

## UDP25G-IP Core

June 2, 2021

Product Specification

Rev1.0



### Design Gateway Co.,Ltd

E-mail: ip-sales@design-gateway.com

URL: www.design-gateway.com

### Features

- UDP/IP stack implementation
- Support IPv4 protocol
- Support Full-duplex transfer, Tx port and Rx port independently assigned
- Support more sessions by using multiple UDP25G IPs
- Support Jumbo frame
- Transmit packet size aligned to 128-bit, bus size of transmitted data
- Total receive data size aligned to 128-bit, bus size of received data
- Several Transmit/Receive buffer sizes
- Simple data interface by 128-bit FIFO interface
- Simple control interface by single-port RAM interface
- 64-bit AXI4 stream to interface for 10G/25G Ethernet MAC
- User clock frequency must be more than or equal to 195.3125 MHz for 25Gb Ethernet/156.25 MHz for 10Gb Ethernet
- Support 10G/25GbE by using 10G/25G Ethernet MAC and PCS
- Reference design available on KCU116 board/FB2CGHH@KU15P card
- Support IP fragmentation
- Customized service for following features
  - Multicast IP
  - Unaligned 128-bit data transferring
  - Network parameter assignment by other methods

Core Facts	
Provided with Core	
Documentation	Reference design manual Demo instruction manual
Design File Formats	Encrypted HDL
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference design manual
Additional Items	Demo on KCU116 board/ FB2CGHH@KU15P card
Support	
Support Provided by Design Gateway Co., Ltd.	

**Table 1: Example Implementation Statistics for Ultrascale device**

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB <sup>1</sup>	IOB	BRAMTile <sup>2</sup>	Design Tools
Kintex UltraScale+	XCKU5P-FFVB676-2-E	350	2407	2310	468	-	16	Vivado2019.1

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Block memory resources are based on 16kB Tx data buffer size, 16kB Tx packet buffer size, and 32kB Rx data buffer size.

June 2, 2021

## Applications

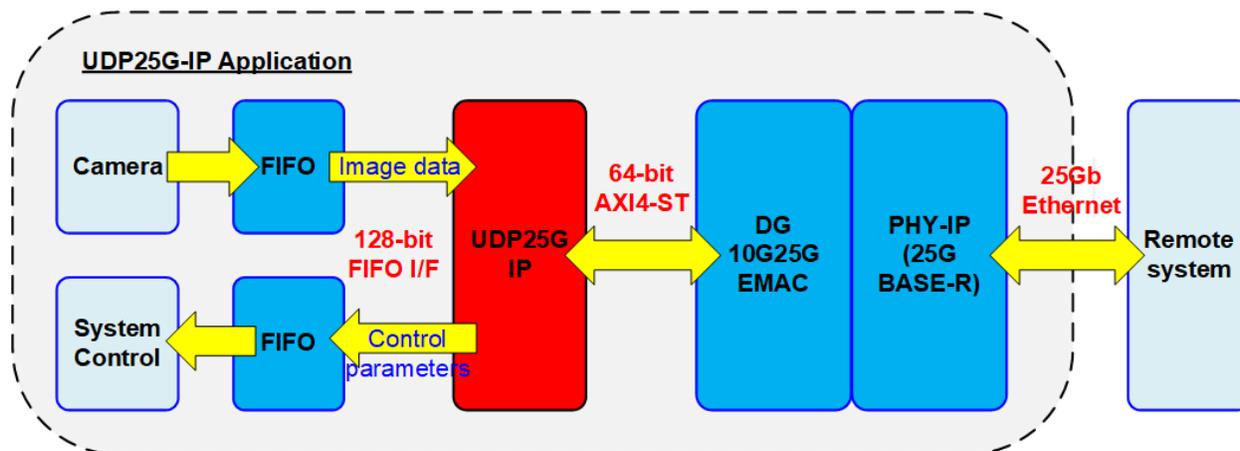
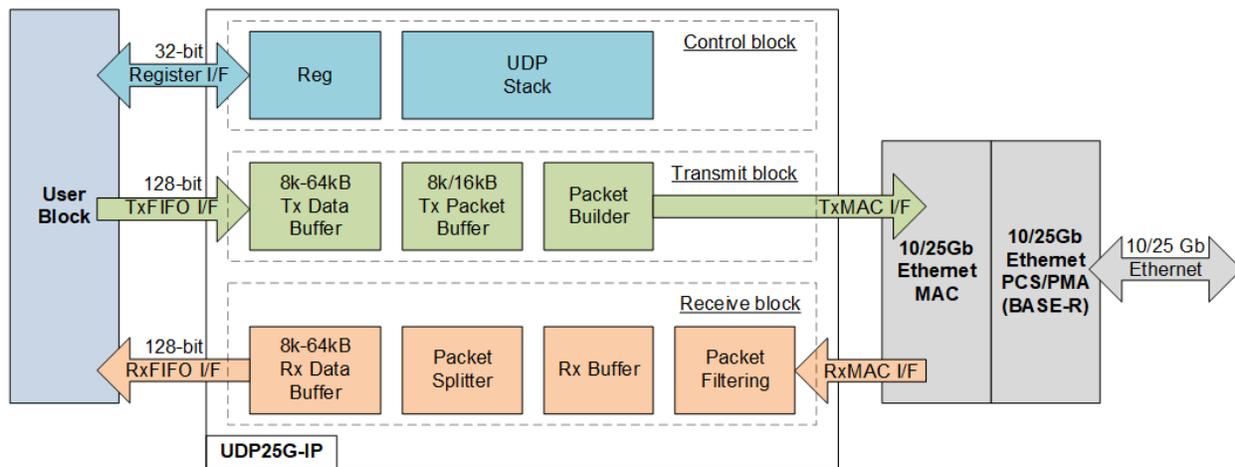


Figure 1: UDP25G IP Application

25Gb Ethernet is the communication channel which can transfer data at very high speed with remote controlling system. UDP/IP protocol is well-known protocol for transferring data stream in one-by-one system or broadcast system. By using UDP/IP with 25Gb Ethernet, big data can be transferred at very high speed rate. UDP25G IP is the IP core to be integrated in the FPGA for transferring 25Gb data following UDP/IP protocol without using CUP and external memory. Therefore, the IP can fit with the application that requires to transfer data at high-speed rate such as video data streaming from camera and the real-time monitoring system.

Figure 1 shows the example application of video camera system. The video stream from camera is forwarded to the FIFO, connected between Camera and UDP25G IP. After that, UDP25G IP creates the packet including the camera data and sends to Remote system via 25Gb Ethernet hardware. At the same time, UDP25G IP supports to receive the data from Remote system which can be assigned by different port number. Therefore, Remote system can update some parameters or control data as real-time controlling system via 25Gb Ethernet.

## General Description



**Figure 2: UDP25G IP Block Diagram**

UDP25G 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 UDP25G IP consists of two interfaces, i.e., Register interface for control signals and FIFO interface for data signals.

Register interface has 5-bit address for accessing up to 32 registers, consisting of network parameters, command register, and system parameters. The IP uses two sessions for transmitting data and receiving data at the same time, one session for one direction. The network parameters assigned in both sessions are similar, except the port number on target device. The parameters of UDP25G IP and the target device are assigned by the user before starting IP initialization. After that, the network parameters cannot be changed. The reset process is necessary to change some network parameters. The initialization process has three 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 128-bit data size. When the data is received from the target, the user reads the received data from the IP via RxFIFO interface.

The buffer size inside the IP can be assigned by the user. In Tx path, two buffers can be adjusted, Tx data buffer and Tx packet buffer. In Rx path, one buffer is available, named Rx data buffer. The buffer size must be much enough to store at least two data packets. If user logic does not pause data transferring with UDP25G IP, Tx data buffer and Rx data buffer will be always ready status. Therefore, the buffer size can be set as minimum value. Bigger buffer size is applied to store the data when the user logic is sometimes not ready for transferring data with UDP25G IP.

## Functional Description

UDP25G 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 has 5-bit address signals and 32-bit data signals. 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 2.

**Table 2: Register map Definition**

RegAddr [4:0]	Reg Name	Dir	Bit	Description
00000b	RST	Wr /Rd	[0]	Reset IP. '0': No reset, '1': Reset. Default value is '1'. <b>After all network parameters are assigned, the user sets '0' to this register for loading parameter and starting system initialization. User must set this register to '1' and '0' respectively when some network parameters are changed. The network parameters controlled by RST register are SML, SMH, DML, DMH, DIP, SIP, DPN, SPN and SRV register.</b>
00001b	CMD	Wr	[0]	User command. Set '1' to start sending data. <b>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.</b>
		Rd	[0]	System busy flag. '0': Idle, '1': IP is busy during initialization or send command. This signal is also mapped as IP output signal, Busy.
00010b	SML	Wr /Rd	[31:0]	Define 32-bit lower MAC address (bit [31:0]) for this IP. <b>To update this value, the IP must be reset by RST register.</b>
00011b	SMH	Wr /Rd	[15:0]	Define 16-bit upper MAC address (bit [47:32]) for this IP. <b>To update this value, the IP must be reset by RST register.</b>
00100b	DIP	Wr /Rd	[31:0]	Define 32-bit target IP address. <b>To update this value, the IP must be reset by RST register.</b>
00101b	SIP	Wr /Rd	[31:0]	Define 32-bit IP address for this IP. <b>To update this value, the IP must be reset by RST register.</b>
00110b	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. <b>To update this value, the IP must be reset by RST register.</b>
00111b	SPN	Wr /Rd	[15:0]	Define 16-bit port number for this IP. <b>To update this value, the IP must be reset by RST register.</b>
01000b	TDL	Wr	[31:0]	Define 32 lower bit (bit[31:0]) of 48-bit tx data length in byte unit. The length must be aligned to 16-byte (data bus size). Valid from 16-0xFFFF_FFFF_FFF0 (Bit[3:0] is ignored by the IP). The 16 upper bit (bit[47:32]) is assigned in TDH register (01101b). <b>User needs to set this register before setting CMD register = 1. TDL/TDH register are read when CMD register is set. After the IP runs Send data command (Busy='1'), the user can set the new value of TDL/TDH register for the next command. The user does not need to set TDL/TDH register again when the next command uses the same total data length.</b>
		Rd	[31:0]	32 lower bit of 48-bit remaining transfer length in byte unit which does not transmit.

RegAddr [4:0]	Reg Name	Dir	Bit	Description
01001b	TMO	Wr	[31:0]	Define timeout value for waiting ARP reply packet during IP initialization in Client mode. The counter is run under Clk input, so timer unit is equal to 1/Clk. IntOut is asserted to '1' when the ARP reply packet is not received in time. This value depends on latency time in the system. Typical value is more than 0x6000 to set more than 100 msec timeout.
		Rd		The details of timeout interrupt are shown in TMO[0] and TMO[10:8]. [0]-Timeout from not receiving ARP reply packet After timeout, the IP resends ARP request until ARP reply is received. [8]-Asserted when Rx data buffer is full. After that, all received packet are ignored until the buffer is not full. [9]-Asserted when UDP checksum of the received packet is error. [10]-Asserted when rx_axis_tuser shows error status.
01010b	PKL	Wr /Rd	[15:0]	UDP data length of one Tx packet in byte unit, but the length must be aligned to 16-byte. Valid from 16-8960. Default value is 1472 byte which is the maximum size of non-jumbo frame that is aligned to 16-byte. Bit[3:0] of this register is ignored by the IP. <b>During running Send data command (Busy='1'), the user must not set this register.</b> <b>Similar to TDL/TDH register, the user does not need to set PKL register again when the next command uses the same packet length.</b>
01101b	TDH	Wr	[15:0]	Define 16 upper bit (bit[47:32] of 48-bit tx data length in byte unit, as described in TDL register.
		Rd		16 upper bit of 48-bit remaining transfer length in byte unit which does not transmit, as described in TDL register.
01110b	SRV	Wr /Rd	[1:0]	"00": 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. "01": Server mode. After RST register changes from '1' to '0', the IP waits for 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'. "1x": Fixed MAC Mode. After RST register changes from '1' to '0', the IP updates the parameters and then de-asserts IP busy to '0'. Target MAC address is loaded by DML/DMH register. <b>Note: In Server mode, when RST register changes from '1' to '0', the target device needs to resend ARP request for UDP25G IP completing the IP initialization.</b>
01111b	VER	Rd	[31:0]	IP version
10000b	DML	Wr /Rd	[31:0]	Define 32 lower bit of target MAC address (bit [31:0]) for this IP when SRV[1:0]="1x" (Fixed MAC). <b>To update this value, the IP must be reset by RST register.</b>
10001b	DMH	Wr /Rd	[15:0]	Define 16 upper bit of target MAC address (bit [47:32]) for this IP when SRV[1:0]="1x" (Fixed MAC). <b>To update this value, the IP must be reset by RST register.</b>

- **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 three modes for running the initialization phase, set by SRV[1:0] register, i.e., Client mode, Server mode, and Fixed MAC 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.

UDP25G IP can send data and receive data with the target device at the same time. The port number of target device for transferring data in each direction can be assigned independently. 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.

**Table 3: TxBuf/TxPac/RxBufBitWidth Parameter description**

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

## Transmit Block

There are two buffers in Transmit block - 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 be 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 transferred 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/TDH register respectively.

- **Tx Data Buffer**

This buffer size is set by "TxBufBitWidth" parameter of the IP. The valid value is 9-12 which is equal to the address size of 128-bit buffer, as shown in Table 3. The buffer size should be more than or equal to two times of Tx packet size, set by PKL register to achieve the best performance. When using bigger buffer size, the IP uses longer time to transfer all data in Tx data buffer until it is empty. Before the buffer is empty, the user logic can switch to handle other tasks. If Tx data buffer always has the data, the transmit performance will be very high.

- **Tx Packet Buffer**

The buffer size is set by "TxPacBitWidth" parameter of the IP. The valid value is 9-10 and the description of the parameter is shown in Table 3. This buffer must store at least one transmit packet. Therefore, the buffer size must be more than Tx packet size, set by PKL register. The maximum value of PKL register is equal to (Tx Packet Buffer size<byte> – 64).

- **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 buffer has free space enough and the header in the packet is matched to the expected value, set by the network parameters inside Reg module. Also, the IP and UDP checksum in the packet must be correct. Otherwise, the received packet is rejected.

- **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 verified 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 Reg module, 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 ignored when the packet is fragment*
- (6) For fragment packet, the packet must be received in the correct order. The packet is rejected when the fragment offset is skipped value.

- **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 9-12 for 8Kbyte – 64Kbyte buffer size. The minimum size of Rx data buffer for transferring data continuously is two times of data size in received packet. To achieve the best performance, this buffer must not be full during receiving a packet. Similar to Tx data buffer, when Rx data buffer size is big, the user logic can switch to handle other tasks before returning to read the data from buffer. If the buffer is full, the packet will be lost and received performance will be dropped.

## User Block

The user 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 UDP25G IP is 64-bit AXI4 stream. UDP25G IP can directly connect with DG 10G25GEMAC IP, but the additional logic with small FIFO 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 DG 10G25GEMAC IP Core are described in following website.

[https://dgway.com/products/IP/GEMAC-IP/dg\\_10g25gemacip\\_data\\_sheet\\_xilinx.pdf](https://dgway.com/products/IP/GEMAC-IP/dg_10g25gemacip_data_sheet_xilinx.pdf)

Xilinx provides 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/ef-di-25gemac.html>

## 10/25G Ethernet PCS/PMA (10/25GBASE-R)

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

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

## Core I/O Signals

Descriptions of all parameters and I/O signals are provided in Table 4 - Table 6. The EMAC interface is 64-bit AXI4 stream interface.

**Table 4: Core Parameters**

Name	Value	Description
TxBufBitWidth	9-12	Tx data buffer size. The value is the address bus size of this buffer.
TxPacBitWidth	9-10	Tx packet buffer size. The value is the address bus size of this buffer.
RxBufBitWidth	9-12	Rx data buffer size. The value is the address bus size of this buffer.

**Table 5: User I/O Signals (Synchronous to Clk)**

Signal	Dir	Description
<b>Common Interface Signal</b>		
RstB	In	Reset IP core. Active Low.
Clk	In	User clock for running UDP25G IP. The frequency range is MacClk/2 – MacClk. When running with 25Gb Ethernet, MacClk frequency is 390.625 MHz. Therefore, the valid range of Clk frequency is 195.3125 – 390.625 MHz.
<b>User Interface</b>		
RegAddr[4:0]	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 when running initialization or send command.
IntOut	Out	IP Interrupt. Assert to high for 1 clock cycle when timeout is detected or received packet has error. More details of Interrupt status are monitored from TMO[10:0] register.
<b>Tx Data Buffer Interface</b>		
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[127:0]	In	Write data to Tx data buffer. Valid when UDPTxFfWrEn='1'.
<b>Rx Data Buffer Interface</b>		
UDPRxFfRdCnt[11:0]	Out	Data counter of Rx data buffer to show the number of received data in 128-bit unit.
UDPRxFfLastRdCnt[3:0]	Out	Remaining byte of the last data in Rx data buffer when total received data in the buffer is not aligned to 16-byte unit. User cannot read the data until all 16-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	Asserted to '1' to read data from Rx data buffer.
UDPRxFfRdData[127:0]	Out	Data output from Rx data buffer. Valid in the next clock cycle after UDPRxFfRdEn is asserted to '1'.

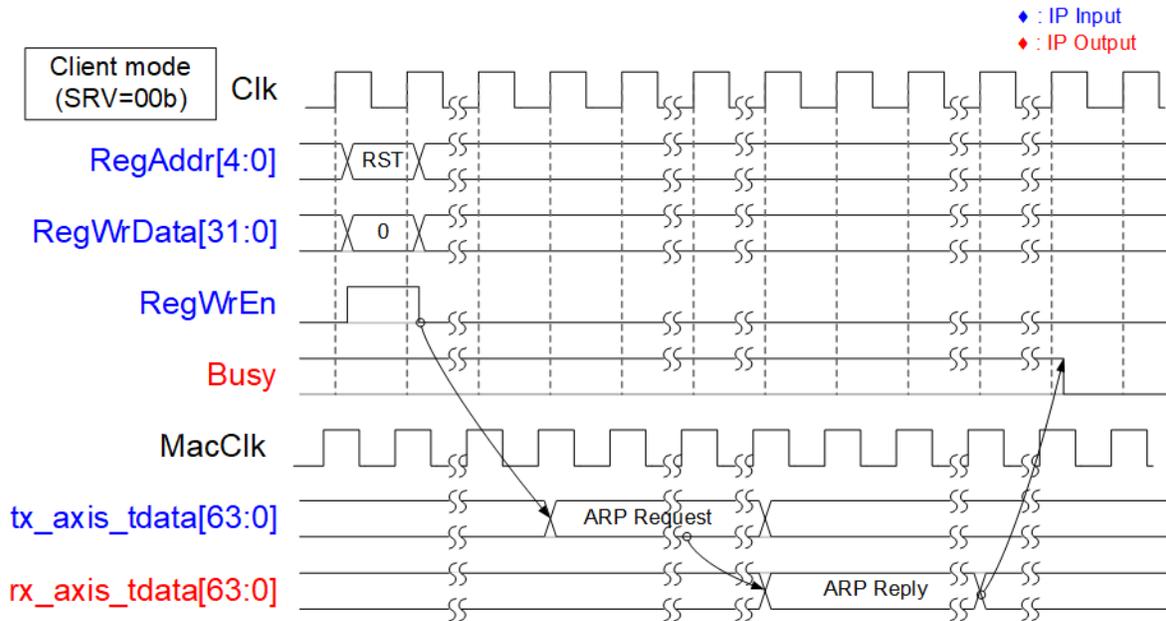
**Table 6: EMAC I/O Signals (Synchronous to MacClk)**

Signal	Dir	Description
MacClk	In	Receive clock from EMAC core which is equal to 390.625MHz for 25Gb Ethernet.
tx_axis_tdata[63:0]	Out	Transmitted data. Valid when tx_axis_tvalid='1'.
tx_axis_tkeep[7:0]	Out	Transmitted data byte enable. Valid when tx_axis_tvalid='1'.
tx_axis_tvalid	Out	Valid signal of transmitted data.
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	Valid signal of received data. 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. '0': normal packet, '1': 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

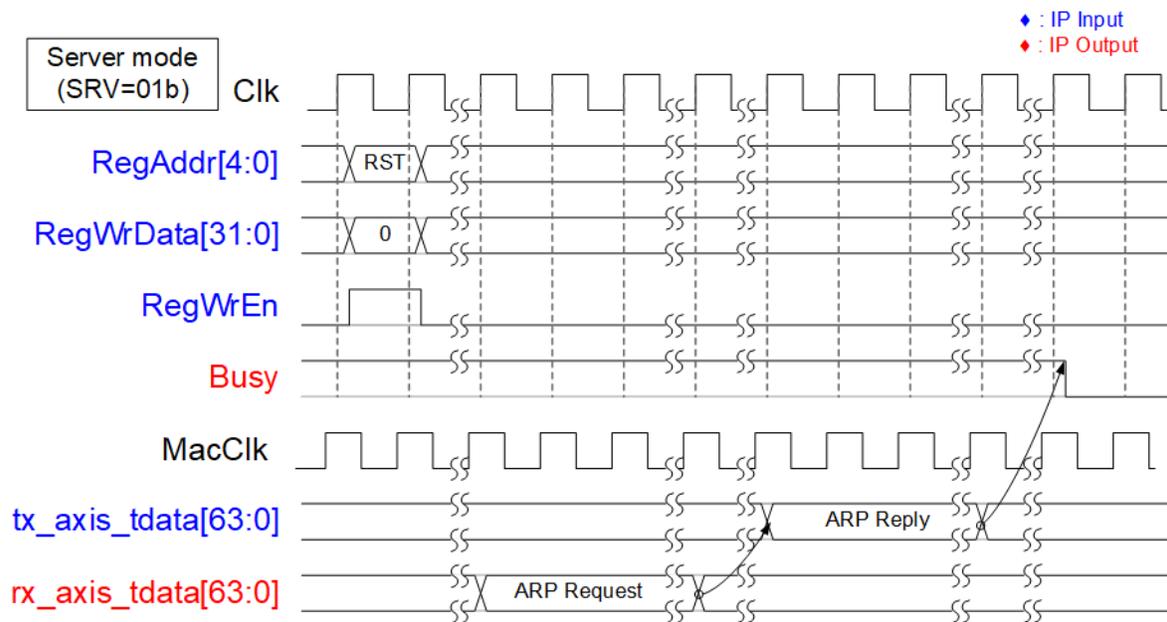
### IP Initialization

The initialization process begins after user changes RST register from '1' to '0'. UDP25G IP can run in three modes, set by SRV register, i.e., Client mode (SRV="00"), Server mode (SRV="01"), and Fixed MAC mode (SRV="1x"). The details of each mode are shown in the following timing diagram.



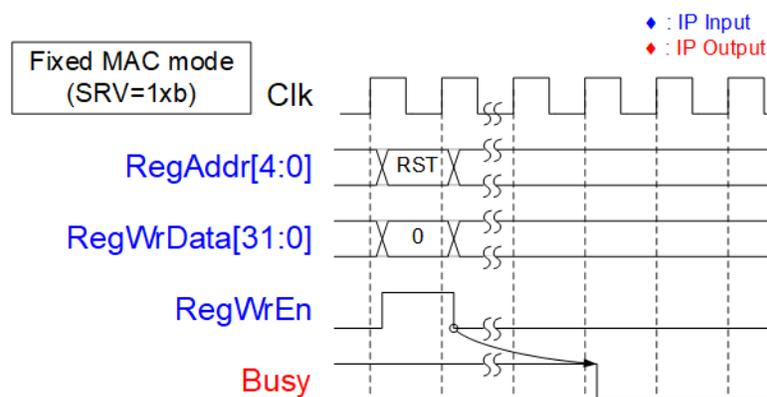
**Figure 3: IP Initialization in Client mode**

As shown in Figure 3, in Client mode UDP25G IP sends ARP request and waits until 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, UDP25G IP waits for ARP request sent by the target device. After that, UDP25G IP returns ARP reply to the target. Target MAC address is extracted from ARP request packet. Finally, Busy signal is de-asserted to '0'.



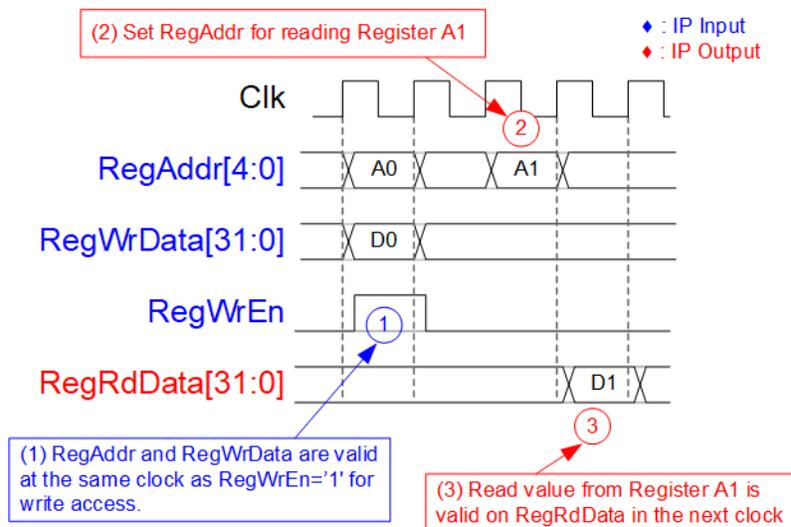
**Figure 5: IP Initialization in Fixed mode**

As shown in Figure 5, after reset process in Fixed MAC mode, UDP25G IP updates all parameters from the registers. Target MAC address is loaded from DML and DMH register. After finishing, Busy signal is de-asserted to '0'.

### Register Interface

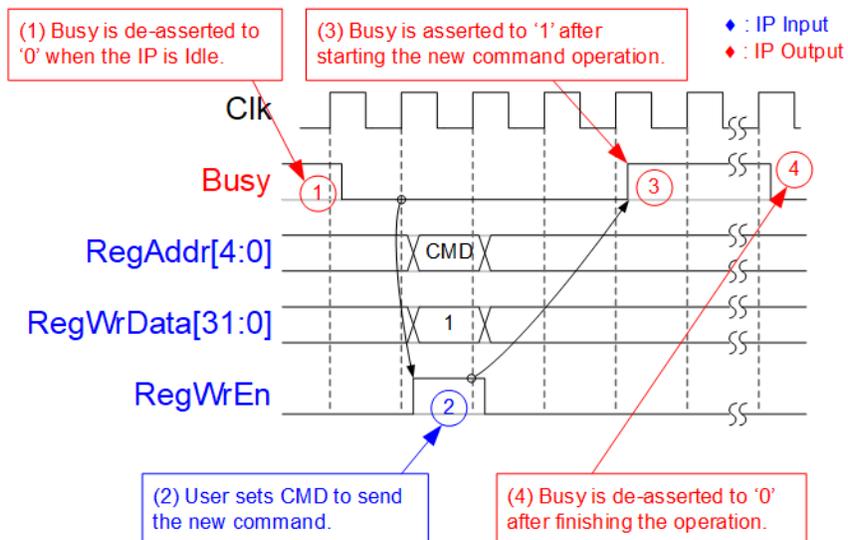
All control signals and the network parameters for the operation are set and monitored via Register interface. Timing diagram of Register interface is similar to Single-port RAM which shares the address bus for write and read access. Read latency time of the read data from the address is one clock cycle. Register map is defined in Table 2.

As shown in Figure 6, 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 6: Register interface timing diagram**

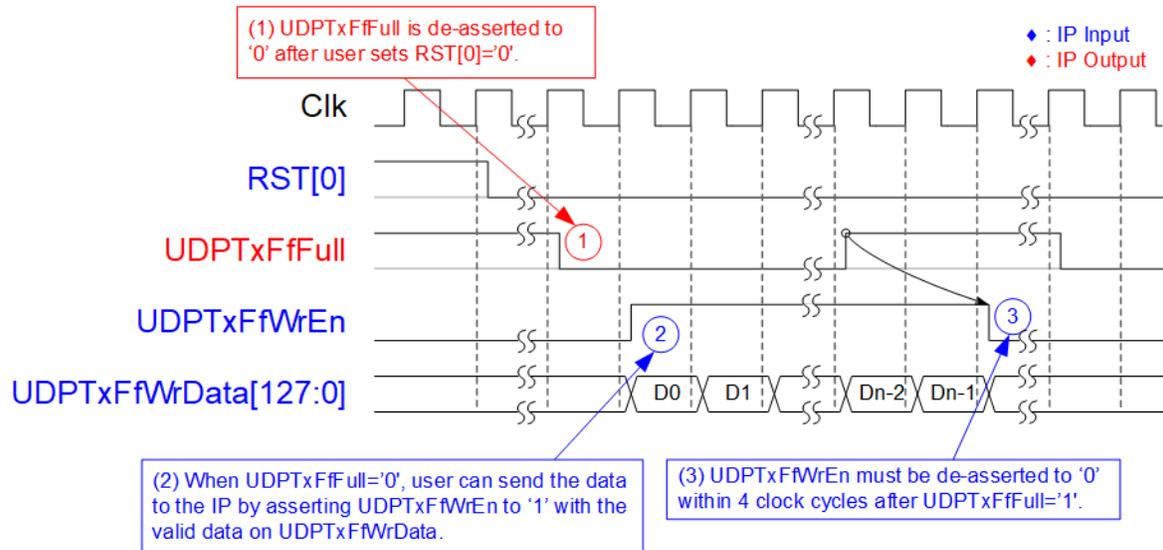
As shown in Figure 7, before the user sets CMD register to start the new command operation, Busy flag must be equal to '0' to confirm that IP is in Idle status. After CMD register is set, Busy flag is asserted to '1'. Busy is de-asserted to '0' when the command is completed.



**Figure 7: 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 Figure 8.

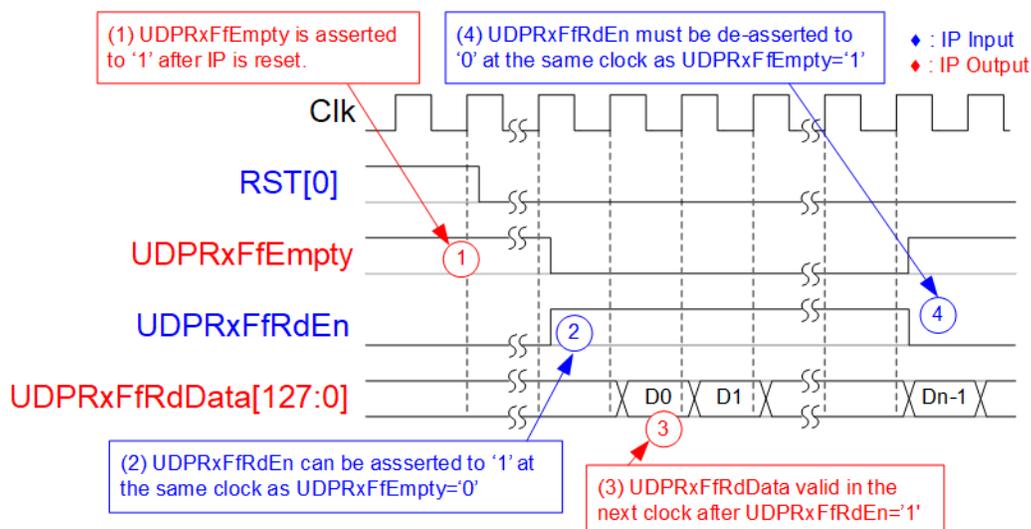


**Figure 8: Tx FIFO interface timing diagram**

- (1) When the IP is in reset state (RST[0] register='1'), full flag (UDPTxFfFull) is asserted to '1' to block the data from user. After the reset is de-asserted (RST[0]='0'), UDPTxFfFull is de-asserted to '0'. The user can write the data to the IP.
- (2) To write the data, UDPTxFWrEn is asserted to '1'. At the same clock, UDPTxFWrData is valid to send the data.
- (3) If UDPTxFfFull is asserted to '1', the user must pause sending data by de-asserting UDPTxFWrEn to '0' within 4 clock cycles.

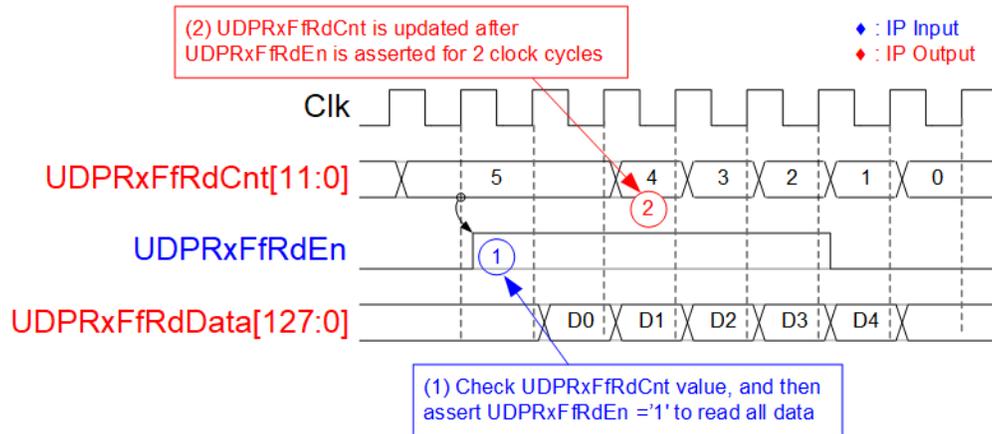
### Rx FIFO Interface

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. Read enable is asserted to '1' to read the data, similar to read interface of general FIFO, as shown in Figure 9.



**Figure 9: Rx FIFO interface timing diagram by using Empty flag**

- (1) When the IP is reset by RST[0], all data in Rx FIFO is flushed. Therefore, UDPRxFfEmpty is asserted to '1'.
- (2) To receive the data, user logic waits until UDPRxFfEmpty is de-asserted to '0'. If the received packet is valid and includes the data, UDPRxFfEmpty is de-asserted to '0' to allow the user logic reading. To read the data, UDPRxFfRdEn can be asserted to '1' at the same clock as UDPRxFfEmpty de-asserted.
- (3) Similar to general FIFO, the read data (UDPRxFfRdData) is valid in the next clock after asserting read enable.
- (4) If the FIFO is empty (UDPRxFfEmpty='1'), the user must pause reading data at the same clock.

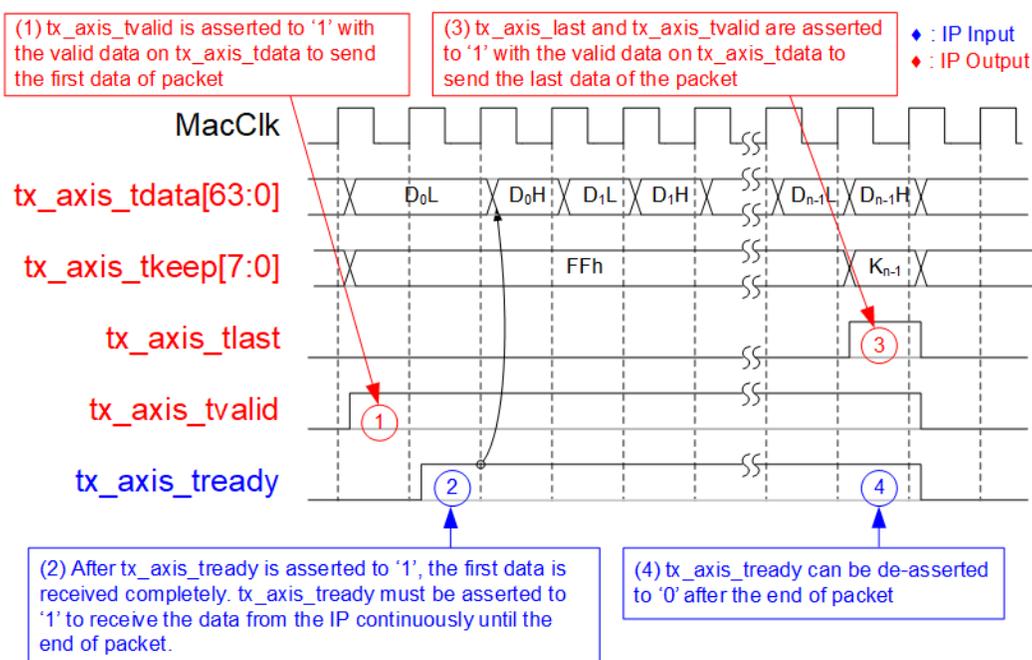


**Figure 10: Rx FIFO interface timing diagram by using read counter**

If user logic reads data as burst mode, UDP25G IP has read counter signal to show the total data stored in Rx FIFO interface as 128-bit unit. For example, Figure 10 shows 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

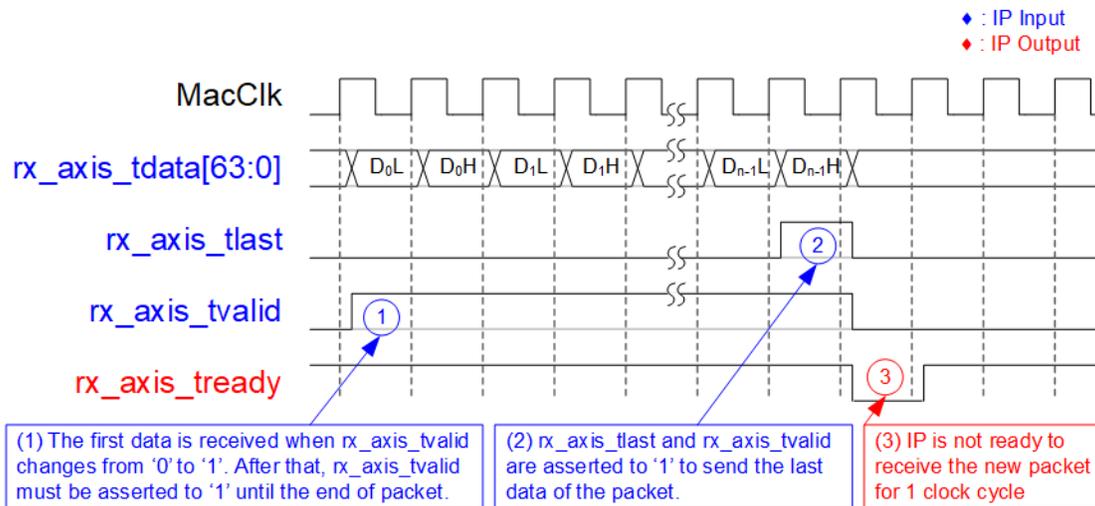
EMAC interface of UDP25G IP is designed by using 64-bit AXI4-stream interface. The limitation is that UDP25G IP cannot pause data transmission when the packet does not end. Therefore, `tx_axis_tready` must be asserted to '1' during transmitting a packet. `tx_axis_tready` can be de-asserted to '0' after the last data in the packet is transferred, as shown in Figure 11.



**Figure 11: Transmit EMAC interface timing diagram**

- (1) UDP25G 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 by EMAC, `tx_axis_tready` must be asserted to '1' to accept all remaining data in the packet from UDP25G 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 12.



**Figure 12: Receive EMAC interface timing diagram**

- (1) UDP25G 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 valid on rx\_axis\_tdata.
- (3) After that, UDP25G IP de-asserts rx\_axis\_tready for 1 clock cycle to complete the packet post-processing. Therefore, EMAC must support to pause the data packet transmission for 1 clock cycle.

The limitation of Tx and Rx EMAC interface of UDP25G IP is that a packet must transfer continuously. However, Xilinx EMAC IP cannot guarantee that the data of each packet for both Tx and Rx interface is transferred continuously. Therefore, UDP25G IP cannot connect with Xilinx EMAC IP (10/25G Ethernet Subsystem IP) directly. The buffer with the additional logic is required to store the data when Xilinx EMAC is not ready for transferring the data for both Tx and Rx interface.

The default reference design of UDP25G IP uses DG 10G25GEMAC IP which can directly connect without the additional logic and buffer. However, the reference design of UDP25G IP by using Xilinx EMAC IP can be requested.

## Example usage

### Client mode (SRV[1:0]="00")

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/TDH register (total transmit length) and PKL register (packet size). Next, set CMD register = '1' (Send data) to start data transmission. The user sends the data to UDP25G IP via Tx FIFO interface before or after setting CMD register. When the command is finished, busy flag is de-asserted to '0'. The user can set the new value to TDL/TDH/PKL register and then set CMD register = "Send data" to start the next transmission.  
b. For data reception, user monitors Rx FIFO status and reads data until Rx FIFO is empty.

### Server mode (SRV[1:0]="01")

Comparing to Client mode which MAC address is decoded from ARP reply packet after UDP25G IP sends ARP request packet, Server mode decodes MAC address from ARP request packet. The process for transferring data is similar to 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 initialization, busy signal is de-asserted to '0'.
- 4) Similar to step 4 of Client mode.

### Fixed MAC mode (SRV[1:0]="1x")

In Fixed MAC mode, MAC Address of the target device is loaded by DML and DMH register. The process for transferring the data is similar to Client and Server mode. The example steps of Fixed MAC mode are shown as follows

- 1) Set RST register='1' to reset the IP.
- 2) Set SML/SMH for MAC address of UDP25G IP, DML/DMH for MAC address of the target device, DIP/SIP for IP address, and DPN/SPN for port number.
- 3) Set RST register='0' to start the IP initialization process. After finishing the initialization, busy signal is de-asserted to '0'.
- 4) Similar to step 4 of Client mode.

## Verification Methods

The UDP25G IP Core functionality was verified by simulation and also proved on real board design by using KCU116 evaluation board and Silicom FB2CGHH@KU15P board.

## Recommended Design Experience

User must be familiar with HDL design methodology to integrate this IP into their 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
1.0	2-Jun-21	New release