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

Core Facts

Provided with Core	
Documentation	User Guide, Design Guide
Design File Formats	Encrypted hdl File
Verification	Test Bench, Simulation Library
Instantiation Templates	VHDL
Reference Designs & Application Notes	QuartusII Project, See Reference Design Manual
Additional Items	Demo on Arria10 GX/Arria10 SoC Development Board
Simulation Tool Used	
ModelSim-Altera 10.1e	
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- UDP/IP stack implementation
- Support IPv4 protocol
- Support one session per one UDP10G-IP (Multisession can be implemented by using multiple UDP10G-IP)
- Transmit data bus size is 64-bit, so transmit packet size must be aligned 64-bit
- Received data bus size is 64-bit, so total received size must be aligned 64-bit
- Transmit/Receive buffer size, adjustable for optimized resource and performance
- Simple data interface by standard FIFO interface
- Simple control interface by standard register interface
- 64-bit Avalon stream to interface with 10-Gbps Ethernet MAC from Intel
- One clock domain interface by fixed 156.25 MHz clock frequency
- Reference design available on Arria 10 GX/Arria10 SoC Development Board
- Support IP fragmentation

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	ALMs	Registers[1]	Pin	Block Memory bit[2]	Design Tools
Arria10 GX	10AX115S2F45I1SG	156.25	1,347	1,985	-	1,179,648	QuartusII 16.0
Arria10 SX	10AS066N3F40E2SG	156.25	1,339	1,975	-	1,179,648	QuartusII 16.0

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Block memory resource s are based on 64k Tx data buffer size, 16k Tx packet buffer size, and 16k Rx data buffer size. Minimum size of each buffer are 4k Tx data buffer size, 2k Tx packet buffer size, and 2k Rx data buffer size.

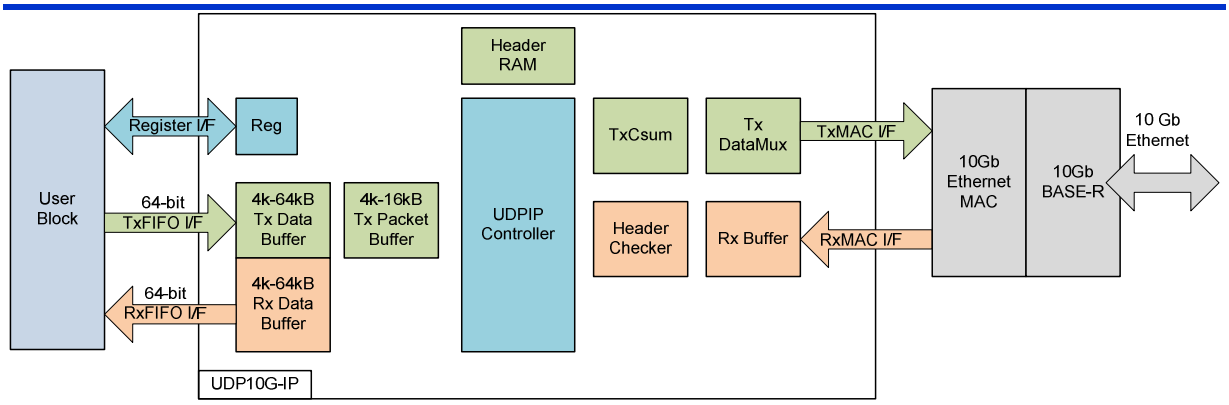

Figure 1: UDP10G-IP Block Diagram

Applications

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

General Description

UDP10G-IP core operating with Intel FPGA 10Gb EMAC IP and 10Gb BASE-R PHY can implement as UDP/IP stack, Transport layer, Internet layer, Link layer, and Physical layer for network data transmission. User can send and receive 10 Gb Ethernet data with some network devices through UDP/IP protocol by using this system.

There are three types of user interface, i.e. control signal by register access, transmit and received data signal by FIFO access. During initializing system, user needs to set up system parameters such as packet size, port number, IP address through register interface. After that, user can send command to transfer data from Tx Data buffer to external network device. For the receive direction, if the header of UDP packet is valid, UDP10G-IP will extract only UDP data to store in internal buffer to forward to the user logic.

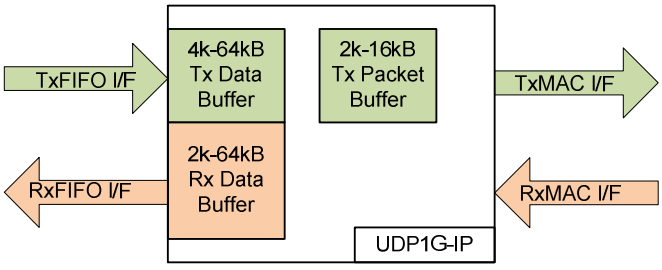

Figure 2: Adjustable Tx/Rx Buffer Size

Three buffers in UDP10G-IP, i.e. Tx Data buffer, Tx Packet buffer, and Rx Data buffer can set the size by setting parameter of the IP. The different size is provided to optimize resource utilization for user application. The setting value of Tx Data buffer size and Tx Packet buffer size are related to transmit packet size which user can set through register interface. Tx Packet Buffer must be more than the Tx packet size while Tx Data Buffer size should be at least two times of the Tx packet size.

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

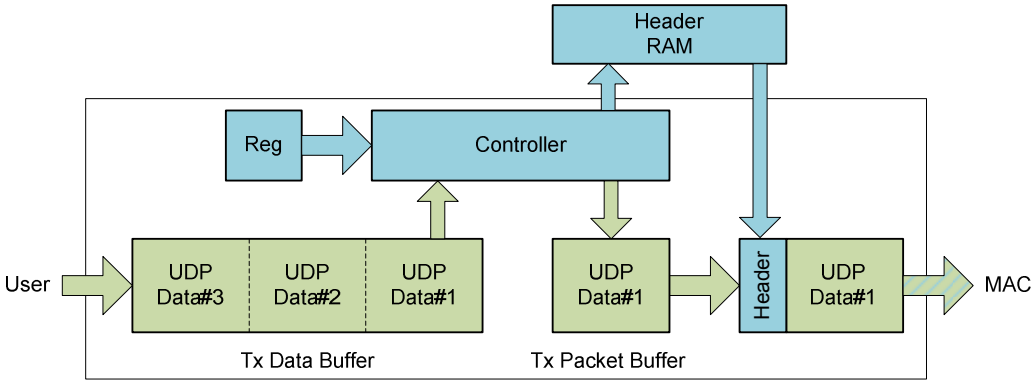

Figure 3: Transmit Data Flow

To transmit data, data of one packet from Tx Data buffer is forwarded to Tx Packet buffer. Parameter value within Reg is used to calculate the header of the packet and stored to Header RAM. Data output from Tx Packet buffer is combined with header data in Header RAM to create UDP packet for sending out to EMAC. UDP checksum and IP checksum are calculated within UDP1G-IP. Busy flag within register and output port are de-asserted to '0' after end of all data transfer. Total transmit size and packet size of each command are specified from register interface.

After IP is in Idle, user can change transmit packet size and total transmit size for new transfer.

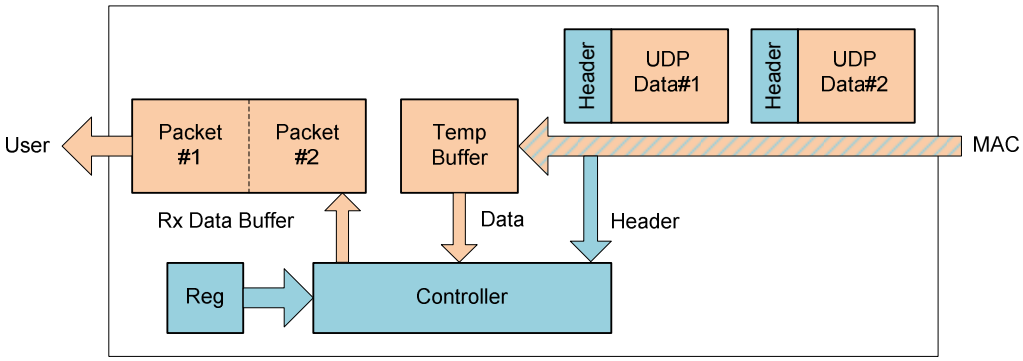

Figure 4: Receive Data Flow

For receiving data, Rx packet is stored in temp buffer firstly. Header and checksum within Rx packet are verified by using setting parameter from register interface. If header or checksum is error, the packet will be rejected and not stored to Rx Data buffer. Data of valid packet is extracted from UDP packet and stored to Rx Data buffer for user reading.

Functional Description

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

Control Block

- **Reg**

User can set parameters for UDP/IP operation by using register interface. Register address of this interface is equal to 4-bit. The description of each register address is defined as shown in Table 2**.** After system reset is released, all internal parameters are updated by new value.

- **UDPIP Controller**

After intialization complete for ARP request/reply transferring to get MAC address of the target from IP address, the IP will be in Idle status and ready to receive command from user. There is only one command for UDP10G-IP, sending data.

Table 2: Register map Definition

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

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

Table 3: TxBuf/TxPac/RxBufBitWidth Parameter description

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

Transmit Block

- **Tx Data Buffer**

This buffer size is set by "TxBufBitWidth" parameter of the IP. The valid value is 9-13 which is equal to the address size of 64-bit buffer, as shown in Table 3.

The buffer size should be at least two times of Tx Packet Size (PKL register in Table 2) for sending data continuously. During sending operation, one packet data is forwarded from this buffer to Tx Packet Buffer and the data of the new packet should be received from the user at the same time. Data of the buffer is flushed after the packet has already sent out to EMAC. So, at least two times of packet size are required for sending current packet and preparing next packet at the same time. Bigger buffer size can help the user logic to control data flow with UDP10G-IP. If there are many data storing in the buffer, user logic will have much time to operate other task during IP sending the data.

- **Tx Packet Buffer**

The size is set by "TxPacBitWidth" parameter of the IP. The valid value is 9-11 and the description of the parameter is shown in Table 3. This buffer size must be more than or equal to Tx Packet size (setting in PKL register) to store at least one packet data splitting from Tx Data Buffer. Data in Tx Packet Buffer is stored until EMAC ready to receive data before sending out. At most, data is stored in this buffer about two times of packet size, so bigger size than two times of the biggest packet size for this buffer is not necessary.

- **Header RAM**

This RAM is applied to store header part of Transmit packet. Parameter in Header RAM are updated by value in registers when user releases RST register. Some parameters such as Target MAC address are updated by ARP Reply (Client mode)/Request (Server mode).

- **TxCsum**

This module is designed to calculate checksum of Tx packet before sending out.

- **TxDataMux**

This module is designed to combine header from Header RAM to data from Tx Packet Buffer, and then send out to EMAC.

Receive Block

- **Rx Buffer**

This is temporary buffer to store each Rx packet from EMAC for waiting Header Checker processing.

- **Header Checker**

Header in Rx packet is compared with parameter in register. Packet will be ignored if some parameter are not matched or checksum is error. UDP checksum is not verified when received packet is IP fragmented packet. Only UDP data without the header is stored into Rx Data Buffer.

- **Rx Data Buffer**

This buffer size is set by “RxBufBitWidth” parameter of the IP. The valid value is 9-13 and the description of the parameter is shown in Table 3. This is the data buffer between user logic and UDP10G-IP. If buffer is full, new receive packet will be ignored. So, the buffer size should be at least two times of receive packet size to store the new packet and transfer the previous packet to user logic at the same time.

User Block

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

10 Gb Ethernet MAC and 10 Gb BASE-R

Both blocks are softIPcore provided by Intel FPGA. Please read more details from following website.
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-10gbps-ethernet-mac.html
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-10gbase-r-pcs.html

Core I/O Signals

Descriptions of all parameters and signal I/O are provided in Table 4 and Table 5. MAC Interface signals are Avalon stream interface to connect with Intel FPGA 10 Gb EMAC.

Table 4: Core Parameters

Generic 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 5: Core I/O Signals

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

Signal	Dir	Description
MAC Interface		
MacRxData[63:0]	In	Received data bus.
MacRxValid	In	Received data valid signal. Synchronous with MacRxData. This signal must be asserted to ‘1’ continuously during start and end of received packet.
MacRxEOP	In	Control signal to indicate the final word in the frame.
MacRxError	In	Control signal asserted at the end of received frame to indicate that the frame has an error. ‘0’: normal packet, ‘1’: error packet.
MacRxReady	Out	Asserted to ‘0’ after receive the last data in the frame (MacRxEOP=’1’). This signal is de-asserted to '0' for 1-clock to pause received data of the next packet.
MacTxData[63:0]	Out	Transmitted data.
MacTxEmpty[2:0]	Out	Specify the number of bytes which are unused of the final word in the frame.
MacTxValid	Out	Transmitted data valid signal to EMAC. Synchronous with MacTxData.
MacTxSOP	Out	Control signal to indicate the first word in the frame.
MacTxEOP	Out	Control signal to indicate the final word in the frame.
MacTxReady	In	Handshaking signal. Asserted when MacTxData has been accepted. This signal must be asserted to ‘1’ continuously during start and end of transmitted packet.

Timing Diagram

IP Initialization

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

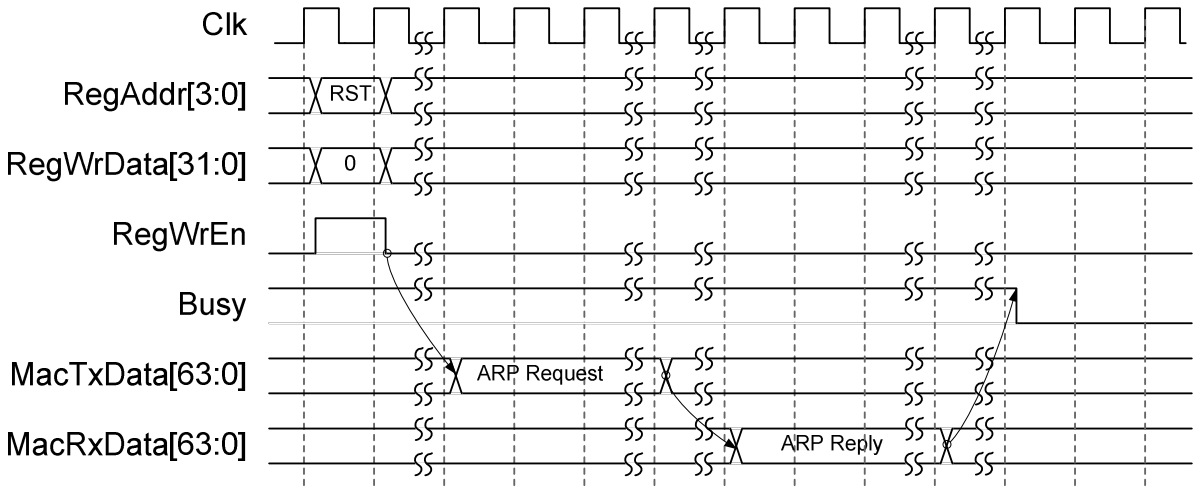

Figure 5: IP Initialization in Client mode

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

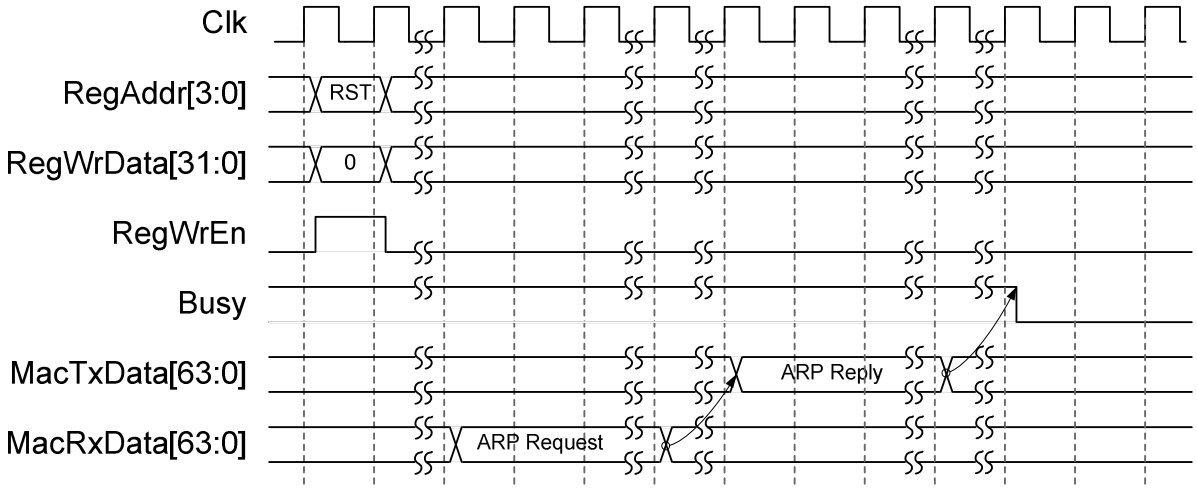

Figure 6: IP Initialization in Server mode

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

Register Interface

User can write/read control signal with UDP10G-IP by using Register interface which has timing diagram as shown in Figure 7. Register map address is designed as shown in Table 2. To write control signal, User needs to set RegWrEn='1' with valid value of RegAddr and RegWrData. To read control signal, User set only RegAddr value and then RegRdData is valid in the next clock period.

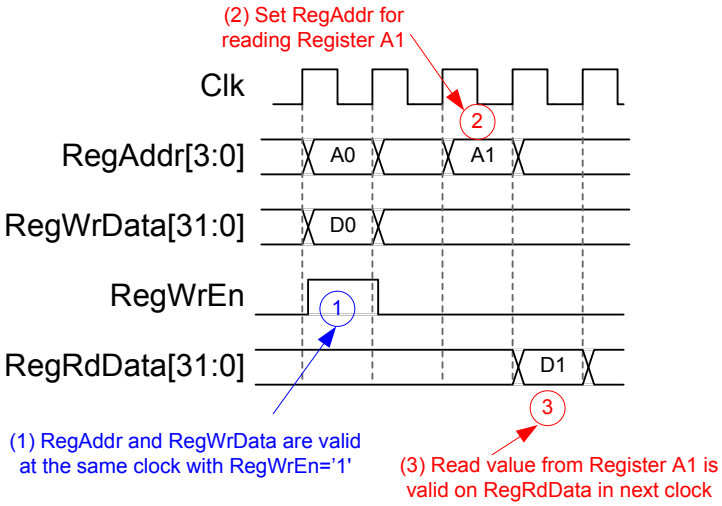

Figure 7: Register Interface Timing Diagram

To start sending data, Busy flag must be monitored through register access or output signal before writing CMD register, as shown in Figure 8. After new command is received, busy is asserted.

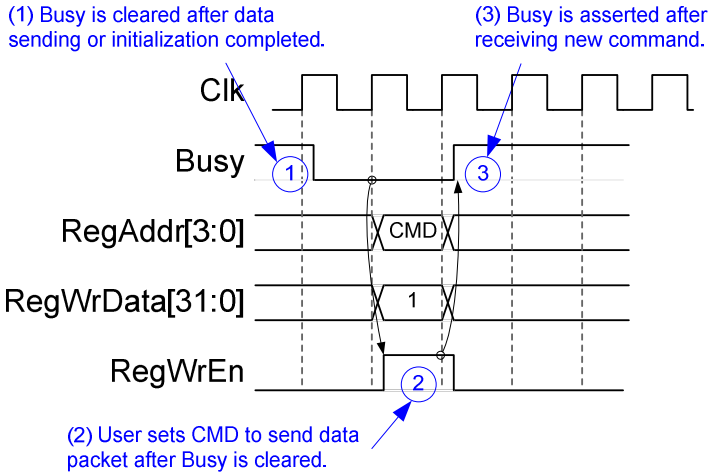

Figure 8: Set CMD register when Busy is de-asserted

Tx FIFO Interface

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

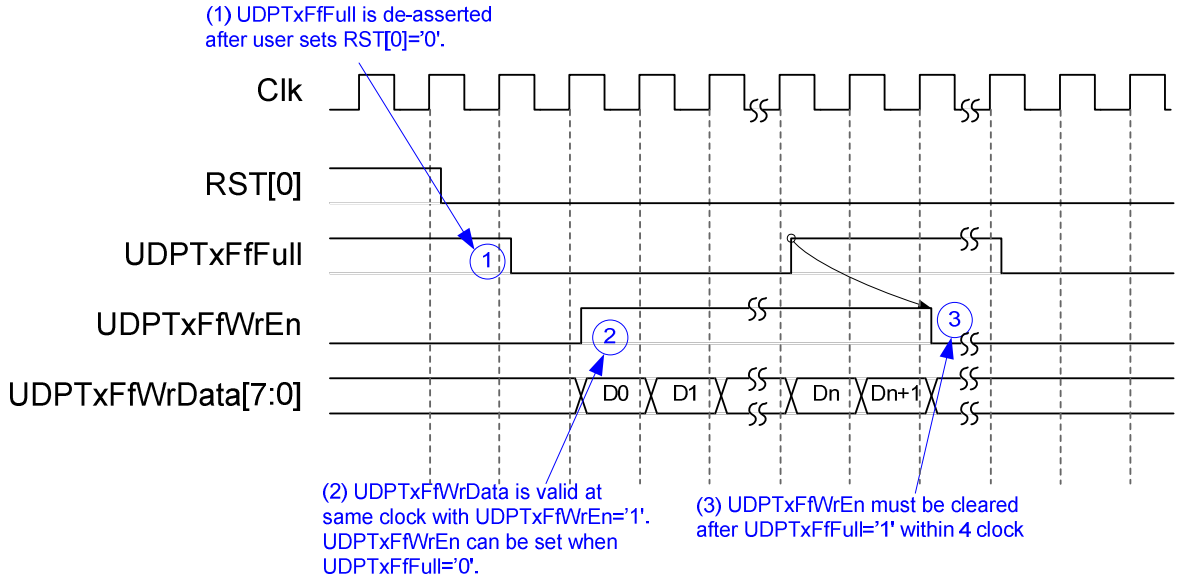

Figure 9: Tx Data Buffer Interface Timing Diagram

Rx FIFO Interface

When IP core receives data from the external device, valid data is stored in Rx Data buffer. User can read data from this buffer by using FIFO interface, as shown in Figure 10. User can monitor data available status from UDPRxFfEmpty. Data can be read by setting UDPRxFfRdEn='1' when UDPRxFfEmpty is cleared to '0'. UDPRxFfRdData is valid in the next clock period. UDPRxFfRdEn must be de-asserted to '0' at the same clock with UDPRxFfEmpty = '1'. Similar to Tx data buffer, Rx data buffer is flushed when IP is reset. UDPRxFfEmpty is asserted to '1' during reset condition.

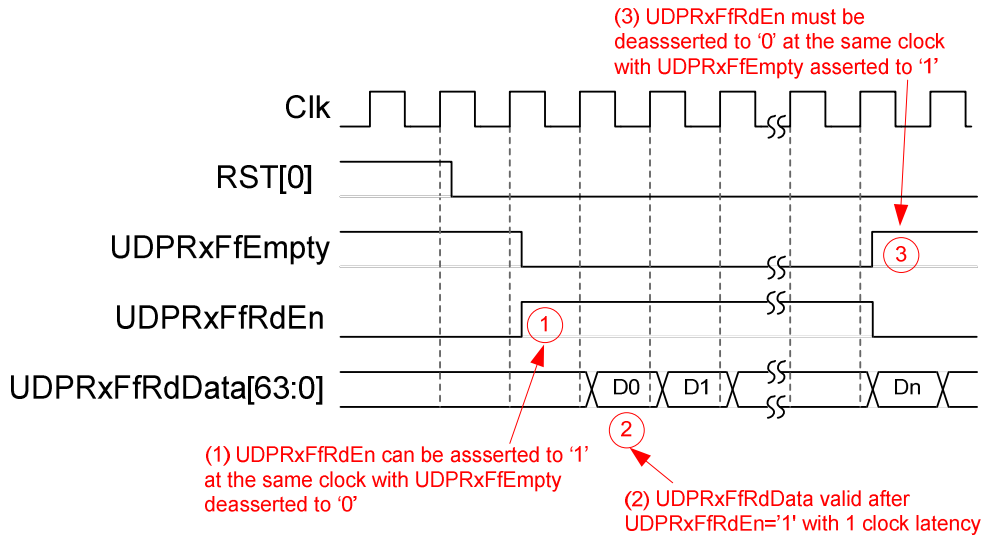

Figure 10: Rx Data Buffer Interface by Empty flag Timing Diagram

Rx data buffer status can be also monitored by using UDPRxFfRdCnt. This signal shows total 64-bit data in Rx data buffer. So, user can assert UDPRxFfRdEn='1' for time period equal to total number of data, as shown in Figure 11. If UDPRxFfLastRdCnt is not equal to 0, it means that current total size is not aligned to 64-bit. The unaligned received size is shown in UDPRxFfLastRdCnt signal.

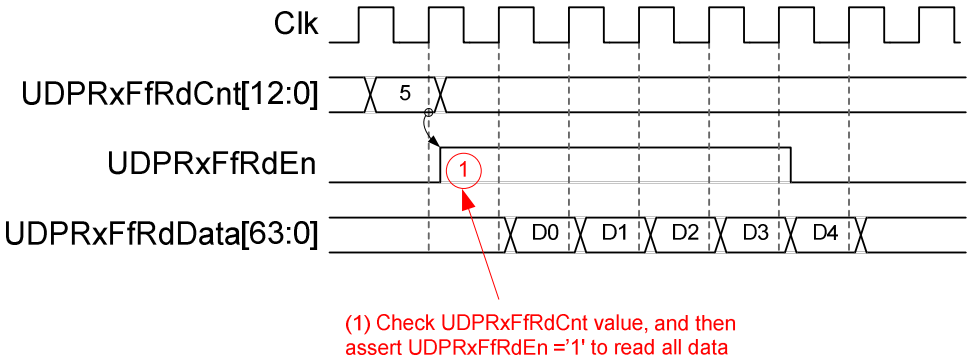

Figure 11: Rx Data Buffer Interface by Read counter Timing Diagram

EMAC Interface

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

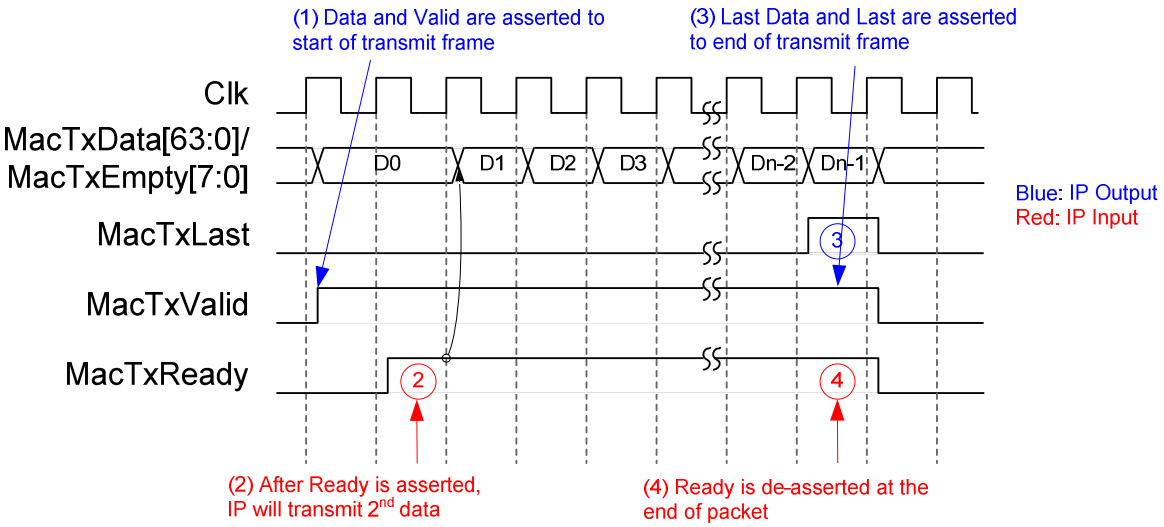

Figure 12: Transmit EMAC Interface

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

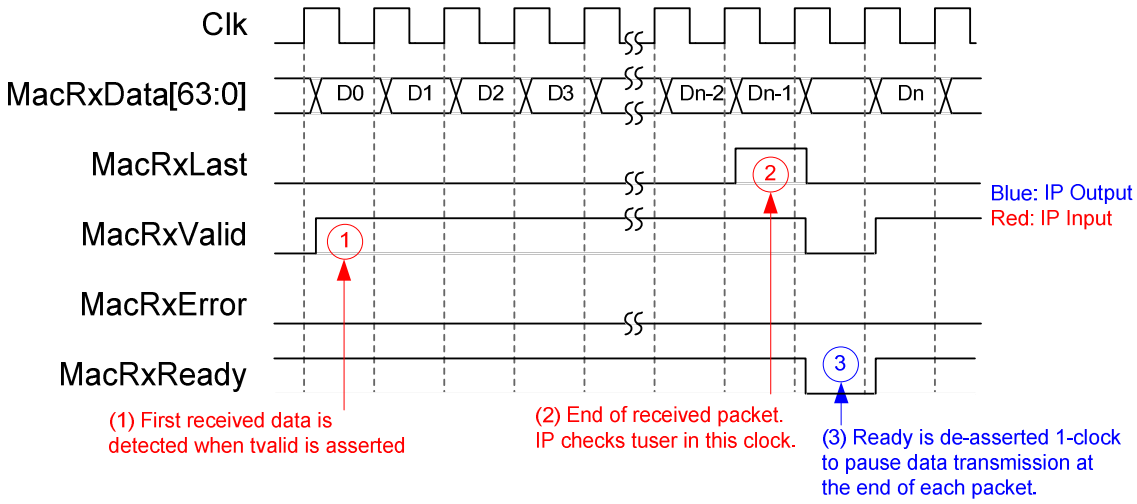

Figure 13: Receive EMAC Interface

Example usage

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

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

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

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

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

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

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

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

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

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

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

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

4) Data process is same as Client mode

Verification Methods

The UDP10G-IP Core functionality was verified by simulation and also proved on real board design by using Arria10 GX/Arria10 SoC development board.

Recommended Design Experience

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

Ordering Information

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

Revision History

Revision	Date	Description
1.0	Sep-19-2017	New release
1.1	Mar-22-2018	Support A10 SoC