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
Constraints Files	User constraint file
Verification	Test Bench, Simulation Library
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on KC705/VC707/ZC706/ KCU105/ZCU102/VCU118
Simulation Tool Used	
ModelSim SE	
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- TCP/IP stack implementation
- Support IPv4 protocol
- Support one session per one TOE10G IP (Multisession can be implemented by using multiple TOE10G IP)
- Support both Server and Client mode (Passive/Active open and close)
- Support Jumbo frame
- Packet size for transmit must be aligned to 64 bit because Transmit data bus size is 64 bit
- Received data bus size is 64-bit, so total received size must be aligned to 64-bit
- Transmitted/Received buffer size, adjustable to optimize resource and performance
- Simple data interface by standard FIFO interface
- Simple control interface by standard register interface
- 64-bit AXI4 stream to interface for 10-Gbps Ethernet MAC
- One clock domain interface by fixed 156.25 MHz clock frequency
- Reference designs available on KC705/VC707/ZC706/KCU105/ZCU102/VCU118 board
- Not support data fragmentation feature

Table 1: Example Implementation Statistics for 7-Series device

Family	Example Device	Fmax (MHz)	Slice Regs	Slice LUTs	Slices[1]	IOB	BRAMTile[2]	Design Tools
Kintex-7	XC7K325TFFG900-2	156.25	3101	3827	1326	-	36	Vivado2017.4
Zynq-7000	XC7Z045FFG900-2	156.25	3101	3827	1361	-	36	Vivado2017.4
Virtex-7	XC7VX485TFFG1761-2	156.25	3101	3828	1347	-	36	Vivado2017.4

Table 2: Example Implementation Statistics for Ultrascale device

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB[1]	IOB	BRAMTile[2]	Design Tools
Kintex-Ultrascale	XCKU040FFVA1156-2E	156.25	3106	3808	755	-	34.5	Vivado2017.4
Zynq-Ultrascale+	XCZU9EG-FFVB1156-2-I	156.25	3106	3806	736	-	34.5	Vivado2017.4
Virtex-Ultrascale+	XCU9P-FLGA2104-2L	156.25	3106	3807	704	-	34.5	Vivado2017.4

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Block memory resources are based on 64kB Tx data buffer size, 16kB Tx packet buffer size, and 64kB Rx data buffer size. Minimum size of each buffer are 4kB Tx data buffer size, 4kB Tx packet buffer size, and 16kB Rx data buffer size for jumbo frame.

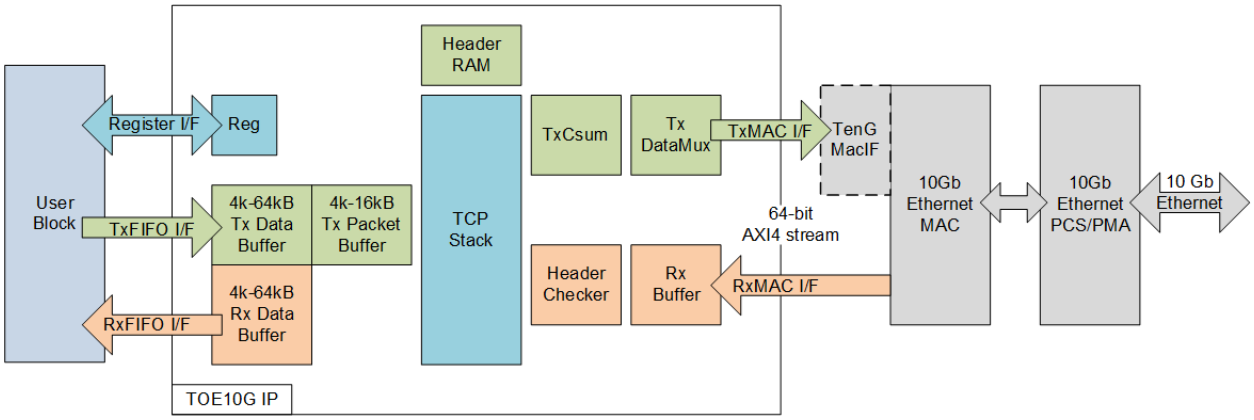

Figure 1: TOE10G IP Block Diagram

Applications

TOE10G IP is designed for network application to achieve data reliability and ultra speed performance by using TCP/IP protocol over 10 Gb Ethernet. Using TOE10G IP, the system easily transfers 10Gb Ethernet data with other network devices without using CPU and external memory. The application such as Ethernet data logger can be designed by using TOE10G IP.

General Description

TOE10G IP core implements TCP/IP stack, so it needs to connect with 10 Gb EMAC IP and 10 Gb Ethernet PCS/PMA to support the lower layer protocol for network data transmission. User sends and receives 10 Gb Ethernet data with some network devices through TCP/IP protocol by using this system.

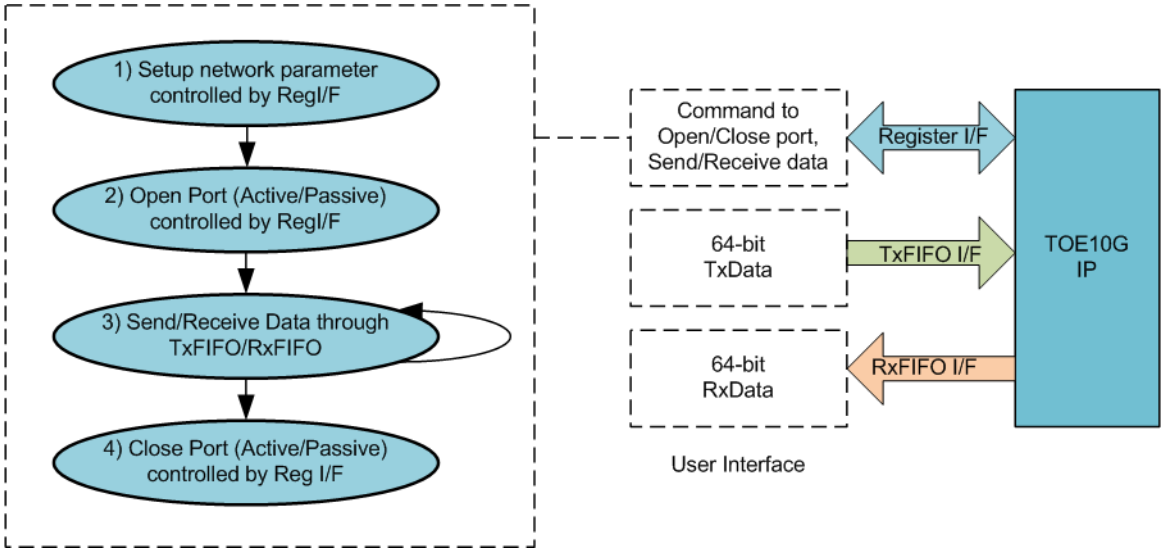

Figure 2: TOE10G IP User Interface and operation sequence

There are two types of TOE10G IP user interface, i.e. control signals by register access and data signals to transmit and receive data by FIFO access. During initializing system, user needs to set up the system parameters such as MAC address, port number, and IP address through register interface. After that, port can be opened by user logic (Active mode) or by external device (Passive mode) to create the data channel. Next, the user sends or receives data through FIFO interface which can be designed by using simple logic. After finishing all data transferring, the port can be closed by user logic (Active mode) or by external device (Passive mode) to destroy the data channel.

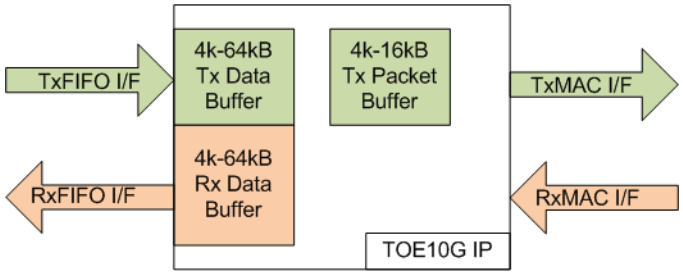

Figure 3: Adjustable Tx/Rx Buffer Size

There are three buffers in TOE10G IP, i.e. Tx Data buffer, Tx Packet buffer, and Rx Data buffer. The buffer size can be set as IP parameter which must be a trade-off between optimizing resource utilization and achieving the better performance for the user application. The bigger Tx Data buffer size and Tx Packet buffer size increase the transmit performance while the bigger Rx Data buffer size increases the receive performance.

The minimum size of Tx Data buffer and Tx Packet buffer is limited by transmit packet size, set through PKL register. Tx Packet Buffer size must be more than the value of PKL register while Tx Data Buffer size should be more than or equal to at least two times of the value of PKL register.

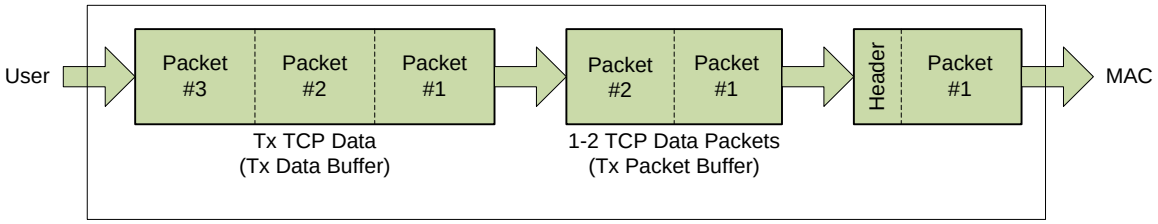

Figure 4: Transmitted Data Flow

To transmit data, data from Tx Data buffer is split into packet size and then fed to Tx Packet buffer. Data output from Tx Packet buffer is merged with the header data in Header RAM before sending out to EMAC. TCP and IP checksum are auto calculated within TOE10G IP. During transmitting data, Acknowledge number of Rx packet is decoded to be the data pointer which has already received completely. Tx Data buffer flushes the data which has received completely.

According to TCP standard, if the acknowledge number in the received ACK packet is same as the value in the previous packet, the sender will translate that the receiver detects data lost. To recover data lost, IP will retransmit the lost packet following the acknowledge number. Busy flag (monitored from CMD register) is de-asserted to '0' after total data (set by TDL register) are transferred completely. User can monitor busy flag to check transfer status.

User can change the packet size (PKL register) and total data size (TDL register) for the next request without closing the port after the IP is in Idle state

For receiving data, Rx packet is stored to the temp buffer firstly to verify Header and checksum. If the header or checksum is error, the packet will be ignored and will not be stored to Rx Data buffer. When the valid data packet is received, the data is stored to Rx Data buffer. At the same time, the acknowledge packet is sent by TOE10G IP to show the pointer of the completely received data to the source network device. After that, TOE10G IP backs to Idle state (Busy flag='0'). If the received data is not in correct sequence, the duplicate ACK will be generated to request the lost data packet.

Functional Description

TOE10G IP core can be divided into three parts, i.e. control block, transmitted block, and received block.

Control Block

- **Reg**

  User can set parameters for TCP/IP operation by using register interface. Register address of this is interface is equal to 4-bit (Maximum supported register = 16). The description of each register is defined as shown in Table 3. After system reset is released to '0', all internal parameters are updated by the set value from user.

- **TCP Stack**

  When user sends command to TOE10G IP (Active mode), the command is decoded by TCP Stack. After that, the transmit block creates the new packet to open port, close port, or send data following the command. At the same time, the received block decodes and monitors the acknowledge packet. TCP Stack controls the sequence and the parameter of the transmit block and the received block are in good status. If not, TCP Stack will stop the current operation and generate the new parameter set for the transmit block and received block for data recovery.

  The passive mode is run when the new packet is received for IP decoding in Idle status. TCP Stack decodes the received packet from the received block and decides the packet type and parameters to return the packet by the transmit block. For example, if the received packet is ARP request, TCP Stack will send the request to the transmit block to create ARP reply packet.

Table 3: Register map Definition

RegAddr [3:0]	Reg Name	Dir	Bit	Description
0000b	RST	Wr /Rd	[0]	Reset IP. '0': No reset, '1': Reset. Default value is '1'. **After all parameters are assigned, the user sets '0' to this register for loading parameter and starting system intialization. Reset needs to be set/clear again to reload parameter when user changes the value of SML, SMH, DIP, SIP, DPN, or SPN register.**
0001b	CMD	Wr	[1:0]	User command in active mode. "00": Send data, "10": Open connection (active), "11": Close connection (active), "01": Undefined. **Before setting this register to send command (active mode), user needs to confirm the system busy flag is equal to '0' by reading bit[0] of this register.** The command operation starts after the user sets this register.
		Rd	[0]	System busy flag. '0': Idle, '1': IP is busy.
			[3:1]	Current operation. "000": Send data, "001": Idle, "010": Active open connection, "011": Active close connection, "100": Receive data, "101": Initialization, "110": Passive open connection, "111": Passive close connection.
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	[15:0]	Define 16 bit target port number. **To run active open command, user needs to set this register before clearing RST register.** For passive open, this register does not need to set. Target port number is auto defined by decoding from the header of the valid received packet which has the correct network parameters (the parameters are matched to the value in SML, SMH, DIP, SIP, and SPN 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 in byte unit, but the size must be aligned to 8 byte. Valid from 8-0xFFFFFFF8 (Bit[2:0] is ignored by the IP). **User needs to set this register before setting CMD register = "00" (Send data). This value is latched to the internal logic when CMD register is set. After the command is run (Busy='1'), the user can set the new length for the next send command.** If the next transmission uses the same length, this register will not need to set again. TOE10G IP uses the latest value for the next transmission.
		Rd	[31:0]	Remaining 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 Rx packet during running the command. The counter is run under 156.25 MHz, so timer unit is equal to 6.4 ns. This value is recommended to be more than 0x6000.
		Rd		[0]-Timeout from not receiving ARP reply packet After timeout, IP resends ARP request until ARP reply is received. [1]-Timeout from not receiving SYN and ACK flag during active open operation After timeout, IP resends SYN packet for 16 times and then sends FIN packet to close connection. [2]-Timeout from not receiving ACK flag during passive open operation After timeout, IP resends SYN/ACK packet for 16 times and then sends FIN packet to close connection. [3]-Timeout from not receiving FIN and ACK flag during active close operation After the 1st timeout, IP sends RST packet to close connection. [4]-Timeout from not receiving ACK flag during passive close operation After timeout, IP resends FIN/ACK packet for 16 times and then sends RST packet to close connection. [5]-Timeout from not receiving ACK flag during data transmit operation After timeout, IP resends the previous data packet. [6]-Timeout from Rx packet lost, Rx data FIFO full, or wrong sequence number IP generates duplicate ACK to request data retransmission. [7]-Timeout from too small received window size (received window size is less than packet size) when running Send data command. After timeout, IP retransmits data packet (like TMO[5] phenomenon). Data retransmission in this situation is controlled by PSH[2]. If PSH[2] is not set to '1', this bit will not been asserted to '1'. [21]-Lost flag when the sequence number of the received ACK packet is skipped. After that, Timeout is found with asserting TMO[6] to '1'. [22]-FIN flag is detected during sending opertion. [23]-Rx packet is ignored due to Rx data buffer full (fatal error). [27]-Rx packet lost detected [30]-RST flag is detected in Rx packet [31],[29:28],[26:24]-Internal test status
1010b	PKL	Wr /Rd	[15:0]	Data length of Tx packet in byte unit, but the packet length must be aligned to 8 byte. Valid from 8-16000. Default value is 1456 byte (Maximum size with 8 byte alignment for non-jumbo frame). Bit[2:0] of this register is ignored by the IP. **This value must not be changed when Send data command still be run (Busy='1').** **If the next command uses the same packet size, this register will not need to set again.** **TOE10G IP uses the lastest value for the next transmission like TDL register.**

RegAddr [3:0]	Reg Name	Dir	Bit	Description
1011b	PSH	Wr /Rd	[2:0]	Sending mode when running Send data command. [0]-Disable to retransmit packet. Set '0' to generate the duplicated data packet for the last data packet in Send data command. (Default = '0'). [1]-Enable to set PSH flag in the transmit packet. Set '1' to insert PSH flag in TCP header of all transmitted packet (Default = '0'). [2]-Enable to retransmit data packet when Send data is paused until timeout from too small received windows size. Set '1' to enable this feature. This flag is designed to solve the problem when the window update packet is lost. (Default = '0'). Data retransmission activates the destination device to regenerate ACK packet. The following conditions must be found to start data retransmission. (1) PSH[2] is set to '1'. (2) The current command is Send data and all data are still not completely sent. (3) The received window size is smaller than the packet size. (4) Timer set by TMO register is overflowed.
1100b	WIN	Wr /Rd	[5:0]	Threshold value in 1Kbyte unit for sending windows update packet. Default value is 0 (Not enable window update feature). The IP transmits the window update packet when the free space of the received buffer is increased from the value in the latest transmit packet more than the threshold value. For example, the user sets WIN="000001b" or 1 Kbyte. Assume that the IP sends the packet and the window size in the packet (refer to the free space of the received buffer in the IP) is equal to 2 Kbyte. After the user reads 1 Kbyte data from the IP, the free space of the received buffer is increased to 3 Kbyte. The IP detects that the window size is incresed more than the threshold. As a result, the IP sends the window update packet to update the window size value to be 3 Kbyte.
1101b	ETL	Wr	[31:0]	Extended total Tx data length in byte unit. The size must be aligned to 8 byte. Bit[2:0] is ignored by the IP. User sets this register during running send command (CMD="00") to increase total Tx data length. So, the data can be transmitted continuously without re-sending the new command to IP. The caution point is that 1) ETL register must be programmed when read value of TDL is not less than 128 Kbyte. 2) The set value of ETL must be less than (0xFFFFFFF8 – read value of TDL). For example, the user sets TDL = 3.5 Gbyte. After the IP sends 2 Gbyte data (remaining size = 1.5 Gbyte), the user sets ETL register = 1.5 Gbyte.The total length is increased to 5 Gbyte (3.5 Gbyte which is the original value of TDL + 1.5 Gbyte which is the ETL value).
1110b	SRV	Wr/ Rd	[0]	'0': Client mode. After IP reset is changed from '1' to '0', the IP sends ARP request to get Target MAC address from the IP address. When the IP receives ARP reply, IP busy is deasserted to '0'. '1': Server mode. After IP reset is changed from '1' to '0', the IP waits ARP request from the Target to get Target MAC address. When the IP receives ARP request, the IP generates ARP reply. After that, the IP busy is deasserted to '0'. Default value is '0' (Client mode) **Note: In Server mode, when the user reset the IP, the Target needs to resend ARP request to TOE10G IP to complete the IP initialization.**

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

Transmitted 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 4. The buffer size should be more than or equal to two times of Tx Packet Size, set in PKL register. Transmitted data from user is stored to this buffer. Data in the buffer is flushed after the target returns acknowledge packet to confirm that data is received completely. Data from this buffer is forwarded to Tx Packet Buffer which is the buffer to store the next transmitt packet.

This buffer size is effect to the total performance. If the size is much enough, IP can send data out continuously without waiting acknowledge packet returned from the target. So, data latency from all processes and the carrier is not effect to the performance.

If total data from user is more than the total transmit size, remaining data will be available in the buffer for the next transfer. The data in the buffer is flushed when the connection is closed or reset. If the data in the buffer is not enough for the current transaction, the IP will not send out the packet. The IP waits until the data from user is much enough for creating one packet.

- **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 4. This buffer size must be more than Tx Packet size (setting in PKL register) to store at least one packet data from Tx Data Buffer. Maximum value of PKL register is equal to (Tx Packet Buffer size<byte> – 24). Data in Tx Packet Buffer is sent out when EMAC and the target are ready to receive data. At the same time as reading data to send the packet, the next packet is received from Tx Data buffer to continue data sending.

- **Header RAM**

This RAM is applied to store the header part of transmitted packet. During IP initialization, the network parameters set by register are loaded to Header RAM. Some parameters such as Target MAC address and Target port number are updated by ARP reply packet (Client mode), ARP request packet (Server mode), and Passive open packet.

- **TxCsum**

This module is desigend to calculate checksum of Tx packet before sending out. After finishing checksum calculation, the result is loaded to Header RAM.

- **TxDataMux**

This module is designed to merge the header from Header RAM to the data from Tx Packet buffer for creating Ethernet packet.

Received Block

- **Rx Buffer**

All received packets from EMAC are stored to this buffer. Only TCP data in the valid packet is forwarded from Rx Buffer to Rx Data Buffer by Header Checker module.

- **Header Checker**

The header in Rx packet are verifed by this module to validate the packet. The packet is valid when the following conditions are met.

(1) Network parameters are matched to the set valid in register, i.e. MAC address, IP address, and Port number.
(2) The packet is ARP packet or TCP/IPv4 packet without data fragment flag.
(3) IP header length and TCP header length are valid (IP header length is equal to 20 bytes and TCP header length is from 20 to 60 bytes).
(4) IP checksum and TCP checksum are correct.
(5) The data pointer decoded by the sequence number is in valid range.
(6) The acknowledge number is in valid range.

- **Rx Data Buffer**

This buffer size is set by “RxBufBitWidth” parameter of the IP. The valid value is 9-13, as shown in Table 4. Free space size of this buffer size is applied to be the received window size in transmitted packet. Setting bigger size of this buffer may increase the received performance because the data source continues sending data without waiting the acknowledge packet returned from TOE10G IP which may be delayed from many factors such as the network routing, the process within the data source, and the received buffer full. Otherwise, bigger buffer increases the chance to rearrange the received data when the received packet sequence is swapped from network routing.

User Block

This is the user module to interface with TOE10G IP. The parameters are set and monitored through the register interface. The data interface is standard FIFO interface. User sends data to TOE10G IP through TxFIFO interface and read data from TOE10G IP through RxFIFO interface. This module can be designed by simple hardware logic.

10G Ethernet MAC

Ethernet MAC implements the MAC layer for 10Gb Ethernet. The user interface to connect with TOE10G IP is 64-bit AXI4 stream while the PHY interface is 64-bit XGMII interface. Design Gateway provides 10G EMAC IP which optimizes the resource and minimizes the data latency when connecting with TOE10G IP. More details of DG 10G EMAC IP Core are described in following website.

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

Otherwise, Xilinx provides 10G Ethernet MAC including many features which has more details provided in following website.

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

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

TOE10G IP can connect with DG EMAC IP directly while the special logic with small FIFO (TenGMacIF in Figure 1) must be designed to connect TOE10G IP with Xilinx EMAC IP.

10G Ethernet PCS/PMA (10GBASE-R)

This module is a no charge Xilinx LogiCORE which can read more details from following website.

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 the address bus size of this buffer.
TxPacBitWidth	9-11	Setting Tx Packet buffer size. The value is the address bus size of this buffer.
RxBufBitWidth	9-13	Setting Rx Data buffer size. The value is the 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 to synchronous with the user interface and EMAC interface.
User Interface		
RegAddr[3:0]	In	Register address bus
RegWrData[31:0]	In	Register write data bus. Synchronous to RegAddr signal for write process.
RegWrEn	In	Register write enable pulse. Synchronous to RegAddr and RegWrData signals.
RegRdData[31:0]	Out	Register read data bus. Valid in the next clock after RegAddr is valid.
ConnOn	Out	Connection Status (‘1’: connection is opened, ‘0’: connection is closed)
TimerInt	Out	Timer interrupt. Assert to high for 1 clock cycle when timeout is detected. More details of Interrupt status could be checked from TMO[7:0] register.
RegDataA1[31:0]	Out	32 bit read value of CMD register (RegAddr=0001b)
RegDataA8[31:0]	Out	32 bit read value of TDL register (RegAddr=1000b)
RegDataA9[31:0]	Out	32 bit read value of TMO register (RegAddr=1001b)
Tx Data Buffer Interface		
TCPTxFfFlush	Out	Transmit buffer within IP is reset. Assert to ‘1’ only 1 clock cycle when the connection is closed or the IP is reset.
TCPTxFfFull	Out	Transmitted buffer full flag. User needs to stop writing data within 4 clock period after this flag is asserted to ‘1’.
TCPTxFfWrEn	In	Transmitted buffer write enable. Assert to ‘1’ to write data to Transmit buffer
TCPTxFfWrData[63:0]	In	Transmitted buffer write data bus. Synchronous with TCPTxFfWrEn.
Rx Data Buffer Interface		
TCPRxFfFlush	Out	Received buffer within the IP is reset. Assert to ‘1’ only 1 clock cycle when the connection is opened.
TCPRxFfRdCnt[12:0]	Out	Received buffer data counter to show total received data in the buffer as 64 bit unit.
TCPRxFfLastRdCnt[2:0]	Out	Remaining byte of the last data in the received buffer when total received data in the buffer is not aligned to 8 byte unit.
TCPRxFfRdEmpty	Out	Received buffer empty flag. User needs to stop reading data immediately when this signal is asserted to ‘1’.
TCPRxFfRdEn	In	Received buffer read enable. Assert to ‘1’ to read data from Received buffer.
TCPRxFfRdData[63:0]	Out	Received buffer read data bus. Valid in the next clock cycle after TCPRxFfRdEn is asserted to ‘1’.

Signal	Dir	Description
MAC Interface		
tx_axis_tdata[63:0]	Out	Transmitted data.
tx_axis_tkeep[7:0]	Out	Tranmistted data byte enable. Synchronous with tx_axis_tdata.
tx_axis_tvalid	Out	Transmitted data valid signal. Synchronous with tx_axis_tdata.
tx_axis_tlast	Out	Control signal to indicate the final word in the frame.
tx_axis_tuser	Out	Control signal to indicate an error condition. This signal is always '0'.
tx_axis_tready	In	Handshaking signal. Asserted to '1' when tx_axis_tdata has been accepted. This signal must not be de-asserted to '0' when packet is transmitting.
rx_axis_tdata[63:0]	In	Received data.
rx_axis_tvalid	In	Received data valid signal. Synchronous with rx_axis_tdata. rx_axis_tvalid must be asserted to '1' continuously when packet is transmitting.
rx_axis_tlast	In	Control signal to indicate the final word in the frame.
rx_axis_tuser	In	Control signal asserted at the end of received frame to indicate that the frame has an error. '1': normal packet, '0': error packet.
rx_axis_tready	Out	Handshaking signal. Asserted to '1' when rx_axis_tdata has been accepted. rx_axis_tready is de-asserted to '0' for 2 clock cycles to be the gap size between each received packet.

Timing Diagram

IP Initialization

For initialization process after user changes RST register from ‘1’ to ‘0’, TOE10G IP supports to run in two modes depending on SRV register setting, i.e. Client mode (SRV=’0’) and Server mode (SRV=’1’).

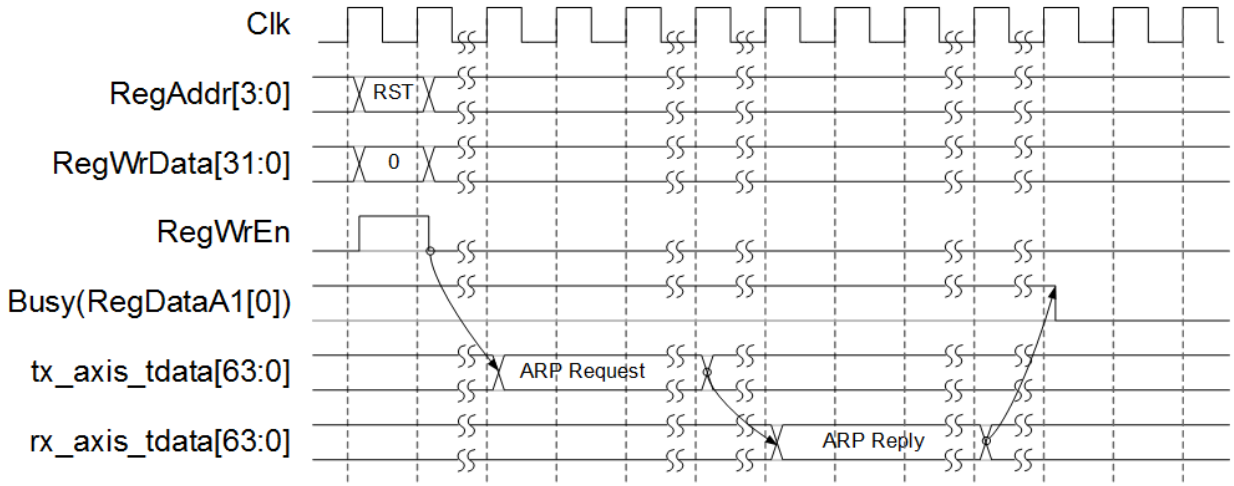

Figure 5: IP Initialization in Client mode

In Client mode, TOE10G 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’.

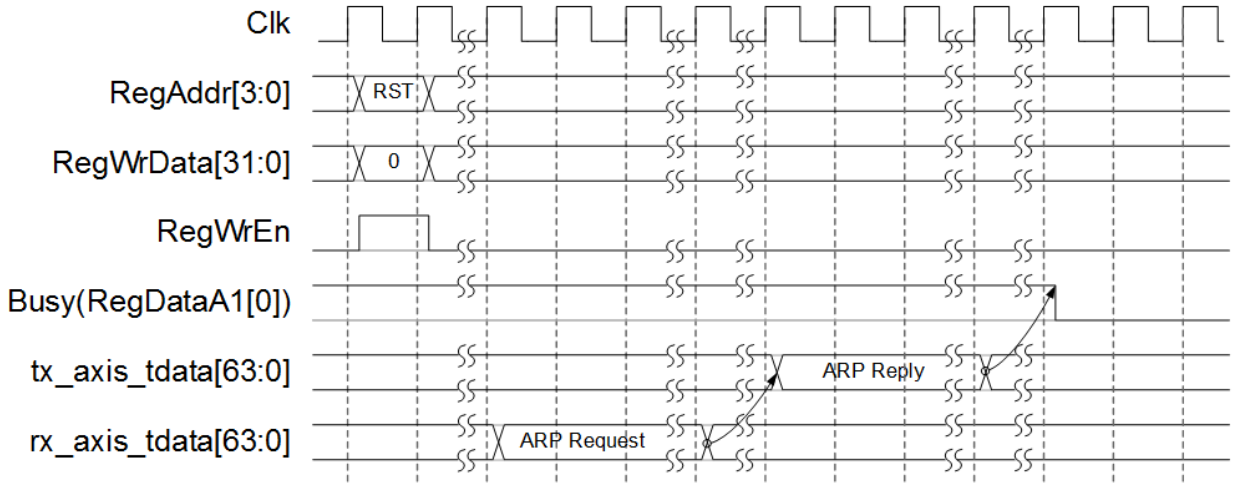

Figure 6: IP Initialization in Server mode

In Server mode, after TOE10G IP reset is changed from ‘1’ to ‘0’, TOE10G IP waits ARP request from the target. After receiving ARP request which has the matched network parameters in the header, TOE10G IP returns ARP reply to the target. Target MAC address is extracted from ARP request packet. Finally, Busy signal is de-asserted to ‘0’.

Register Interface

Network parameters, command, and status signals of TOE10G IP are set and monitored through Register interface. Timing diagram of the register interface is shown in Figure 7.

Register map address is designed as shown in Table 3. To write the register, the user sets RegWrEn='1' with the valid value of RegAddr and RegWrData. To read the register, the user sets only RegAddr and then RegRdData is valid in the next clock cycle.

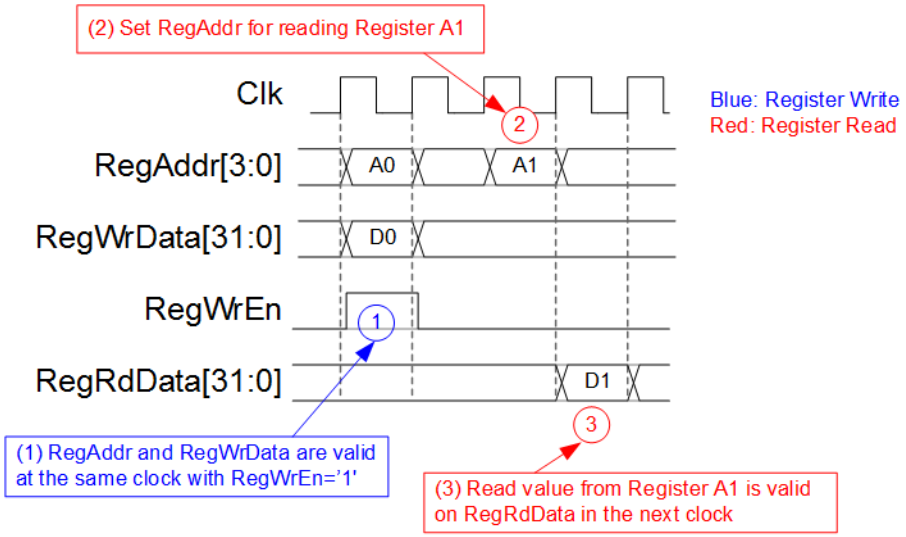

Figure 7: Register Interface Timing Diagram

Before the user sets CMD register to start the new command operation, Busy flag must be equal to '0' to confirm that IP is in Idle status. After CMD register is set, Busy flag is asserted to '1', as shown in Figure 8. Busy is de-asserted to '0' when the command is completed.

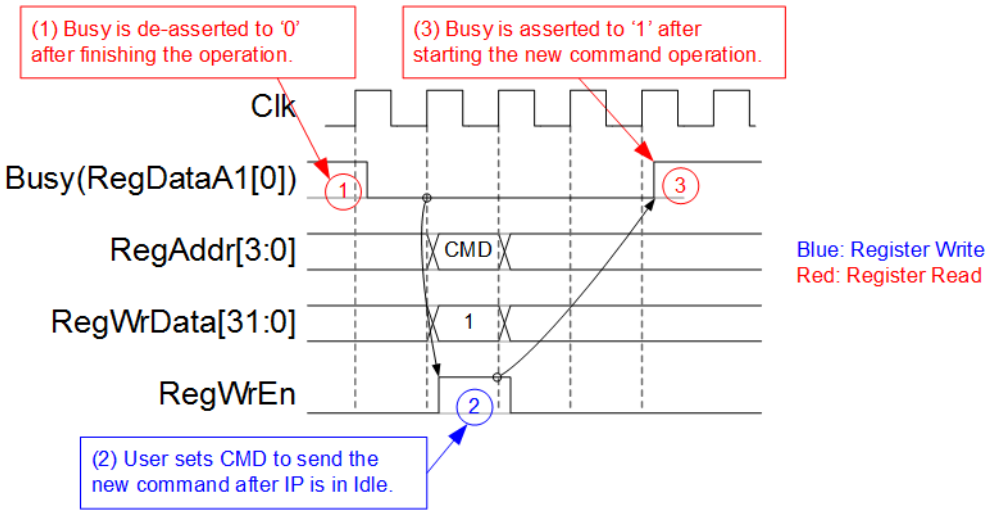

Figure 8: Set CMD register when busy is de-asserted

Tx FIFO Interface

User sends data to IP core by using FIFO interface, as shown in Figure 9.

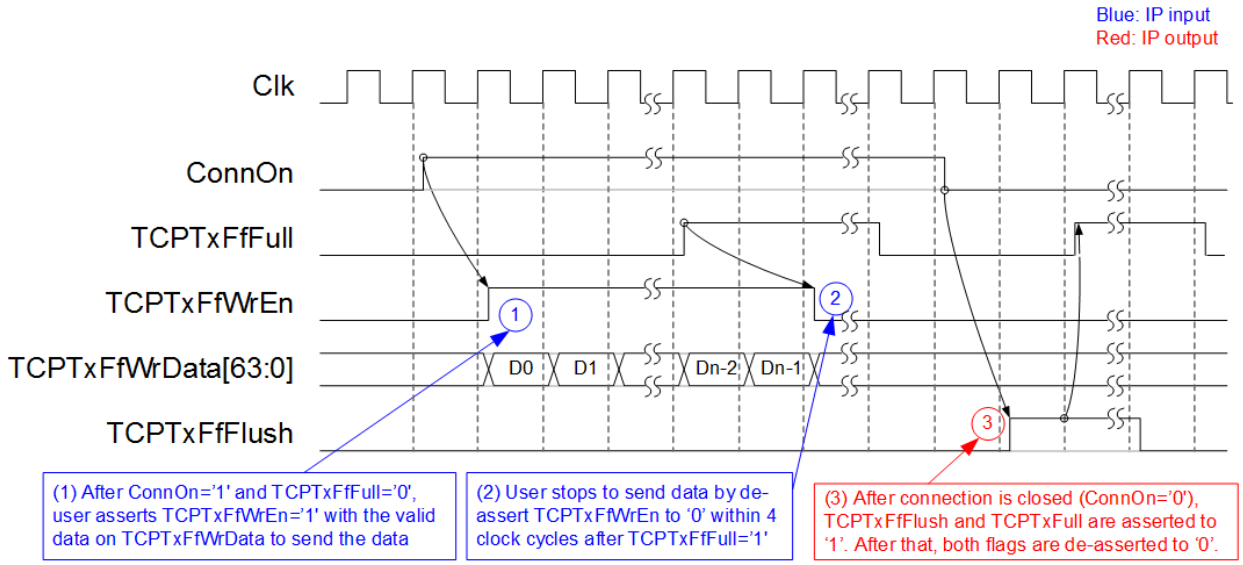

Figure 9: Tx Data Buffer Interface Timing Diagram

(1) Before sending data, user needs to check two conditions. First, full flag (TCPTxFfFull) is not asserted to '1'. Second, ConnOn must be equal to '1'. After two signal conditions are met, the user sets TCPTxFfWrEn='1' with valid value of TCPTxFfWrData.

(2) TCPTxFfWrEn must be de-asserted to '0' within 4 clock cycles to stop data sending after TCPTxFfFull is asserted to '1'.

(3) The port is closed by TOE10G IP or the destination device when there is no data for transferrring. ConnOn changes from '1' to '0' when connection is closed. After that, TCPTxFfFlush is asserted to '1' to flush the data inside TxFIFO and TCPTxFfFull is asserted to '1'.

Rx FIFO Interface

The received data extracted from the valid received packet is stored in Rx Data buffer. User can read the data from the buffer through FIFO interface, as shown in Figure 10.

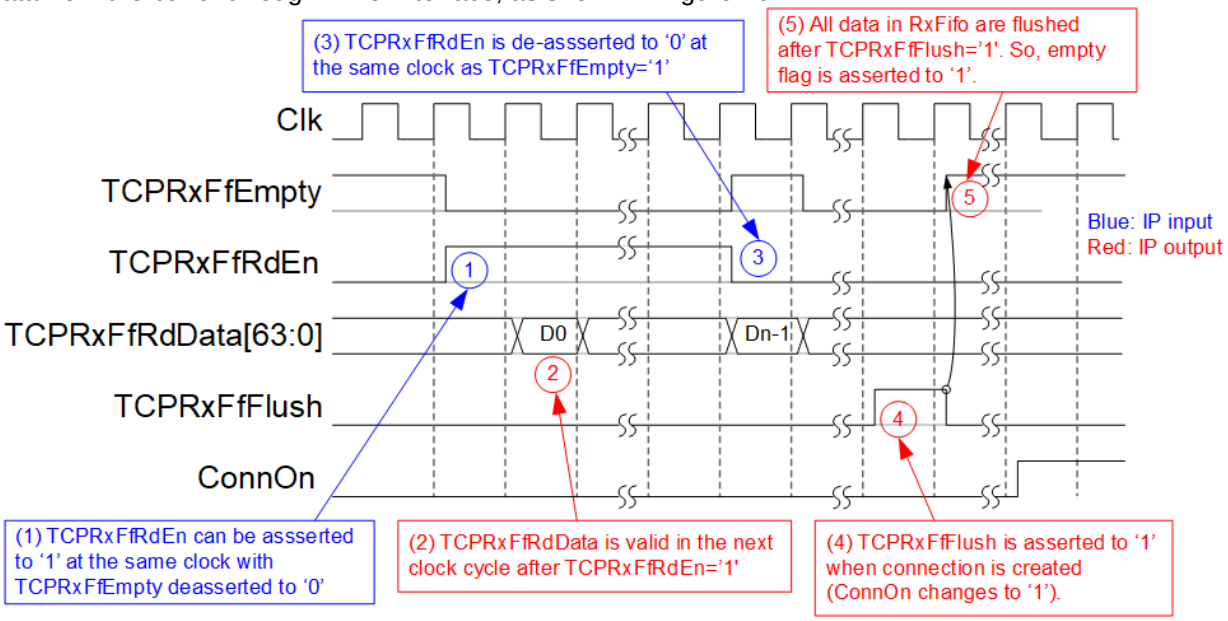

Figure 10: Rx Data Buffer Interface by Empty flag Timing Diagram

(1) User monitors the available data available from TCPRxFfEmpty. Data can be read when TCPRxFfEmpty is cleared to '0'. TCPRxFfRdEn is asserted to '1' to read data from Rx data buffer.
(2) TCPRxFfRdData is valid in next clock cycle.
(3) Data reading must be immediately stopped by de-asserting TCPRRxFfRdEn='0' when TCPRxFfEmpty='1'.
(4) Rx data buffer flushes all data when the connection is new created (ConnOn changes from '0' to '1'). Flush operation of Rx data buffer is monitored through TCPRxFfFlush signal.
(5) After Flush operation, TCPRxFfEmpty is asserted to '1'.

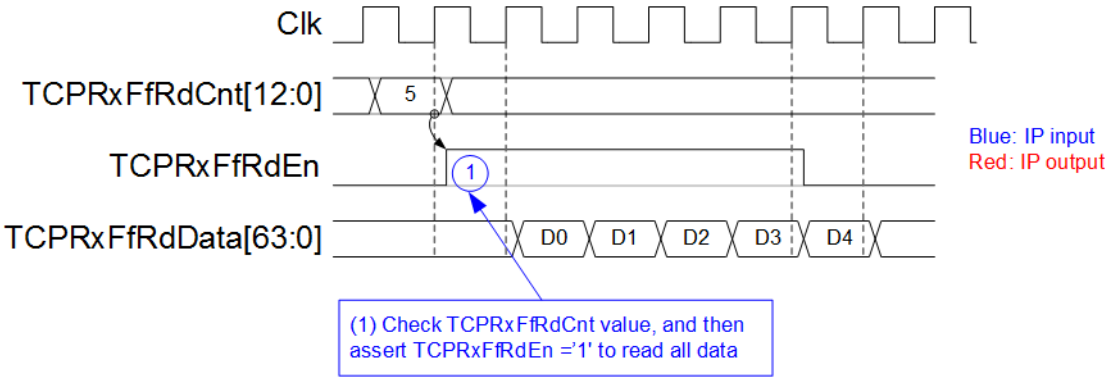

Figure 11: Rx Data Buffer Interface by Read counter Timing Diagram

Otherwise, Rx data buffer status can be also monitored by using TCPRxFfRdCnt. This signal shows total data in Rx data buffer as 64 bit unit. So, user can assert TCPRxFfRdEn='1' for many clock cycles, following the value of TCPRxFfRdCnt as shown in Figure 11.

EMAC Interface

EMAC interface of TOE10G IP is designed by using 64-bit AXI4-stream interface. The limitation is that TOE10G IP cannot pause data transmission when the packet still not end. So, tx_axis_tready must be asserted to '1' during packet transmission. Tx_axis_tready can be de-asserted to '0' after the last data in the packet is transferred, as shown in Figure 12.

From the limitation, TOE10G IP can connect with DG 10G EMAC IP core directly, but special logic with small FIFO must be added to interface between TOE10G IP and Xilinx 10G EMAC IP.

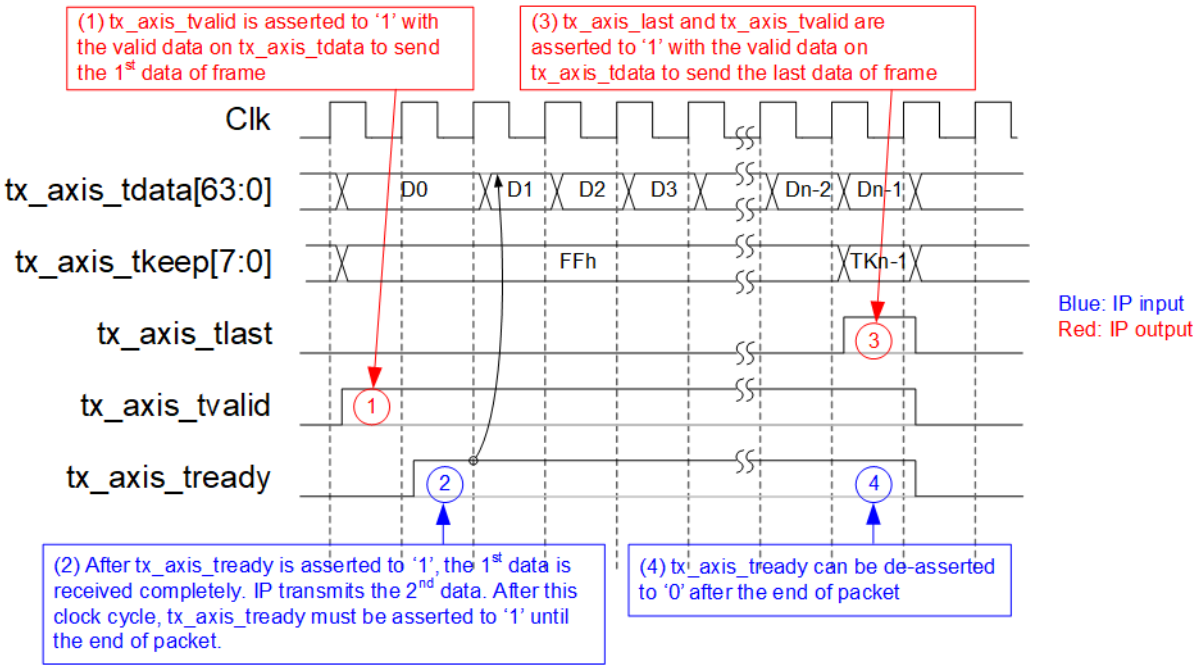

Figure 12: Transmitted EMAC Interface

(1) TOE10G IP asserts tx_axis_tvalid with the first data of the packet. All signals are latched until tx_axis_tready is asserted to '1' to acknowledge the data transmit request.
(2) TOE10G IP sends the next data after tx_axis_tready is asserted to '1'. The data is sent continuously until the end of frame. So, tx_axis_tready must be asserted to '1' until the end of frame.
(3) tx_axis_tlast and tx_axis_tvalid are asserted to '1' with the last transmit data to show end-of-packet status.
(4) After end of frame, tx_axis_tready can be asserted to '0' to pause the next packet transmission.

Timing diagram of EMAC interface for sending data to TOE10G IP is shown in Figure 13. Similar to tx_axis_* interface, data of one packet transferring through rx_axis_* interface must be sent continuously until the end of packet. Rx_axis_tvalid and rx_axis_tready must be always asserted to '1' between start of packet and end of packet.

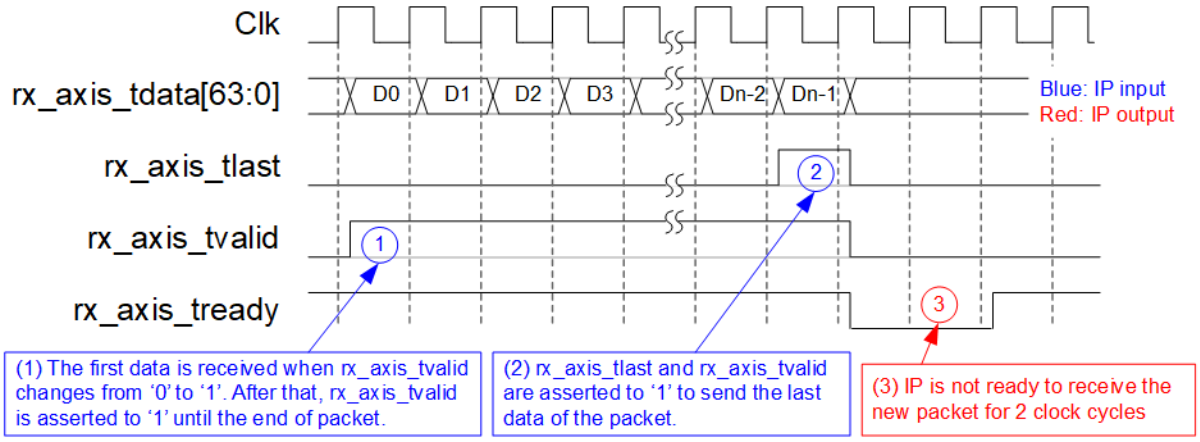

Figure 13: Received EMAC Interface

(1) TOE10G IP detects start of the received frame when rx_axis_tvalid changes from '0' to '1'. At the same time, the 1st received data is valid on rx_axis_tdata. After rx_axis_tready is asserted to '1' to acknowledge the 1st data, the next data in the same packet will be sent to TOE10G IP continuously. rx_axis_tvalid must be asserted to '1' until the end of frame.

(2) End of the received frame is detected when rx_axis_tlast='1' and rx_axis_tvalid='1'. The last data of the frame is available on rx_axis_tdata.

(3) After that, TOE10G IP de-asserts rx_axis_tready for 2 clock cycles to complete the packet post processing. So, EMAC must support to pause each data packet for 2 clock cycles.

Example usage

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

The example sequence to set register for sending and receiving data 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. (DPN is optional setting when the port is opened by IP or active open).
3) Set RST register='0' to clear the reset. After that, IP starts initialization by sending ARP request packet to get Target MAC address from ARP reply packet. Busy signal is cleared to '0' when finishing the initialization process.
4) The new connection is created by two modes.
   a. Active open: Write CMD register = "Open connection" to create the connection (SYN packet is firstly sent from TOE10G IP).
   b. Passive open: Wait until "ConnOn" signal = '1' (the target device sends SYN packet to TOE10G IP firstly).
5) a. For data transmission, set TDL register (total transmitted length) and PKL register (packet size). Next, set CMD register = "Send Data" to start data transmission. The user sends the data to TOE10G IP through TxFIFO interface before or after setting CMD register. When the command is finished, busy flag changes to'0'. The user can set the new value to TDL/PKL register and then set CMD register = "Send Data" to start the next transmission.
   b. For data reception, user monitors RxFIFO status and reads data until RxFIFO is empty.
6) Similar to creating the connection, the connection is destroyed by two modes.
   a. Active close: Set CMD register = "Close connection" to close the connection (FIN packet is firstly sent by TOE10G IP).
   b. Passive close: Wait until "ConnOn" signal = '0' (FIN packet is sent from the target to TOE10G IP firstly).

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 packet after TOE10G IP sends ARP request packet. In Server mode, MAC address is decoded from ARP request packet 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 clear the reset. IP starts initialization process by waiting ARP request packet to get Target MAC address. Next, the IP creates ARP reply packet returned to the Target. After finishing the intialization, busy signal is cleared to '0'.
4) Remaining steps are similar to step 4 – 6 of Client mode

Verification Methods

The TOE10G IP Core functionality was verified by simulation and also proved on real board design by using KC705/VC707/ZC706/KCU105/ZCU102/VCU118 evaluation board.

Recommended Design Experience

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

Ordering Information

This product is available directly from Design Gateway Co., Ltd. Please contact Design 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	May-29-2014	New release
1.1	Sep-9-2014	Update IP to support full-duplex
1.2	Sep-24-2014	Update valid value of Rx Data Buffer in page7
1.3	Nov-14-2014	Add ZC706 board support
1.4	Oct-20-2015	Add PSH, WIN, and ETL register, and RegDataAx port
1.5	Nov-3-2015	Update read value of CMD[3:1] register
1.6	Dec-23-2015	Update register to support readback
1.7	Feb-23-2017	Add KCU105 support and TCPRxFfLastRdCnt signal
1.8	Jan-18-2018	Add SRV register and rx_axis_tready signal
1.9	Aug-21-2019	Add support VCU118 board and DG EMAC IP