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 CycloneV E/ ArriaV 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 UDP1G-IP (Multisession can be implemented by using multiple UDP1G-IP)
- Transmit/Receive buffer size, adjustable for resource and performance optimization
- Simple data interface by standard FIFO interface
- Simple control interface by standard register interface
- 8-bit Avalon stream to interface with Triple-Speed Ethernet MAC from Intel
- One clock domain interface by fixed 125 MHz clock frequency
- Reference design available on CycloneV E/ArriaV GX/Arria 10 SoC Development Board
- Not support data fragmentation feature

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	ALMs	Registers[1]	Pin	Block Memory bit[2]	Design Tools
StratixIV GX	EP4SGX230KF40C2	125	1,125	1,553	-	1,181,696	QuartusII 14.0
CycloneV E	5CEFA7F31I7	125	1,048	1,698	-	1,181,696	QuartusII 15.1
ArriaV GX	5AGXFB3H4F35C5	125	1,047	1,686	-	1,181,696	QuartusII 14.0
Arria10 SX	10AS066N3F40E2SGE2	125	996	1,635	-	1,181,696	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.

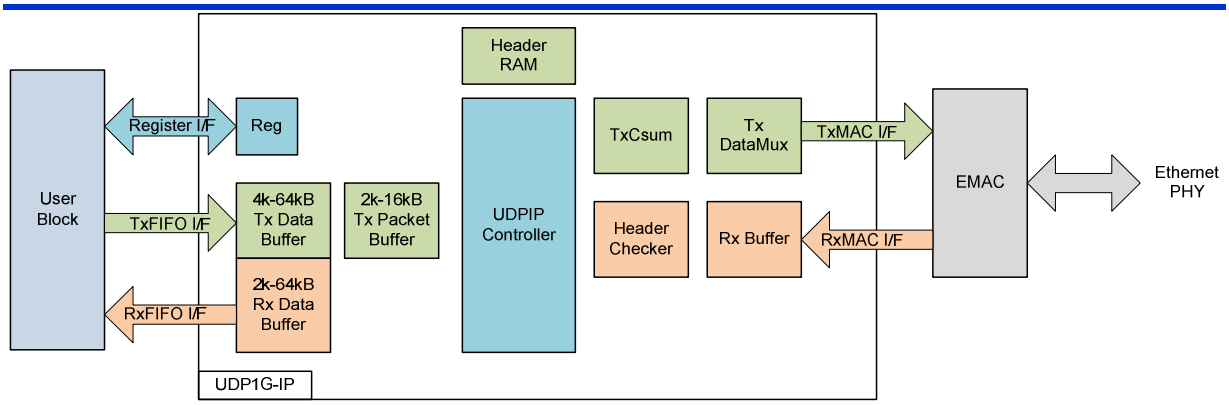

Figure 1: UDP1G-IP Block Diagram

Applications

UDP1G-IP is designed for network application by using UDP/IP protocol to transfer high speed data through 1 Gb Ethernet. By using this IP, user can easily transfer data with any device through UDP/IP protocol without CPU and external meomory in the system.

General Description

UDP1G-IP core operating with Intel Triple-Speed Ethernet MAC can implement UDP/IP stack, Transport layer, Internet layer, and Link layer for network data transmission. User can send and receive 1 Gb Ethernet data with any network device through UDP/IP protocol by using this system and external PHY chip.

There are three types of user interface, i.e. control signal by register access, transmit and receive data signal by FIFO access. During initializing system, user needs to set up system parameter 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, UDP1G-IP will extract only UDP data to store in internal buffer for forwarding to the user.

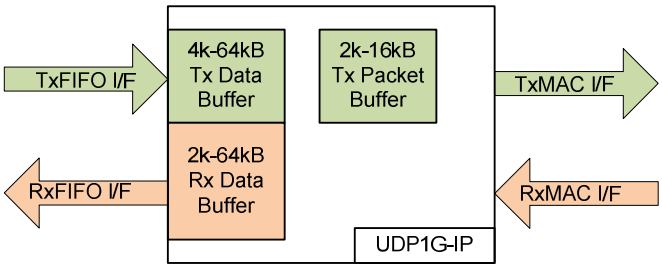

Figure 2: Adjustable Tx/Rx Buffer Size

Three buffers in UDP1G-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.

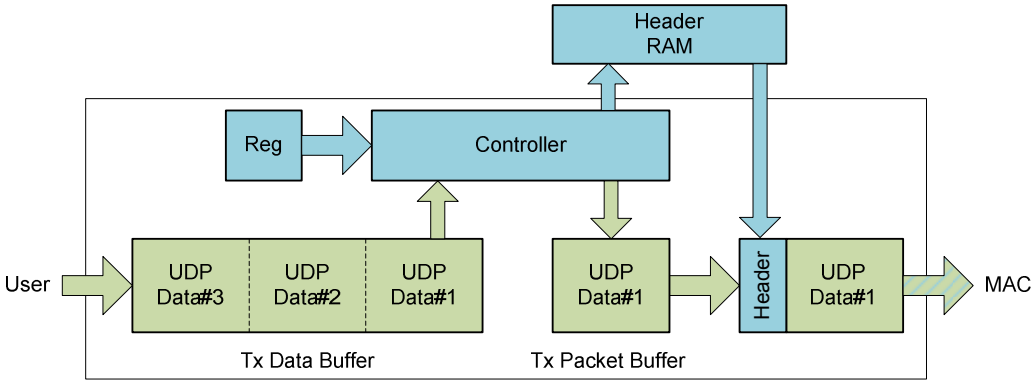

Figure 3: Transmit Data Flow

To transmit data, data of one packet from Tx Data buffer will be forwarded to Tx Packet buffer. Parameter value within Reg are used to calculate the header of the packet and store to Header RAM. Data output from Tx Packet buffer will be combined with header data in Header RAM to create UDP packet for sending out to EMAC. UDP and IP checksum will be calculated within UDP1G-IP. Busy flag within register and output port will be cleared 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.

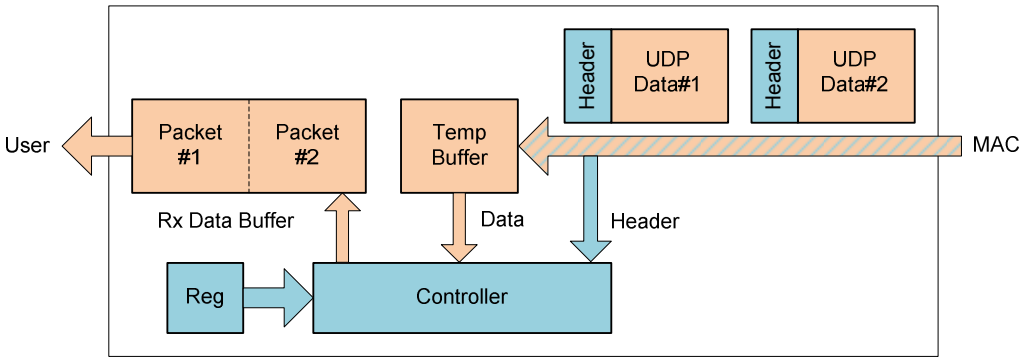

Figure 4: Receive Data Flow

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

Functional Description

UDP1G-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 will be updated by new value.

- **UDPIP Controller**

After release reset, IP will send ARP request to get MAC address of the target from IP address. Then, it will wait command from user to send data packet or receive data packet with external device.

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/clear again to reload parameter if the value of SML, SMH, DIP, SIP, DPN, or SPN register is changed by user.**
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 operation.**
		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.**
1000b	TDL	Wr	[31:0]	Total Tx data length transfer in byte unit. Valid from 1-0xFFFFFFFF. **User needs to set this register before setting CMD register. This value will be 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 will not need to set again. Previous value will be used from internal latch.
		Rd	[0]	Remaining data transfer length in byte unit which still not transmit.

RegAddr [3:0]	Reg Name	Dir	Bit	Description
1001b	TMO	Wr	[31:0]	Define timeout value for waiting ARP reply packet after sending ARP request. This is programmed to timeout counter which runs in 125 MHz, so timer unit is about 8 ns. This value should be more than 0x6000.
		Rd		[0]-Timeout from not receiving ARP reply packet After timeout, IP will resent 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. Valid from 1-16000. Default value is 1472 byte (Maximum size for non-jumbo frame). **This value must not be changed during data transmission not complete (Busy='1'). If next transmit use same packet size, user will not need to set this register because the previous value is latched in the logic.**

Note:

1. Target Mac address is defined from returned value in ARP Reply packet, so user doesn't need to set this parameter.

Table 3: TxBuf/TxPac/RxBufBitWidth Parameter description

Value of BitWidth	Buffer Size	TxBufBitWidth	TxPacBitWidth	RxBufBitWidth
11	2kByte	No	Valid	Valid
12	4kByte	Valid	Valid	Valid
13	8kByte	Valid	Valid	Valid
14	16kByte	Valid	Valid	Valid
15	32kByte	Valid	No	Valid
16	64kByte	Valid	No	Valid

Transmit Block

- **Tx Data Buffer**

This buffer size is set by "TxBufBitWidth" parameter of the IP. The valid value is 12-16 which is equal to the address size of buffer, as shown in Table 3.

The buffer size must be at least two times of Tx Packet Size in PKL register for sending data continuously. During sending operation, one packet data will be forwarded from this buffer to Tx Packet Buffer and the data of the new packet will be received from the user at the same time. Data of the buffer will be 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 will help the user logic to control data flow with UDP1G-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 11-14 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 all data from the IP. If EMAC is not ready for long time, this buffer will store at most ~2 data packets. So, the remaining space in the buffer which is more than 2 packets will not be used.

- **Header RAM**

This RAM is applied to store header part of Transmit packet. Parameter in Header RAM will be updated by register value when release RST register. Some parameters such as Target MAC address will be updated by ARP Reply.

- **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 with 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 will be compared with parameter in register. Packet will be ignored if any parameter is not matched or checksum is error. Only UDP data without the header will be stored into Rx Data Buffer.

- **Rx Data Buffer**

This buffer size is set by “RxBufBitWidth” parameter of the IP. The valid value is 11-16 and the description of the parameter is shown in Table 3. This is the data buffer between user logic and UDP1G-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 and monitoring register interface, writing data to Tx FIFO, and reading data from Rx FIFO. This module can be designed by simple hardware logic.

Triple-Speed Ethernet MAC

This block is softIPcore provided by Intel. Please read more details from following website.
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-ethernet-mac.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 Triple Speed EMAC.

Table 4: Core Parameters

Generic Name	Value	Description
TxBufBitWidth	12-16	Setting Tx Data buffer size. The value is referred to address bus size of this buffer.
TxPacBitWidth	11-14	Setting Tx Packet buffer size. The value is referred to address bus size of this buffer.
RxBufBitWidth	11-16	Setting Rx Data buffer size. The value is referred to address bus size of this buffer.

Table 5: Core I/O Signals

Signal	Dir	Clk	Description
Common Interface Signal			
RstB	In		Reset IP core. Active Low.
Clk	In		125 MHz fixed clock frequency input for user interface and MAC transmit interface for 1 Gbps mode
User Interface			
RegAddr[3:0]	In	Clk	Register address bus
RegWrData[31:0]	In	Clk	Register Write data bus
RegWrEn	In	Clk	Register Write enable pulse. Assert with valid value of RegAddr and RegWrData signals.
RegRdData[31:0]	Out	Clk	Register Read data bus. Available after asserting RegAddr with 1 Clk period latency
Busy	Out	Clk	IP busy status (‘0’-Idle, ‘1’-IP Intialization or IP sending data)
IntOut	Out	Clk	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	Clk	Transmit buffer full flag. User needs to stop writing data within 4 clock period after this flag is asserted to high.
UDPTxFfWrEn	In	Clk	Transmit buffer write enable. Assert to write data to Transmit buffer.
UDPTxFfWrData[7:0]	In	Clk	Transmit buffer write data bus. Synchronous with UDPTxFfWrEn.
Rx Data Buffer Interface			
UDPRxFfRdCnt[15:0]	Out	Clk	Receive buffer data counter to show total number of receive data in buffer.
UDPRxFfRdEmpty	Out	Clk	Receive buffer empty flag. User needs to stop reading data immediately.
UDPRxFfRdEn	In	Clk	Receive buffer read enable. Assert to read data from Receive buffer.
UDPRxFfRdData[7:0]	Out	Clk	Receive buffer read data bus. Valid after UDPRxFfRdEn assert with 1 Clk period latency.
UDPRxFfRdCnt[15:0]	Out	Clk	Receive buffer data counter to show total number of receive data in buffer.

MAC Interface			
MacTxSOP	Out	Clk	Transmit start of packet. Assert this signal when the first byte in the frame is driven on MacTxData.
MacTxData[7:0]	Out	Clk	Transmit data.
MacTxEOP	Out	Clk	Transmit end of packet. Assert this signal when the last byte in the frame is driven on MacTxData.
MacTxValid	Out	Clk	Transmit data valid. Assert this signal to indicate that the data on the following signals are valid: MacTxSOP, MacTxData, and MacTxEOP.
MacTxReady	In	Clk	Mac ready. Assert this signal to indicate that MAC is ready to accept data.
MacRxClk	In		Receive clock from MAC.
MacRxSOP	In	MacRxClk	Receive start of packet. This signal is unused.
MacRxData[7:0]	In	MacRxClk	Receive data.
MacRxEOP	In	MacRxClk	Receive end of packet. Assert when the last byte of frame is driven on MacRxData.
MacRxValid	In	MacRxClk	Receive data valid. Assert when data on the following signals are valid: MacRxSOP, MacRxData, and MacRxEOP.
MacRxError	In	MacRxClk	Receive error. Assert with the final byte in the frame to indicate that receive frame has the error.

Timing Diagram

Register Interface

User can write/read control signal with UDP1G-IP by using Register interface which has timing diagram as shown in Figure 5. 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 will be valid in next clock period.

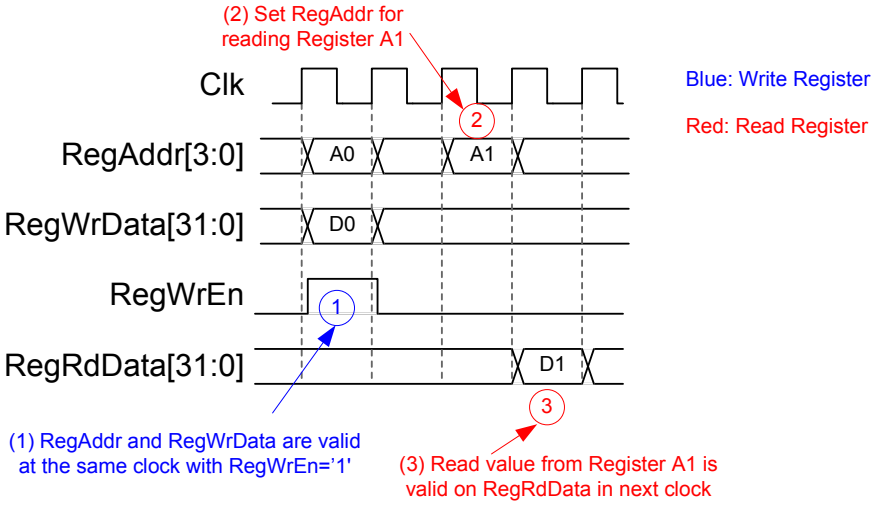

Figure 5: Register Interface Timing Diagram

To start sending data, Busy flag must be monitored through register access or output signal before writing CMD register, as shown in Figure 6. After new command is received, busy will be asserted.

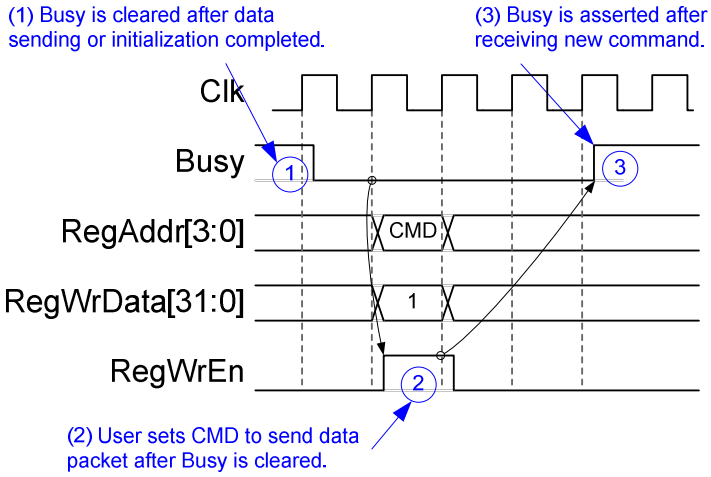

Figure 6: Set CMD register when Busy is de-asserted

Tx FIFO Interface

User can send data to IP core by using FIFO interface, as shown in Figure 7. 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 period to stop data sending after UDPTxFfFull is asserted to '1'. During IP is in reset condition, UDPTxFfFull will be asserted and all data in FIFO will be flushed.

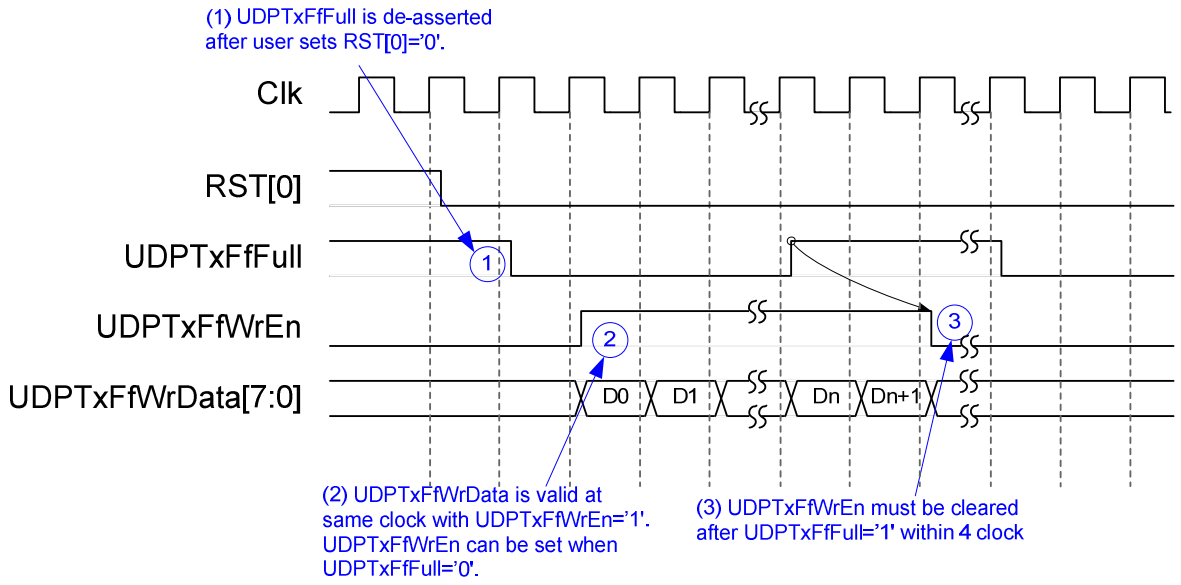

Figure 7: Tx Data Buffer Interface Timing Diagram

Rx FIFO Interface

When IP core receives data from the external device, valid data will be stored in Rx Data buffer. User can read data from this buffer by using FIFO interface, as shown in Figure 8. User can monitor data available status from UDPRxFfEmpty. Data can be read when UDPRxFfEmpty is cleared to '0'. UDPRxFfRdEn can be set to '1' to read data from Rx data buffer and UDPRxFfRdData will be valid in next clock period. Reading data must be stopped by clearing UDPRRxFfRdEn='0' at the same clock with UDPRxFfEmpty = '1'. Similar to Tx data buffer, Rx data buffer will be flushed when IP is reset. UDPRxFfEmpty is asserted to '1' during reset condition.

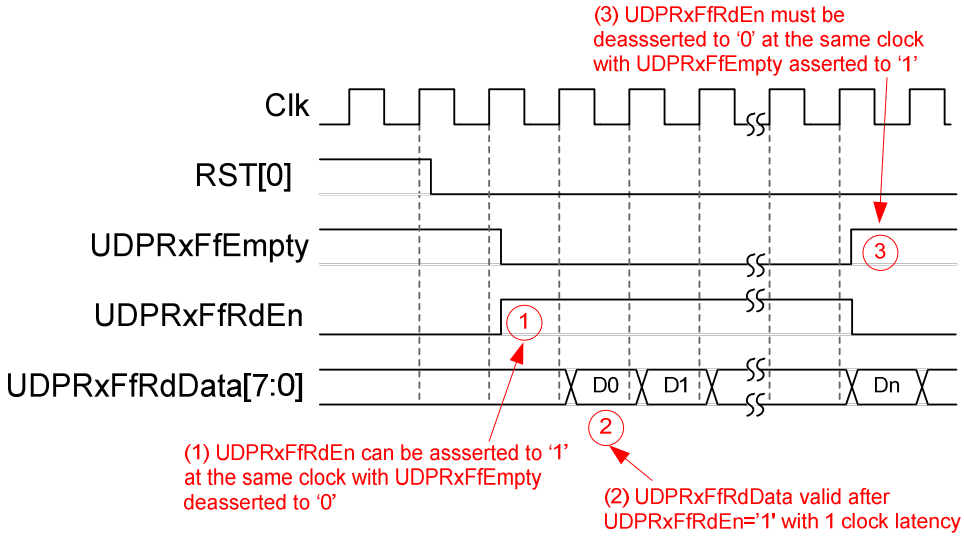

Figure 8: Rx Data Buffer Interface by Empty flag Timing Diagram

Rx data buffer status can be also monitored by using UDPRxFfRdCnt. This signal shows total number of available data of Rx data buffer in byte unit. So, user can assert UDPRxFfRdEn='1' for time period equal to total number of data, as shown in Figure 9.

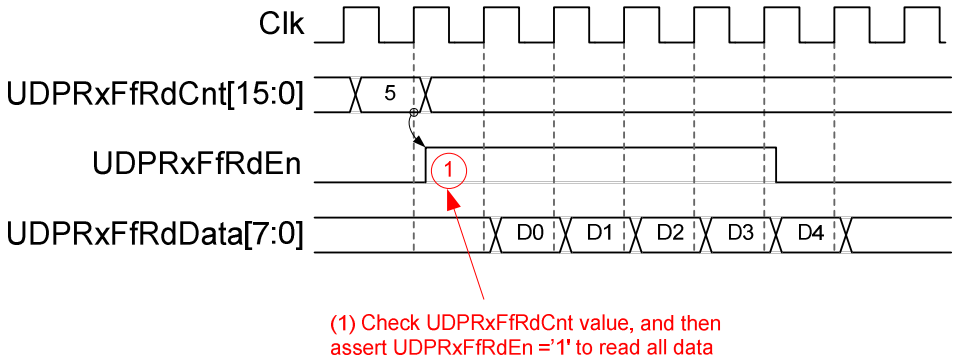

Figure 9: Rx Data Buffer Interface by Read counter Timing Diagram

EMAC Interface

To transmit packet, the IP will assert MacTxSOP and MacTxValid with the first data of the packet on MacTxData. The signals will be latched until MacTxReady output from EMAC is asserted to '1' to acknowledge data transmit request. MacTxReady must be asserted to '1' until the packet is end of transmission. So, MacTxValid is asserted to '1' to send data on MacTxData continuously until the end of the packet. MacTxEOP and MacTxValid will be asserted to '1' with the last transmit data to show end-of-packet status.

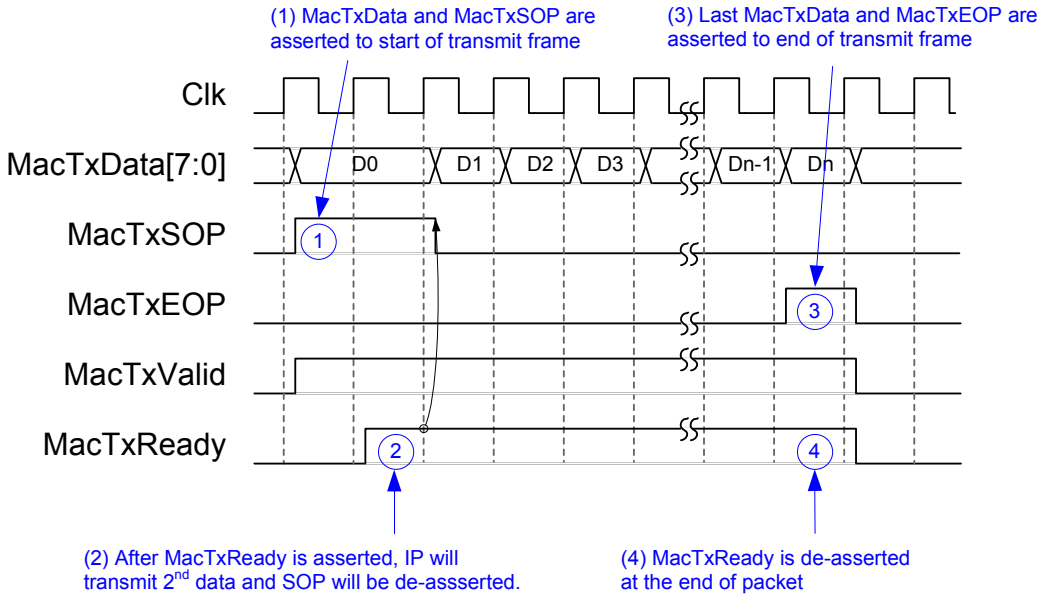

Figure 10: Transmit EMAC Interface

Figure 11 shows timing diagram of Receive side. UDP1G-IP monitors start of receive frame from MacRxValid which changes from '0' to '1'. MacRxData will be received continuously until MacRxEOP is asserted to be end of packet. Last MacRxData (D_n) must be available on the bus after D_{n-1} because the IP will read D_0-D_n continuously. MacRxEOP is used to be valid signal of MacRxError for checking packet validation.

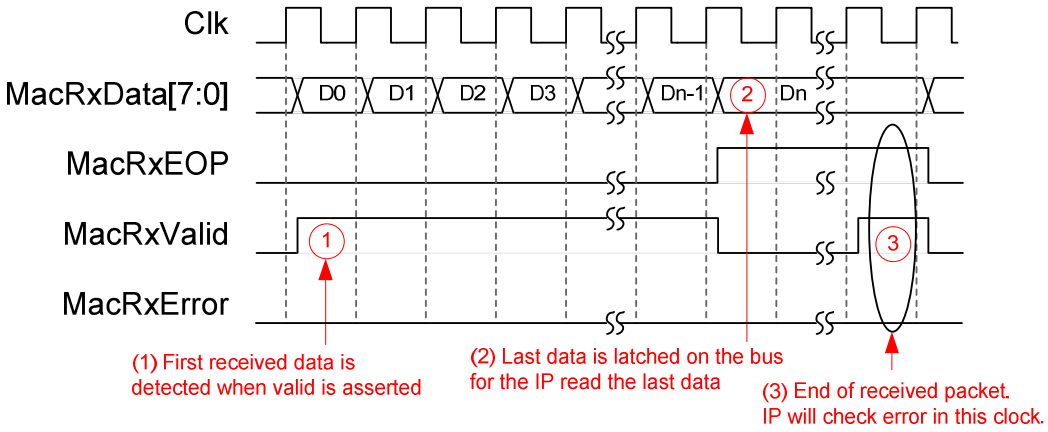

Figure 11: Receive EMAC Interface

Example usage

The example of register setting sequence for data transmission and reception is 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.

Verification Methods

The UDP1G-IP Core functionality was verified by simulation and also proved on real board design by using CycloneV E/ArriaV 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	Feb-27-2017	New release
1.1	Mar-1-2017	Update Figure11