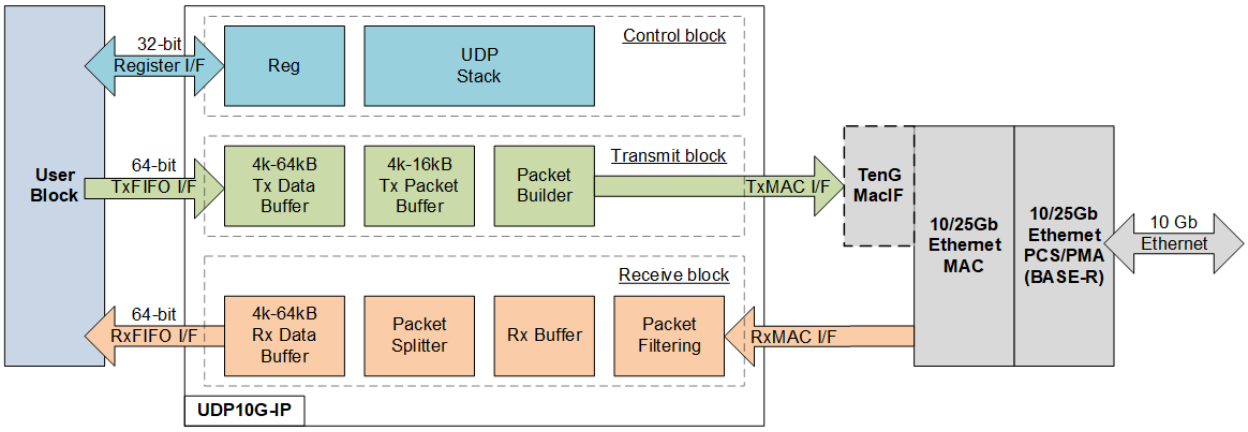

Figure 2: UDP10G IP Block Diagram

UDP10G IP core implements UDP/IP stack by hardware logic and connects with 10/25 Gb EMAC IP and PCS/PMA (BASE-R) module as the lower layer hardware. User interface of UDP10G IP consists of two interfaces, i.e. Register interface for control signals and FIFO interface for data signals.

Register interface has 4-bit address for accessing up to 16 registers, consisting of network parameters, command register and system parameters. The IP uses two sessions for transferring data in bi-directional, one session for one direction. All network parameters of both sessions must be similar, except the port number on the target device. The network parameters are assigned by the user to be fixed value for both UDP10G IP and the target device before starting IP initialization. The reset process is necessary when some network parameters must be changed. The initialization process has two modes to get MAC address of the target device. After finishing the initialization process, the IP is ready for transferring data with the target device.

To send the data, the user sets total transfer size and packet size to the IP and then transfers the data via TxFIFO interface which is 64-bit data size. When the data is received from the target, the user reads the received data from the IP via RxFIFO interface.

To meet the user system requirement which may be sensitive on the memory resource or the performance, the buffer size inside the IP can be assigned by the user. In Tx path, two buffers can be adjusted, i.e. Tx data buffer and Tx packet buffer. In Rx path, one buffer is available, named Rx data buffer. Buffer size in the IP is applied to be the data buffer between user logic and EMAC. Using bigger buffer size, the user logic can switch to run other task for longer time before switching back to transfer the data with UDP10G IP.

Functional Description

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

Control Block

- **Reg**

  All parameters of the IP are set via register interface which uses 4-bit address and 32-bit data bus. Timing diagram of register interface is similar to single-port RAM interface. The address is shared for both write and read directions.The description of each register is defined as shown in Table 3.

Table 3: Register map Definition

RegAddr [3:0]	Reg Name	Dir	Bit	Description
0000b	RST	Wr /Rd	[0]	Reset IP. '0': No reset, '1': Reset. Default value is '1'. **After all parameters are assigned, the user sets '0' to this register for loading parameters and start system initialization. User must set this register to '1' and '0' respectively when some network paramters are changed. The network parameters controlled by RST register are SML, SMH, DIP, SIP, DPN, SPN and SRV register.**
0001b	CMD	Wr	[0]	User command. Set '1' to start sending data. **Before setting this register to start new operation, user needs to confirm that the system is in Idle status by checking busy signal de-asserted to '0'. Busy signal is the IP output and can be read by bit[0] of CMD register.**
		Rd	[0]	System busy flag. '0': Idle, '1': IP is busy from initialization or send command. This signal is also mapped as Busy (IP output signal).
0010b	SML	Wr /Rd	[31:0]	Define 32-bit lower MAC address (bit [31:0]) for this IP. **To update this value, the IP must be reset by RST register.**
0011b	SMH	Wr /Rd	[15:0]	Define 16-bit upper MAC address (bit [47:32]) for this IP. **To update this value, the IP must be reset by RST register.**
0100b	DIP	Wr /Rd	[31:0]	Define 32-bit target IP address. **To update this value, the IP must be reset by RST register.**
0101b	SIP	Wr /Rd	[31:0]	Define 32-bit IP address for this IP. **To update this value, the IP must be reset by 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. **To update this value, the IP must be reset by RST register.**
0111b	SPN	Wr /Rd	[15:0]	Define 16-bit port number for this IP. **To update this value, the IP must be reset by RST register.**
1000b	TDL	Wr	[31:0]	Total Tx data length in byte unit, but the length must be aligned to 8-byte (data bus size). Valid range is 8-0xFFFFFFF8 (bit[2:0] is ignored by the IP). **User needs to set this register before setting CMD register='1'. This register is read when CMD register is set. After the IP runs Send data command (Busy='1'), the user can set the new value of TDL register for the next command. The user does not need to set TDL register again when the next command uses the same total data length.**
		Rd		Remaining transfer length in byte unit which does not transmit.

RegAddr [3:0]	Reg Name	Dir	Bit	Description
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. IntOut is asserted to '1' when the ARP reply is not received in time. This value is recommended to be more than 0x6000.
		Rd		The details of timeout interrupt. [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]	UDP data length of one Tx packet in byte unit, but the length must be aligned to 8-byte. Valid from 8-16000. Default value is 1472 byte which is the maximum size for non-jumbo frame that is aligned to 8-byte. Bit[2:0] of this register is ignored by the IP. **During running Send data command (Busy='1'), the user must not set this register. Similar to TDL register, the user does not need to set PKL register again when the next command uses the same packet length.**
1110b	SRV	Wr/ Rd	[0]	'0': Client mode (default). After RST register changes from '1' to '0', the IP sends ARP request to get Target MAC address from the ARP reply returned by the target device. IP busy is deasserted to '0' after receiving ARP reply. '1': Server mode. After RST register changes from '1' to '0', the IP waits ARP request from the Target to get Target MAC address. After receiving ARP request, the IP generates ARP reply and then de-asserts IP busy to '0'. **Note: In Server mode, when RST register changes from '1' to '0', the target device needs to resend ARP request for UDP10G IP completing the IP initialization.**
1111b	VER	Rd	[31:0]	IP version

- **UDP Stack**

  UDP stack is the main controller of the IP for controlling the other modules in every process. The IP operation has two phases, i.e. IP initialization phase and data transferring phase.

  After RST register changes from '1' to '0', the initialization phase begins. There are two modes for running the initialization phase, set by SRV[0] register, i.e. Client mode or Server mode. The parameters from Reg module is read by UDP Stack and then set to Transmit block and Receive block for transferring the packet to complete the initialization process, following the mode. After that, the IP changes to data transferring phase.

  UDP10G IP supports full-duplex transfer, so UDP10G IP can send data to the target device and receive data from the target device at the same time. Busy signal is asserted to '1' during sending data and de-asserted to '0' after finishing sending data.

  To send the data, the data from the user is stored in Tx data buffer and Tx packet buffer. After the network parameters are read to build UDP header by Packet Builder, Transmit block sends UDP packet including the data from the user to the target device via Ethernet MAC.

  When the data is received by Receive block, Busy signal is not asserted. UDP Stack does not need to send any packet to confirm the received data, so UDP Stack is available for running Send command. Full-duplex transfer can be run with full performance.

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

There are two buffers in Transmit block, i.e. Tx data buffer and Tx packet buffer which the size can be adjusted by parameter assignment. The minimum size of Tx data buffer and Tx packet buffer is limited by the transmit packet size, set by PKL register. Data from Tx data buffer is split to one packet size stored in Tx packet buffer. UDP header is prepared and then combined with UDP data stored in Tx packet buffer to build complete UDP packet. The data in Tx data buffer can be flushed after the data is forwarded to EMAC. After finishing the send data command, the user can change the packet size and total data size for the new send data command by updating PKL and TDL register respectively.

- **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 more than or equal to two times of Tx Packet Size, set by PKL register. If Tx data buffer always stores at least two packet data, UDP10G IP can transfer data to EMAC continuously and the system can get the best transmit performance on 10Gb Ethernet. Also, this buffer is the data buffer for the user logic. Using the bigger size allows the user logic to have longer time for switching to run other tasks before going back to handle with the data interface of Tx data buffer.

- **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). At most, two packets are stored in Tx Packet buffer, so the user can set the size of Tx packet buffer to fit with the system requirement.

- **Packet Builder**

  UDP packet consists of the header and the data. Packet builder receives network parameters, set in Reg module, and then prepares UDP header. Also, IP and UDP checksum are calculated to be UDP header. After that, all UDP header is built, the header combining with the data from Tx packet buffer is transmitted to EMAC.

Receive Block

In the receive block, Rx data buffer is included to store the received data from the target device. The data is stored in the buffer when the header in the packet is matched and Rx data buffer size has free space enough. The header in the packet is compared with the network paramters, set by Reg module, and IP/UDP checksum must be correct. Otherwise, the received packet will be ignored.

- **Rx Buffer**

  This is temporary buffer to store the received packets from EMAC when the previous packet is not completely processed.

- **Packet Filtering**

  The header in Rx packet are verifed by this module to validate the packet. The packet is valid when the following conditions are met.

(1) Network parameters are matched to the value in register, i.e. MAC address, IP address and Port number.
(2) The packet is ARP packet or UDP/IPv4 packet.
(3) IP header length is valid (IP header length is equal to 20 bytes).
(4) IP data length and UDP data length must be matched.
(5) IP checksum and UDP checksum are correct or disabled.

Note: UDP checksum is not verifed for fragment packet.

(6) In case of fragment packet, the packet must be received in a good sequence, not swapped.

- **Packet Splitter**

  This module is designed to remove the packet header and split only UDP data to store to Rx data buffer.

- **Rx Data Buffer**

  This buffer size is set by “RxBufBitWidth” parameter of the IP. The valid value is 9-13 for 4Kbyte – 64Kbyte buffer size. This is the data buffer between user logic and UDP10G IP for receiving data operation. If user logic does not read data from Rx data buffer for long time until the buffer is full, the new received packet will be ignored. The minimum size of Rx data buffer is recommended to be equal to two times of receive packet size or more. Bigger buffer size can store more data when the user logic is not available to read data from UDP10G IP. The proper size depends on the unavailable time of the user logic.

User Block

This module can be designed by using state machine to set the command and the parameters via register interface. Also, the status can be monitored to confirm the operation is finished without any error. The data path can connect with the FIFO for sending or receiving data with the IP.

10G/25G Ethernet MAC

Ethernet MAC implements the MAC layer for 10/25Gb Ethernet. The user interface to connect with UDP10G IP is 64-bit AXI4 stream. UDP10G IP can directly connect with DG 10G25GEMAC IP, but the additional logic with small FIFO (TenGMacIF in Figure 2) must be designed when connecting with Xilinx 10/25G EMAC IP.

Design Gateway provides 10G25GEMAC IP which can be applied with 10G/25G Ethernet solution. The resource can be optimized with less latency time. More details of the IP are described in following website.

https://dgway.com/products/IP/10GEMAC-IP/dg_10g25gemacip_data_sheet_xilinx_en.pdf

Xilinx provides 10G or 10/25G Ethernet Subsystem (Ethernet MAC and Ethernet PCS/PMA) with many features, described in the following website.

https://www.xilinx.com/products/intellectual-property/do-di-10gemac.html

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

10G Ethernet PCS/PMA (10GBASE-R)

This module is a no charge Xilinx LogiCORE which can read more details from following website.

https://www.xilinx.com/products/intellectual-property/10gbase-r.html

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. The EMAC Interface is 64-bit AXI4 stream interface.

Table 5: Core Parameters

Name	Value	Description
TxBufBitWidth	9-13	Setting Tx Data buffer size. The value is the address bus size of this buffer.
TxPacBitWidth	9-11	Setting Tx Packet buffer size. The value is the address bus size of this buffer.
RxBufBitWidth	9-13	Setting Rx Data buffer size. The value is the 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 to synchronous with the user interface and EMAC interface.
User Interface		
RegAddr[3:0]	In	Register address bus. In Write access, RegAddr is valid when RegWrEn='1'.
RegWrData[31:0]	In	Register write data bus. Valid when RegWrEn='1'.
RegWrEn	In	Register write enable. Valid at the same clock as RegAddr and RegWrData.
RegRdData[31:0]	Out	Register read data bus. Valid in the next clock after RegAddr is valid.
Busy	Out	IP busy status ('0'-Idle, '1'-IP is busy from the intialization process or sending data)
IntOut	Out	Timer interrupt. 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[10:0] register.
Tx Data Buffer Interface		
UDPTxFfFull	Out	Asserted to '1' when Tx data buffer is full. User needs to stop writing data within 4 clock cycles after this flag is asserted to '1'.
UDPTxFfWrEn	In	Write enable to Tx data buffer. Asserted to '1' to write data to Tx data buffer.
UDPTxFfWrData[63:0]	In	Write data to Tx data buffer. Valid when UDPTxFfWrEn='1'.
Rx Data Buffer Interface		
UDPRxFfRdCnt[12:0]	Out	Data counter of Rx data buffer to show the number of received data in 64-bit unit.
UDPRxFfLastRdCnt[2:0]	Out	Remaining byte of the last data in Rx data buffer when total received data in the buffer is not aligned to 8-byte unit. User cannot read the data until all 8-byte data is received.
UDPRxFfRdEmpty	Out	Asserted to '1' when Rx data buffer is empty. User needs to stop reading data immediately when this signal is asserted to '1'.
UDPRxFfRdEn	In	Assert to '1' to read data from Rx data buffer.
UDPRxFfRdData[63:0]	Out	Data output from Rx data buffer. Valid in the next clock cycle after UDPRxFfRdEn is asserted to '1'.

Signal	Dir	Description
MAC Interface		
tx_axis_tdata[63:0]	Out	Transmit data. Valid when tx_axis_tvalid='1'.
tx_axis_tkeep[7:0]	Out	Transmit data byte enable. Valid when tx_axis_tvalid='1'.
tx_axis_tvalid	Out	Transmit data valid signal.
tx_axis_tlast	Out	Control signal to indicate the final word in the frame. Valid when tx_axis_tvalid='1'.
tx_axis_tuser	Out	Control signal to indicate an error condition. This signal is always '0'.
tx_axis_tready	In	Handshaking signal. Asserted to '1' when tx_axis_tdata has been accepted. This signal must not be de-asserted to '0' when a packet is transmitting.
rx_axis_tdata[63:0]	In	Received data. Valid when rx_axis_tvalid='1'.
rx_axis_tvalid	In	Received data valid signal. rx_axis_tvalid must be asserted to '1' continuously for transferring a packet.
rx_axis_tlast	In	Control signal to indicate the final word in the frame. Valid when rx_axis_tvalid='1'.
rx_axis_tuser	In	Control signal asserted at the end of received frame (rx_axis_tvalid='1' and rx_axis_tlast='1') to indicate that the frame has CRC error. '1': normal packet, '0': error packet.
rx_axis_tready	Out	Handshaking signal. Asserted to '1' when rx_axis_tdata has been accepted. rx_axis_tready is de-asserted to '0' for 1 clock cycles to be the gap size between each received packet.

Timing Diagram

IP Initialization

The initialization process begins after user changes RST register from '1' to '0'. UDP10G IP can run in two modes, set by SRV[0] register, i.e. Client mode (SRV[0]='0') and Server mode (SRV[0]='1'). The details of each mode are shown in the following timing diagram.

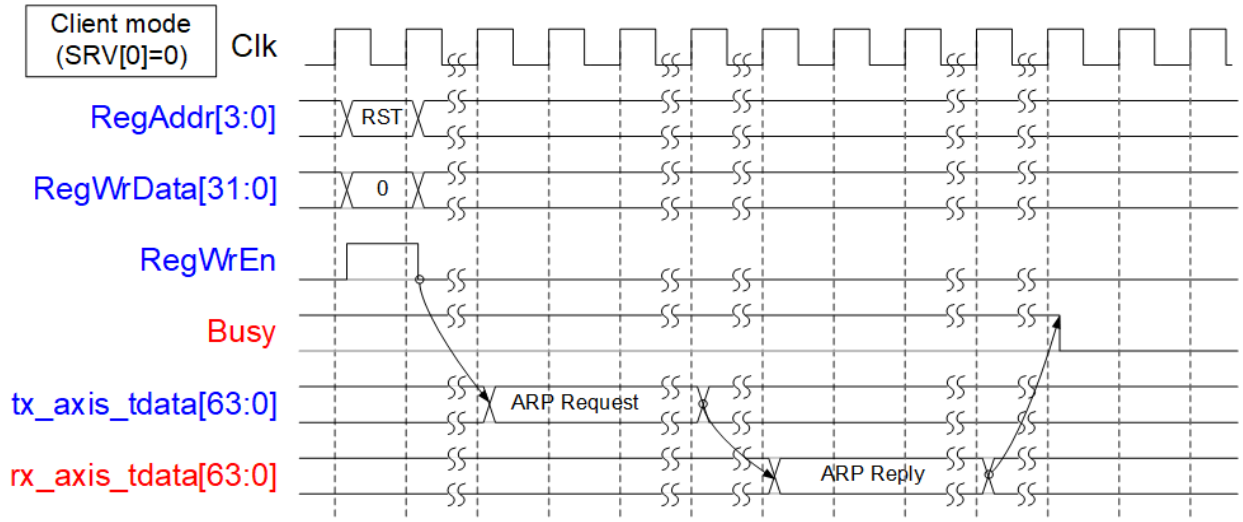

Figure 3: IP Initialization in Client mode

As shown in Figure 3, in Client mode, UDP10G IP sends ARP request and waits ARP reply returned from the target device. Target MAC address is extracted from ARP reply packet. After finishing, Busy signal is de-asserted to '0'.

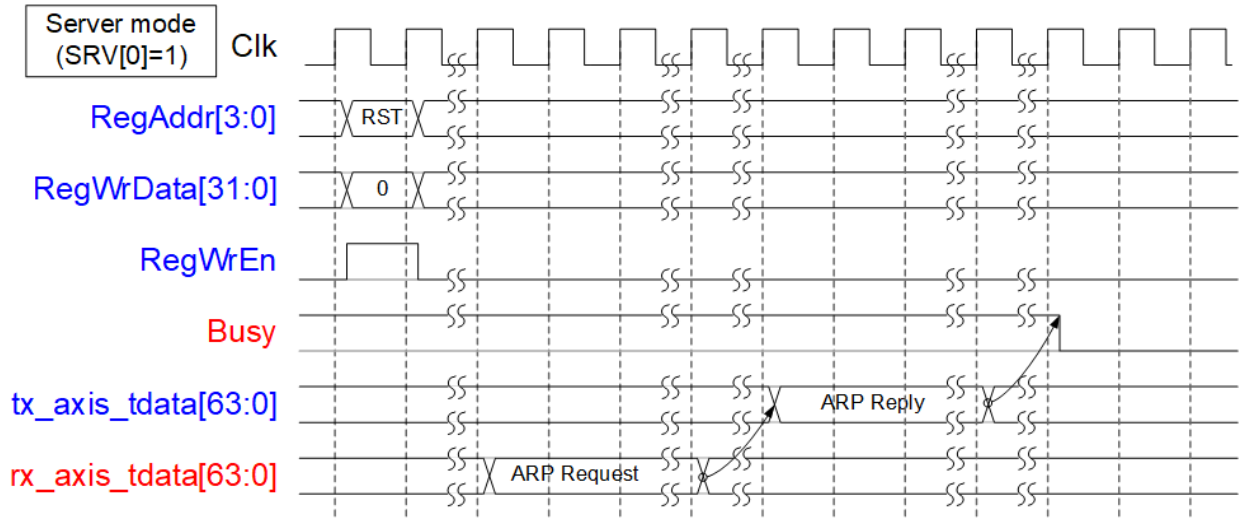

Figure 4: IP Initialization in Server mode

As shown in Figure 4, after reset process in Server mode, UDP10G IP waits ARP request sent by target device. After that, UDP10G IP returns ARP reply to the target. Target MAC address is extracted from ARP request packet. Finally, Busy signal is de-asserted to '0'.

Register Interface

All control signals and the network parameters for the operation are set and monitored via Register interface. Timing diagaram of Register interface is similar to Single-port RAM which shares the address bus for write and read access. Read latency time of the read data from the address is one clock cycle. Register map is defined in Table 3.

As shown in Figure 5, to write the register, the user sets RegWrEn='1' with the valid value of RegAddr and RegWrData. To read the register, the user sets only RegAddr and then RegRdData is valid in the next clock cycle.

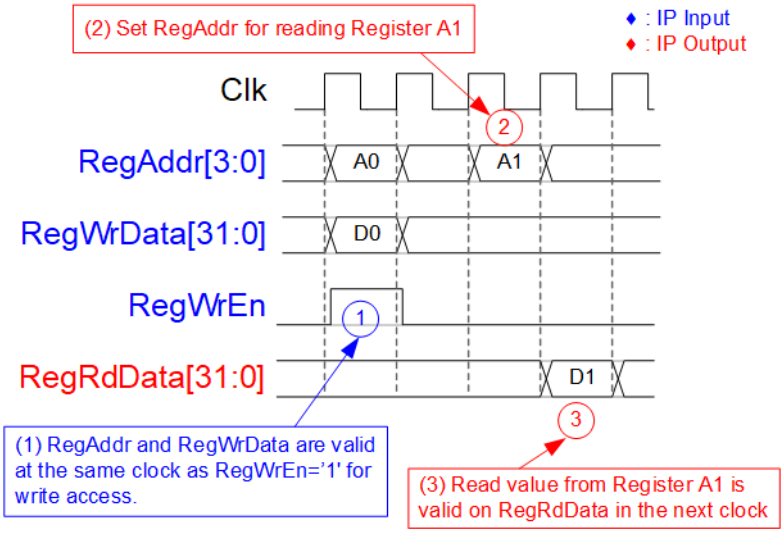

Figure 5: Register interface timing diagram

As shown in Figure 6, before the user sets CMD register to start the new command operation, Busy flag must be equal to '0' to confirm that IP is in Idle status. After CMD register is set, Busy flag is asserted to '1'. Busy is de-asserted to '0' when the command is completed.

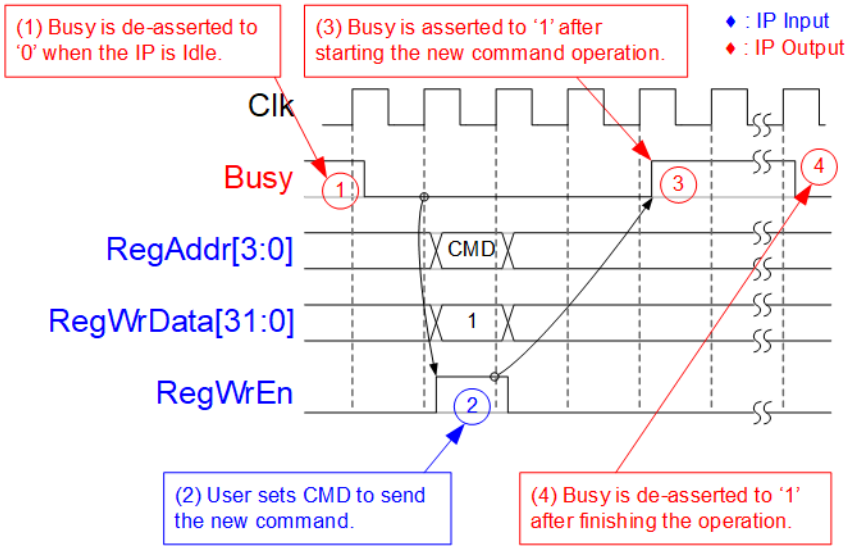

Figure 6: CMD register timing diagram

Tx FIFO Interface

To send the data to IP core via Tx FIFO interface, Full flag is monitored to be flow control signal. The write signals are similar to write interface of general FIFO by using write data and write enable as shown in as shown in Figure 7.

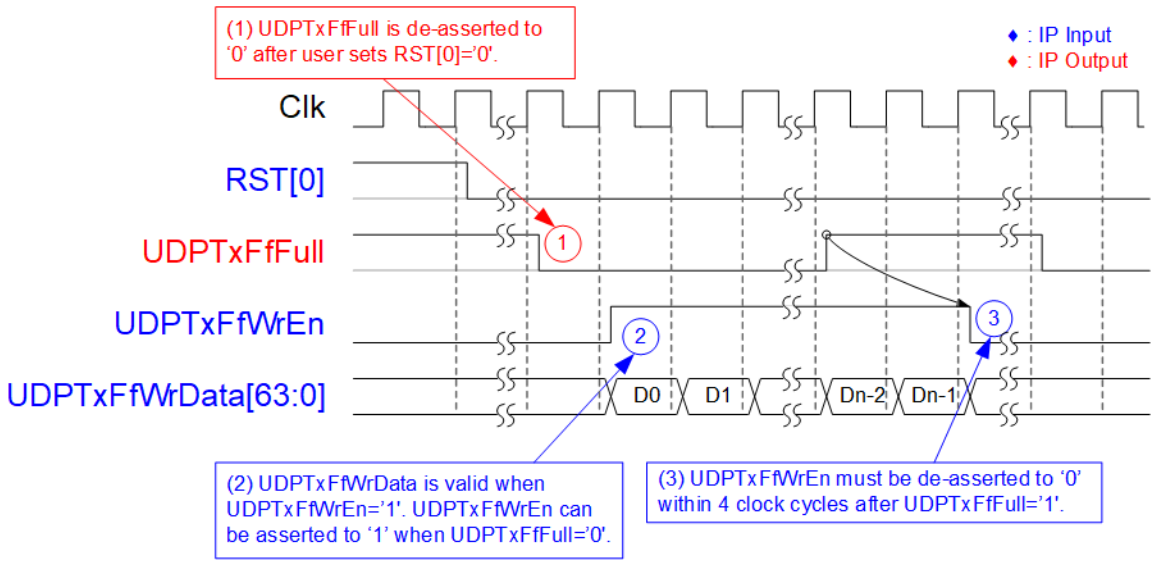

Figure 7: Tx FIFO interface timing diagram

(1) During IP is in reset condition, UDPTxFfFull is asserted to '1' and all data in FIFO are flushed.
(2) Before sending data, user needs to confirm that full flag (UDPTxFfFull) is not asserted to '1'. After that, UDPTxFfWrEn can be asserted to '1' with valid value of UDPTxFfWrData.
(3) UDPTxFfWrEn must be de-asserted to '0' within 4 clock cycles to pause data sending after UDPTxFfFull is asserted to '1'.

Rx FIFO Interface

After the received data is stored in Rx data buffer, the user can read the data from Rx data buffer by using Rx FIFO interface. Empty flag is monitored to check data available status and then asserts read enable signal to read the data, similar to read interface of general FIFO, as shown in Figure 8.

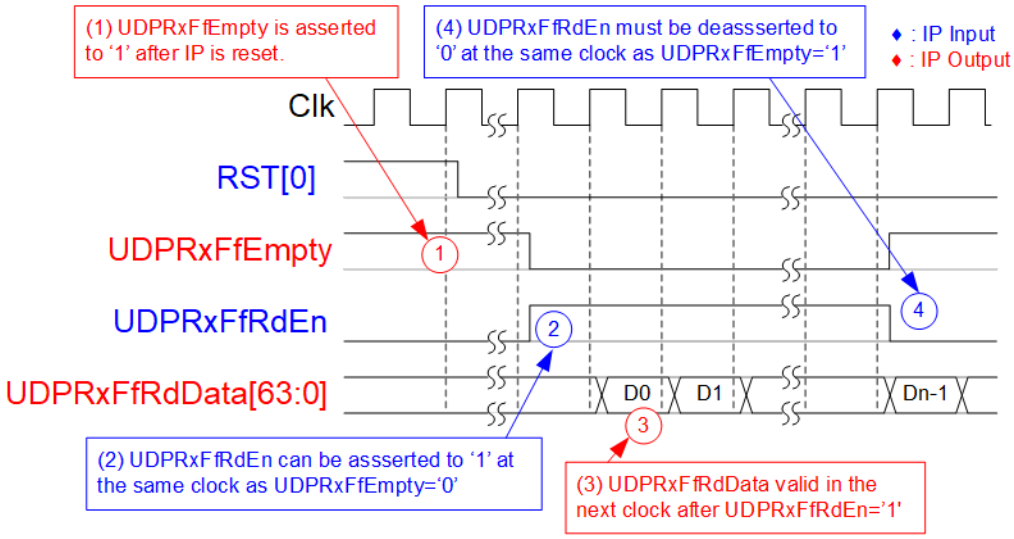

Figure 8: Rx FIFO interface timing diagram by Empty flag

(1) After the IP finishes reset process, there is no data in Rx data buffer (UDPRxFfEmpty='1').
(2) UDPRxFfEmpty is monitored to check data available status. When data is ready (UDPRxFfEmpty ='0'), UDPRxFfRdEn can be asserted to '1' to read data from Rx data buffer.
(3) UDPRxFfRdData is valid in the next clock cycle.
(4) Reading data must be immediately paused by de-asserting UDPRxFfRdEn='0' when UDPRxFfEmpty ='1'.

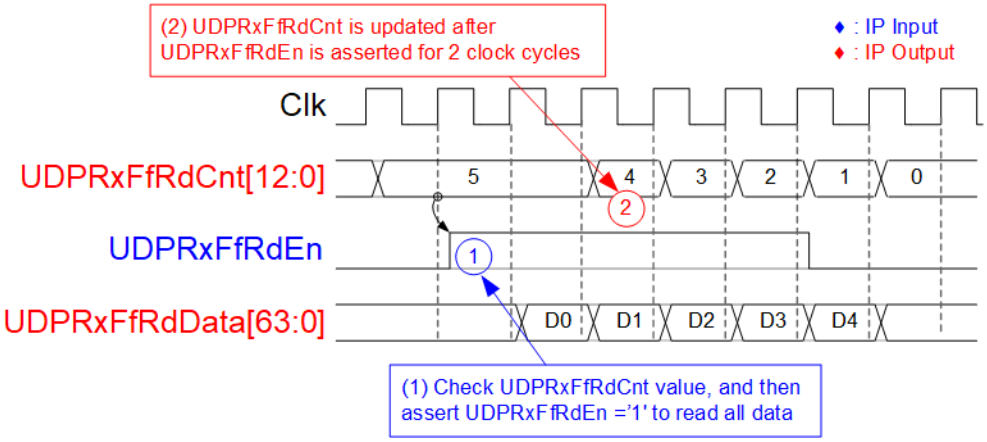

Figure 9: Rx FIFO interface timing diagram by using read counter

If user logic reads data as burst mode, UDP10G IP has read counter signal to show the total data stored in Rx FIFO interface as 64-bit unit. For example in Figure 9, there are five data available in Rx data buffer. So, user can assert UDPRxFfRdEn to '1' for 5 clock cycles to read all data from Rx data buffer. The latency time between read counter (UDPRxFfRdCnt) and read enable (UDPRxFfRdEn) is 2 clock cycles.

EMAC Interface

EMAC interface of UDP10G IP is designed by using 64-bit AXI4-stream interface. The limitation is that UDP10G IP cannot pause data transmission when the packet does not end. So, tx_axis_tready must be asserted to '1' during transmitting packet. Tx_axis_tready can be de-asserted to '0' after the last data in the packet is transferred, as shown in Figure 10.

From the limitation, UDP10G IP can connect with DG 10G25GEMAC IP core directly, but special logic with small FIFO must be added to interface between UDP10G IP and Xilinx 10G/25G EMAC IP.

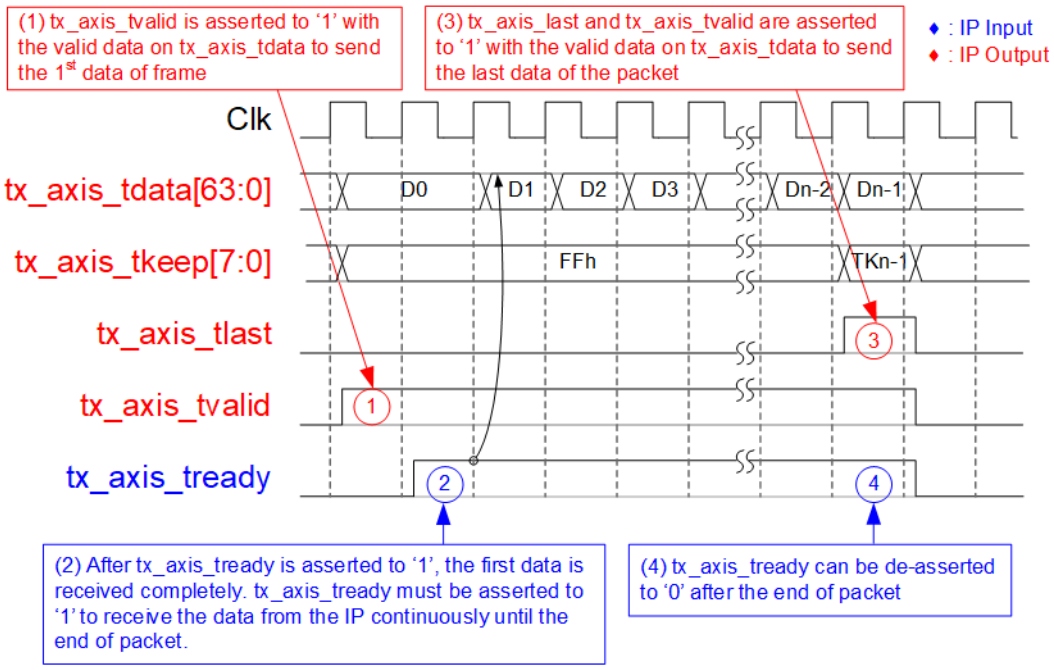

Figure 10: Transmit EMAC interface timing diagram

(1) UDP10G IP asserts tx_axis_tvalid with the first data of the packet. All signals are latched until tx_axis_tready is asserted to '1' to accept the first data.
(2) After the first data is accepted, tx_axis_tready must be asserted to '1' to accept all remaining data in the packet from UDP10G IP until end of packet. The IP sends all data of one packet continuously.
(3) tx_axis_tlast and tx_axis_tvalid are asserted to '1' when the last data of the packet is transmitted.
(4) After the end of the packet, tx_axis_tready can be asserted to '0' to pause the next packet transmission.

Similar to Transmit EMAC interface, the data of one packet must be received continuously in Receive EMAC interface. Valid signal must be asserted to '1' from the start of the packet to the end of the packet, as shown in Figure 11.

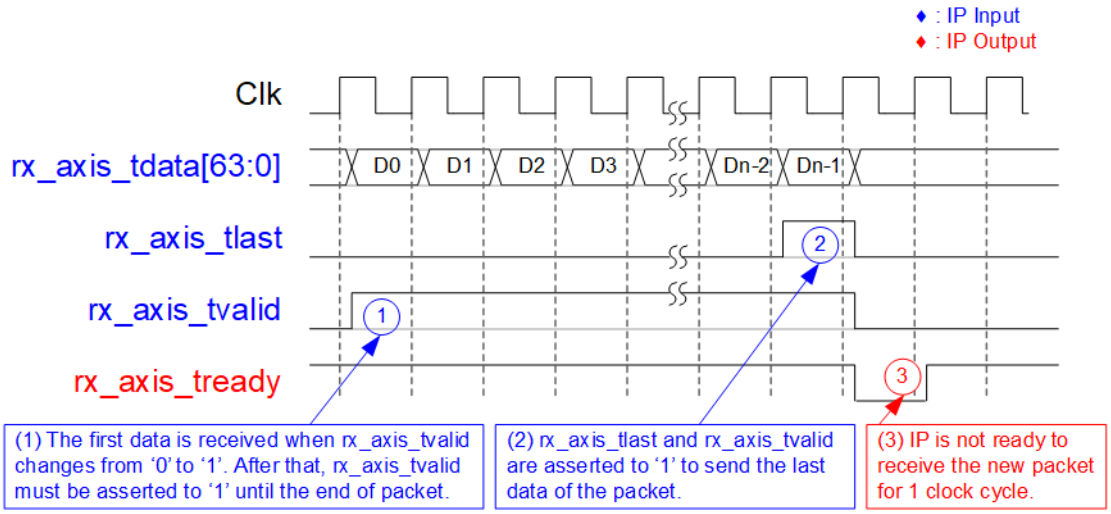

Figure 11: Receive EMAC interface timig diagram

(1) UDP10G IP detects start of the received frame when rx_axis_tvalid changes from '0' to '1' and the first data is valid on rx_axis_tdata. After that, rx_axis_tready is asserted to '1' to accept all data until the end of the packet. rx_axis_tvalid must be asserted to '1' for sending the data of one packet continuously.

(2) The end of the packet is detected when rx_axis_tlast='1' and rx_axis_tvalid='1'. At the same clock, the last data is vakud on rx_axis_tdata.

(3) After that, UDP10G IP de-asserts rx_axis_tready for 1 clock cycle to complete the packet post processing. So, EMAC must support to pause the data packet transmission for 1 clock cycle.

Example usage

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

The example step to set register for transferring data 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 start the IP initialization process by sending ARP request packet to get Target MAC address from ARP reply packet. Busy signal is de-asserted to '0' after finishing the initialization process.
4) a. For data transmission, set TDL register (total transmit length) and PKL register (packet size). Next, set CMD register = '1' to start data transmission. The user sends the data to UDP10G IP via TxFIFO interface before or after setting CMD register. When the command is finished, busy is de-asserted to '0'. The user can set the new value to TDL/PKL register and then set CMD register = '1' to start the next transmission.
   b. For data reception, user monitors RxFIFO status and reads data until RxFIFO is empty.

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

Comparing to Client mode which MAC address is decoded from ARP reply packet after UDP10G IP sends ARP request packet, Server mode decodes MAC address from ARP request packet. The process for transferring data is the same as Client mode. The example step 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 start the IP initialization process by waiting ARP request packet to get Target MAC address. Next, the IP creates ARP reply packet returned to the target device. After finishing the intialization, busy signal is de-asserted to '0'.
4) Remaining steps are similar to step 4 of 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 Gateway 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	15-Aug-2017	Draft release
1.0	15-Sep-2017	Update IP specification
1.1	8-Mar-2019	Support ZCU102
1.2	20-Aug-2020	Support DG 10G25GEMAC IP
1.3	22-Aug-2020	Update company information
1.4	2-Oct-2020	Correct information