Design Gateway Co.,Ltd
E-mail: ip-sales@design-gateway.com
URL: design-gateway.com

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/Alveo U250/ FB2CGHH@KU15P card
Support	
Support Provided by Design Gateway Co., Ltd.	

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 UDP100G IPs
- Support Jumbo frame
- Transmit packet size aligned to 512-bit, bus size of transmitted data
- Total receive data size aligned to 512-bit, bus size of received data
- Two Transmit/Receive buffer sizes - 32 Kbytes and 64 Kbytes
- Simple data interface by 512-bit FIFO interface
- Simple control interface by single-port RAM interface
- 512-bit AXI4 stream interface with 100G Ethernet MAC
- At least 240 MHz user clock frequency
- Reference design available on KCU116 board/Alveo U250 card/FB2CGHH@KU15P card
- Support IP fragmentation
- Customized service for following features
  - Multicast IP
  - Unaligned 512-bit data transferring
  - Network parameter assignment by other methods

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB[1]	IOB	BRAMTile[2]	Design Tools
Kintex UltraScale+	XCKU5P-FFVB676-2-E	350	7380	6673	1392	-	53	Vivado2019.1
Alveo	U250	350	7354	6643	1388		53	Vivado2019.1

Notes:

1) Actual logic resource dependent on percentage of unrelated logic
2) Block RAM resources are based on 64kB Tx data buffer size and 64kB Rx data buffer size which is the maximum size. However, using 32kB buffer size can achieve the best transfer performance

Applications

UDP/IP protocol is a well-known protocol for streaming data for real-time application. By using 100Gb Ethernet connection, the data stream can transfer more than 10 Gbyte/s which is enough for transferring many video stream in the studio or many sensor data from high-speed A/D converter in satellite system.

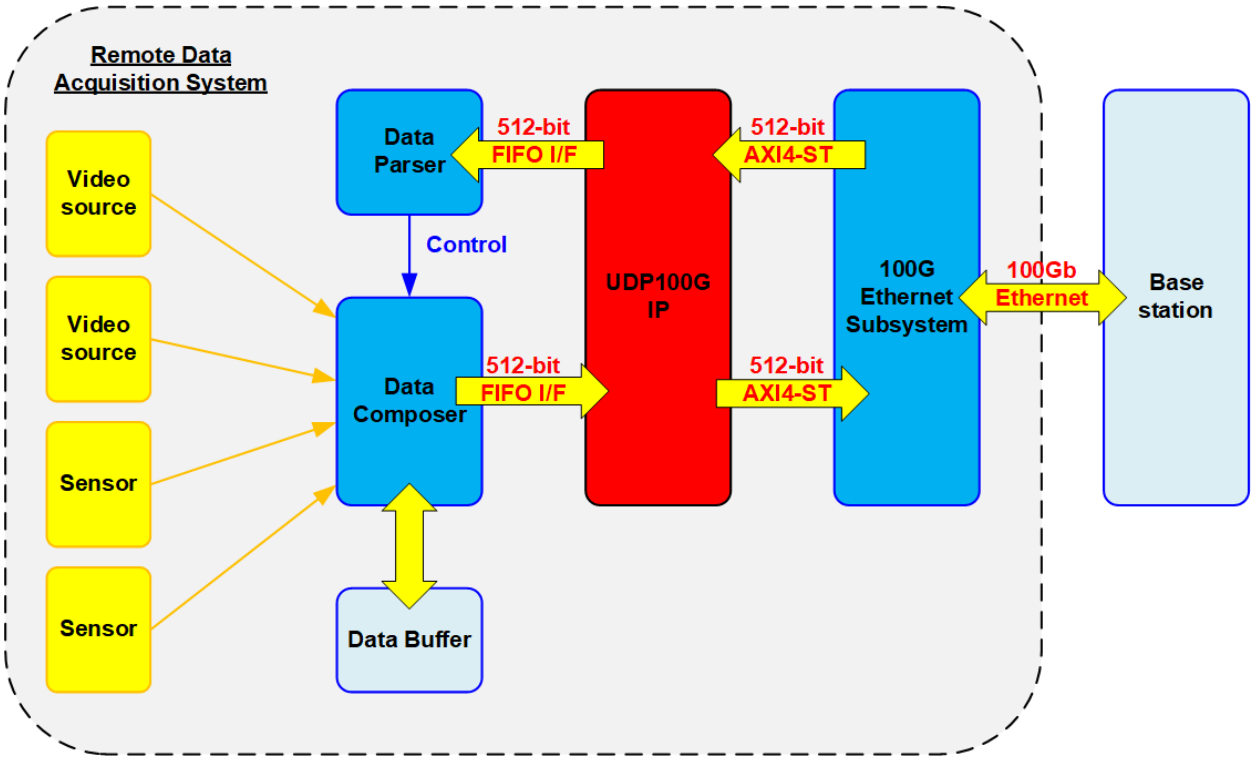

Figure 1: Remote data acquisition application

Figure 1 shows the example application to use UDP100G IP for sending ultra-speed data which consists of many data types such as video stream and sensor data by using 100Gb Ethernet. The data from many sources are connected to Data composer for constructing the data block in the data buffer. When the data is ready, the data block from buffer is transferred to UDP100G IP. UDP100G IP constructs UDP packet which includes UDP payload data from Data composer and then forwards to Remote base station via 100Gb Ethernet Subsystem and 100Gb Ethernet.

At the same time, base station can set the control parameters to UDP100G IP by using 100Gb Ethernet. UDP100G IP supports full duplex data transferring. Therefore, the packet that includes system parameters from base station can be decoded by Data parser via 512-bit of receive FIFO interface of UDP100G IP.

From above solution, the data acquisition system can be operated at ultra-speed performance with real-time controlling and monitoring.

General Description

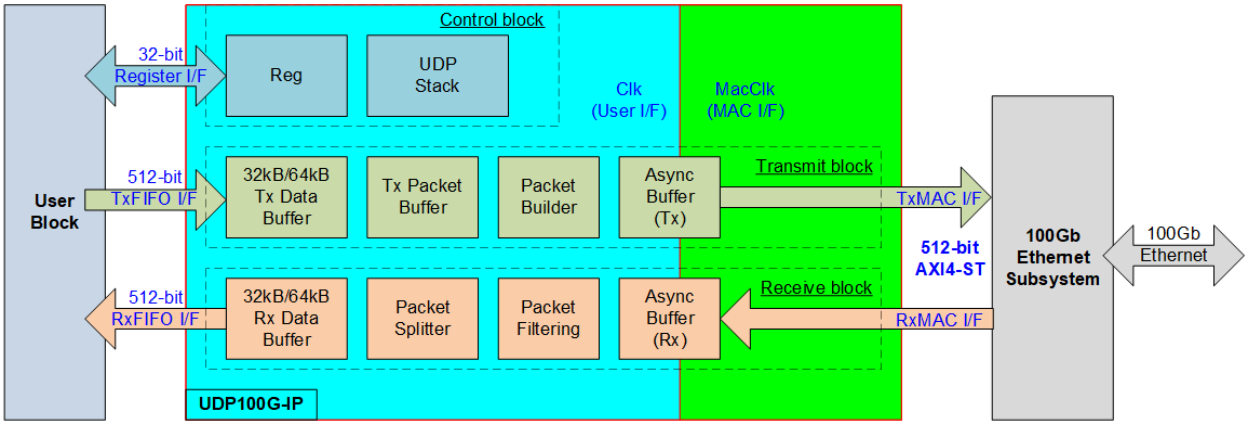

Figure 2: UDP100G IP Block Diagram

UDP100G IP core implements UDP/IP stack by hardwire logic and connects with 100Gb Ethernet Subsystem module for the low-layer hardware. User interface of UDP100G IP consists of two interfaces, i.e., Register interface for control signals and FIFO interface for data signals. There are two clock domains applied in UDP100G IP – Clk for user interface and MacClk for EMAC interface.

Register interface uses 5-bit address to access up to 32 registers, consisting of the network parameters, command register, and system parameters. The IP supports to set the different port number of the target device for transmitting data and receiving data at the same time. However, the same value of target port number is applied if the target device uses UDP100G IP. The parameters of UDP100G 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 512-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 of Tx data buffer and Rx data buffer inside the IP can be assigned by the user. Bigger buffer size is applied to store the data when the user logic is sometimes not ready for transferring data with UDP100G IP.

UDP100G IP is designed to connect with 100Gb Ethernet subsystem which uses 512-bit AXI4-ST to be user interface. Ethernet subsystem, provided by Xilinx, includes EMAC, PCS, and PMA function. The clock frequency of user interface of 100Gb Ethernet subsystem is equal to 322.265625 MHz.

There are two Async buffers inside UDP100G IP, so the user interface can run in independent clock. It is recommended to use 240 MHz or more to be user clock frequency. Using slower clock, Async buffer may be full and some packets may be lost.

Functional Description

As shown in Figure 2, UDP100G IP can be divided into three blocks - control block, transmit block, and receive block. The details of each block are described as follows.

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, as shown in Figure 6. The address for writing data and reading data is shared. 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'. **After all network parameters are assigned, the user sets '1' and then sets '0' to this register for loading parameter and starting system initialization. To update some parameters, user must set this register to '1' and '0' respectively again. The network parameters controlled by RST register are SML, SMH, DML, DMH, DIP, SIP, DPN, SPN, and SRV register.**
00001b	CMD	Wr	[0]	User command. Set '1' to start sending data. **Before setting this register to start new operation, the system must be in Idle state. User must confirm that busy is equal to '0' by reading IP output (Busy) or 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 IP output signal, Busy.
00010b	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.**
00011b	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.**
00100b	DIP	Wr /Rd	[31:0]	Define 32-bit target IP address. **To update this value, the IP must be reset by RST register.**
00101b	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.**
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. **To update this value, the IP must be reset by RST register.**
00111b	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.**
01000b	TDL	Wr	[31:0]	Define 32 lower bits (bit[31:0] of 48-bit tx data length in byte unit. The length must be aligned to 64 bytes (data bus size). Valid from 64-0xFFFF_FFFF_FFC0 (Bit[5:0] is ignored by the IP). The 16 upper bit (bit[47:32]) is assigned in TDH register (01101b). **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 and Busy is asserted to '1', the user can set the new value to 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.**
		Rd	[31:0]	32 lower bits 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 each Tx packet in byte unit, but the length must be aligned to 64-byte. Valid from 64-8960. Default value is 1472 byte which is the maximum size of non-jumbo frame that is aligned to 64-byte. Bit[5:0] 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/TDH register, the user does not need to set PKL register again when the next command uses the same packet length.**
01101b	TDH	Wr	[15:0]	Define 16 upper bits (bit[47:32] of 48-bit tx data length in byte unit), as described in TDL register.
		Rd		16 upper bits 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 de-asserted 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. **Note: In Server mode, when RST register changes from '1' to '0', the target device needs to resend ARP request for UDP100G IP completing the IP initialization.**
01111b	VER	Rd	[31:0]	IP version
10000b	DML	Wr /Rd	[31:0]	Define 32 lower bits of target MAC address (bit [31:0]) for this IP when SRV[1:0]="1x" (Fixed MAC mode). **To update this value, the IP must be reset by RST register.**
10001b	DMH	Wr /Rd	[15:0]	Define 16 upper bits of target MAC address (bit [47:32]) for this IP when SRV[1:0]="1x" (Fixed MAC mode). **To update this value, the IP must be reset by RST register.**

- **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 are read by UDP Stack and then set to Transmit block and Receive block for building and verifying the header of the packet. After finishing IP initialization, the IP changes to data transferring phase.

  UDP100G 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 is assigned independently. Busy signal is asserted to ‘1’ during sending data and de-asserted to ‘0’ after finishing sending data.

  To send the data, UDP payload data from the user is stored in Tx data buffer and Tx packet buffer. While the network parameters are applied to build UDP header by Packet Builder. After that, Transmit block merges UDP header with UDP payload data to construct the complete UDP packet for sending to the target device.

  The processor for sending data and receiving data is run independently. Busy signal is not asserted to ‘1’ when the data is received by Receive block.

Table 3: TxBuf/RxBufBitWidth Parameter description

Value of BitWidth	Buffer Size	TxBufBitWidth	RxBufBitWidth
9	32kByte	Valid	Valid
10	64kByte	Valid	Valid

Transmit Block

The data buffer size can be adjusted by parameter assignment for buffering data between user logic and UDP100G IP. Data from Tx data buffer is split into packet size and then stored in Tx packet buffer. UDP header is prepared from the network parameters in Reg module and then combined with UDP data from Tx data buffer to build the complete UDP packet. The transmitted data in Tx data buffer is flushed after finishing transferring to EMAC. After finishing Send data command, the user can change the packet size (PKL) and total data size (TDL/TDH) for the next Send data command.

- **Tx Data Buffer**

  This buffer size is set by “TxBufBitWidth” parameter of the IP which can be equal to 9 or 10. The parameter is the address size of 512-bit buffer, as shown in Table 3. This buffer stores the data from the user for preparing the transmit packet sent to the target device.

  If total data from user is more than the value of TDL register, the remained data will be stored in the buffer for the next command. All data in the buffer is flushed when the IP is reset. Please note that the IP cannot send the packet if the data stored in the buffer is less than the packet size. The IP must wait until the data from user is enough for creating one packet.

- **Tx Packet Buffer**

  This buffer stores at least one transmit packet before forwarding a packet to Async buffer.

- **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 a part of UDP header. After all UDP header is completely built, the header combining with the data from Tx packet buffer is transmitted to Async buffer (Tx).

- **Async Buffer (Tx)**

  Async buffer (Tx) is designed to forward the packet from Clk domain to MacClk domain. Also, it includes the logic to interface with 100G EMAC. It is recommended to use at least 240 MHz for user clock domain. If using too low frequency, the transmitted packet sent to EMAC will be always paused. Therefore, the transmit performance will be dropped.

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 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.

- **Async Buffer (Rx)**

  Async buffer (Rx) is designed to forward EMAC packet from MacClk domain to Clk domain. Also, the logic for interface with 100G EMAC is included. It is recommended to use at least 240 MHz for user clock domain. If using too low frequency, the buffer will be full and the received packet from EMAC will be lost.

- **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 not verified 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. UDP payload data is extracted 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 for 32 Kbyte or 10 for 64Kbyte buffer size. Rx data buffer is the buffer for transferring data between the user logic and UDP100G IP.

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 if the operation is finished without any error. The data path can connect with the FIFO for sending or receiving data with the IP.

100G Ethernet Subsystem

100G Ethernet Subsystem implements the MAC layer and Physical layer for 100Gb Ethernet. The user interface to connect with UDP100G IP is 512-bit AXI4 stream. Xilinx provides 100G Ethernet Subsystem (Ethernet MAC and Ethernet PCS/PMA) with many features, described in the following website.

https://www.xilinx.com/products/intellectual-property/cmac_usplus.html

Core I/O Signals

Descriptions of all parameters and I/O signals are provided in Table 4 - Table 6. The EMAC interface is 512-bit AXI4 stream interface.

Table 4: Core Parameters

Name	Value	Description
TxBufBitWidth	9-10	Tx data buffer size. The value is the address bus size of this buffer.
RxBufBitWidth	9-10	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
Common Interface Signal		
RstB	In	Reset IP core. Active Low.
Clk	In	User clock for running UDP100G IP. Clock frequency should be at least 240 MHz to achieve the best performance. *Note: 240 MHz is the clock frequency that can compensate overhead time inside UDP100G IP for transferring 1472-byte packet in both directions. To support smaller packet size without reducing transmit performance or without lost packet in receive direction, please contact our sales support to recommend the clock frequency.*
User Interface		
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'- IP is Idle, '1'-IP is busy (when running initialization or Send command).
IntOut	Out	IP Interrupt. Asserted to high for 1 clock cycle when timeout is detected or received packet has an error. More details of Interrupt status are monitored 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[511:0]	In	Write data to Tx data buffer. Valid when UDPTxFfWrEn='1'.
Rx Data Buffer Interface		
UDPRxFfRdCnt[9:0]	Out	Data counter of Rx data buffer to show the number of received data in 512-bit unit.
UDPRxFfLastRdCnt[5:0]	Out	Remaining byte of the last data in Rx data buffer when total number of received data in the buffer is not aligned to 64-byte unit. User cannot read the data until all 64-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[511:0]	Out	Data output from Rx data buffer. Valid in the next clock cycle after UDPRxFfRdEn is asserted to '1'.

Table 6: Tx EMAC Signals (Synchronous to MacClk)

Signal	Dir	Description
MacClk	In	Clock source from EMAC core which is equal to 322.265625MHz for 100Gb Ethernet.
tx_axis_tdata[511:0]	Out	Transmitted data. Valid when tx_axis_tvalid='1'.
tx_axis_tkeep[63:0]	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 equal to '0'.
tx_axis_tready	In	Handshaking signal. Asserted to '1' when tx_axis_tdata has been accepted.
rx_axis_tdata[511:0]	In	Received data. Valid when rx_axis_tvalid='1'
rx_axis_tvalid	In	Valid signal of received data.
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. Typcially, rx_axis_tready is always asserted to '1'. If Clk frequency is too low until free space of Async buffer is not enough, rx_axis_tready will be de-asserted to '0' after receiving end of packet. The signal is re-asserted to '1' when free space of Async buffer is enough for storing one packet.

Timing Diagram

IP Initialization

The initialization process begins after user changes RST register from ‘1’ to ‘0’. UDP100G 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.

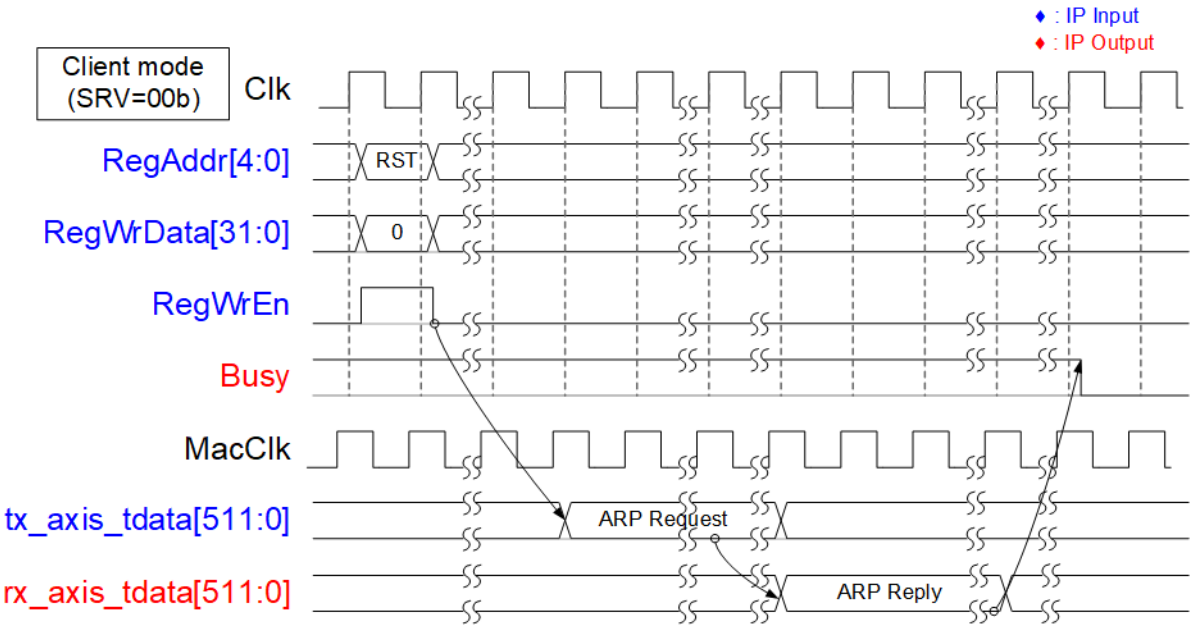

Figure 3: IP Initialization in Client mode

As shown in Figure 3, in Client mode UDP100G 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’.

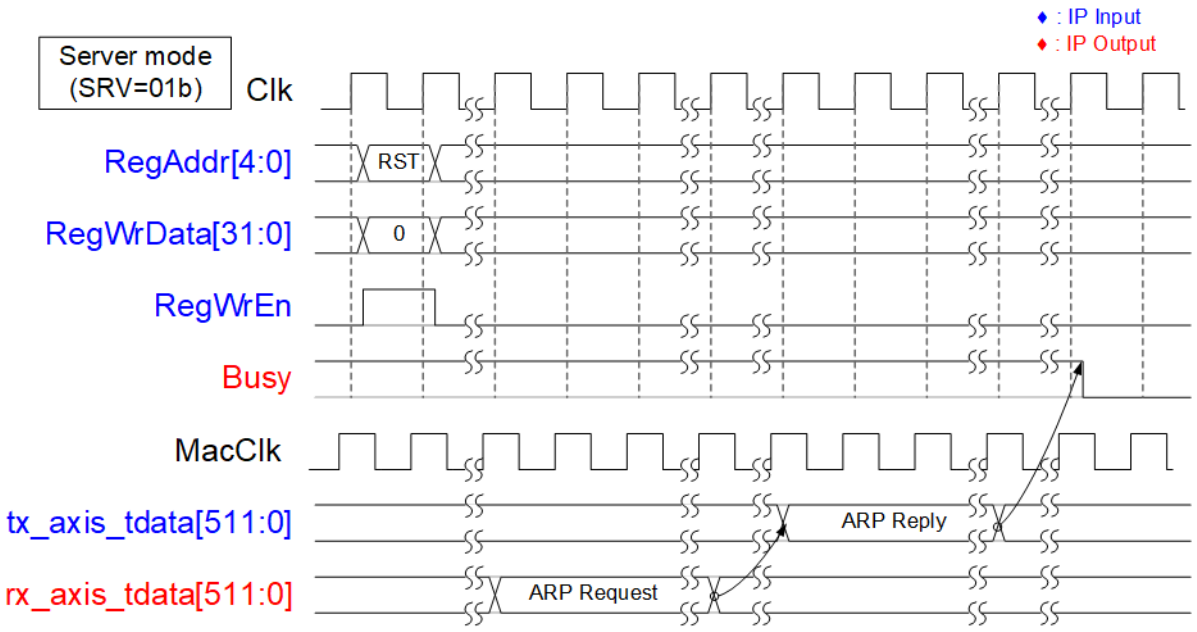

Figure 4: IP Initialization in Server mode

As shown in Figure 4, after finishing reset process in Server mode, UDP100G IP waits until ARP request sent by the target device. After that, UDP100G IP returns ARP reply to the target. Target MAC address is extracted from ARP request packet. Finally, Busy signal is de-asserted to '0'.

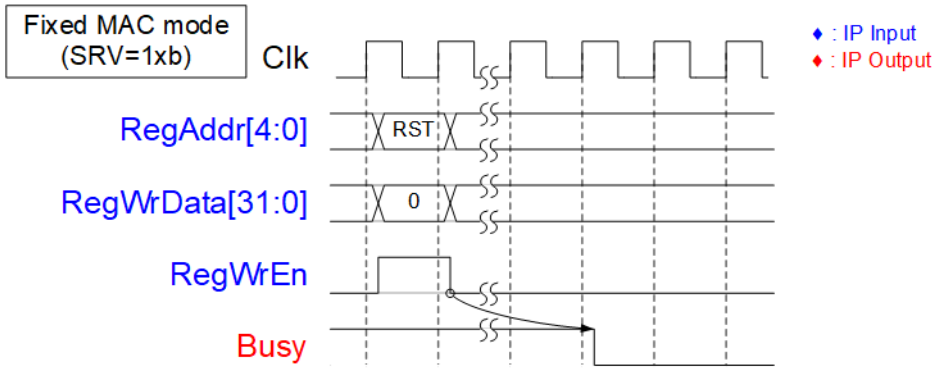

Figure 5: IP Initialization in Fixed mode

As shown in Figure 5, after finishing reset process in Fixed MAC mode, UDP100G 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 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 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.

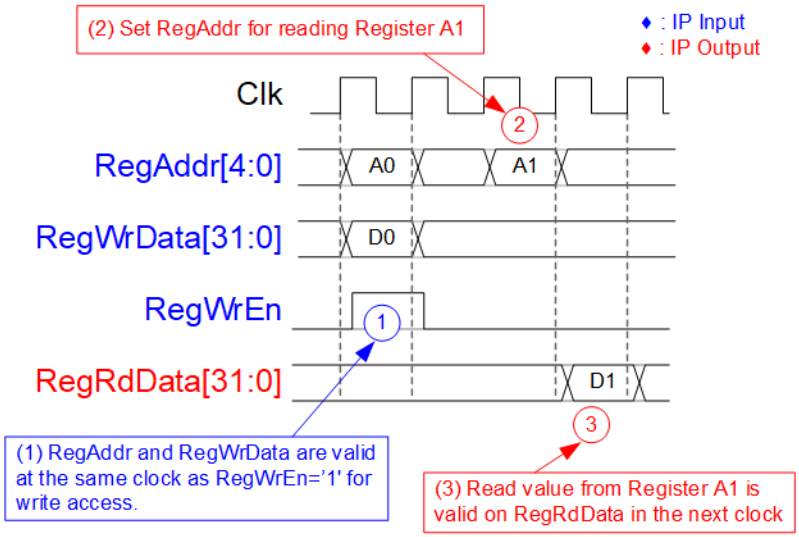

Figure 6: Register interface timing diagram

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.

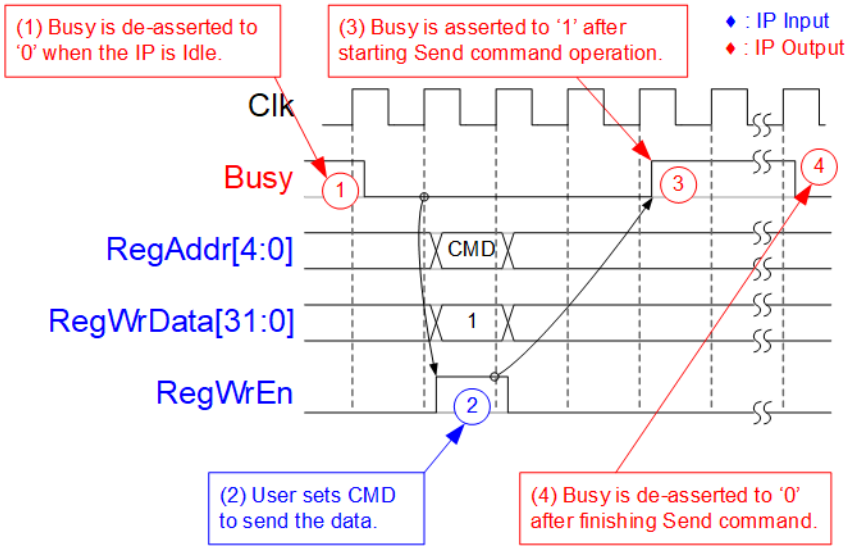

Figure 7: CMD register timing diagram

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.

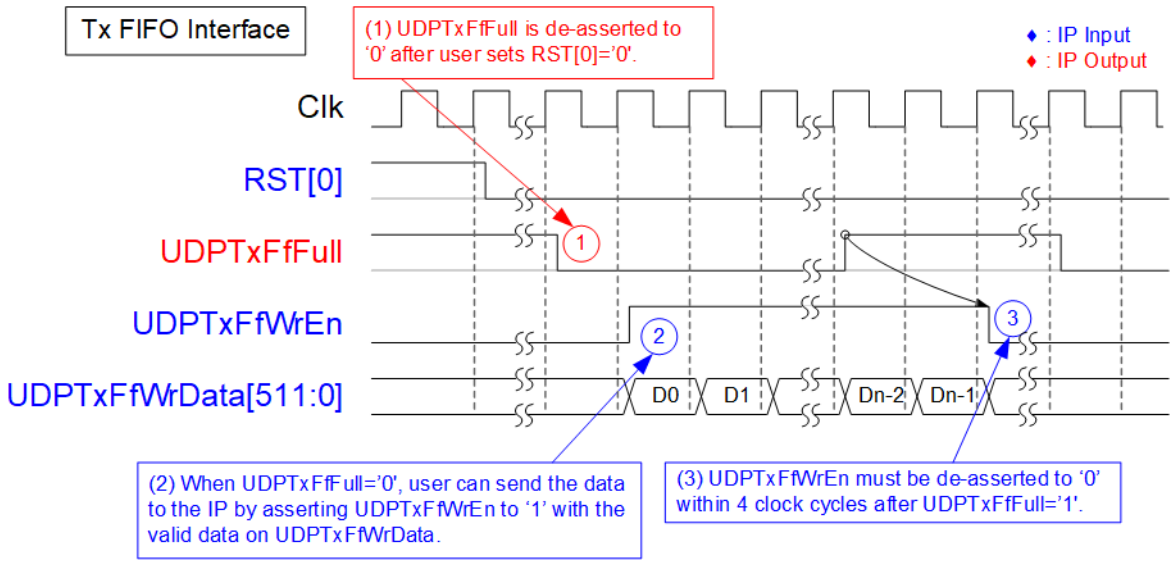

Figure 8: Tx FIFO interface timing diagram

(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'. After that, the user can write the data to the IP.

(2) To write the data, UDPTxFfWrEn is asserted to '1'. At the same clock, UDPTxFfWrData is valid to send the data.

(3) If UDPTxFfFull is asserted to '1', the user must pause sending data by de-asserting UDPTxFfWrEn 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 and then asserts read enable signal to read the data, similar to read interface of general FIFO, as shown in Figure 9.

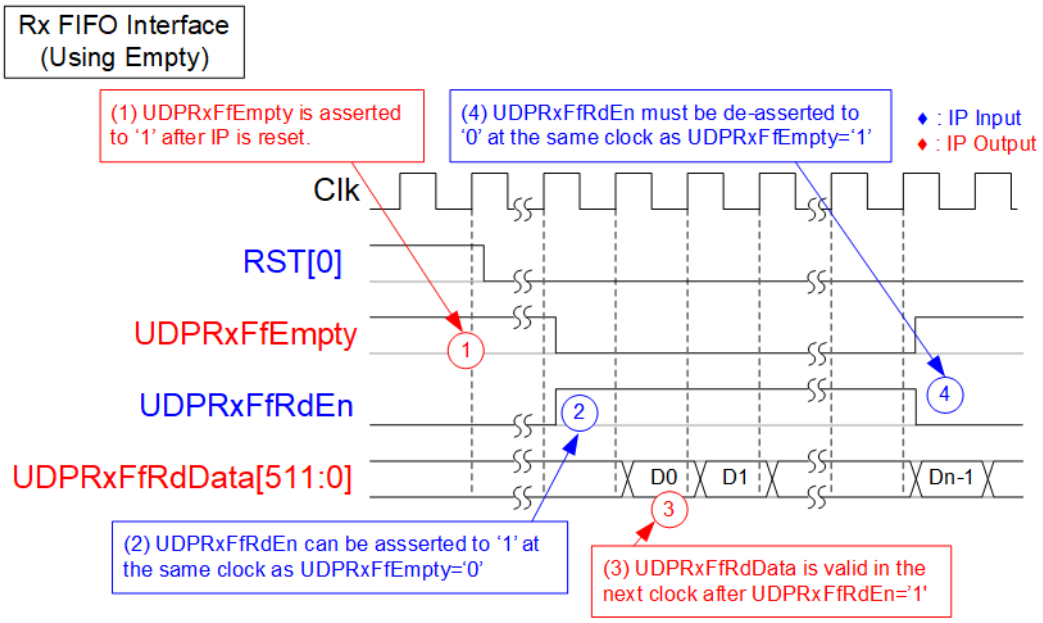

Figure 9: Rx FIFO interface timing diagram by using Empty flag

(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.

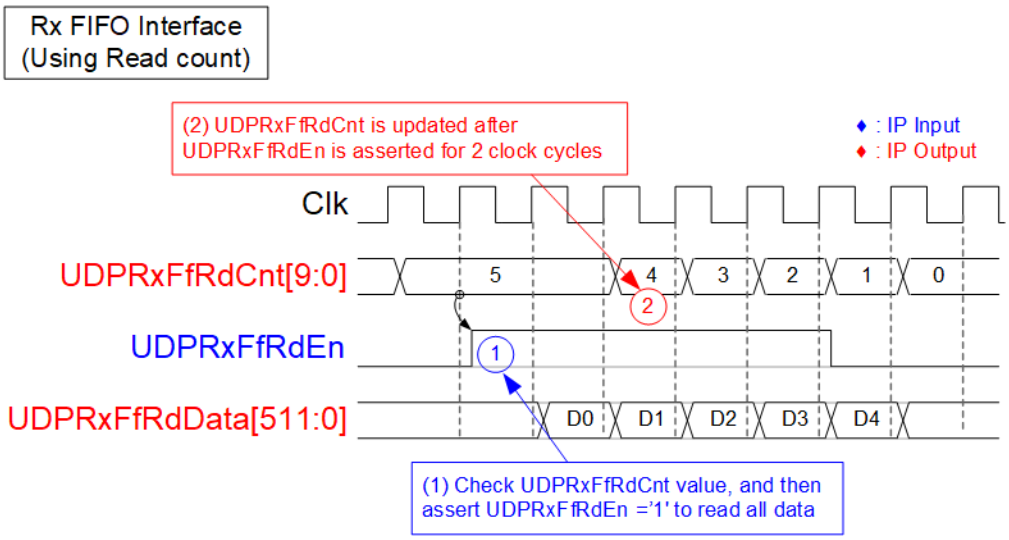

Figure 10: Rx FIFO interface timing diagram by using read counter

If user logic reads data as burst mode, UDP100G IP has read counter signal to show the total number of data stored in Rx FIFO interface as 512-bit unit. For example, Figure 10 shows five data available in Rx data buffer. Therefore, user can assert UDPRxFfRdEn to '1' for 5 clock cycles to read all data from Rx data buffer. The latency time to update read counter (UDPRxFfRdCnt) after asserting read enable (UDPRxFfRdEn) is 2 clock cycles.

EMAC Interface

EMAC interface of UDP100G IP is designed by using 512-bit AXI4-stream interface. The details of EMAC interface for Transmit and Receive direction are shown in Figure 11 - Figure 13.

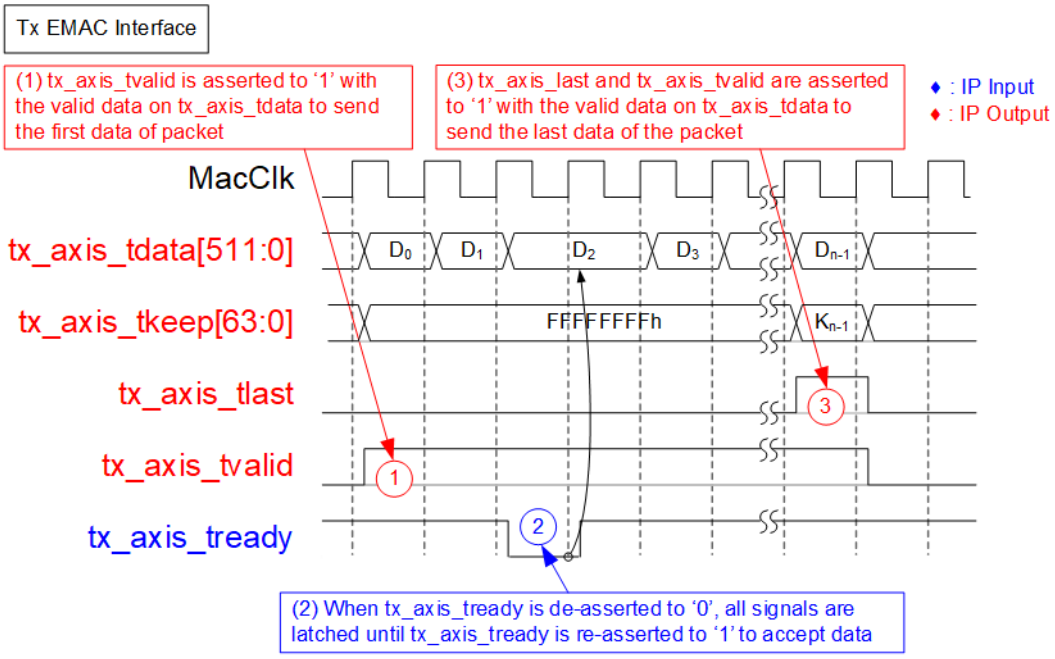

Figure 11: Transmit EMAC interface timing diagram

(1) To send the first data of a packet, UDP100G IP asserts tx_axis_tvalid with the valid data on tx_axis_tdata.

(2) During transmitting the packet, EMAC may be not ready to receive the data and de-asserts tx_axis_tready to '0'. UDP100G IP holds the same value of all signals to wait until tx_axis_tready is re-asserted to '1'. After that, the next data is transmitted.

(3) To send the last data of the packet, tx_axis_tlast and tx_axis_tvalid are asserted to '1' with the valid data on tx_axis_tdata. According to EMAC specification, tx_axis_tvalid must be always asserted to '1' during a packet transmission. It cannot be de-asserted to '0' before end of packet is transmitted.

To receive the packet from EMAC, if the user clock frequency is too low, it will be possible that rx_axis_tready which is output from UDP100G IP is de-asserted to '0' after receiving end of packet. rx_axis_tready is de-asserted to '0' when Async buffer (Rx) has too less free space, as shown in Figure 13. After the free space is enough, rx_axis_tready is re-asserted to '1'. Therefore, if the user clock frequency is high, rx_axis_tready is always asserted to '1' for receiving the data from EMAC, as shown in Figure 12.

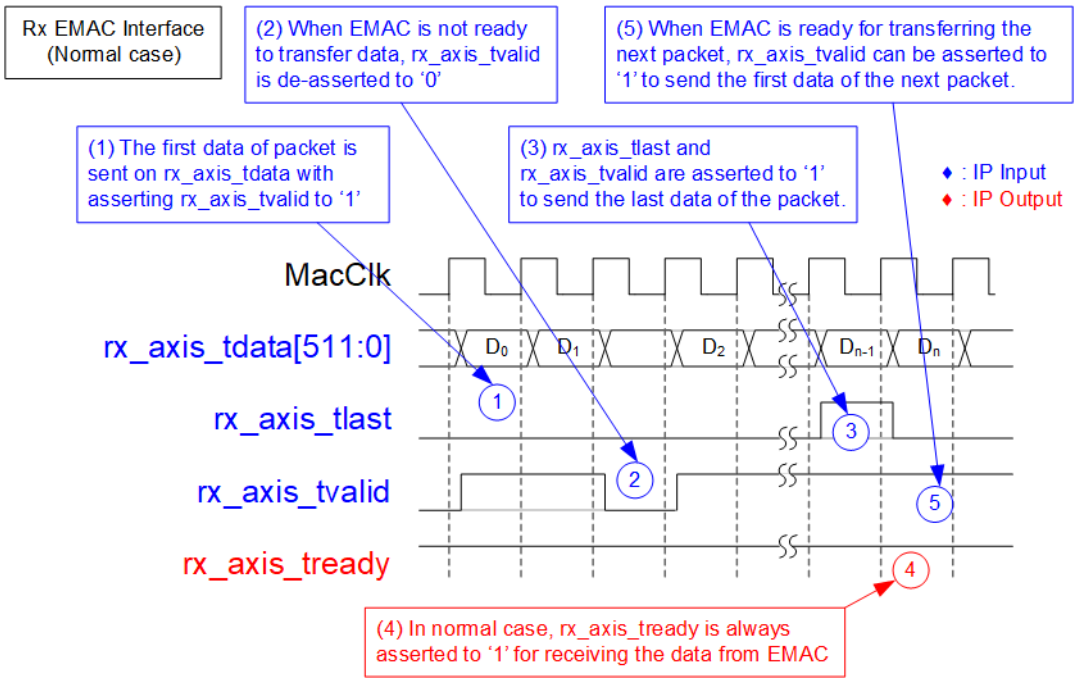

Figure 12: Receive EMAC interface timing diagram (Normal)

(1) To send the first data of a packet, EMAC asserts rx_axis_tvalid to '1' with the valid data on rx_axis_tdata. After that, EMAC can transmit the data to UDP100G IP continuously.
(2) When EMAC is not ready to send the data, rx_axis_tvalid is de-asserted to '0' to pause data transmission.
(3) To send the last data of the packet, EMAC asserts rx_axis_tlast and rx_axis_tvalid to '1' with the valid data on rx_axis_tdata.
(4) In normal case, rx_axis_tready is always asserted to '1' for receiving the next packet after receiving the last data.
(5) If EMAC is ready by asserting rx_axis_tready, rx_axis_tvalid can be asserted to '1' to send the next packet.

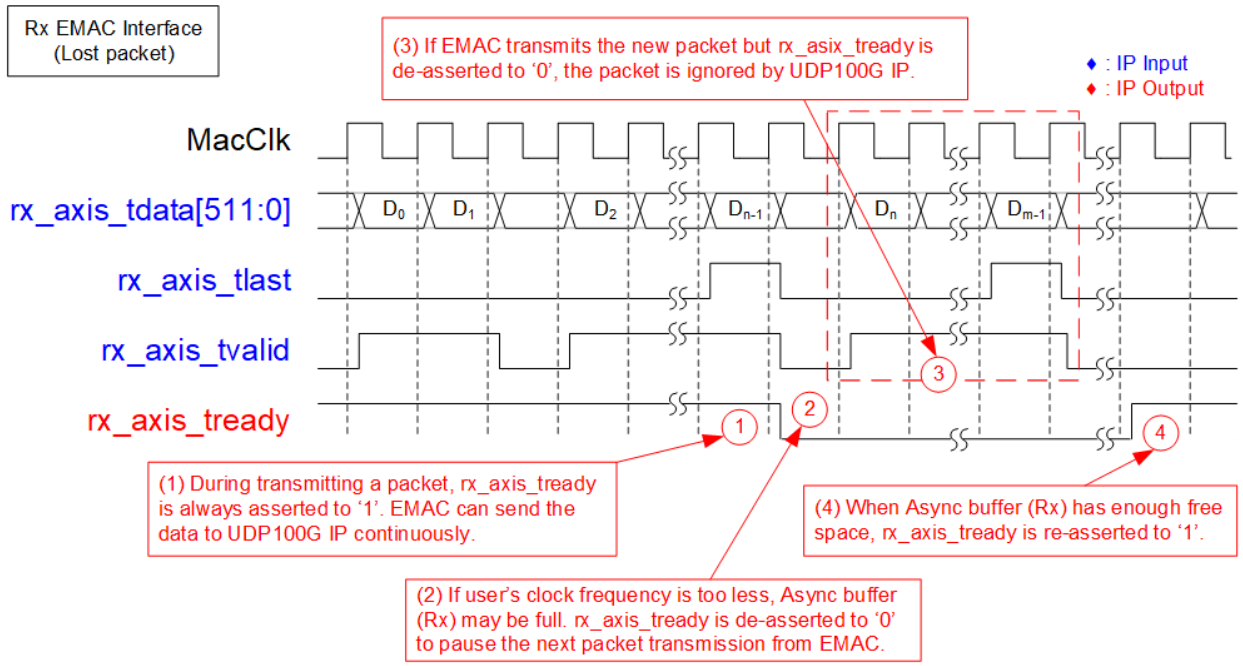

Figure 13: Receive EMAC interface timing diagram (Data lost)

(1) "rx_axis_tready" is not de-asserted to '0' before receiving end of packet. Therefore, the packet will be received completely when rx_axis_tready is asserted to '1'.

(2) If clock frequency of user's clock (Clk signal) is too low until Async buffer (Rx) inside UDP100G IP does not have enough free space for storing the next packet, rx_axis_tready is de-asserted to '0' after receiving end-of-packet (rx_axis_tlast='1' and rx_axis_tvalid='1').

(3) During de-asserting rx_axis_tready to '0', if EMAC sends the new packet by asserting rx_axis_tvalid to '1', UDP100G IP will ignore the packet.

(4) When Async buffer (Rx) has enough free space, rx_axis_tready is re-asserted to '1'. Now UDP100G IP is ready to receive the new packet from EMAC.

Example usage

Client mode (SRV[1:0]=”00”)

The example steps to set register for transferring data in Client mode are 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) When the data is transferred in each direction, the operation of the IP is described as follows.
   a. For data transmission,
      i. Set TDL/TDH register (total transmit length) and PKL register (packet size).
      ii. Set CMD register = ‘1’ (Send data) to start data transmission.
      Note: The user sends the data to UDP100G IP via TxFIFO interface before or after setting CMD register.
      iii. When the command is finished, busy flag is de-asserted to ’0’.
      iv. The user can set the new value to TDL/TDH/PKL register and return to step i.
   b. For data reception, user monitors RxFIFO status and reads data until RxFIFO is empty.

Server mode (SRV[1:0]=”01”)

Comparing to Client mode which MAC address is decoded from ARP reply packet after UDP100G 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 steps for running in Server mode are 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 for running in Fixed MAC mode are shown as follows

1) Set RST register=’1’ to reset the IP.
2) Set SML/SMH for MAC address of UDP100G 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 UDP100G IP Core functionality was verified by simulation and also proved on real board design by using KCU116 evaluation board, Alveo U250 Accelerator card, 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	4-Aug-21	New release
1.1	24-Nov-21	Correct feature