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

Core Facts

Provided with Core	
Documentation	User Guide, Design Guide
Design File Formats	Encrypted File
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on AC701, KC705, VC707, ZC706, Zynq Mini-ITX(Z100)
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- TCP/IP stack implementation
- Support IPv4 protocol
- Support one session per one TOE1G IP (Multisession can be implemented by using multiple TOE1G IPs)
- Support both Server and Client mode (Passive/Active open and close)
- Various Transmit/Receive buffer size (2KB, 4KB, 8KB, 16KB, 32KB, and 64KB)
- Simple data interface by standard FIFO interface at 8-bit data bus
- Simple control interface by 32-bit single port RAM interface
- 8-bit AXI4 stream to interface with Xilinx Ethernet MAC
- One clock domain interface by fixed 125 MHz clock frequency
- Reference designs available on AC701/KC705/VC707/ZC706/Zynq Mini-ITX(Z100) board
- Not support data fragmentation feature
- Customized service for following features
  - Jumbo frame support
  - Buffer size extension by using Windows Scaling feature
  - Network parameter assignment by other methods

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	Slice Regs	Slice LUTs	Slices[1]	RAMB36E1[2]	RAMB18E1[2]	Design Tools
Artix-7	XC7A200T-2FBG676	125	2616	2689	985	36	3	Vivado2017.4
Kintex-7	XC7K325T-2FFG900	125	2608	2694	1059	36	3	Vivado2017.4
Zynq-7000	XC7Z045-2FFG900	125	2608	2688	981	36	3	Vivado2017.4
Virtex-7	XC7VX485T-2FFG1761	125	2608	2691	1021	36	3	Vivado2017.4

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

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

Applications

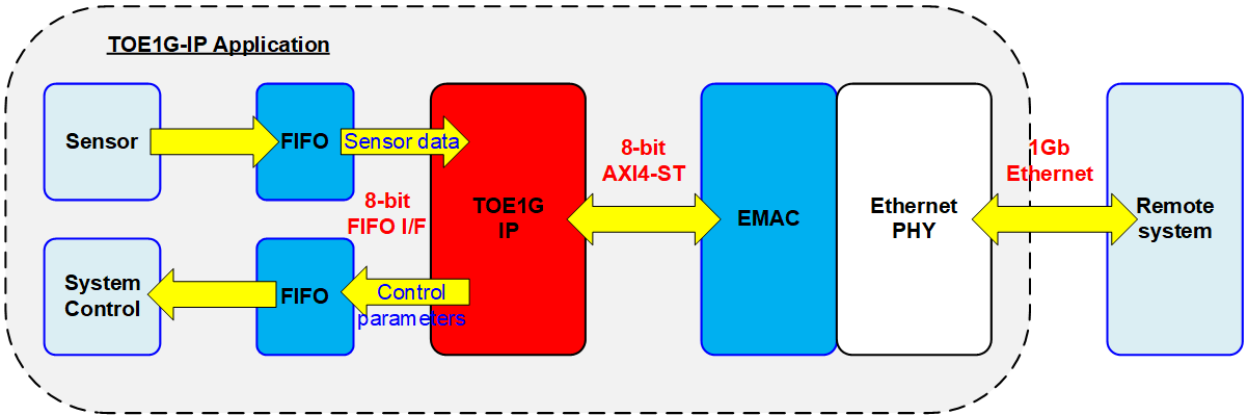

Figure 1: TOE1G IP Application

1 Gb Ethernet is the well-known communication channel for transferring data with remote controlling system. When combined with the TCP/IP protocol, this system can ensure reliable data transfer. TOE1G IP is an integrated IP that enables data transfer via 1Gb Ethernet without requiring the use of CPU or external memory. It is suitable for applications that require high-speed data transfer, such as video data streaming and sensor monitoring system using FPGA solutions.

Figure 1 shows an example of a sensor monitoring system, where data from the sensor is stored in a FIFO and forwarded to a remote system via 1Gb Ethernet using the TOE1G IP. The TOE1G IP supports full-duplex transfer in the same TCP session, allowing the remote system to send parameters for controlling the sensor monitoring system via 1Gb Ethernet. Our website also offers an FTP server demo that utilizes the TOE1G IP. For more information about this demo, please contact us.

TOE1G IP is designed to transfer data at the high speed, resulting a latency time that is increased by the internal pipeline register and buffer. For low-latency applications such as FinTech, we recommend the user of our low-latency IP. More details about this IP can be found on our website.

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

General Description

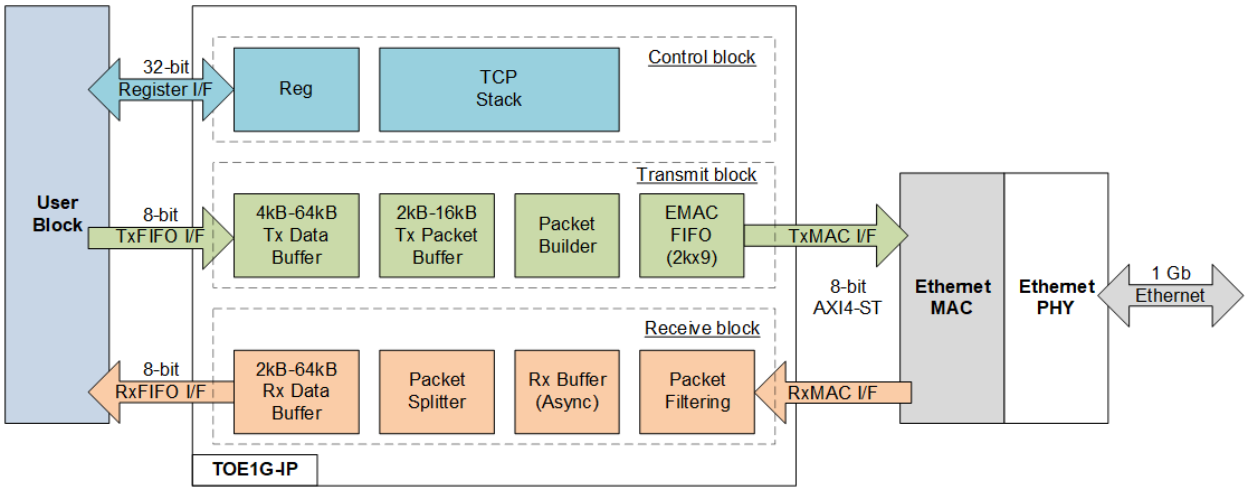

Figure 2: TOE1G IP Block Diagram

The TOE1G IP core is a hardwired logic implementation of the TCP/IP stack that connects with the EMAC IP and external PHY module to form the lower layer hardware. The user interface of TOE1G IP consists of two interfaces, i.e., the Register interface for control signals and the FIFO interface for data signals.

The Register interface uses a 4-bit address for accessing up to 16 registers, including network parameters, command registers, and system parameters. One TOE1G IP operates one TCP session for communicating with a single target device. Network parameters must be set before de-asserting the reset signal to execute IP initialization. After finishing the reset operation and parameter initialization, the IP is ready to transfer data with the target device. Network parameters cannot be changed without a reset process. TOE1G IP has two initialization modes for obtaining the MAC address of the target device. Further details of each mode can be found in the IP Initialization topic.

In accordance with TCP/IP standard, connection establishment is the first step before transferring data. TOE1G IP supports both active open (the IP opens the port) and passive open (the target device opens the port) modes. After a successful connection, data can be transferred via the new connection. To send TCP payload data, the user must set the total transfer size, packet size, and send command to the IP. The TCP payload data is transferred via the TxFIFO interface. Conversely, when the TCP packet is received from the target, the TCP payload data is extracted and stored in the Rx data buffer. The user logic monitors FIFO status to detect the amount of received data and then asserts read enable to read the data via the RxFIFO interface. When there is no more data to transfer, the connection can be terminated by closing the port. TOE1G IP supports both active close (the IP closes the port) and passive close (the target device closes the port) modes.

To meet the user system requirements, which may be sensitive to the memory resources or the performance, the buffer size inside the IP can be assigned by the user. There are three buffers whose sizes can be adjusted, i.e., Tx data buffer, Tx packet buffer, and Rx data buffer. Using a bigger buffer size may increase the transfer performance in each direction. Further details about the hardware inside the IP are described in the next topic.

Functional Description

A shown in Figure 2, TOE1G-IP core can be divided into three parts, i.e., control block, transmit block and receive block. The details of each block are described as follows.

Control Block

- **Reg**

  All parameters of the IP are set via Register interface that consists of 4-bit address signals and 32-bit data signals. The timing diagram of the Register interface is similar to a single-port RAM interface, as shown in Figure 5. The write and read addresses are the same signal. Table 2 provides a description of each register.

Table 2: Register map Definition

RegAddr [3:0]	Reg Name	Dir	Bit	Description
0000b	RST	Wr /Rd	[0]	Reset IP. 0b: No reset, 1b: Reset. Default value is 1b. **Once the network parameters have been assigned, the user can execute system initialization by setting this register to 1b and then 0b. This action loads the parameters into the IP. If the user needs to update certain parameters, this process must be repeated by setting this register to 1b and then 0b again. The RST register controls the following network parameters: SML, SMH, DIP, SIP, DPN, SPN, and SRV.**
0001b	CMD	Wr	[1:0]	User command. 00b: Send data, 10b: Open connection (active), 11b: Close connection (active), 01b: Undefined. The command operation begins after the user sets CMD register. **In order to start a new operation by setting this register, the system must first be in the Idle state. To confirm that the system is not busy, the user should read bit[0] of CMD register, which should be equal to 0b.**
		Rd	[0]	System busy flag. 0b: Idle, 1b: IP is busy.
			[3:1]	Current operation. 000b: Send data, 001b: Idle, 010b: Active open, 011b: Active close, 100b: Receive data, 101b: Initialization, 110b: Passive open, 111b: Passive close.
0010b	SML	Wr /Rd	[31:0]	Define 32-bit lower MAC address (bit [31:0]) for this IP. **To update this value, the IP must be reset by RST register.**
0011b	SMH	Wr /Rd	[15:0]	Define 16-bit upper MAC address (bit [47:32]) for this IP. **To update this value, the IP must be reset by RST register.**
0100b	DIP	Wr /Rd	[31:0]	Define 32-bit target IP address. **To update this value, the IP must be reset by RST register.**
0101b	SIP	Wr /Rd	[31:0]	Define 32-bit IP address for this IP. **To update this value, the IP must be reset by RST register.**
0110b	DPN	Wr /Rd	[15:0]	Define 16-bit target port number. Unused when the port is opened in passive mode. **To update this value, the IP must be reset by RST register.**
0111b	SPN	Wr /Rd	[15:0]	Define 16-bit port number for this IP. **To update this value, the IP must be reset by RST register.**
1000b	TDL	Wr	[31:0]	Total Tx data length in byte unit. Valid range is 1-0xFFFFFFFF. **The user must first set this register before setting CMD register = Send data (00b). When the IP executes the 'Send data' command and asserts Busy to 1b, the system will read this register, allowing the user to subsequently set the TDL register for the next command. If the same TDL is used in the subsequent command, the user is not required to set TDL again.**
		Rd		Remaining transfer length in byte unit which does not transmit.

RegAddr [3:0]	Reg Name	Dir	Bit	Description
1001b	TMO	Wr	[31:0]	Define timeout value for awaiting the return of an Rx packet from the target. The counter for the timer unit operates at 125 MHz, resulting in a timer unit equal to 8 ns. If the packet is not received within the specified time, TimerInt will be asserted to 1b. For further information of TimerInt, please refer to the Read value of TMO[7:0] register. It is recommended that the TMO is set to a value greater than 0x6000.
		Rd		The details of timeout interrupt are shown in TMO[7:0]. Other bits are read for IP monitoring. [0]-Timeout from not receiving ARP reply packet After timeout, the IP resends ARP request until ARP reply is received. [1]-Timeout from not receiving SYN and ACK flag during active open operation After timeout, the 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, the 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, the IP sends RST packet to close connection. [4]-Timeout from not receiving ACK flag during passive close operation After timeout, the 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, the IP resends the previous data packet. [6]-Timeout from Rx packet lost, Rx data FIFO full or wrong sequence number The IP generates duplicate ACK to request data retransmission. [7]-Timeout from too small receive window size when running Send data command and setting PSH[2] to 1b. After timeout, the IP retransmits data packet, similar to TMO[5] recovery process. [21]-Lost flag when the sequence number of the received ACK packet is skipped. As a result, TimerInt is asserted and TMO[6] is equal to 1b. [22]-FIN flag is detected during sending operation. [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]	TCP data length of each Tx packet in byte unit. Valid from 1-16000. Default value is 1460 byte which is the maximum size of non-jumbo frame. **During running Send data command (Busy=1b), the user must not set this register.** **Similar to TDL register, the user does not need to set PKL register again if the next command uses the same packet length.**
1011b	PSH	Wr /Rd	[2:0]	Sending mode when running Send data command. [0]-Disable to retransmit packet. 0b: Generate the duplicate data packet for the last data packet in Send data command when TDL value is not equal to N times of PKL value to accelerate ACK packet (default). 1b: Disable the duplicate data packet. [1]-PSH flag value in TCP header for all transmitted packet. 0b: PSH flag = 0b (default). 1b: PSH flag = 1b.

RegAddr [3:0]	Reg Name	Dir	Bit	Description
1011b	PSH	Wr/ Rd	[2:0]	[2]-Enable to retransmit data packet when Send data command is paused until timeout, caused by the receive window size being smaller than the packet size. This flag is designed to resolve the system hang problem resulting from lost window update packet. Activating data retransmission prompts the target device to regenerate the lost window update packet. All following conditions must be met to initiate data retransmission. (1) PSH[2] is set to 1b. (2) The current command is 'Send data' and all data are not completely sent. (3) The receive window size is smaller than the packet size. (4) Timer set by TMO register is overflowed. 0b: Disable the feature (default),1b: Enable the feature.
1100b	WIN	Wr /Rd	[5:0]	Threshold value in 1Kbyte unit to initiate window update packet transmission. Default value is 0 (Not enable window update transmission). The IP sends the window update packet when the free space in the receive buffer increases by an amount greater than the threshold value from the value in the most recently transmitted packet. For example, if the user sets WIN=000001b (1 Kbyte) and the the window size of the most recently transmitted packet is 2 Kbyte, when the user reads 1 Kbyte data from the IP and the free space in the receive buffer is updated from 2 Kbyte to be 3 Kbyte, the IP detects that the increased window size is greater than the threshold value of 1 Kbyte (3 KB – 2 KB). As a result, the IP sends the window update packet to update the receive buffer size.
1110b	SRV	Wr/ Rd	[0]	0b: Client mode (default). When the RST register changes from 1b to 0b, the IP sends an ARP request to obtain the Target MAC address from the ARP reply returned by the target device. The IP busy signal is de-asserted to 0b after receiving the ARP reply. 1b: Server mode. When RST register changes from 1b to 0b, the IP waits for an ARP request from the target to obtain Target MAC address. After receiving the ARP request, the IP generates an ARP reply and then de-asserts the IP busy signal to 0b. **Note: In Server mode, when RST register changes from 1b to 0b, the target device needs to resend the ARP request for the TOE1G IP to complete the IP initialization.**
1111b	VER	Rd	[31:0]	IP version

- **TCP Stack**

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

  To transfer data between the TOE1G IP and the target device, three processes are involved: port opening, data transfer, and port closing. The IP supports active open or close by sending SYN or FIN packets when the user sets the CMD register to 10b (port opening) or 11b (port closing). Alternatively, the port can be opened or closed by the target device (passive mode) when the TCP Stack receives SYN or FIN packet. While the port is being opened or closed, the Busy flag is asserted to 1b. Once all packets are transferred, Busy is de-asserted to 0b. The ConnOn signal can be applied to check if the port status is completely opened or closed. The data can be transferred when ConnOn is asserted to 1b (indicating that the port is completely opened).

  To send the data, user data is stored in the Tx data and Tx packet buffers. Packet Builder uses the network parameters set by the user to build TCP header, and then the data of Tx data buffer is appended to the TCP packet. The Transmit block then sends the TCP packet to the target device via Ethernet MAC. If the target device receives the data correctly, an ACK packet is returned to Receive block. The TCP Stack monitors the status of the Transmit and Receive blocks to confirm that the data has been sent successfully. If the data is lost, the TCP Stack pauses the current data transmission and initiates the data retransmission process in Transmit block.

  When the Receive block receives data, TCP Stack checks the order of the received data. If the data is in the correct order, a normal ACK packet is generated by the Transmit block. Otherwise, the TCP Stack starts the lost data recovery process by instructing the Transmit block to generate duplicate ACKs to the target device.

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

Transmit block contains two buffers - Tx data buffer and Tx packet buffer – whose sizes can be adjusted through parameter assignment. A larger buffer size may improve transmit performance. However, the minimum size of theses buffer is limited by the transmit packet size set by the PKL register. Data from the Tx data buffer is split into packets based on the packet size and stored in the Tx packet buffer. TCP header is constructed using the network parameters from the Reg module and then combined with the TCP data from the Tx packet buffer to form a complete TCP packet. The data in the Tx data buffer is flushed after the target device sends an ACK packet. Once the Send data command is completed, the user can initiate the next command.

- **Tx Data Buffer**

The size of this buffer is determined by the “TxBufBitWidth” parameter of the IP, with valid value ranging from 12-16, which corresponds to the address size of an 8-bit buffer as shown in Table 3. The buffer size should be at least twice the size of the Tx Packet Size set in the PKL register. This buffer stores data from the user to prepare the transmit packet sent to the target device. Data is removed from the buffer when the target device confirms that the data has been completely received. When the buffer size is large enough, the IP can send multiple data packets to the target device without waiting for an ACK packet to clear the buffer. The user can continuously store new data in the Tx data buffer without waiting for long periods. This results in the best transmit performance on a 1Gb Ethernet connection. However, if there is significant latency time due to the carrier, networking interface, or target system, all the data in the Tx data buffer may be transferred before an ACK packet is returned to flush the buffer. In such cases, the user must pause filling the buffer with new data, resulting in reduced transmit performance.

If the total user data is greater than the value of the TDL register, the buffer will still have remaining data after completing the current Send command. This data can be applied for the next Send command. All data in the buffer is flushed when the connection is closed or the IP is reset.

Note: The IP cannot send the packet if the data stored in the buffer is less than the transmit size. The IP must wait until the user data is sufficient to create one packet.

- **Tx Packet Buffer**

The size of the buffer size is determined by the “TxPacBitWidth” parameter of the IP. Its valid range is 11-14, and the details of the parameter are shown in Table 3. To store at least one transmit packet, the buffer size must be larger than the Tx packet size set by the PKL register. Note that the maximum value of the PKL register is equal to the Tx Packet Buffer size (in bytes) minus 4.

- **Packet Builder**

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

- **EMacFIFO**

EMacFIFO is 2k x 9-bit FIFO to support flow control of transmit data to EMAC.

Receive Block

The Receive block contains the Rx data buffer, which stores the data received from the target device. The received data is stored in the buffer when the header in the packet matches the expected value, set by the network parameters inside the Reg module, and when the IP and TCP checksum are correct. If any of these conditions are not met, the received packet is rejected. Increasing the size of the Rx data buffer may improve the receive performance. Additionally, the TOE1G IP can reorder packets if only one packet is out of order. For example, if the packet order is #1, #3, #2 and #4 (where packet #2 is interchanged with packet#3), the TOE1G IP can fix the order. However, if more than one packet is out of order, such as in the case of packet#1, #3, #4, and #2 (where packet #3 and #4 are received before packet#2), the TOE1G IP is unable to reorder the packets. In this scenario, the data needs to be retransmitted, and duplicate ACK packets must be generated.

- **Rx Buffer**

This is temporary buffer that is used to hold incoming packets from EMAC in cases where the previous packet has not yet been completely processed. Furthermore, the buffer is designed as an asynchronous buffer, which helps to convert the clock domain of the received data stream from the receive clock to be the transmit clock.

- **Packet Filtering**

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

(1) The network parameters must match the values set in the Reg module, such as the MAC address, IP address, and Port number.
(2) The packet must either be an ARP packet or a TCP/IPv4 packet without a data fragment flag.
(3) The IP header length and TCP header length must be valid, with the IP length being equal to 20 bytes and the TCP header length being between 20 and 60 bytes.
(4) Both the IP checksum and TCP checksum must be correct.
(5) The data pointer, as decoded by the sequence number, must be within a valid range.
(6) The acknowledge number must be within a valid range.

- **Packet Splitter**

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

- **Rx Data Buffer**

The size of the Rx data buffer is determined by the “RxBufBitWidth” parameter of the IP and can range from 11 – 16 (2KB to 64 KB). The size of the Rx data buffer is also applied as the window size of the transmit packet. When the Rx data buffer is sufficiently large, the target device can send multiple data packets to the TOE1G IP without having to wait for an ACK packet, which may be delayed by the networking system. Consequently, a larger Rx data buffer can improve the receive performance.

The data is stored in the buffer until it is read by the user. If the user does not read the data from the buffer for long time, the buffer becomes full, and the target device can no longer send data to the IP, resulting in reduced performance. To achieve optimal receive performance, it is recommended that the user logic reads the data from the IP as soon as it is available. By doing so, the Rx data buffer will not become full, and the receive performance will not be affected by the full window size.

User Block

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

Ethernet MAC

The reference design of the IP uses Tri-Mode Ethernet Media Access Controller (TEMAC) core from Xilinx. TOE1G IP can directly connect to Xilinx EMAC IP. Additional information of Xilinx IP is available on the website listed below.

https://www.xilinx.com/products/intellectual-property/temac.html

Core I/O Signals

Descriptions of all parameters and signal I/O are provided in Table 4 - Table 6 . The EMAC interface is 8-bit AXI4-stream standard for connecting with Xilinx EMAC 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 (Synchronous to Clk)

Signal	Dir	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	Register address bus. Valid when RegWrEn=1b in Write process.
RegWrData[31:0]	In	Register write data bus. Valid when RegWrEn=1b.
RegWrEn	In	Register write enable pulse. Synchronous to RegAddr and RegWrData signals.
RegRdData[31:0]	Out	Register Read data bus. Available the next clock after RegAddr is valid.
ConnOn	Out	Connection Status (1b: connection is opened, 0b: connection is closed)
TimerInt	Out	Timer interrupt. Asserted to high for 1 clock cycle when timeout is detected. More details of Interrupt status could be checked from TMO[7:0] register.
Busy	Out	IP busy status (0b: IP is idle, 1b: IP is busy). This signal is similar to bit[0] of CMD register.
Tx Data Buffer Interface		
TCPTxFfFlush	Out	Tx data buffer within the IP is reset. Asserted to 1b for 1 clock cycle when the connection is closed or the IP is reset.
TCPTxFfFull	Out	Asserted to 1b when Tx data buffer is full. User needs to stop writing data within 4 clock cycles after this flag is asserted to 1b.
TCPTxFfWrCnt[15:0]	Out	Data counter in byte unit of Tx data buffer to show the amount of data in Tx data buffer.
TCPTxFfWrEn	In	Write enable to Tx data buffer. Asserted to 1b to write data to Tx data buffer.
TCPTxFfWrData[7:0]	In	Write data to Tx data buffer. Valid when TCPTxFfWrEn=1b.
Rx Data Buffer Interface		
TCPRxFfFlush	Out	Rx data buffer within the IP is reset. Asserted to 1b when the new connection is opened.
TCPRxFfRdCnt[15:0]	Out	Data counter of Rx data buffer to show the amount of received data in byte unit.
TCPRxFfRdEmpty	Out	Asserted to 1b when Rx data buffer is empty. User needs to stop reading data immediately when this signal is asserted to 1b.
TCPRxFfRdEn	In	Assert to 1b to read data from Rx data buffer.
TCPRxFfRdData[7:0]	Out	Data output from Rx data buffer. Valid in the next clock cycle after TCPRxFfRdEn is asserted to 1b.

Table 6: EMAC Interface

Signal	Dir	Description
Transmit MAC Interface (Synchronous to Clk)		
MacTxReset	In	Active-High Tx software reset from EMAC core. This signal is unused.
MacTxValid	Out	Transmitted data valid signal to EMAC. Synchronous with MacTxData.
MacTxData[7:0]	Out	Transmitted data to EMAC core. Valid when MacTxValid=1b.
MacTxLast	Out	Control signal to indicate the final byte in a frame. Valid when MacTxValid=1b.
MacTxUser	Out	Control signal to indicate an error condition. This signal is always 0b.
MacTxReady	In	Handshaking signal. Asserted to 1b when MacTxData has been accepted.
Receive MAC Interface (Synchronous to MacRxClk)		
MacRxClk	In	Received clock from EMAC core.
MacRxReset	In	Active-High Rx software reset from EMAC core. This signal is unused.
MacRxValid	In	Received data valid signal from EMAC. Synchronous with MacRxData. MacRxValid must be asserted to 1b until end of packet for transferring a packet continuously.
MacRxData[7:0]	In	Received data bus from EMAC core. Valid when MacRxValid=1b.
MacRxLast	In	Control signal to indicate the final byte in the frame. Valid when MacRxValid=1b.
MacRxUser	In	Control signal asserted at the end of received frame to indicate that the frame has an error. 0b: normal packet, 1b: error packet. Valid when MacRxValid=1b and MacRxLast=1b.
MaxRxReady	Out	Handshaking signal. Asserted to 1b when MacRxData has been accepted. MacRxReady is de-asserted to 0b for 3 clock cycles to be the gap size between each received packet.

Timing Diagram

IP Initialization

After the RST register value is changed from 1b to 0b, the initialization of TOE1G IP is initialized. Two modes can be executed, Client mode (SRV[0]=0b) or Server mode (SRV[0]=1b). The information on each mode is presented in the timing diagram below.

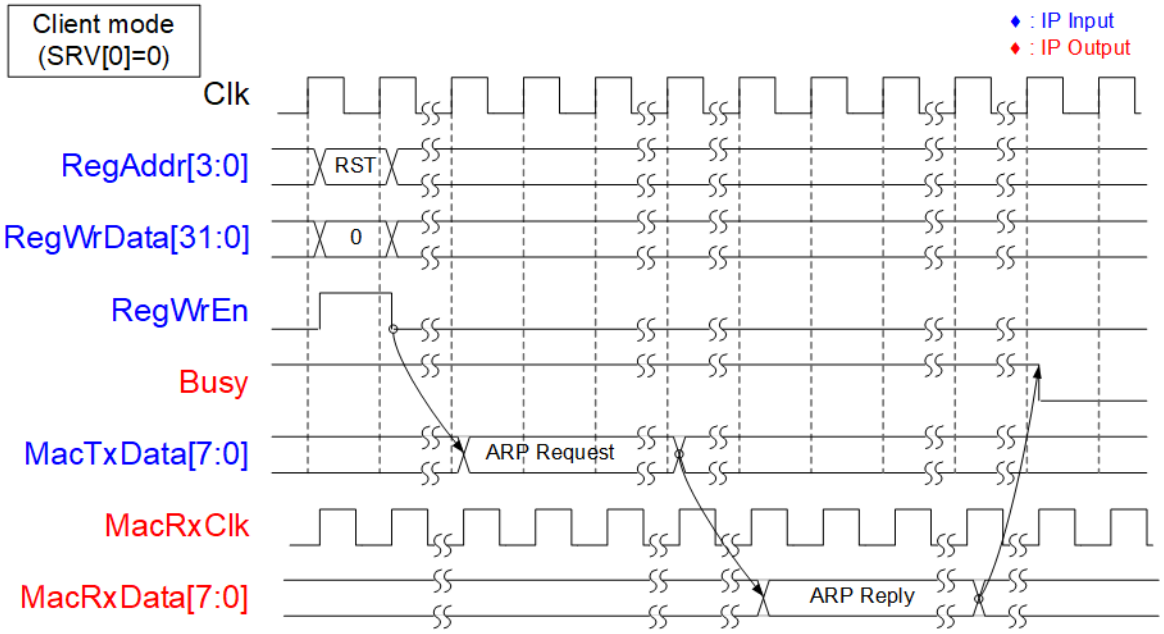

Figure 3: IP Initialization in Client mode

As shown in Figure 3, in Client mode, the TOE1G IP sends an ARP request packet and waits for an ARP reply packet returned from the target device. Target MAC address is extracted from ARP reply packet. Upon completion, the Busy signal is de-asserted to 0b.

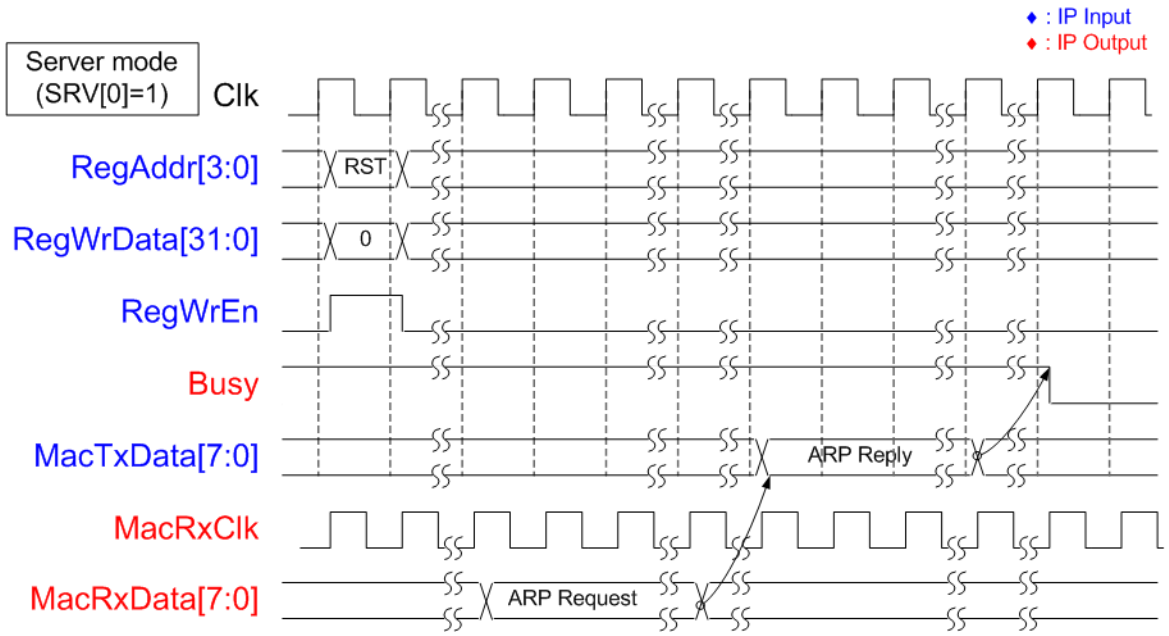

Figure 4: IP Initialization in Server mode

As shown in Figure 4, after reset process in Server mode is completed, the TOE1G IP waits for an ARP request packet from the target device. Upon receipt, the TOE1G IP generates an ARP reply packet. The Target MAC address is extracted from ARP request packet. Once the ARP reply packet has been transmitted, the Busy signal is de-asserted to 0b.

Register Interface

The Register interface is responsible for setting and monitoring all control signals and network parameters during operation. The timing diagram of the interface is similar to that of Single-port RAM, which shares the address bus for write and read access, and has a read latency time of one clock cycle. A Register map of this interface is provided in Table 2.

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

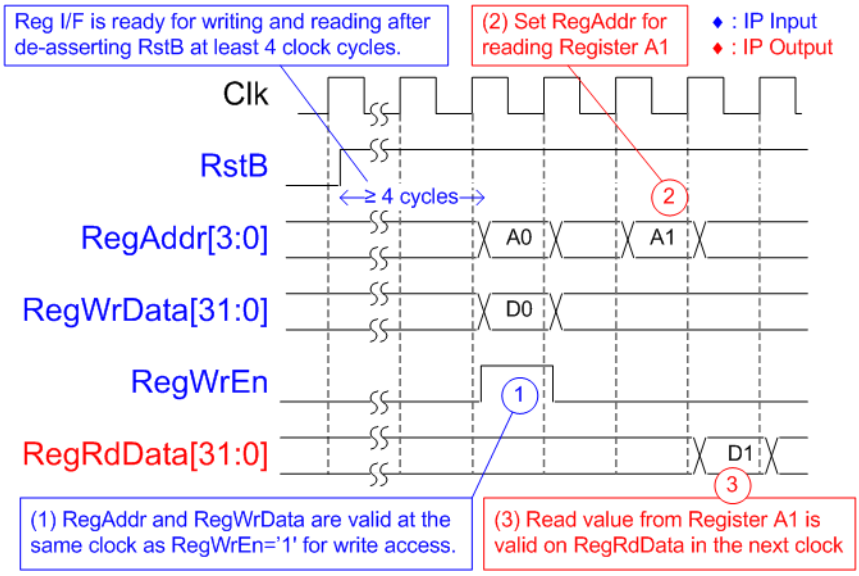

Figure 5: Register interface timing diagram

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

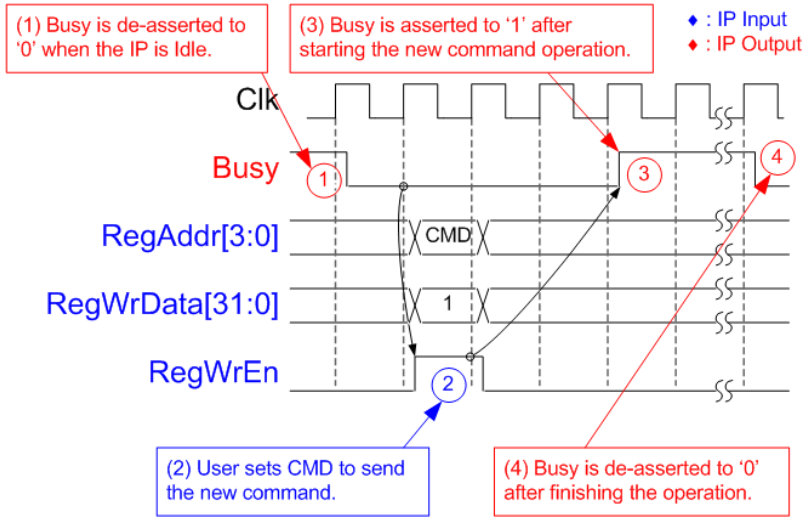

Figure 6: CMD register timing diagram

Tx FIFO Interface

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

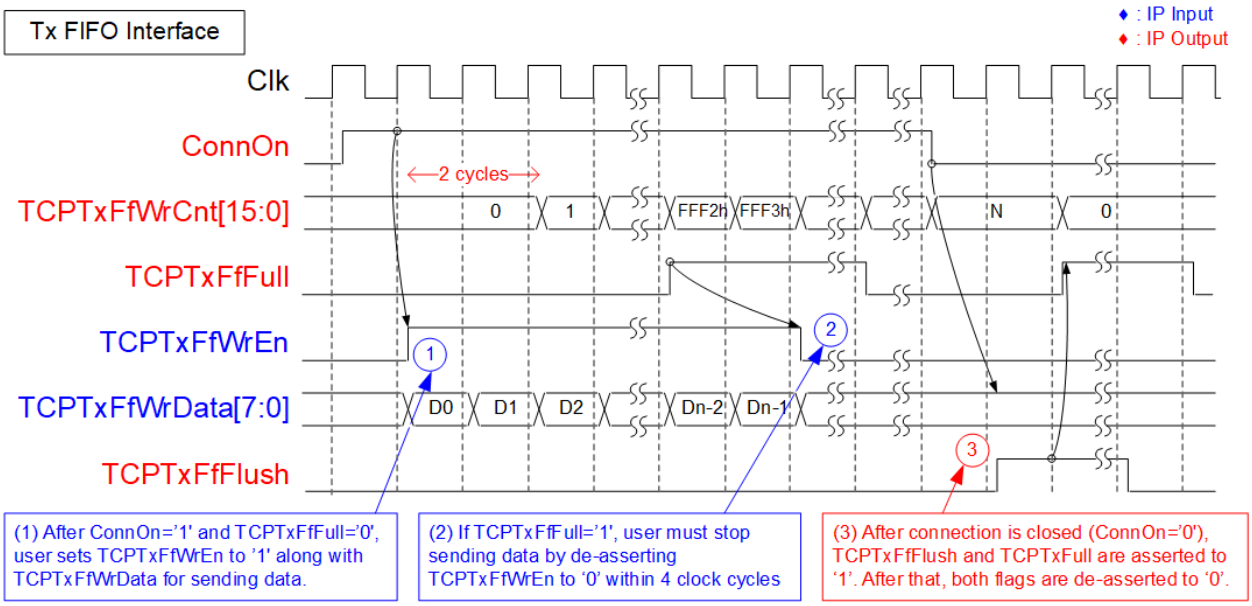

Figure 7: Tx FIFO interface timing diagram

(1) Before asserting TCPTxFfWrEn to 1b to write the data to TOE1G IP, the full flag (TCPTxFfFull) must not be asserted to 1b and ConnOn must be equal to 1b. To write the data, assert TCPTxFfWrEn to 1b along with TCPTxFfWrData.

(2) If TCPTxFfFull is asserted to 1b, TCPTxFfWrEn must be de-asserted to 0b within four clock cycles to pause sending data.

(3) When there is no more data for transferring, the connection may be terminated by active or passive mode. After the port is closed, the following situations are found.
   a) ConnOn changes from 1b to 0b.
   b) TCPTxFfFlush is asserted to 1b to flush all data inside TxFIFO for a while and then de-asserted to 0b.
   c) TCPTxFfWrCnt is reset to 0.
   d) TCPTxFfFull is asserted to 1b to block the new user data and then de-asserted to 0b, similar to TCPTxFfFlush.

While RST register value is set to 1b, TCPTxFfFull is asserted to 1b to block the new user data. The TCPTxFfFull can be de-asserted to 0b after the RST register value is set to 0b to start the IP initialization process, as shown in Figure 8.

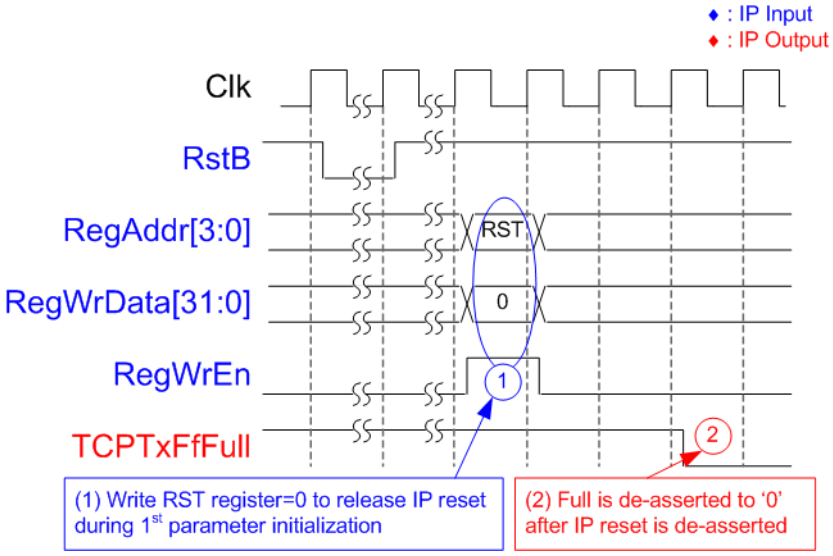

Figure 8: TCPTxFfFull de-asserted from IP reset

Rx FIFO Interface

The Rx FIFO interface is used to retrieve data stored in the Rx data buffer. To determine if data is available for reading, the Empty flag (TCPRxFfEmpty) is monitored, and the read enable signal (TCPRxFfRdEn) is then asserted to access the data, like a typical FIFO read interface, as illustrated in Figure 9.

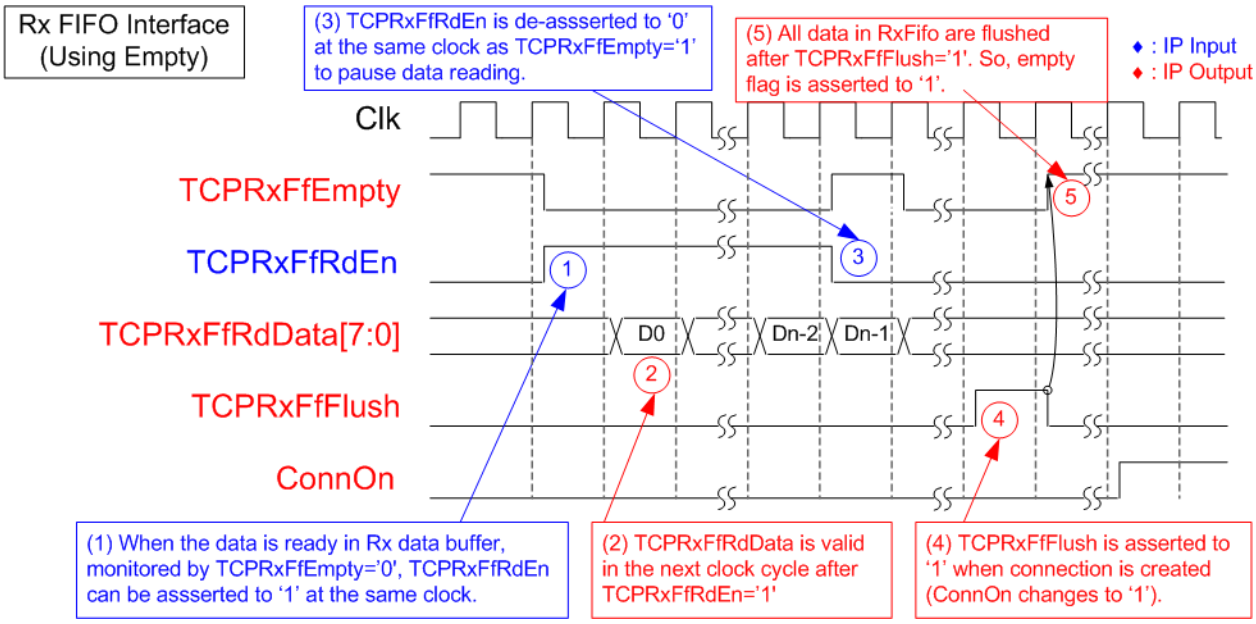

Figure 9: Rx FIFO interface timing diagram by Empty flag

(1) Check the TCPRxFfEmpty flag to confirm data availability. When data is ready (TCPRxFfEmpty=0b), set TCPRxFfRdEn to 1b to read data from the Rx data buffer.
(2) The TCPRxFfRdData signal is valid in the next clock cycle.
(3) Reading data must be immediately paused by setting TCPRRxFfRdEn=0b when TCPRxFfEmpty is equal to 1b.
(4) The user must read all data from the Rx data buffer before creating a new connection. When a new connection is established, all data in the Rx data buffer is flushed, and TCPRxFfFlush is set to 1b. Once the new connection is completed, the ConnOn value changes from 0b to 1b.
(5) After finishing the Flush operation, TCPRxFfEmpty is asserted to 1b.

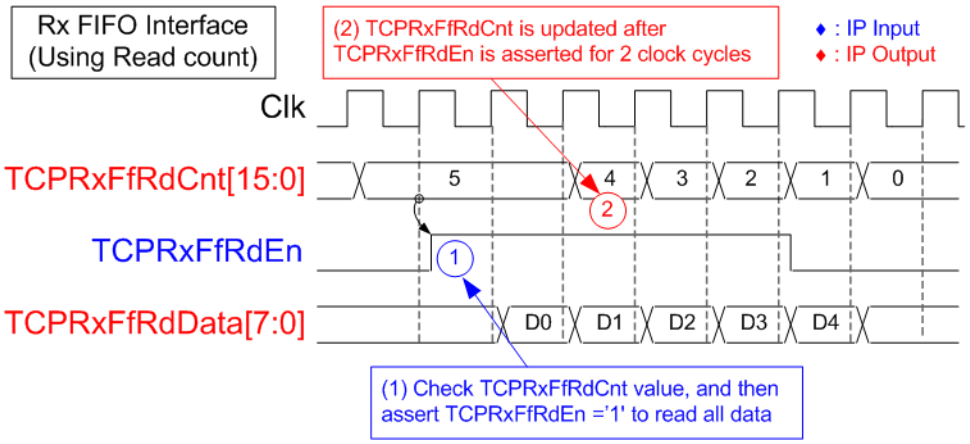

Figure 10: Rx FIFO interface timing diagram by using read counter

When the user logic reads data in burst mode, the TOE1G IP provides a read data counter signal to indicate the total amount of data stored in the Rx data buffer in bytes. For instance, in Figure 10, there are five units of data available in the Rx data buffer. Therefore, the user can set TCPRxFfRdEn to 1b for five clock cycles to read all the data from the Rx data buffer. The latency time to update TCPRxFfRdCnt after setting TCPRxFfRdEn to 1b is two clock cycles.

EMAC Interface

The timing diagram of EMAC interface for TOE1G IP is compatible to Xilinx TEMAC IP core. Figure 11 shows an example to transmit a packet from TOE1G IP to EMAC.

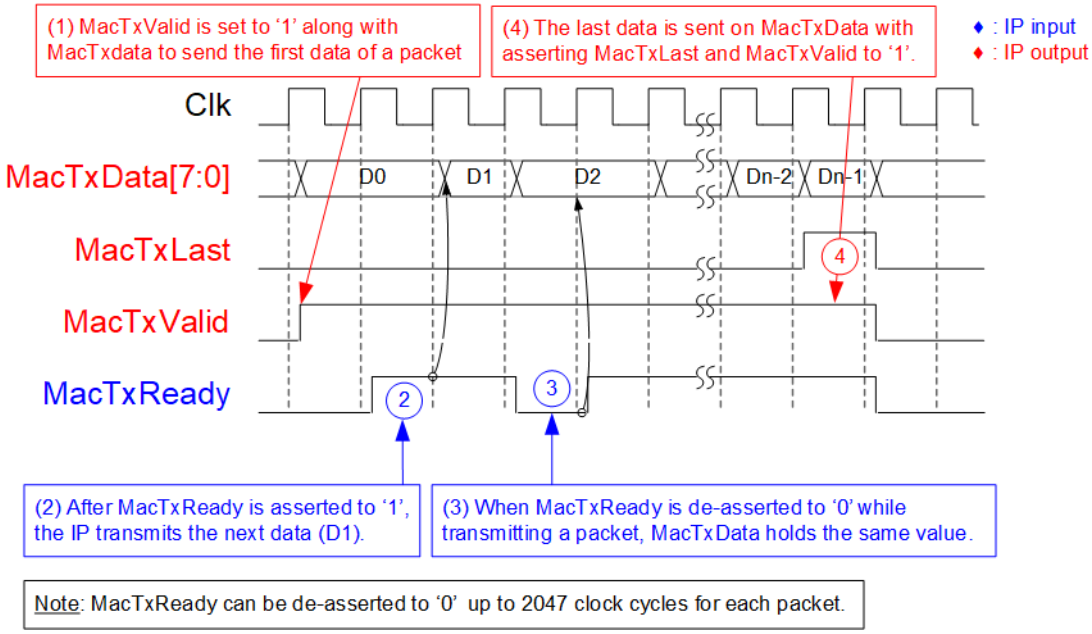

Figure 11: Transmit EMAC Interface

(1) To send the first data of a packet on MacTxData, TOE1G IP asserts MacTxValid and latches all signals until MacTxReady is asserted to 1b. The acknowledgment for the first data sent is detected.

(2) TOE1G IP sends the second data (D1) only after MacTxReady has been asserted to 1b.

(3) During packet transmission, MacTxReady may be de-asserted to 0b according to Xilinx EMAC behavior. In this case, MacTxData retains its value until MacTxReady is re-asserted to 1b.

Note: The EMacFIFO inside TOE1G IP can store Tx data stream from Transmit Block when EMAC is not ready. With a depth of 2047, TOE1G IP supports MacTxReady being de-asserted to 0b for less than 2047 cycles during packet transmission.

(4) To transmit the last data in each packet, MacTxLast and MacTxValid are both asserted to 1b with the last data on MacTxData.

To receive data via the Receive EMAC interface, valid signal must be asserted to 1b continuously from the start of packet to the end of the packet. The timing diagram for the Receive EMAC interface has two formats, depending on whether it is connected with Xilinx EMAC IP or other modules using AXI4-St interface (as shown in Figure 12 and Figure 13, respectively).

TOE1G-IP needs to waits for 3-clock cycle gap between each received packet to perform post processing. When connected with Xilinx EMAC IP, MacRxReady is always asserted to 1b as Xilinx EMAC IP de-asserts the valid signal to pause data transmission for four clock cycles before asserting end-of-packet.

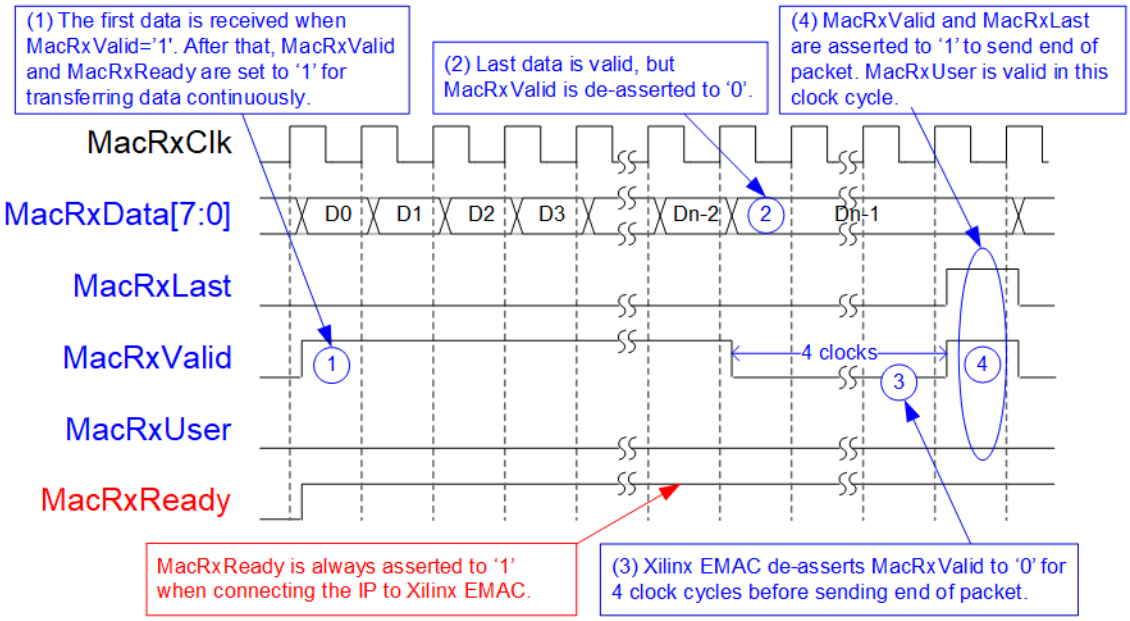

Figure 12: Receive EMAC Interface when connecting with Xilinx EMAC

(1) The start of a received frame is detected when MacRxValid changes from 0b to 1b. TOE1G IP reads the first received data (D0) from MacRxData and starts packet processing. MacRxReady and MacRxValid remain asserted to transfer a packet continuously.
(2) TOE1G IP continuously reads data of one packet and requires the last data to be valid on MacRxData, although the control signal to show the last data may not have been asserted by EMAC yet.
(3) EMAC de-asserts MacRxValid for 4 clock cycles to receive and validate the 32-bit CRC of the Ethernet packet.
(4) To signal the end of the packet, MacRxLast and MacRxValid are asserted to 1b. TOE1G IP reads MacRxUser to check if the packet has a good CRC status. If MacRxUser is asserted, TOE1G IP ignores the packet.

Some user applications do not connect TOE1G IP to Xilinx EMAC, but connecting to other modules through AXI4-ST bus. MacRxValid signal needs to be continuously asserted until the end of the packet. In this case, MacRxReady is de-asserted to 0b for 3 clock cycles to pause the next packet, as shown in Figure 13.

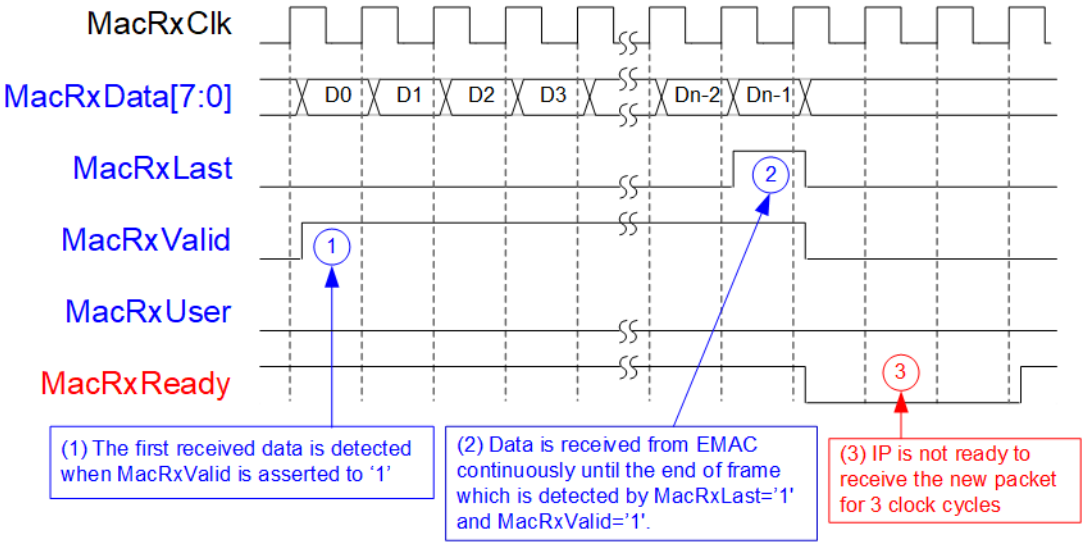

Figure 13: Receive EMAC Interface when connecting through other modules

(1) TOE1G IP detects the start of the received frame when MacRxValid changes from 0b to 1b. The first data is valid on MacRxData and MacRxReady is asserted to 1b to accept all data until the end of the packet. MacRxValid is continuously asserted to 1b to transmit the data of one packet continuously.

(2) The end of the packet is detected when MacRxLast=1b and MacRxValid=1b. At the same clock, the last data is valid on MacRxData.

(3) After that, TOE1G IP de-asserts MacRxReady for 3 clock cycles to complete the packet post-processing.

Example usage

Client mode (SRV[0]=0b)

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

1) Set the RST register=1b to reset the IP.
2) Set the SML/SMH for MAC address, DIP/SIP for IP address, and DPN/SPN for port number.
   Note: DPN is an optional setting when the port is opened by IP (Active open).
3) Set RST register=0b to start the IP initialization process. The TOE1G IP will send an ARP request packet to get the Target MAC address from the ARP reply packet. The Busy signal is de-asserted to 0b after completing the initialization process.
4) The new connection can be created by two modes.
   a. Active open: Write CMD register = “Open connection” to create the connection (SYN packet is firstly sent by TOE1G IP). After that, wait until Busy flag is de-asserted to 0b.
   b. Passive open: Wait until “ConnOn” signal = 1b (the target device sends SYN packet to TOE1G IP firstly).
5) a. For sending data, set TDL register (total transmit length) and PKL register (packet size). Then, set CMD register = “Send Data” to start data transmission. The user can send the data to TOE1G IP via the TxFIFO interface before or after setting the CMD register. Once the command is finished, the Busy flag is de-asserted to 0b. The user can set a new value to the TDL/PKL register and then set CMD register = “Send Data” to start the next transmission.
   b. For receiving data, the user should monitor RxFIFO status and read the data until RxFIFO is empty.
6) Similar to creating the connection, the connection can be terminated by two modes.
   a. Active close: Set CMD register = “Close connection” to close the connection (FIN packet is firstly sent by TOE1G IP). After that, wait until Busy flag is de-asserted to 0b.
   b. Passive close: Wait until “ConnOn” signal = 0b (FIN packet is sent from the target to TOE1G IP firstly).

Server mode (SRV[0]=1b)

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

1) Set RST register=1b to reset the IP.
2) Set SML/SMH for MAC address, DIP/SIP for IP address, and DPN/SPN for port number.
3) Set RST register=0b to being the IP initialization process by waiting for an ARP request packet to get the Target MAC address. The IP then creates an ARP reply packet to return to the target device. Once the initialization process is completed, the Busy signal is de-asserted to 0b.
4) The remaining steps are the same as steps 4 – 6 in Client mode

PKL and TDL setting in Send command

When executing the Send command, the TOE1G IP can operate in two modes based on the value of TDL compared to N times of PKL. The details for each mode are described as follows

TDL = N times of PKL

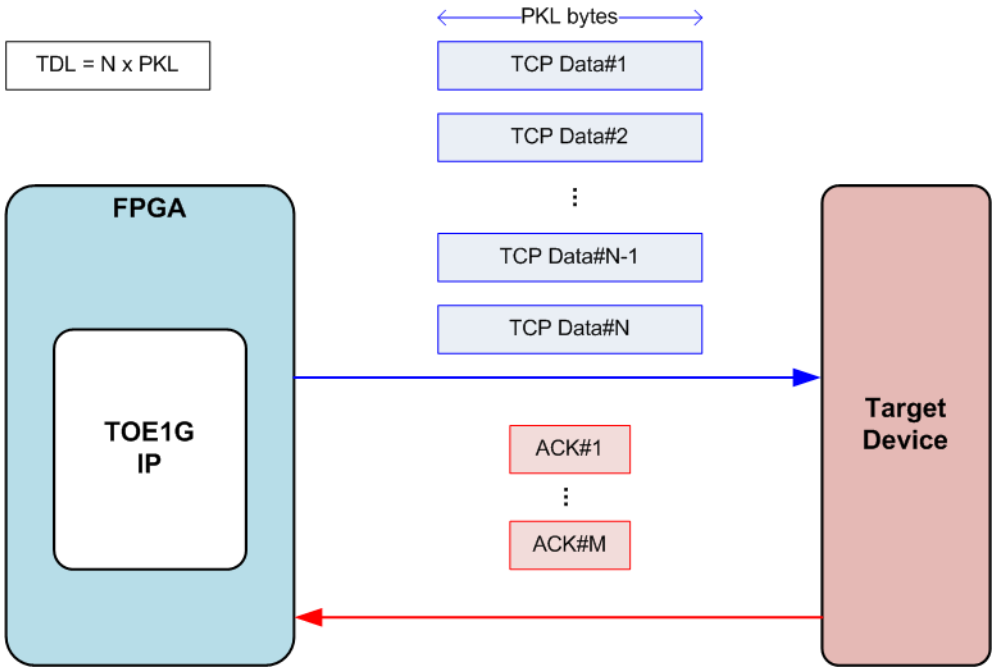

Figure 14: TCP packet when TDL = N times of PKL

If TDL value is equal to N times of PKL value, the user data is split into N packets and transmitted to the target device, as shown in Figure 14. If the target device responds with an ACK packet for each TCP packet, there will be N ACK packets in the network system. To improve network performance, several ACK packets can be combined into be one packet using the TCP delayed ACK technique. Therefore, the number of ACK packets returned from the target device (M) may be less than the number of data packets from TOE1G IP (N) when running the Send command. The PSH[0] value does not affect this condition. The last data packet (TCP Data#N) is sent only once.

TDL = N times of PKL + Residue

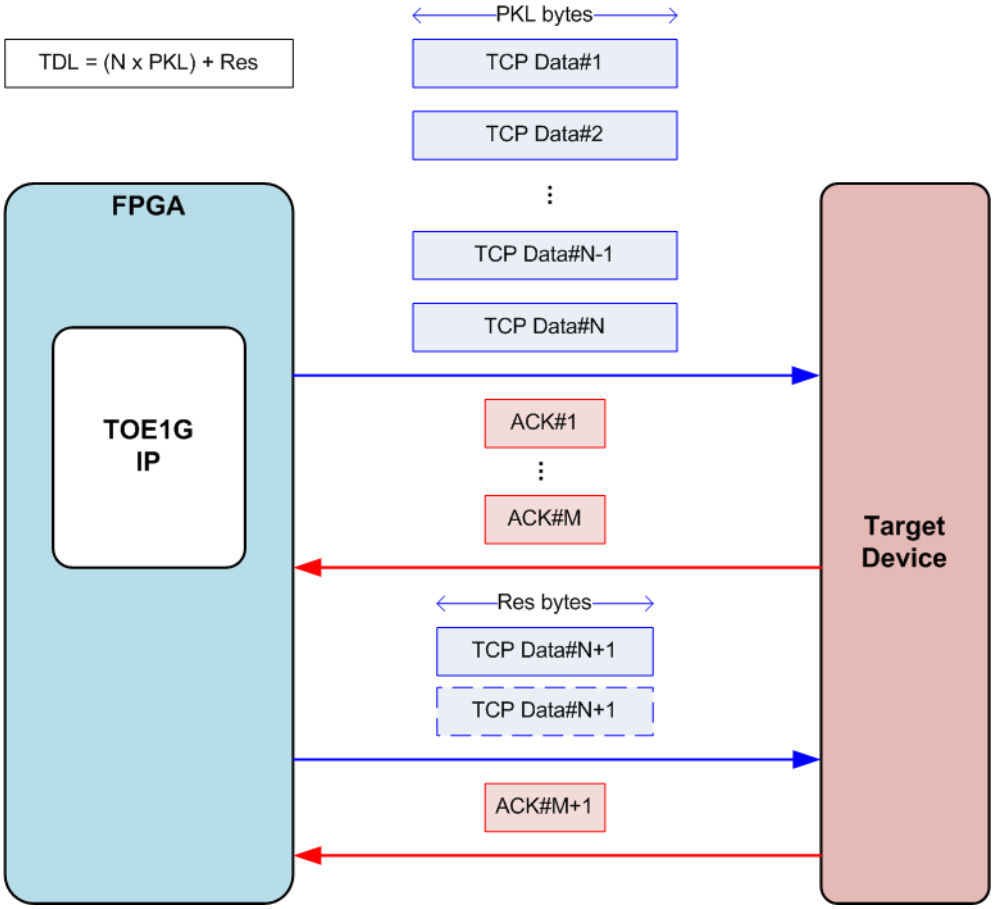

Figure 15: TCP packet when TDL = (N times of PKL) + Residue

If TDL value is not equal to N times of PKL value, the data sent to the target device is split into N packets of PKL-byte data and one last packet that contains Res-byte data, as shown in Figure 15. The first step is similar to the condition where TDL is equal to N times of PKL. The IP needs to receive an ACK packet from the target device to confirm that all N-packets have been received completely. After that, the last packet, which contains the residue byte data, is sent to the target device. If the PSH[0] register is set to 0b (default value), the residue packet is sent twice. Otherwise, the last packet is sent only once. The Send command is completed when the target returns an ACK to confirm that the last packet have been received.

Note: If target device is running on an OS that enables the delayed ACK feature, the ACK#M packet, which confirms the acceptance of TCP Data#N, may arrive too late due to timeout condition in some conditions. Therefore, the target device needs to disable the delayed ACK feature or the TDL value should be aligned to PKL value in systems that are sensitive to this latency time.

Connection termination of unusual case

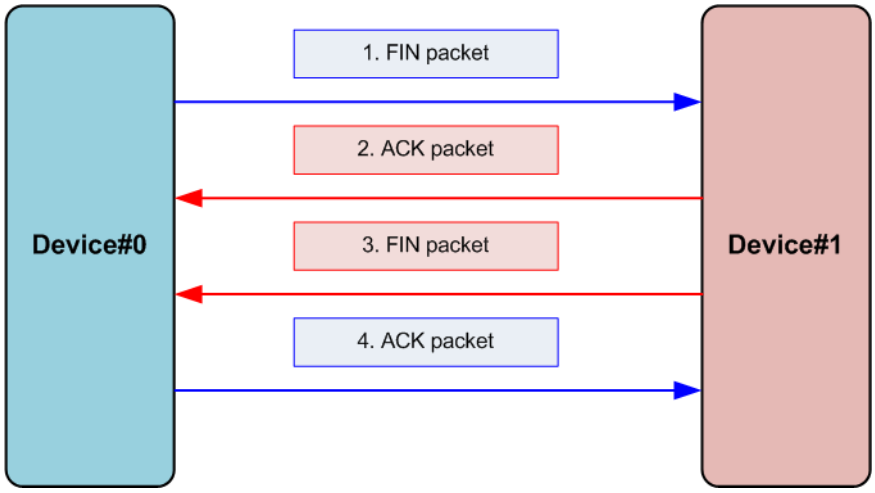

Figure 16: Terminate connection sequence

The process of terminating a connection in the normal case is illustrated in Figure 16, where four packets are exchanged between two devices. The first device (Device#0) initiates the connection termination by sending a FIN packet. If the second device (Device#1) agrees to terminate the connection, it responds with an ACK and FIN packet, which may be sent together in one packet or in separate packets. Finally, Device#0 confirms the termination by sending an ACK packet. The TOE1G IP can execute the close connection in two modes, Active and Passive. This section describes the operation of TOE1G IP in some unusual cases.

1. In the Active mode, TOE1G IP sends a FIN packet to initiate the close and expects to receive ACK and FIN packets from the target. Assumed that a FIN packet sets sequence number (SeqNum) to be N and an acknowledge number (AckNum) to be M, the expected ACK and FIN packet must contain SeqNum=M and AckNum=N+1. If TOE1G IP does not receive the expected packets until timeout (set by the TMO register), it sends a RST packet to terminate the connection immediately without 16 retry times. TOE1G IP also asserts TimerInt and TMO[3] to 1b.
2. If TOE1G IP receives new data from the target while executing the Active close command, it rejects the data and still waits for the expected ACK and FIN packets. Similar to the first case, if the expected packets are not received until the timeout, TOE1G IP sends the RST packet to terminate the connection.
3. In the Passive mode, while TOE1G IP is transmitting data to the target, it receives a FIN packet from the target to terminate the connection. TOE1G IP sends an ACK and FIN packet in response, with SeqNum set to the most recently confirmed data acceptance value. After the termination of the connection, the ConnOn and Busy outputs are set to 0b. The user can check the amount of untransmitted data in the TDL register.

Verification Methods

The TOE1G-IP Core functionality was verified by simulation and also proved on real board design by using AC701/KC705/VC707/ZC706/Zynq Mini-ITX evaluation board.

Recommended Design Experience

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

Ordering Information

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

Revision History

Revision	Date	Description
2.10	23-Feb-2023	Add TCPTxFfWrCnt signal. Add PKL and TDL setting in Send command and Connection termination of unusual case sections.
2.09	2-Oct-2020	Update company info
2.08	30-Jul-2020	Add PSH[2] register and TMO description
2.07	1-Aug-2018	Add SRV register and MacRxReady
2.06	11-May-2017	Add EMACIF block to support new version of Xilinx EMAC
2.05	14-Nov-2016	Add Zynq Mini-ITX support and add WIN register
2.04	2-Sep-2016	IP core renamed from TOE2-IP to TOE1G-IP
2.03	24-Dec-2015	Add Busy status and register readback
2.02	19-Jan-2015	Support Zynq device and add PSH register
2.01	20-Nov-2014	Add TCPTxFfFull condition
2.00	7-Aug-2014	Update IP to support full-duplex
1.05	24-Apr-2014	Add VC707 Support
1.04	13-Mar-2014	Update port name and add more description for Transmit operation
1.03	28-Nov-2013	Add AC701 Support
1.02	19-Nov-2013	Update Figure1
1.01	11-Dec-2012	Update IP port and buffer size description
1.00	3-Dec-2012	New release