Design Gateway Co.,Ltd
54 BB Building 14th Fl., Room No.1402 Sukhumvit 21 Rd. (Asoke), Klongtoey-Nua, Wattana, Bangkok 10110
Phone: (+66) 02-261-2277
Fax: (+66) 02-261-2290
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
Constraints Files	User constraint file
Verification	Simulation model
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on AC701
Simulation Tool Used	
Modelsim	
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- UDP/IP stack implementation
- Support IPv4 protocol
- Support one port connection
- Transmit/Receive buffer size, adjustable for optimized resource and performance
- Simple data interface by standard FIFO interface
- Simple control interface by standard register interface
- One clock domain interface by fixed 125 MHz clock frequency
- Reference designs available on AC701 evaluation board
- Not support data fragmentation feature

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	Slice Regs	Slice LUTs	Slices[1]	IOB[2]	RAMB36E1	RAMB18E1	Design Tools
Artix-7	XC7A200FBG676-2	125	1553	1416	609	133	36	1	Vivado2014.4
Kintex-7	XC7K325TFFG900-2	125	1543	1417	602	133	36	1	Vivado2014.4
Virtex-7	XC7VX485TFFG1761-2	125	1543	1417	627	133	36	1	Vivado2014.4
Zynq-7000	XC7Z045FFG900-2	125	1543	1418	611	133	36	1	Vivado2014.4

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Assuming all core I/Os and clocks are routed off-chip

3) Block memory resources are based on 64k Tx data buffer size, 16k Tx packet buffer size, and 64k 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.

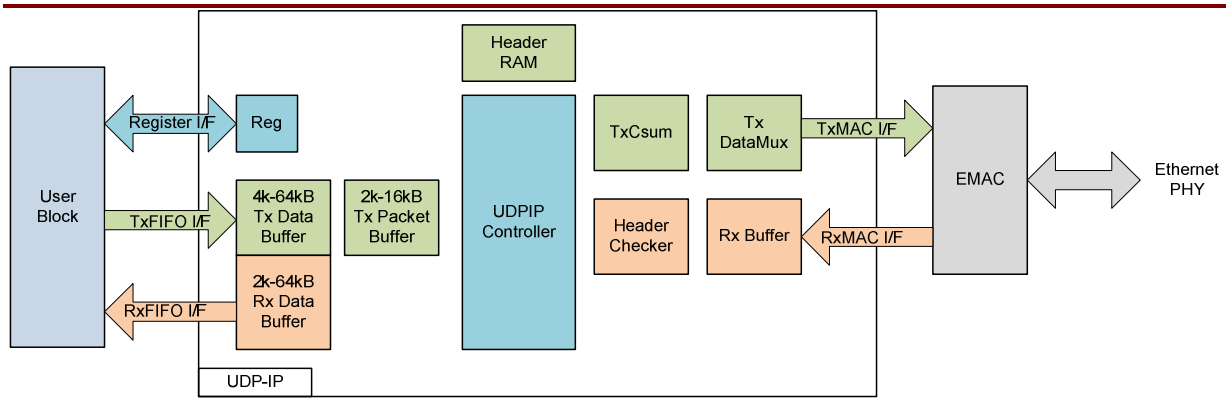

Figure 1: UDP-IP Block Diagram

Applications

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

General Description

UDP-IP core operating with Xilinx EMAC IP core can operate UDP/IP stack, Transport layer, Internet layer, and Link layer for network data transmission. User can send and receive 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 received data signal by FIFO access. During initializing system, user needs to set up system parameter such as packet size, port number, IP number through register interface. User can send command to transfer data from Tx Data buffer to external network device. Also, UDP data from external device will be stored to Rx data buffer within UDP-IP.

The size of three buffers in UDP-IP (Tx Data buffer, Tx Packet buffer, and Rx Data buffer) can be selected by setting parameter of the IP. The different size is provided to optimize resource utilization for user application. 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. Also, Rx Data buffer size must be more than two times of received packet size.

To transmit data, data from Tx Data buffer will be split into packet size and then fed to Tx Packet buffer. Data output from Tx Packet buffer will be added by header data in Header RAM before sending out to EMAC. UDP and IP checksum will be auto calculated within UDP-IP. Busy flag within register and output port will be cleared after completed data transfer size is equal to setting value from user. User can monitor this busy flag to check transfer status.

For receiving data, Rx packet will be stored to temp buffer firstly. Header and checksum within Rx packet will be verified. If header or checksum is error, the packet will be rejected and not store to Rx Data buffer. Only valid data will be stored to Rx Data buffer.

Functional Description

UDP-IP core can be divided into three parts, i.e. control block, transmit block, and received 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 for 11 registers. The description of each register address is defined as shown in Table 2. After system reset is released, all internal parameters will be updated following setting value of each register.

- **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 start data transferring to 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 again if user will change value of SML, SMH, DIP, SIP, DPN, or SPN register.**
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]	System busy flag. '0': Idle, '1': System operating. 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 will transmit data with same length, this register doesn't need to set again. Previous value will be used from internal latch.
		Rd	[31:0]	Remain 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 register is used by 125 MHz counter, so timer unit is about 8 ns. This value should be more than 0x6000.
		Rd		[0]-Timeout from not receiving ARP reply packet [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 still use same packet size, user does 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 should be at least two times of Tx Packet Size in PKL register. Transmit data from user will be stored within this buffer. To send data output from user command, one packet data will be read out from this buffer to store to Tx Packet Buffer to wait next processing.

Buffer size inside UDP-IP is not effect to transfer performance. It is used to be data buffer for user logic interface with the IP.

- **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 one packet data splitting from Tx Data Buffer. Data in Tx Packet Buffer is stored to wait until EMAC ready to receive data, and then data will be sent out.

- **Header RAM**

This RAM is applied to store header part of Transmit packet. Parameter in Header RAM will be updated by register value when user release RST register. Some parameters will be updated after receiving ARP Reply.

- **TxCsum**

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

- **TxDataMux**

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

Received 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 will be splitted out and store 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 buffer for user logic. Setting big buffer can store much received data for user logic when the logic is not available to dump data out.

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.

GEMAC

This block is softIPcore provided by Xilinx.

Core I/O Signals

Descriptions of all parameters and signal I/O are provided in Table 4 and Table 5. All signals in MAC Interface group are designed to connect to Xilinx EMAC port directly.

Table 4: Core Parameters

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.
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	Received buffer data counter to show total number of received data in buffer.
UDPRxFfRdEmpty	Out	Clk	Received buffer empty flag. User needs to stop reading data immediately.
UDPRxFfRdEn	In	Clk	Received buffer read enable. Assert to read data from Received buffer.
UDPRxFfRdData[7:0]	Out	Clk	Received buffer read data bus. Valid after UDPRxFfRdEn assert with 1 Clk period latency.

Signal	Dir	Clk	Description
MAC Interface			
MacRxClk	In		Received clock from EMAC core.
MacRxReset	In		Active-High Rx software reset from EMAC core. This signal is unused.
MacRxData[7:0]	In	MacRxClk	Received data bus from EMAC core.
MacRxValid	In	MacRxClk	Received data valid signal from EMAC. Synchronous with MacRxData.
MacRxLast	In	MacRxClk	Control signal to indicate the final byte in the frame.
MacRxUser	In	MacRxClk	Control signal asserted at the end of received frame to indicate that the frame has an error. ‘0’: normal packet, ‘1’: error packet.
MacTxReset	In		Active-High Tx software reset from EMAC core. This signal is unused.
MacTxData[7:0]	Out	Clk	Transmitted data to EMAC core.
MacTxValid	Out	Clk	Transmitted data valid signal to EMAC. Synchronous with MacTxData.
MacTxLast	Out	Clk	Control signal to indicate the final byte in a frame.
MacTxUser	Out	Clk	Control signal to indicate an error condition. This signal is always ‘0’.
MacTxReady	In	Clk	Handshaking signal. Asserted when MacTxData has been accepted.

Timing Diagram

User can write/read control signal with UDP-IP by using Register interface which has timing diagram as shown in Figure 2. 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.

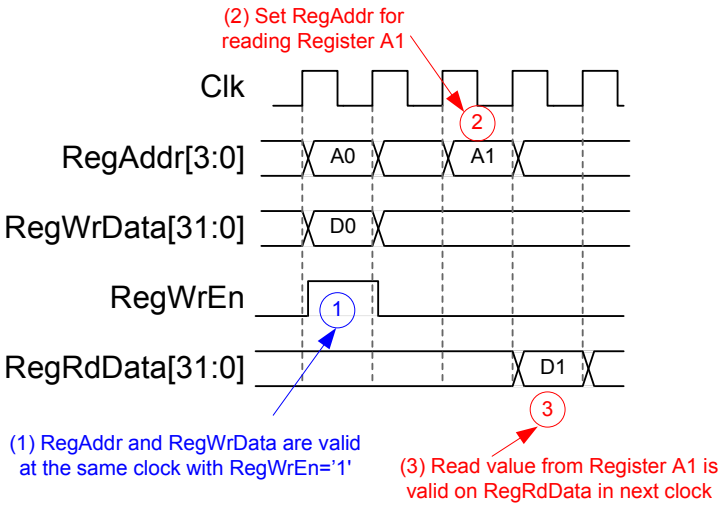

Figure 2: Register Interface Timing Diagram

To start sending data, Busy flag must be monitored through register access or output signal before writing CMD register, as shown in Figure 3. After new command is received, busy will be asserted.

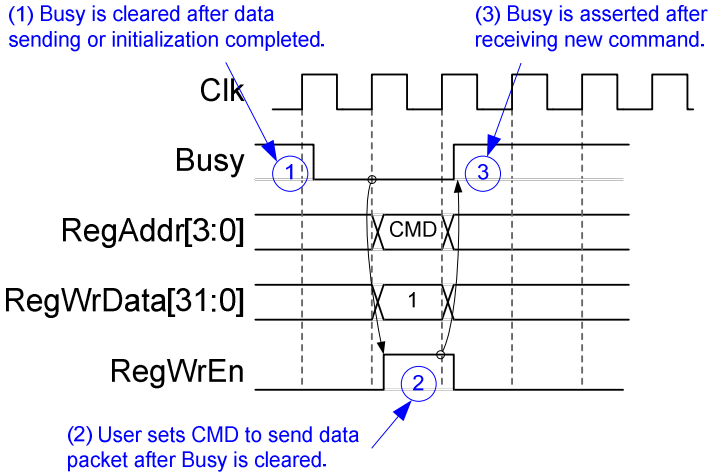

Figure 3: Set CMD register when Busy is de-asserted

User can send data to IP core by using FIFO interface, as shown in Figure 4. 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.

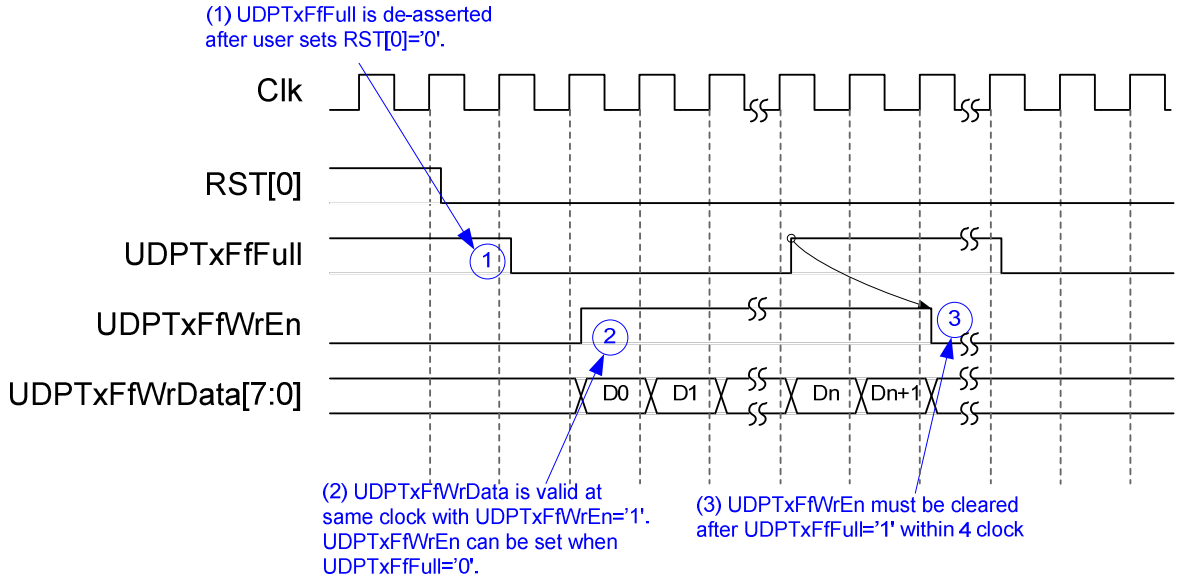

Figure 4: Tx Data Buffer Interface Timing Diagram

When IP core receives data from external, data will be stored in Rx Data buffer. User can read data from this buffer by using FIFO interface, as shown in Figure 5. User can monitor data available status from UDPRxFfEmpty, and assert UDPRxFfRdEn to read data when UDPRxFfEmpty is cleared to '0'. UDPRxFfRdData will be valid in next clock period. UDPRRxFfRdEn must be '0' at the same clock with UDPRxFfEmpty = '1' to stop data reading. Similar to Tx data buffer, Rx data buffer will be flushed when IP is reset. UDPRxFfEmpty is asserted to '1' during reset condition.

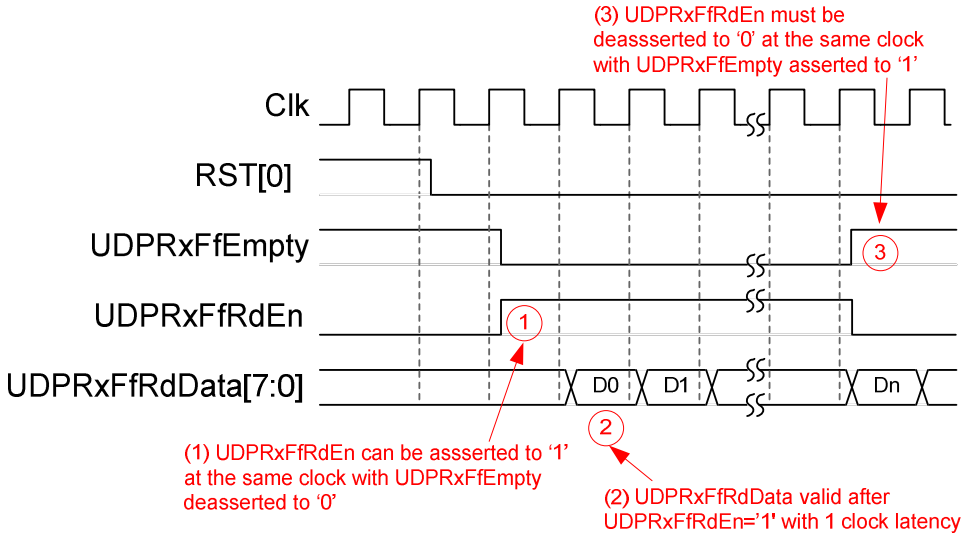

Figure 5: Rx Data Buffer Interface by Empty flag Timing Diagram

Rx data buffer status can be also monitored by using UDPRxFfRdCnt. This signal shows total number of available data in Rx data buffer. So, user can assert UDPRxFfRdEn='1' for time period equal to total number of data, as shown in Figure 6.

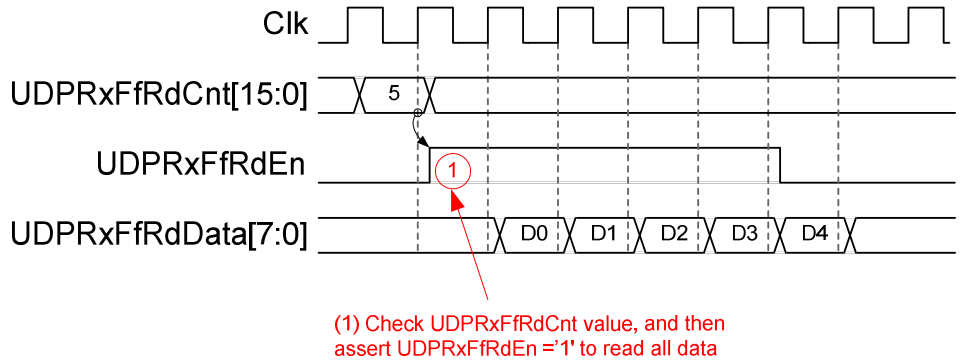

Figure 6: Rx Data Buffer Interface by Read counter Timing Diagram

Verification Methods

The UDP-IP Core functionality was verified by simulation and also proved on real board design by using AC701 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 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	Jan-4-2016	New release