UDP10G-IP Core

March 8, 2023 | Product Specification | Rev1.5

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

Core Facts

Provided with Core	
Documentation	User Guide, Design Guide
Design File Formats	Encrypted File
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on KCU105, KCU116, ZCU102, ZCU111
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 multiple sessions by using multiple UDP10G IPs
- Support Jumbo frame
- Transmit packet size aligned to 64-bit, transmitted data bus size
- Total receive data size aligned to 64-bit, received data bus size
- Transmit/Receive buffer sizes, adjustable to balance resource usage and performance
- Simple data interface by standard FIFO interface at 64-bit data bus
- Simple control interface by 32-bit single-port RAM interface
- 64-bit AXI4 stream interface with 10G/25G Ethernet MAC
- One clock domain interface, 156.25 MHz for 10G Ethernet
- Reference designs available on KCU105, KCU116, ZCU102, ZCU111 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 for 7-series device

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

Notes:
1) Actual logic resource dependent on percentage of unrelated logic
2) Block memory resources are based on 64k Tx data buffer size, 16k Tx packet buffer size, and 64k Rx data buffer size.

Table 2: Example Implementation Statistics for Ultrascale device

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB	IOB[2]	BRAMTile [1]	URAM	Design Tools
Kintex-Ultrascale	XCKU040FFVA1156-2E	156.25	1871	2223	433	-	34.5	-	Vivado2017.4
Kintex-Ultrascale+	XCKU5P-FFVB676-2-E	156.25	1866	2101	439	-	-	5	Vivado2019.1
Zynq-Ultrascale+	XCZU9EG-FFVB1156-2	156.25	1871	2220	433	-	34.5	-	Vivado2017.4
Zynq-Ultrascale+	XCZU28DR-FFVG1517-2-E	156.25	1866	2101	408	-	-	5	Vivado2019.1

Applications

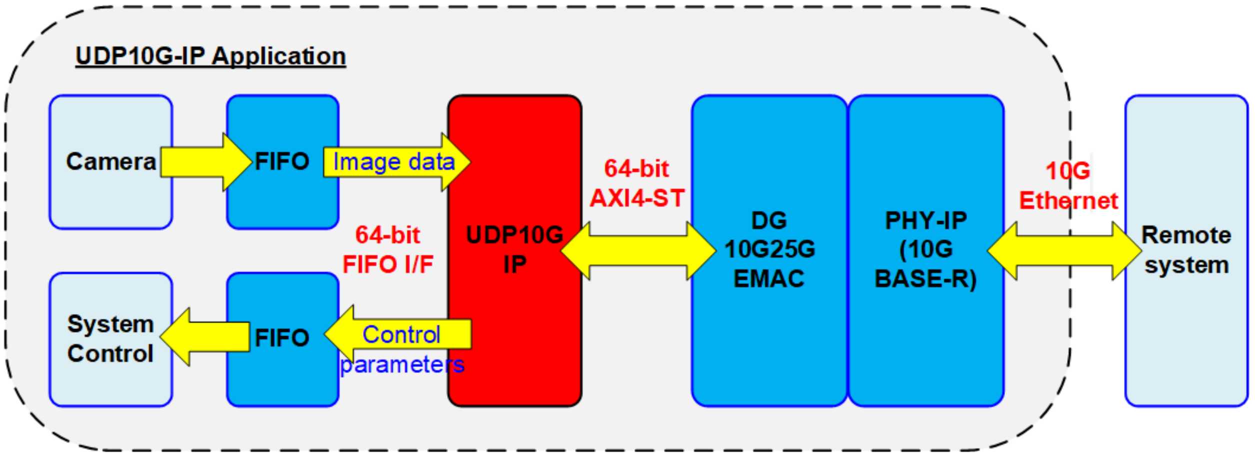

Figure 1: UDP10G IP Application

The 10G Ethernet is a communication channel for transferring data with remote controlling system. When combined with the UDP/IP protocol, this system can transfer data in both Tx and Rx direction at high-speed rate via 10G Ethernet network. Additionally, UDP/IP supports data transfer for both one-by-one and broadcast systems. UDP10G IP is an integrated IP that enables data transfer via 10G Ethernet without requiring the use of CPU or external memory. It is suitable for applications that require high-speed data transfer for both Tx and Rx directions, such as video data streaming and monitoring system using FPGA solutions.

Figure 1 shows an example application of video camera system, where video raw data from the camera is stored in a FIFO and forwarded to a remote system via 10G Ethernet using the UDP10G IP. The UDP10G IP supports full-duplex transfer simultaneously using different port numbers, allowing the Remote system to update the parameters for real-time controlling via 10G Ethernet while receiving the data from UDP10G IP via 10G Ethernet network.

UDP10G IP is designed to transfer data at the high speed, resulting a latency time that is increased by the internal pipeline register and buffer. For low-latency applications such as FinTech, we recommend the user of our low-latency IP. More details about this IP can be found on our website.

https://dgway.com/Lowlatency-IP_X_E.html

General Description

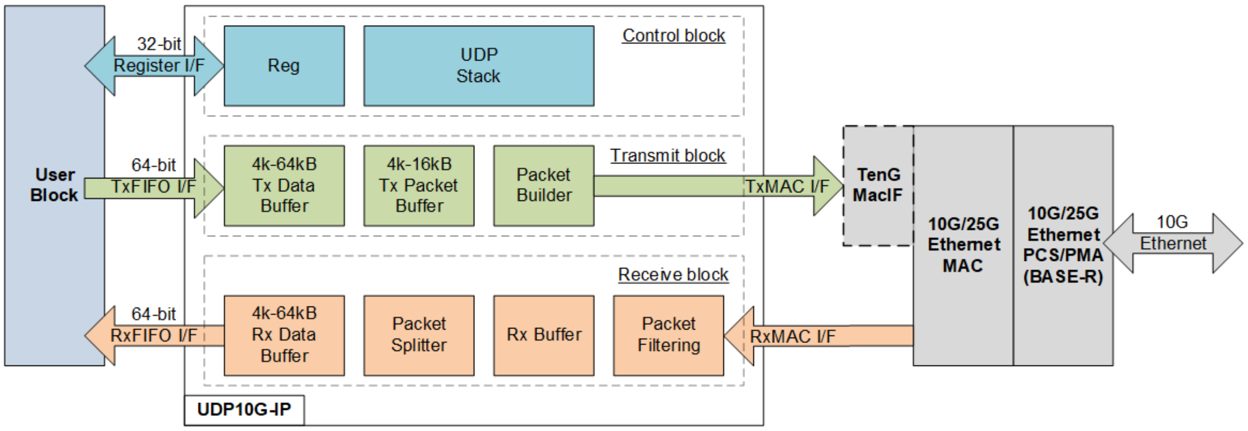

Figure 2: UDP10G IP Block Diagram

The UDP10G IP core is a hardwired logic implementation of the UDP/IP stack that connects with the EMAC IP and PCS/PMA (BASE-R) module to form the lower layer hardware. The user interface of UDP10G IP consists of two interfaces, i.e., the Register interface for control signals and the FIFO interface for data signals.

The Register interface uses a 4-bit address for accessing up to 16 registers, including network parameters, command registers, and system parameters. Bi-directional data transfer is achieved using two sessions for each UDP10G IP, one session assigned for each direction. The network parameters for both sessions must be the same, except the port number on the target device. Network parameters must be set before de-asserting the reset signal to execute IP initialization. After finishing the reset operation and parameter initialization, the IP is ready to transfer data with the target device. Network parameters cannot be changed without a reset process. UDP10G IP has two initialization modes for obtaining the MAC address of the target device. Further details of each mode can be found in the IP Initialization topic.

To send UDP payload data, the user must set the total transfer size, packet size, and send command to the IP. The UDP payload data is transferred via TxFIFO interface which is 64-bit data size. There is no byte enable in the TxFIFO interface, so the transmitted data from the user must be aligned 64-bit. Additionally, the packet length and total amount of transmitted data must also be aligned to 64-bit.

On the other hand, when the UDP packet is received from the target, the UDP payload data is extracted and stored in the Rx data buffer. The user logic monitors FIFO status and detect the amount of received data and then asserts read enable to read the data via RxFIFO interface. The received data on the Rx FIFO I/F can be read when at least one 64-bit data is available in Rx data buffer. If total amount of received data is not aligned to 64-bit, the user cannot read the last data. The user must wait until the next data is received to fill the remaining byte of 64-bit data for reading the Rx data buffer.

To meet the user system requirements, which may be sensitive to the memory resources or the performance, the buffer size inside the IP can be assigned by the user. There are three buffers whose sizes can be adjusted, i.e., Tx data buffer, Tx packet buffer, and Rx data buffer. If the Tx buffer size is too small, the UDP10G IP may transmit all data to the target before the user fills the buffer with new data, causing the UDP10G IP to pause transmission and wait for new data from the user. On the other hand, if the Rx buffer size is too small, the buffer may become full if the user pauses reading data from the UDP10G IP for a long time, causing the UDP10G IP to reject the new received packet from the target.

UDP10G IP uses a 64-bit AXI4-Stream to connect with the Ethernet MAC. When Ethernet MAC is implemented by Xilinx 10G/25G Ethernet Subsystem, the adapter logic (TenGMacIF) must be designed to act as the interface module between UDP10G IP and Ethernet MAC. However, DG 10G25GEMAC can connect to UDP10G IP directly without the adapter logic. More information about the hardware within the IP is described in the following section. More information about the hardware within the IP is described in the following section.

Functional Description

A shown in Figure 2, UDP10G IP core can be divided into three parts, i.e., 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 that consists of 4-bit address signals and 32-bit data signals. The timing diagram of the Register interface is similar to a single-port RAM interface, as shown in Figure 5. The write and read addresses are the same signal. Table 3 provides a description of each register.

Table 3: Register map Definition

RegAddr [3:0]	Reg Name	Dir	Bit	Description
0000b	RST	Wr /Rd	[0]	Reset IP. 0b: No reset, 1b: Reset. Default value is 1b. **Once the network parameters have been assigned, the user can execute system initialization by setting this register to 1b and then 0b. This action loads the parameters into the IP. If the user needs to update certain parameters, this process must be repeated by setting this register to 1b and then 0b again. The RST register controls the following network parameters: SML, SMH, DIP, SIP, DPN, SPN, and SRV.**
0001b	CMD	Wr	[0]	User command. Set 1b to start sending data. **In order to start a new operation by setting this register, the system must first be in the Idle state. To confirm that the system is not busy, the user should read Busy signal or bit[0] of CMD register, which should be equal to 0b.**
		Rd	[0]	System busy flag. 0b: Idle, 1b: 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 using for the UDP10G IP to send data. [31:16]-Define 16-bit target port number using for the UDP10G IP to receive 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.**

RegAddr [3:0]	Reg Name	Dir	Bit	Description
1000b	TDL	Wr	[31:0]	Total Tx data length in byte unit. The value must be aligned to 8-byte because bit[2:0] are not used. Valid range is 8-0xFFFFFFF8. **The user must first set this register before setting CMD register=1b. When the IP executes the command and asserts Busy to 1b, the system will read this register, allowing the user to subsequently set the TDL register for the next command. If the same TDL is used in the subsequent command, the user is not required to set TDL again.**
		Rd		Remaining transfer length in byte unit which does not transmit.
1001b	TMO	Wr	[31:0]	Define timeout value for awaiting the return of an ARP reply packet after sending an ARP request packet. The counter operates at 156.25 MHz, resulting in a timer unit equal to 6.4 ns. If the ARP reply packet is not received in the specified time, IntOut will be asserted to 1b. It is recommended that the TMO is set to a value greater than 0x6000.
		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 MacRxUser shows error status.
1010b	PKL	Wr /Rd	[15:0]	UDP data length of each Tx packet in byte unit. The value must be aligned to 8-byte because bit[2:0] are not used. Valid from 8-8968. Default value is 1472 byte, which is the maximum size of non-jumbo frame and aligned to 8-byte. **During running Send data command (Busy=1b), the user must not set this register. Similar to TDL register, the user does not need to set PKL register again if the next command uses the same packet length.**
1110b	SRV	Wr/ Rd	[0]	0b: Client mode (default). When the RST register changes from 1b to 0b, the IP sends an ARP request to obtain the Target MAC address from the ARP reply returned by the target device. The IP busy signal is de-asserted to 0b after receiving the ARP reply. 1b: Server mode. When RST register changes from 1b to 0b, the IP waits for an ARP request from the target to obtain Target MAC address. After receiving the ARP request, the IP generates an ARP reply and then de-asserts the IP busy signal to 0b. **Note: In Server mode, when RST register changes from 1b to 0b, the target device needs to resend the ARP request for the UDP10G IP to complete the IP initialization.**
1111b	VER	Rd	[31:0]	IP version

- **UDP Stack**

  The UDP stack is responsible for controlling the modules involved in interfacing with the user and transferring packets via EMAC. The IP operation involves two phases - IP initialization and data transfer. After the RST register transitions from 1b to 0b, the initialization phase begins. The SRV[0] is used to set the initialization mode, which can be Client mode or Server mode. The UDP stack reads the parameters from the Reg module and sets them in the Transmit and Receive blocks for packet transfer with the target device. Once initialization is complete, the IP enters the data transfer phase.

  The UDP10G IP allows for simultaneous data transfer in both Tx and Rx directions with the target device. During sending data, the Busy signal is set to 1b. It is de-asserted to 0b once the sending process is completed. To transmit data, the user data is stored in the Tx data and Tx packet buffers. The Packet Builder uses the user-defined network parameters to build UDP header, and then the data of Tx data buffer is appended to the UDP packet. The Transmit block then sends the UDP packet to the target device via Ethernet MAC.

  When the Receive block receives data, the Busy signal is not asserted. The extracted UDP data from the received packet is forwarded to the user without generating any packet to be returned to the target device. The Tx and Rx of UDP/IP are processed individually to achieve optimal performance in both directions.

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

Transmit block contains two buffers - Tx data buffer and Tx packet buffer – whose sizes can be adjusted through parameter assignment. The minimum size of these buffer is limited by the transmit packet size set by the PKL register. Data from the Tx data buffer is split into packets based on the packet size and stored in the Tx packet buffer. UDP header is constructed using the network parameters from the Reg module and then combined with the UDP data from the Tx packet buffer to form a complete UDP packet. The data in the Tx data buffer is flushed after the packet is transmitted to EMAC successfully. Once the Send data command is completed, the user can initiate the next command.

- **Tx Data Buffer**

  The size of this buffer is determined by the "TxBufBitWidth" parameter of the IP, with valid value ranging from 9-13, which corresponds to the address size of a 64-bit buffer as shown in Table 4. To ensure sustainable transmission of data to EMAC, the buffer size should be at least twice the size of the Tx Packet Size set in the PKL register. By filling the buffer with data that is at least two times of the PKL value, the UDP10G IP will have a sufficient amount of data to transmit. This results in the best transmit performance on 10G Ethernet. Increasing the size of the buffer increases the possibility that at least two packet data will be available for UDP10G IP use.

- **Tx Packet Buffer**

  The size of the buffer size is determined by the "TxPacBitWidth" parameter of the IP. Its valid range is 9-11, and the details of the parameter are shown in Table 4. To store at least one transmit packet, the buffer size must be larger than the Tx packet size set by the PKL register. Note that the maximum value of the PKL register is equal to the Tx Packet Buffer size (in bytes) minus 32. Configuring the buffer size to be greater than two times of PKL value does not provide any additional benefits.

- **Packet Builder**

  The UDP packet is comprised of a header and data. The Packet builder first receives network parameters from the Reg module and uses them to construct the UDP header. The UDP and IP checksum are also calculated for the header. Once the header is fully constructed, it is combined with the data from the Tx packet buffer and then transmitted to the EMAC.

Receive Block

The Receive block contains the Rx data buffer, which stores the data received from the target device. The received data is stored in the buffer when the header in the packet matches the expected value, set by the network parameters inside the Reg module, and when the IP and UDP checksum are correct. If any of these conditions are not met, the received packet is rejected. Increasing the size of the Rx data buffer enhances the chances of avoiding lost packets due to user logic pausing for too long while reading data.

- **Rx Buffer**

  This is temporary buffer that is used to hold incoming packets from EMAC in cases where the previous packet has not yet been completely processed.

- **Packet Filtering**

  This module is responsible for verifying the header of the Rx packet to determine its validity. The packet will be valid if all following conditions are met.

(1) The network parameters must match the values set in the Reg module, such as the MAC address, IP address, and Port number.
(2) The packet must either be an ARP packet or a UDP/IPv4 packet.
(3) The IP header length must be valid, with the IP length being equal to 20 bytes.
(4) The IP data length matches to the UDP data length.
(5) Both the IP checksum and UDP checksum must be correct.
Note: UDP checksum is not verified for fragment packet.
(6) In case of fragment packet, the order of received packet must be correct. The packet is rejected when the fragment offset is skipped value.

- **Packet Splitter**

  The purpose of this module is to extract UDP data from incoming packets and store it in the Rx data buffer, after removing the packet header.

- **Rx Data Buffer**

  The size of the Rx data buffer is determined by the “RxBufBitWidth” parameter of the IP and can range from 9 – 13 (4KB to 64KB). It is recommended to set the buffer size to be equal to or greater than two times of the data size in the received packet. To ensure that data is not lost, user logic must read the data from the buffer to keep the free space size of the buffer to be greater than the data size in the received packet. Increasing the size of the buffer increases the possibility of having enough free space to store new data from the target device.

User Block

The core engine of the user module can be designed by state machine to set the command and the parameters through the Register interface. Additionally, the status can be monitored to ensure that the operation has been completed without any errors. The data path can also be connected to the FIFO for sending or receiving data with the IP.

10G Ethernet System

The UDP10G IP operates with Ethernet MAC IP via 64-bit AXI4 stream. There are various solutions for Ethernet MAC IP core. Two solutions are described in this section – Ethernet MAC IP from Design Gateway and Ethernet MAC IP from Xilinx.

DG 10G25GEMAC IP Core

The first solution uses DG 10G25GEMAC IP and Xilinx PCS/PMA module. This solution is designed to optimize IP resource and has minimal latency time. The user interface of DG 10G25GEMAC IP can directly connect with UDP10G IP, and the PCS/PMA module can be created using the Xilinx tool's IP wizard. The Xilinx IP core is no charge IP core. More information about DG 10G25GEMAC IP Core is available at

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

Additionally, more details about the 10G/25G Ethernet PCS/PMA (BASE-R) are available at.

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

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

Xilinx 10G/25G Ethernet Subsystem

The second solution uses Xilinx IP core which may be the 10G or 10G/25G Ethernet Subsystem, which includes both Ethernet MAC and PCS/PMA. When using 10G/25G Ethernet Subsystem, a small FIFO adapter logic is required to connect UDP10G IP, TenGMac IF in Figure 2. More information of these IPs are available at

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

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 I/O signals 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 referred to address bus size of this buffer.
TxPacBitWidth	9-11	Setting Tx Packet buffer size. The value is referred to address bus size of this buffer.
RxBufBitWidth	9-13	Setting Rx Data buffer size. The value is referred to address bus size of this buffer.

Table 6: Core I/O Signals

Signal	Dir	Description
Common Interface Signal		
RstB	In	Reset IP core. Active Low.
Clk	In	156.25 MHz fixed clock frequency input for user interface and EMAC interface.
User Interface		
RegAddr[3:0]	In	Register address bus. Valid when RegWrEn=1b in Write process.
RegWrData[31:0]	In	Register write data bus. Valid when RegWrEn=1b.
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 (0b-Idle, 1b-IP is busy when executing the initialization or send data process)
IntOut	Out	Timer interrupt. Asserted 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 1b when Tx data buffer is full. User needs to stop writing data within 4 clock cycles after this flag is asserted to 1b.
UDPTxFfWrCnt[12:0]	Out	Data counter in 64-bit unit of Tx data buffer to show the amount of data in Tx data buffer.
UDPTxFfWrEn	In	Write enable to Tx data buffer. Asserted to 1b to write data to Tx data buffer.
UDPTxFfWrData[63:0]	In	Write data to Tx data buffer. Valid when UDPTxFfWrEn=1b.
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 amount of 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 1b when Rx data buffer is empty. User needs to stop reading data immediately when this signal is asserted to 1b.
UDPRxFfRdEn	In	Asserted to 1b 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 1b.

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

Timing Diagram

IP Initialization

After the RST register value is changed from 1b to 0b, the initialization of UDP10G IP is initialized. Two modes can be executed, Client mode (SRV[0]=0b) or Server mode (SRV[0]=1b). The information on each mode is presented in the timing diagram below.

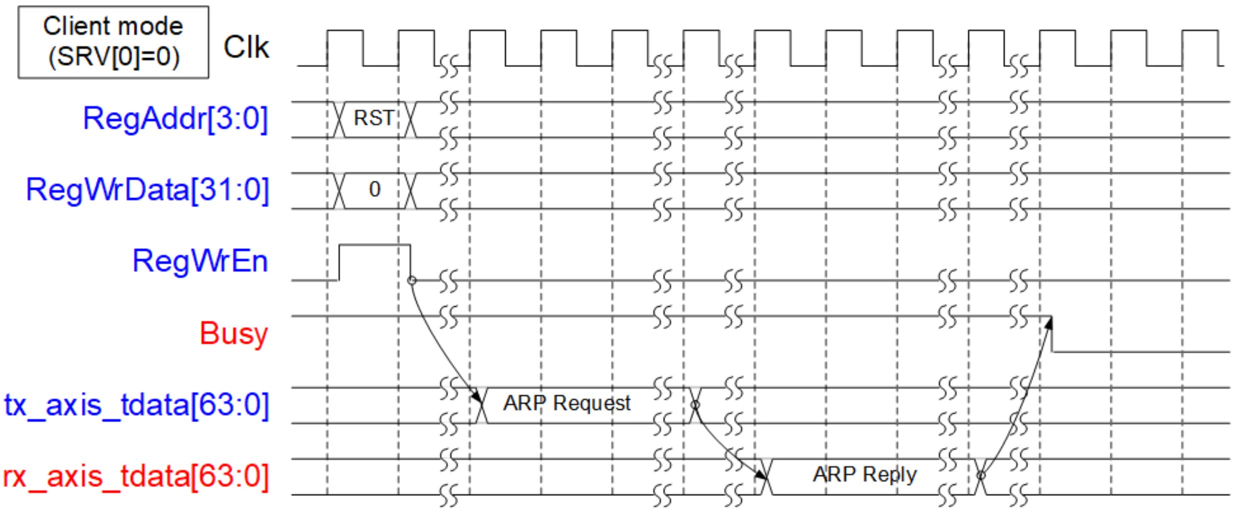

Figure 3: IP Initialization in Client mode

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

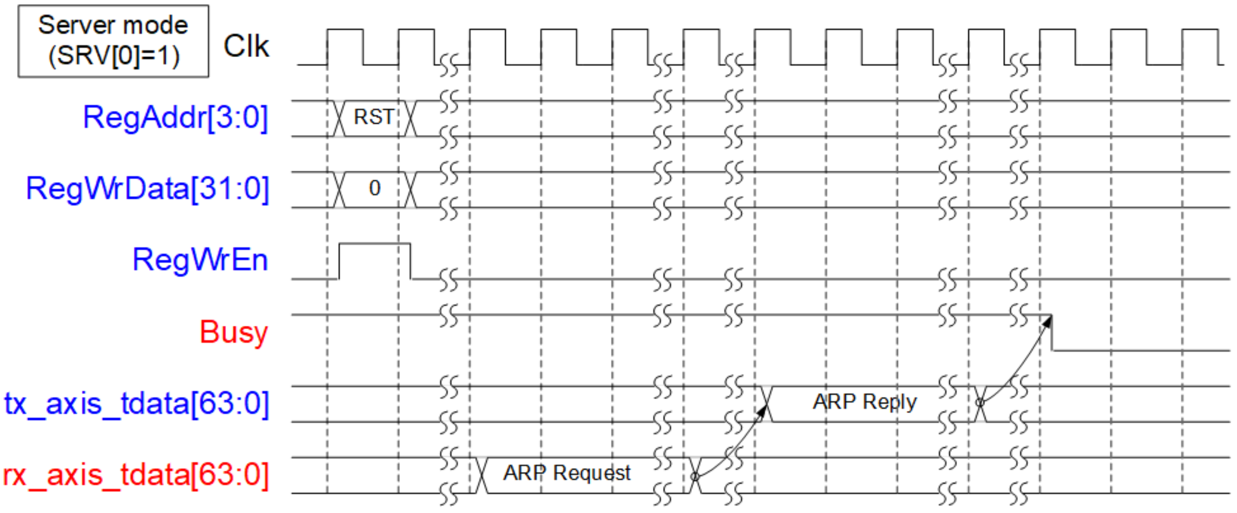

Figure 4: IP Initialization in Server mode

As shown in Figure 4, after reset process in Server mode is completed, the UDP10G IP waits for an ARP request packet from the target device. Upon receipt, the UDP10G IP generates an ARP reply packet. The Target MAC address is extracted from ARP request packet. Once the ARP reply packet has been transmitted, the Busy signal is de-asserted to 0b.

Register Interface

The Register interface is responsible for setting and monitoring all control signals and network parameters during operation. The timing diagram of the interface is similar to that of Single-port RAM, which shares the address bus for write and read access, and has a read latency time of one clock cycle. A Register map of this interface is provided in Table 3.

As shown in Figure 5, to write to the register, the user sets RegWrEn to 1b with the valid values for RegAddr and RegWrData. Before setting RegWrEn to 1b, please confirm that RstB is de-asserted to 1b for at least 4 clock cycles. To read from the register, the user only sets RegAddr, and RegRdData becomes valid in the next clock cycle.

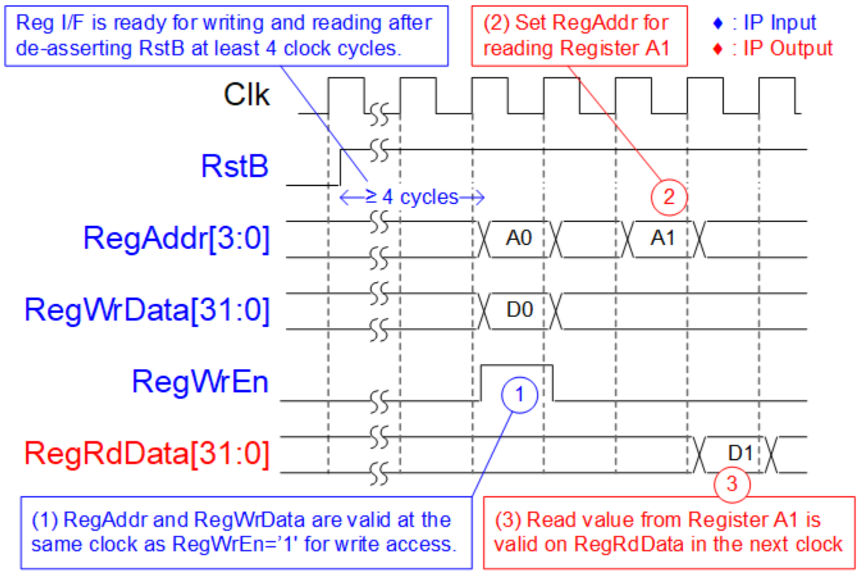
Reg I/F is ready for writing and reading after de-asserting RstB at least 4 clock cycles.
(2) Set RegAddr for reading Register A1
♦ : IP Input
♦ : IP Output
Clk
RstB
≥ 4 cycles
2
RegAddr[3:0]
A0
A1
RegWrData[31:0]
D0
RegWrEn
1
RegRdData[31:0]
D1
3
(1) RegAddr and RegWrData are valid at the same clock as RegWrEn='1' for write access.
(3) Read value from Register A1 is valid on RegRdData in the next clock

Figure 5: Register interface timing diagram

As shown in Figure 6, before setting the CMD register to initiate a new command operation, the Busy flag must be equal to 0b to confirm that IP is in Idle status. After setting the CMD register, the Busy flag is asserted to 1b and de-asserted to 0b when the command is completed.

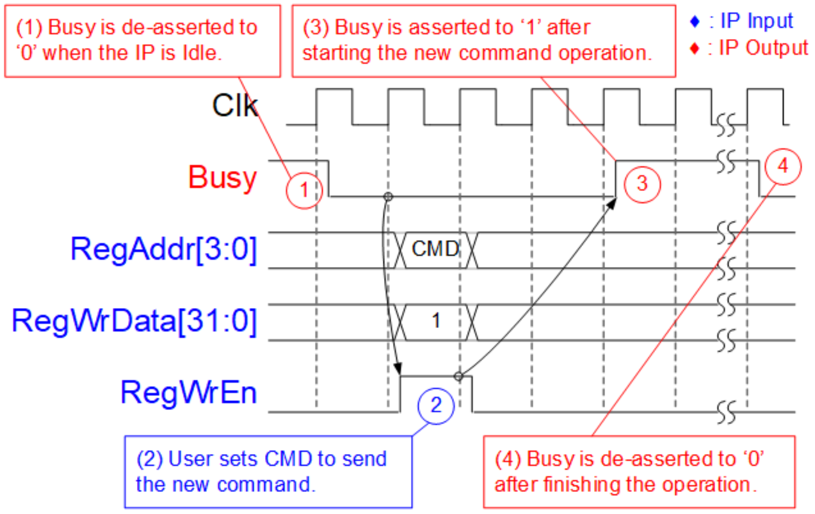
(1) Busy is de-asserted to '0' when the IP is Idle.
(3) Busy is asserted to '1' after starting the new command operation.
♦ : IP Input
♦ : IP Output
Clk
Busy
1
3
4
RegAddr[3:0]
CMD
RegWrData[31:0]
1
RegWrEn
2
(2) User sets CMD to send the new command.
(4) Busy is de-asserted to '0' after finishing the operation.

Figure 6: CMD register timing diagram

Tx FIFO Interface

Tx FIFO interface provides two control signals for the flow control, the full flag (UDPTxFfFull) and the write data counter (UDPTxFfWrCnt). UDPTxFfWrCnt is updated after asserting UDPTxFfWrEn for two clock cycles. UDPTxFfFull serves as an indicator of when the internal buffer is almost full and is asserted before it reaches its capacity. It is recommended to pause sending data within four clock cycles after UDPTxFfFull is asserted. Figure 7 shows an example timing diagram for the Tx FIFO interface.

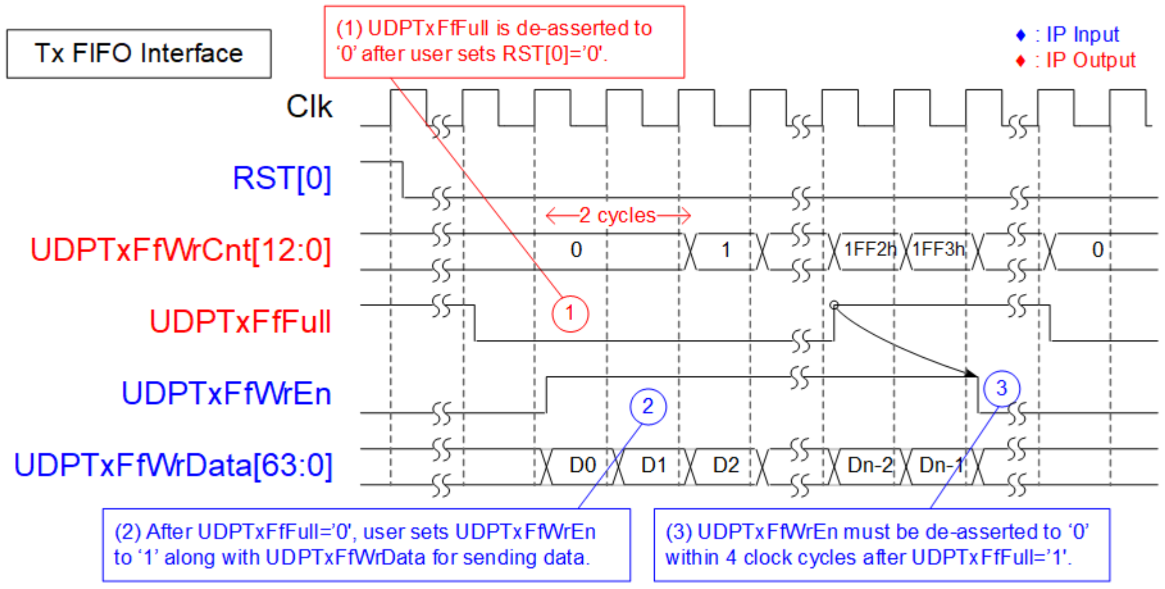
Tx FIFO Interface
(1) UDPTxFfFull is de-asserted to '0' after user sets RST[0]='0'.
♦ : IP Input
♦ : IP Output
Clk
RST[0]
UDPTxFfWrCnt[12:0]
UDPTxFfFull
UDPTxFfWrEn
UDPTxFfWrData[63:0]
2 cycles
0
1
1FF2h
1FF3h
0
D0
D1
D2
Dn-2
Dn-1
1
2
3
(2) After UDPTxFfFull='0', user sets UDPTxFfWrEn to '1' along with UDPTxFfWrData for sending data.
(3) UDPTxFfWrEn must be de-asserted to '0' within 4 clock cycles after UDPTxFfFull='1'.

Figure 7: Tx FIFO interface timing diagram

(1) When the IP is in reset state (RST[0] register=1b), the full flag (UDPTxFfFull) is set to 1b, preventing data from being sent by the user. Once the reset is released (RST[0]=0b), UDPTxFfFull is set to 0b, allowing the user to write data to the IP.

(2) To write data, the user needs to set UDPTxFfWrEn to 1b along with the write data on UDPTxFfWrData signal.

(3) If UDPTxFfFull is set to 1b, the user must pause sending data by de-asserting UDPTxFfWrEn to 0b within 4 clock cycles.

Rx FIFO Interface

The Rx FIFO interface is used to retrieve data stored in the Rx data buffer. To determine if data is available for reading, the Empty flag (UDPRxFfEmpty) is monitored, and the read enable signal (UDPRxFfRdEn) is then asserted to access the data, like a typical FIFO read interface, as illustrated in as shown in Figure 8.

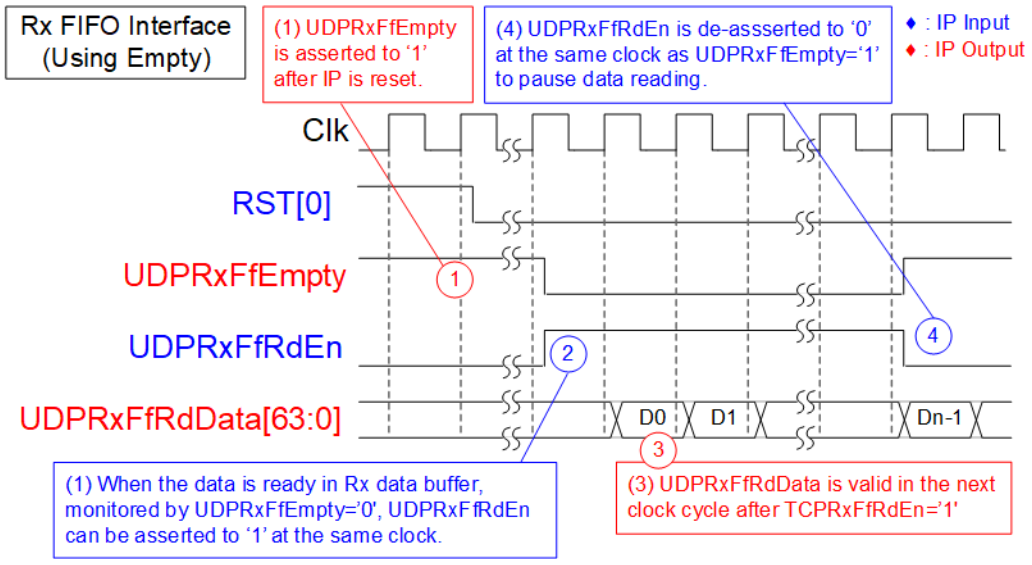
Rx FIFO Interface (Using Empty)
(1) UDPRxFfEmpty is asserted to '1' after IP is reset.
(4) UDPRxFfRdEn is de-assserted to '0' at the same clock as UDPRxFfEmpty='1' to pause data reading.
♦ : IP Input
♦ : IP Output
Clk
RST[0]
UDPRxFfEmpty
UDPRxFfRdEn
UDPRxFfRdData[63:0]
1
2
3
4
D0
D1
Dn-1
(1) When the data is ready in Rx data buffer, monitored by UDPRxFfEmpty='0', UDPRxFfRdEn can be asserted to '1' at the same clock.
(3) UDPRxFfRdData is valid in the next clock cycle after TCPRxFfRdEn='1'

Figure 8: Rx FIFO interface timing diagram using Empty flag

(1) After the IP finishes reset process, there is no data in Rx data buffer and UDPRxFfEmpty is set to 1b.
(2) Check the UDPRxFfEmpty flag to confirm data availability. When data is ready (UDPRxFfEmpty=0b), set UDPRxFfRdEn to 1b to read data from the Rx data buffer.
(3) The UDPRxFfRdData signal is valid in the next clock cycle.
(4) Reading data must be immediately paused by setting UDPRxFfRdEn=0b when UDPRxFfEmpty is equal to 1b.

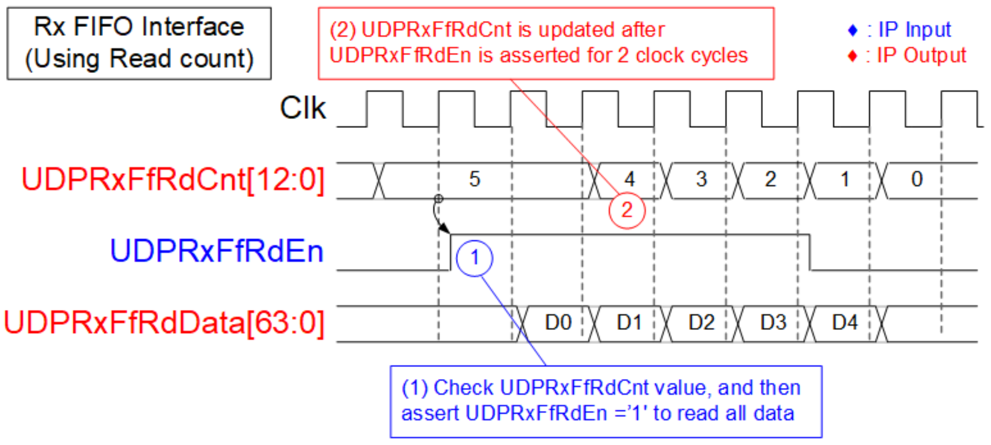
Rx FIFO Interface
(Using Read count)
(2) UDPRxFfRdCnt is updated after UDPRxFfRdEn is asserted for 2 clock cycles
♦ : IP Input
♦ : IP Output
Clk
UDPRxFfRdCnt[12:0]
5
4
3
2
1
0
UDPRxFfRdEn
UDPRxFfRdData[63:0]
D0
D1
D2
D3
D4
1
2
(1) Check UDPRxFfRdCnt value, and then assert UDPRxFfRdEn ='1' to read all data

Figure 9: Rx FIFO interface timing diagram using read counter

When the user logic reads data in burst mode, the UDP10G IP provides a read data counter signal to indicate the total amount of data stored in the Rx data buffer in bytes. For instance, in Figure 9, there are five units of data available in the Rx data buffer. Therefore, the user can set UDPRxFfRdEn to 1b for five clock cycles to read all the data from the Rx data buffer. The latency time to update UDPRxFfRdCnt after setting UDPRxFfRdEn to 1b is two clock cycles.

EMAC Interface

EMAC interface of UDP10G IP utilizes a 64-bit AXI4-stream interface, but it has a limitation that it cannot pause data transmission before transmitting the end of the packet. To transmit a packet, the tx_axis_tready must be asserted to 1b while transmitting a packet. However, once the final data of the packet has been transferred, tx_axis_tready can be de-asserted to 0b to pause the transmission of the next packet.

However, the user interface of Xilinx Ethernet Subsystem may pause data transmission before the final data of each packet is transferred, which is incompatible with the continuous data transfer requirement of the UDP10G IP EMAC interface. Therefore, additional logic, including a small buffer, is necessary to store the data when the Xilinx Ethernet Subsystem is not ready to transfer the data in each packet.

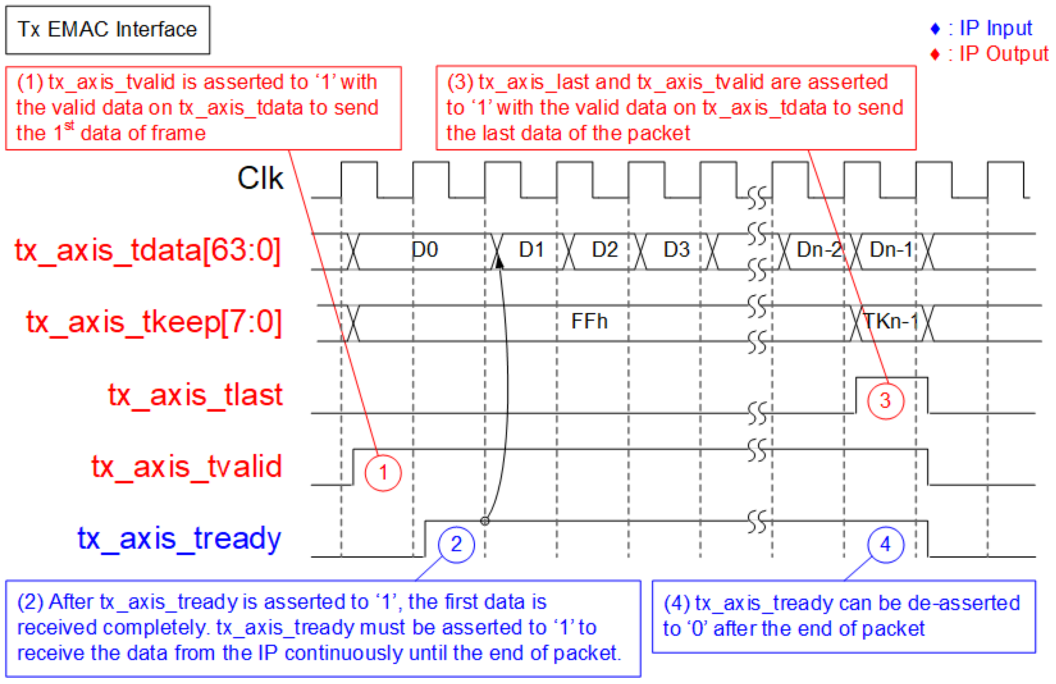
Tx EMAC Interface
♦ : IP Input
♦ : IP Output
(1) tx_axis_tvalid is asserted to '1' with the valid data on tx_axis_tdata to send the 1st data of frame
(3) tx_axis_last and tx_axis_tvalid are asserted to '1' with the valid data on tx_axis_tdata to send the last data of the packet
Clk
tx_axis_tdata[63:0]
tx_axis_tkeep[7:0]
tx_axis_tlast
tx_axis_tvalid
tx_axis_tready
D0
D1
D2
D3
Dn-2
Dn-1
FFh
TKn-1
1
2
3
4
(2) After tx_axis_tready is asserted to '1', the first data is received completely. tx_axis_tready must be asserted to '1' to receive the data from the IP continuously until the end of packet.
(4) tx_axis_tready can be de-asserted to '0' after the end of packet

Figure 10: Transmit EMAC interface timing diagram

(1) UDP10G IP asserts tx_axis_tvalid to 1b with the first data of the packet. All signals are latched until tx_axis_tready is asserted to 1b to accept the first data.
(2) After the first data is accepted by EMAC, tx_axis_tready must remain set to 1b to accept all remaining data in the packet from UDP10G IP until the end of the packet. The IP sends all data of each packet continuously.
(3) tx_axis_tlast and tx_axis_tvalid are both asserted to 1b when the final data of the packet is transmitted.
(4) Once the end of the packet has been transferred, tx_axis_tready can be asserted to 0b to pause the transmission of the next packet.

The Receive EMAC interface also requires continuous receipt of packet data, similar to Transmit EMAC interface. The Valid signal must remain asserted at 1b from the start of the packet to the end of the packet without interruption, as shown in Figure 11.

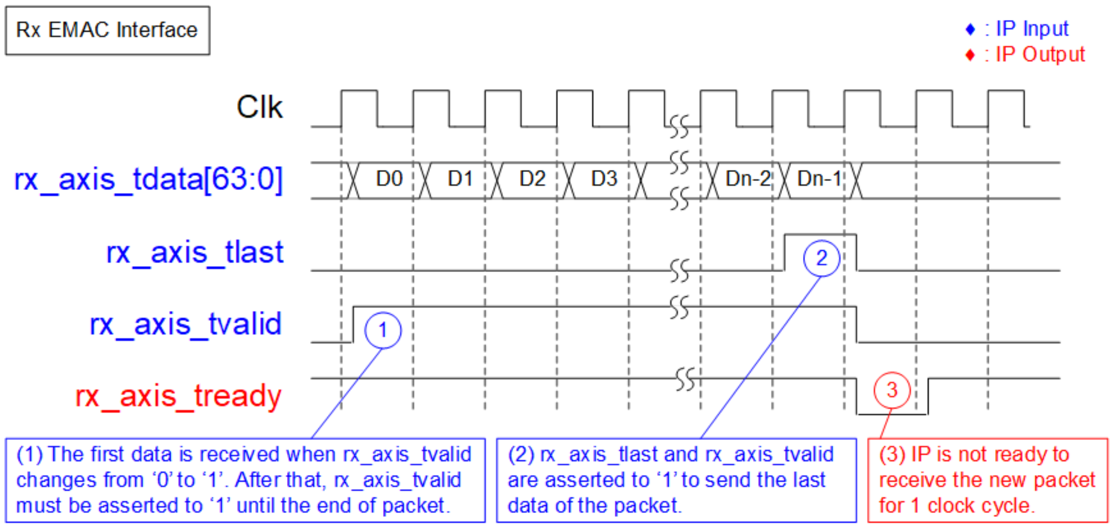
Rx EMAC Interface
♦ : IP Input
♦ : IP Output
Clk
rx_axis_tdata[63:0]
D0
D1
D2
D3
Dn-2
Dn-1
rx_axis_tlast
rx_axis_tvalid
rx_axis_tready
1
2
3
(1) The first data is received when rx_axis_tvalid changes from '0' to '1'. After that, rx_axis_tvalid must be asserted to '1' until the end of packet.
(2) rx_axis_tlast and rx_axis_tvalid are asserted to '1' to send the last data of the packet.
(3) IP is not ready to receive the new packet for 1 clock cycle.

Figure 11: Receive EMAC interface timing diagram

(1) The UDP10G IP detects the start of the receive frame when rx_axis_tvalid changes from 0b to 1b. In this cycle, the first data is valid on rx_axis_tdata. Afterward, rx_axis_tready is asserted to 1b to accept all data of this packet until the end of the packet. To continuously send the data of each packet, rx_axis_tvalid must remain set to 1b.

(2) The end of the packet is detected when rx_axis_tlast=1b and rx_axis_tvalid=1b. In this cycle, the final data of the packet is valid on rx_axis_tdata.

(3) After the final data of the packet has been transferred, UDP10G IP de-asserts rx_axis_tready for 1 clock cycle to complete the packet post-processing. Therefore, EMAC must support pausing data packet transmission for 1 clock cycle.

Example usage

Client mode (SRV[0]=0b)

The steps to set the register for transferring data in Client mode are outlined below.

1) Set the RST register=1b to reset the IP.
2) Set the SML/SMH for MAC address, DIP/SIP for IP address, and DPN/SPN for port number.
3) Set RST register=0b to start the IP initialization process. The UDP10G IP will send an ARP request packet to get the Target MAC address from the ARP reply packet. The Busy signal is de-asserted to 0b after completing the initialization process.
4) a. For sending data, set TDL register (total transmit length) and PKL register (packet size). Then, set CMD register = 1b to start data transmission. The user can send the data to UDP10G IP via the TxFIFO interface before or after setting the CMD register. Once the command is finished, the Busy flag is de-asserted to 0b. The user can set a new value to the TDL/PKL register and then set CMD register = 1b to start the next transmission.
   b. For receiving data, the user should monitor RxFIFO status and read the data until RxFIFO is empty.

Server mode (SRV[0]=1b)

In Server mode, the MAC address is decoded from ARP request packet instead of ARP reply packet as in Client mode. However, the process for transferring data is the same as that of Client mode. The following steps illustrate an example of Server mode.

1) Set RST register=1b 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=0b to begin the IP initialization process by waiting for an ARP request packet to get the Target MAC address. The IP then creates an ARP reply packet to return to the target device. Once the initialization process is completed, the Busy signal is de-asserted to 0b.
4) The remaining steps are the same as steps 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, KCU116, ZCU102, and ZCU111 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
1.5	8-Mar-23	Support KCU116 and ZCU111, add UDPTxFfWrCnt signal, update PKL valid range, and update Figure5.
1.4	2-Oct-20	Correct information
1.3	22-Aug-20	Update company information
1.2	20-Aug-20	Support DG 10G25GEMAC IP
1.1	8-Mar-19	Support ZCU102
1.0	15-Sep-17	Update IP specification
0.1	15-Aug-17	Draft release