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 HDL
Instantiation Templates	VHDL
Reference Designs & Application Notes	QuartusII Project, See Reference Design Manual
Additional Items	Demo on Arria10 SoC board, Arria10 GX board, Cyclone10 GX board, Stratix10 GX board, and Stratix10 MX board
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 IPs)
- Support both Server and Client mode (Passive/Active open and close)
- Support Jumbo frame
- Transmitted packet size aligned to 64-bit, transmitted data bus size
- Total receive data size aligned to 64-bit, received data bus size
- Transmit/Receive buffer size, adjustable for optimizing resource and performance
- Simple data interface by standard FIFO interface at 64-bit data bus
- Simple control interface by single-port RAM interface
- 64-bit Avalon stream to interface with 10 Gbps Ethernet MAC
- One clock domain interface by fixed 156.25 MHz clock frequency
- Reference design available on Arria10 SoC/Arria10 GX/Cylone10 GX/Stratix10 GX/Stratix10 MX board
- Not support data fragmentation feature
- Customized service for following features
  - Unaligned 64-bit data transferring
  - Buffer size extension by using Windows Scaling feature
  - Network parameter assignment by other methods

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	ALMs[1]	Registers[1]	Pin	Block Memory bit[2]	Design Tools
Arria 10 SX	10AS066N3F40E2SGE2	156.25	2,566	3,931	-	1,179,648	QuartusII16.0
Arria 10 GX	10AX115S2F45I2SG	156.25	2,453	3,491	-	1,179,648	QuartusII16.0
Cyclone 10 GX	10CX220Y7F80E5G	156.25	2,247	3,446	-	1,179,648	QuartusII18.0
Stratix 10 GX	1SG280HU2F50E2VGS1	156.25	2,928	3,523	-	1,179,648	QuartusII18.0

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 which are maximum buffer size to achieve the best performance.

Applications

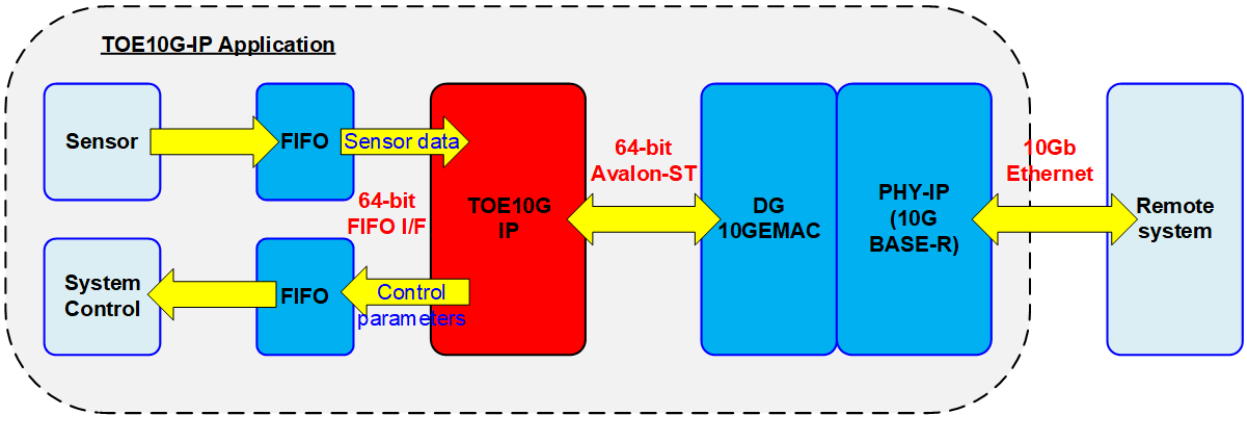

Figure 1: TOE10G IP Application

10 Gb Ethernet is the communication channel which transfers high-speed data with remote controlling system. By using TCP/IP protocol for transferring via 10Gb Ethernet, the system transfers very big data in short time with reliability. TOE10G IP is the IP core which is integrated to the system for transfering data via 10Gb Ethernet without using CPU and external memory. Therefore, the IP fits with the application which needs to transfer reliable data at high-speed rate by using FPGA solution such as video data streaming and sensor monitoring system.

Figure 1 shows the example application of sensor monitoring system. The data from sensor is stored to the FIFO and forwarded to remote system via 10Gb Ethernet by TOE10G IP. TOE10G IP supports full-duplex data transferring by using the same session. Therefore, during transferring data the remote system can send the parameters for real-time controlling the sensor system via 10Gb Ethernet.

Besides, our website purposes FTP server demo by using TOE10G IP. Please contact our sales for more details.

TOE10G IP is designed for transferring data with high-performance feature. Therefore, latency time of data path is much from internal pipeline registers and buffers. For low-latency application such as FinTech, we purposes the low-latency IP that shows more details on our website.

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

General Description

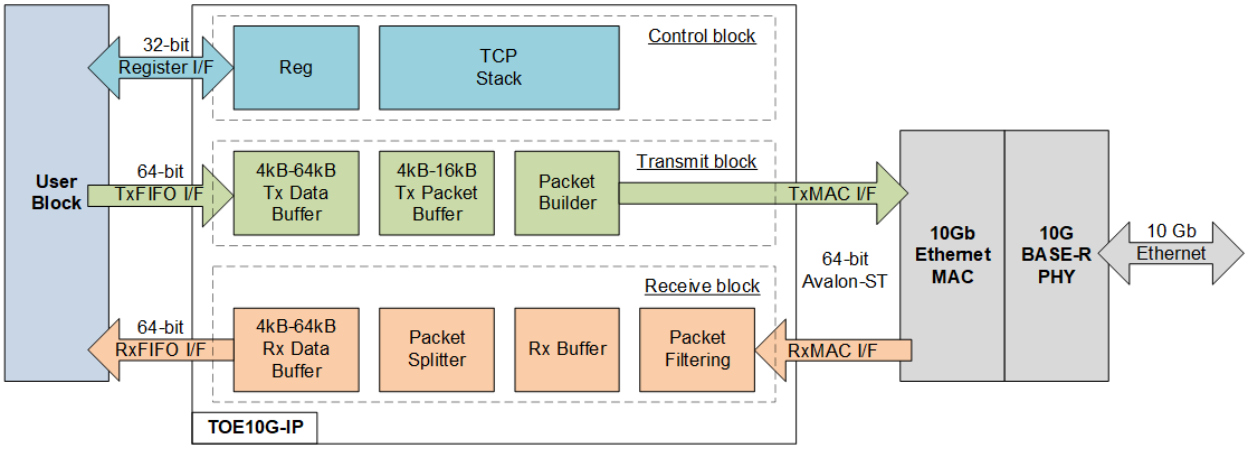

Figure 2: TOE10G IP Block Diagram

TOE10G IP core implements TCP/IP stack by hardwire logic and connects with 10Gb EMAC IP and 10G BASE-R PHY module as the low-layer hardware. User interface of TOE10G IP consists Register interface for control signals and FIFO interface for data signals.

Register interface has 4-bit address for accessing up to 16 registers. The registers stores the network parameters, command register, and system parameters. One TOE10G IP can operate one session for communicating with one target device. Consequently, the network parameters are once set before de-asserting reset signal to start IP initialization. After finishing reset operation and parameter initialization, the IP is ready for transferring data with the target device. The network parameters cannot change without reset process. TOE10G IP has two initialization modes for getting MAC address of the target device. More details of each mode are described in IP Initialization topic.

To transfer data with the user, 64-bit FIFO interface is applied. There is no byte enable in FIFO interface, so the transmitted data from user must be aligned 64-bit. Also, the packet length and total transmit length must be aligned to 64-bit. On the other hand, the received data on Rx FIFO I/F can be read when 64-bit data is valid in Rx data buffer. If the last data is not aligned to 64-bit, the user cannot read it. The user must wait until the next data is received to fill remaining byte in 64-bit data for reading Rx data buffer.

According to TCP/IP standard, the first step before transferring the data is opening the port. TOE10G IP supports both active open (the port is opened by the IP) and passive open (the port is opened by the target device). After that, the data can be transferred via the new connection. To send the data, the user sets total transfer size and packet size to the IP and then transfers the data via TxFIFO interface. When the data is received from the target, the IP extracts the data and stores to Rx data buffer. The user logic monitors FIFO status to detect available data and then asserts read enable to read the data via RxFIFO interface. When there is no more data for transferring, the connection can be terminated by closing the port. TOE10G IP supports both active close (the port is closed by the IP) and passive close (the port is closed by the target device).

To meet the user system requirement which may be sensitive on the memory resource or the performance, the buffer size inside the IP can be assigned by the user. In Tx path, two buffers can be adjusted - Tx data buffer and Tx packet buffer. In Rx path, one buffer is available - Rx data buffer. Using the bigger buffer size may increase the transfer performance in each direction. More details of the hardware inside the IP are described in the next topic.

Functional Description

As shown in Figure 2, TOE10G IP can be divided into three blocks - 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 which uses 4-bit address and 32-bit data. Timing diagram of register interface is similar to single-port RAM interface, as shown in Figure 5. The address for writing data and reading data is shared. The description of each register is defined as shown in Table 2.

Table 2 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 network parameters are assigned, the user sets '1' and then sets '0' to this register for loading parameter and starting system initialization. To update some parameters, user must set this register to '1' and '0' respectively again. The network parameters controlled by RST register are SML, SMH, DIP, SIP, DPN, SPN, and SRV register.**
0001b	CMD	Wr	[1:0]	User command. "00": Send data, "10": Open connection (active), "11": Close connection (active), "01": Undefined. The command operation begins after the user sets CMD register. **Before setting this register to start new operation, the system must be in Idle state. User must confirm that busy is equal to '0' by reading bit[0] of CMD register or bit[0] of RegDataA1 output signal.**
		Rd	[0]	System busy flag. '0': Idle, '1': IP is busy.
			[3:1]	Current operation. "000": Send data, "001": Idle, "010": Active open, "011": Active close, "100": Receive data, "101": Initialization, "110": Passive open, "111": 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, but the length must be aligned to 8-byte (data bus size). Valid range is 8-0xFFFFFFF8 (Bit[2:0] is ignored by the IP). **User needs to set this register before setting CMD register = Send data (00b). This register is read when CMD register is set. After the IP runs Send data command and Busy is asserted to '1', the user can set TDL register for the next command. The user does not need to set TDL register again when the next command uses the same total data length.**
		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 waiting Rx packet returned from the target. The counter is run under 156.25 MHz, so timer unit is equal to 6.4 ns. TimerInt is asserted to '1' when no packet is received until timeout. Timeout status of TimerInt can be read from TMO[7:0] register. It is recommended to set this register to be more 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 PSH[2] is set to '1'. 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 '1'. [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, but the length must be aligned to 8-byte. Valid from 8-16000. Default value is 1456 byte which is the maximum size of non-jumbo frame that is aligned to 8-byte. Bit[2:0] of this register is ignored by the IP. **During running Send data command (Busy='1'), 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 for Send data command. [0]-Disable to retransmit packet. '0': Generate the duplicate data packet for the last data packet in Send data command when the TDL value is not equal to N times of PKL value to accelerate ACK packet (default). '1': Disable the duplicate data packet. [1]-PSH flag value in TCP header for all transmitted packet. '0': PSH flag = '0' (default). '1': PSH flag = '1'.

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 smaller than the packet size. This flag is designed to solve the system hang problem when the window update packet is lost. Data retransmission can activate the target device to regenerate the lost window update packet. All following conditions must be met to start data retransmission. (1) PSH[2] is set to '1'. (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. '0': Disable the feature (default), '1': Enable the feature.
1100b	WIN	Wr /Rd	[5:0]	Threshold value of free space in Rx data buffer, assigned in 1Kbyte unit for sending window update packet. Default value is 0 (disable window update feature). The IP transmits the window update packet when the free space of Rx data buffer is increased from the value in the latest transmitted packet more than the threshold value. For example, the user sets WIN="000001b" (1 Kbyte) and the window size of the latest transmitted packet is equal to 2 Kbyte. After the user reads 1 Kbyte data from the IP, free space of Rx data buffer is updated from 2 Kbyte to be 3 Kbyte. The IP detects the increased free space size is more than 1 Kbyte (3K – 2K) which is the threshold value. As a result, the IP sends the window update packet to update the receive buffer size.
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 can set this register during running Send data command operating (Busy='1') for extending total Tx data length. So, the data can be transmitted continuously without re-sending the new command to IP. The caution points to use this feature are as follows. 1) ETL register must be programmed when read value of TDL is not less than 128 Kbyte to be the safe gap that Busy is not de-asserted to '0' before setting ETL register. 2) The set value of ETL must be less than the max value of TDL (0xFFFFFFF8) – read value of TDL to avoid overflow value. For example, the user sets TDL = 3.5 Gbyte and then set CMD register = Send data. After the IP completes 2 Gbyte data (remaining size = 1.5 Gbyte), the user sets ETL register = 1.5 Gbyte.The total transmit length is equal to 5 Gbyte (3.5 Gbyte of TDL + 1.5 Gbyte of ETL).
1110b	SRV	Wr/ Rd	[0]	'0': Client mode (default). After RST register changes from '1' to '0', the IP sends ARP request to get Target MAC address from the ARP reply returned by the target device. IP busy is deasserted to '0' after receiving ARP reply. '1': Server mode. After RST register changes from '1' to '0', the IP waits for ARP request from the Target to get Target MAC address. After receiving ARP request, the IP generates ARP reply and then de-asserts IP busy to '0'. **Note: In Server mode, when RST register changes from '1' to '0', the target device needs to resend ARP request for TOE10G IP completing the IP initialization.**
1111b	VER	Rd	[31:0]	IP version

- **TCP Stack**

  TCP stack is the main controller to control the other modules for interfacing with user and transferring a packet with EMAC. The IP operation has two phases - IP initialization phase and data transferring phase.

  After RST register changes from ‘1’ to ‘0’, the initialization phase begins. There are two modes for running the initialization phase, set by SRV[0] register, i.e., Client mode and Server mode. The parameters from Reg module is read by TCP Stack and then set to Transmit block and Receive block for transferring the packet to complete the initialization process, following the mode. After that, the IP changes to data transferring phase.

  To transfer data between TOE10G IP and the target device, three processes are run, i.e., opening the port, transferring data, and closing the port. The IP supports to run active open or active close by sending SYN or FIN packet when the user sets CMD register = “10” (port opening) or “11” (port closing). Also, the port can be closed or opened by the target device (passive mode) when TCP Stack detects SYN or FIN packet from Receive block. During port opening or port closing, TCP Stack asserts Busy flag to ‘1’. After finishing transferring all packets and the port is completely opened or closed, Busy flag is de-asserted to ‘0’. 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 ‘1’ (the port is opened completely).

  To send the data, the data from the user is stored in Tx data buffer. The network parameters from user setting are applied to build TCP header by Packet Builder and append the data from Tx data buffer. After that, the complete packet is transmitted to EMAC. If the target receives the data correctly, ACK packet is returned to Receive block. TCP Stack monitors the status of Transmit block and Receive block to confirm that the data is sent successfully. If the data is lost, TCP Stack pauses the current data transmission and then start data retransmission process in Transmit block.

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

Table 3 TxBuf/TxPac/RxBufBitWidth Parameter description

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

Transmit Block

There are two buffers in Transmit block - Tx data buffer and Tx packet buffer which can adjust the size by parameter assignment. Using bigger size may increase the transmit performance. The minimum size of Tx data buffer and Tx packet buffer depends on transmit packet size, set by PKL register. At least one packet data must be stored in both Tx buffer. TCP header is prepared from the network parameters in Reg module and then combined with TCP data from Tx data buffer to build the complete TCP packet. The transmitted data in Tx data buffer is flushed after the target device returns ACK packet. After finishing Send data command, the user can update the packet size (PKL) and total data size (TDL) for the next send data command.

- **Tx Data Buffer**

  This buffer size is set by “TxBufBitWidth” parameter of the IP which can be equal to 9-13. The parameter is the address size of 64-bit buffer, as shown in Table 3. The buffer size should be more than or equal to two times of Tx Packet Size, set in PKL register. This buffer stores the data from the user for preparing the transmit packet sent to the target device. Data can be removed from the buffer after the target device confirms that the data is completely received. Consequently, when the buffer size is large enough, the IP can send many data to the target device without waiting ACK packet returned from the target to clear the buffer. Also, the user can store the new data to Tx data buffer without waiting for a long time. When the buffer is not full, the packet can be transmitted to the target device continuously. As a result, the system can achieve the best transmit performance on 10Gb Ethernet connection. Nevertheless, when the carrier and the networking interface have much latency time, all data in the Tx data buffer may be completely transferred before the ACK packet to flush the buffer is returned. The user cannot fill the new data in that time. Therefore, the transmit performance can be reduced when the latency time is much.

  If total data from user is more than the value of TDL register, the data is remained in the buffer for the next command. All data in the buffer is flushed when the connection is closed or the IP is reset. Please note that the IP cannot send the packet if the data stored in the buffer is less than transmit size. The IP must wait until the data from user is 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 3. This buffer must store at least one transmit packet, so the buffer size must be more than Tx packet size, set by PKL register. The maximum value of PKL register is equal to (Tx Packet Buffer size<byte> – 24).

- **Packet Builder**

  TCP packet consists of the header and the data. Packet builder receives network parameters, set in Reg module, and then prepares TCP header. Also, IP and TCP checksum are calculated to be a part of TCP header. After all TCP header is completely built, the header combining with the data from Tx packet buffer is transmitted to EMAC.

Receive Block

In the receive block, Rx data buffer is included to store the received data from the target device. The data is stored in the buffer when the header in the packet is matched to the expected value, set by the network parameters inside Reg module. Also, the IP and TCP checksum in the packet must be correct. Otherwise, the received packet is rejected. Using bigger size of Rx data buffer may increase the receive performance. Besides, TOE10G IP can support the packet re-ordering when only one packet is swapped. For example, the receive order is packet#1, #3, #2, and #4 (packet #2 is swapped with packet#3). If the packet order is switched more than one packet such as packet#1, #3, #4, and #2 (packet #3 and #4 are received before packet#2), TOE10G IP cannot reorder the data and detect as data lost condition. After that, the data recovery process is run by generating duplicated ACK packet.

- **Rx Buffer**

  This is temporary buffer to store the received packets from EMAC when the previous packet is not completely processed.

- **Packet Filtering**

  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 value in Reg module, 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 equal to 20 – 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.

- **Packet Splitter**

  This module is designed to remove the packet header and split only TCP data to store to Rx data buffer.

- **Rx Data Buffer**

  This buffer size is set by “RxBufBitWidth” parameter of the IP. The valid value is 9-13 for 4Kbyte – 64Kbyte buffer size. Rx data buffer size is applied to be the window size of the transmitted packet. When Rx data buffer is big enough, the target device can send many data to TOE10G IP without waiting ACK packet returned by the IP which may be delayed from the networking system. As a result, the bigger size of Rx data buffer may increase the receive performance.

  The data is stored in the buffer until the user reads it. If the user does not read data out from the buffer for long time, the buffer will be full. Therefore, the target device cannot send more data to the IP and then the receive performance is reduced. To achieve the best receive performance, it is recommended for the user logic to read the data from the IP when the data is ready. If the Rx data buffer is not full, the receive performance will not be dropped by the full window size.

User Block

The user module can be designed by using state machine to set the command and the parameters via register interface. Also, the status can be monitored to confirm the operation is finished without any error. The data path can connect with the FIFO for transferring data with the IP.

10 Gb Ethernet MAC

The user interface of 10G EMAC to connect with TOE10G IP is 64-bit Avalon stream interface while the interface with 10G BASE-R PHY 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_intel_en.pdf

Otherwise, Intel provides low latency Ethernet 10G MAC including many features. TOE10G IP also supports connecting with Intel 10G EMAC IP directly. More details of 10G EMAC Intel FPGA IP are described in following website.

https://www.intel.com/content/www/us/en/programmable/products/intellectual-property/ip/interface-protocols/m-alt-10gbps-ethernet-mac.html

10 Gb BASE-R PHY

This module is the IP core by Intel FPGA for connecting with 10G Ethernet MAC via a standard XGMII interface running at 156.25 Mbps. The IP core allows direct connection with SFP+ optical module. More details are described in following website.

https://www.intel.com/content/www/us/en/programmable/products/intellectual-property/ip/interface-protocols/m-alt-10gbase-r-pcs.html

Core I/O Signals

Descriptions of all parameters and I/O signals are provided in Table 4 and Table 5. The EMAC interface is 64-bit Avalon stream standard.

Table 4: 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 5: Core I/O Signals

Signal	Dir	Description
Common Interface Signal		
RstB	In	Reset IP core. Active Low.
Clk	In	156.25 MHz fixed clock frequency to synchronous with the user interface and EMAC interface.
User Interface		
RegAddr[3:0]	In	Register address bus. In Write access, RegAddr is valid when RegWrEn='1'.
RegWrData[31:0]	In	Register write data bus. Valid when RegWrEn='1'.
RegWrEn	In	Register write enable. Valid at the same clock as RegAddr and RegWrData.
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 are monitored from TMO[7:0] register.
RegDataA1[31:0]	Out	32-bit read value of CMD register (RegAddr=0001b) . Bit[0] is busy flag of TOE10G IP.
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	Tx data buffer within the IP is reset. Asserted to '1' when the connection is closed or the IP is reset.
TCPTxFfFull	Out	Asserted to '1' when Tx data buffer is full. User needs to stop writing data within 4 clock cycles after this flag is asserted to '1'.
TCPTxFfWrEn	In	Write enable to Tx data buffer. Asserted to '1' to write data to Tx data buffer.
TCPTxFfWrData[63:0]	In	Write data to Tx data buffer. Valid when TCPTxFfWrEn='1'.
Rx Data Buffer Interface		
TCPRxFfFlush	Out	Rx data buffer within the IP is reset. Asserted to '1' when the new connection is opened.
TCPRxFfRdCnt[12:0]	Out	Data counter of Rx data buffer to show the number of received data in 64-bit unit.
TCPRxFfLastRdCnt[2:0]	Out	Remaining byte of the last data in Rx data buffer when total received data in the buffer is not aligned to 8-byte unit. User cannot read the data until all 8-byte data is received.
TCPRxFfRdEmpty	Out	Asserted to '1' when Rx data buffer is empty. User needs to stop reading data immediately when this signal is asserted to '1'.
TCPRxFfRdEn	In	Assert to '1' to read data from Rx data buffer.
TCPRxFfRdData[63:0]	Out	Data output from Rx data buffer. Valid in the next clock cycle after TCPRxFfRdEn is asserted to '1'.

Signal	Dir	Description
MAC Interface		
MacTxData[63:0]	Out	Transmitted data. Valid when MacTxValid='1'.
MacTxEmpty[2:0]	Out	Specify the number of bytes which are unused of the final word in the frame.
MacTxValid	Out	Valid signal of transmitted data.
MacTxSOP	Out	Control signal to indicate the first word in the frame. Valid when MacTxValid='1'.
MacTxEOP	Out	Control signal to indicate the final word in the frame. Valid when MacTxValid='1'.
MacTxReady	In	Handshaking signal. Assert to '1' when MacTxData has been accepted. This signal must not be de-asserted to '0' when a packet is transmitting.
MacRxData[63:0]	In	Received data. Valid when MacRxValid='1'.
MacRxValid	In	Valid signal of received data. MacRxValid must be asserted to '1' continuously for transferring a packet.
MacRxEOP	In	Control signal to indicate the final word in the frame. Valid when MacRxValid='1'.
MacRxError	In	Control signal asserted at the end of received frame (MacRxValid='1' and MacRxEOP='1') to indicate that the frame has CRC error. '1': error packet, '0': normal packet. To connect with Intel 10G EMAC, this port connects to avalon_st_rx_error[1] signal.
MacRxReady	Out	Handshaking signal. Asserted to '1' when MacRxData has been accepted. MacRxReady is de-asserted to '0' for 2 clock cycles to be the gap size between each received packet.

Timing Diagram

IP Initialization

The initialization process begins after user changes RST register from ‘1’ to ‘0’. TOE10G IP can run in two modes, set by SRV[0] register - Client mode (SRV[0]=’0’) and Server mode (SRV[0]=’1’). The details of each mode are shown in the following timing diagram.

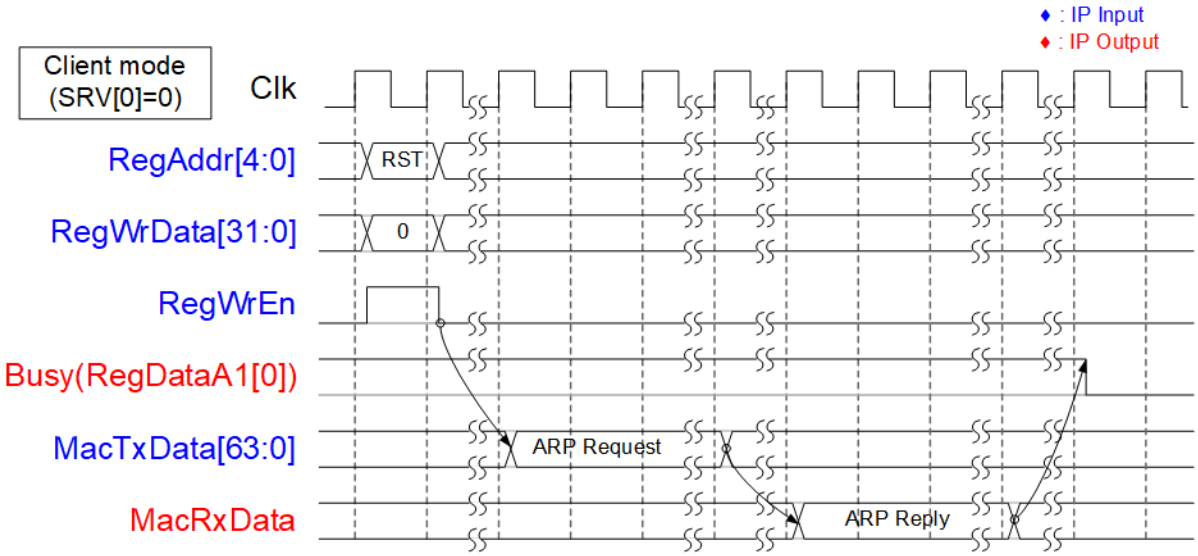

Figure 3: IP Initialization in Client mode

As shown in Figure 3, in Client mode TOE10G IP sends ARP request and waits until ARP reply returned from the target device. Target MAC address is extracted from ARP reply packet. After finishing, Busy signal (bit0 of RegDataA1) is de-asserted to ‘0’.

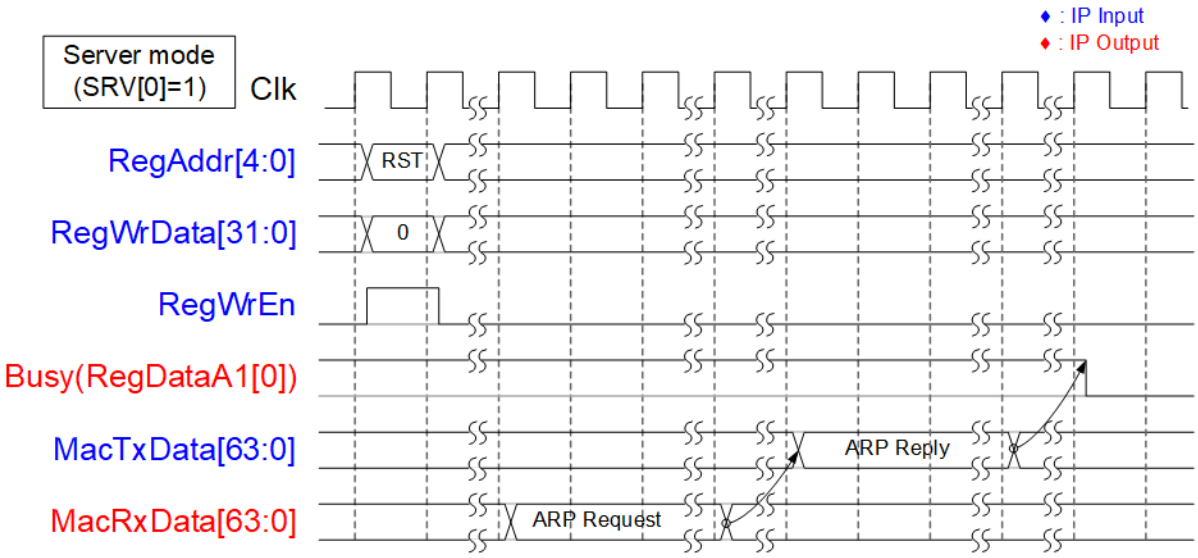

Figure 4: IP Initialization in Server mode

As shown in Figure 4, after finishing reset process in Server mode, TOE10G IP waits until ARP request sent by the target device. After that, 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

All control signals and the network parameters for the operation are set and monitored via Register interface. Timing diagaram of Register interface is similar to Single-port RAM which shares the address bus for write and read access. Read latency time of the read data from the address is one clock cycle. Register map is defined in Table 2.

As shown in Figure 5, 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.

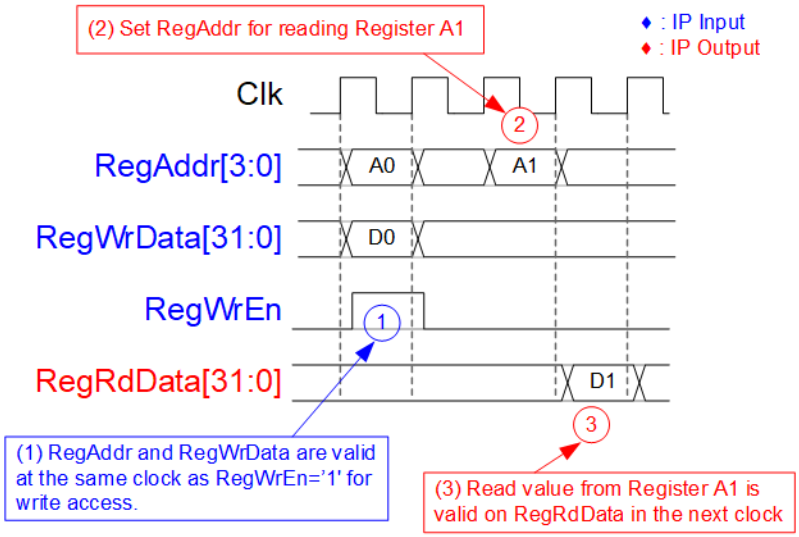

Figure 5: Register interface timing diagram

As shown in Figure 6, 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'. Busy is de-asserted to '0' when the command is completed.

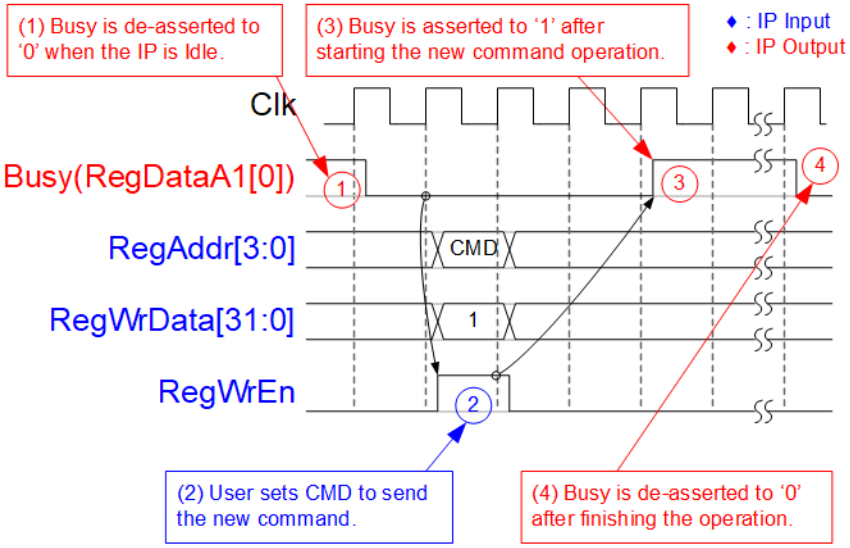

Figure 6: CMD register timing diagram

Tx FIFO Interface

To send the data to IP core via Tx FIFO interface, Full flag is monitored to be flow control signal. The write signals are similar to write interface of general FIFO by using write data and write enable as shown in Figure 7.

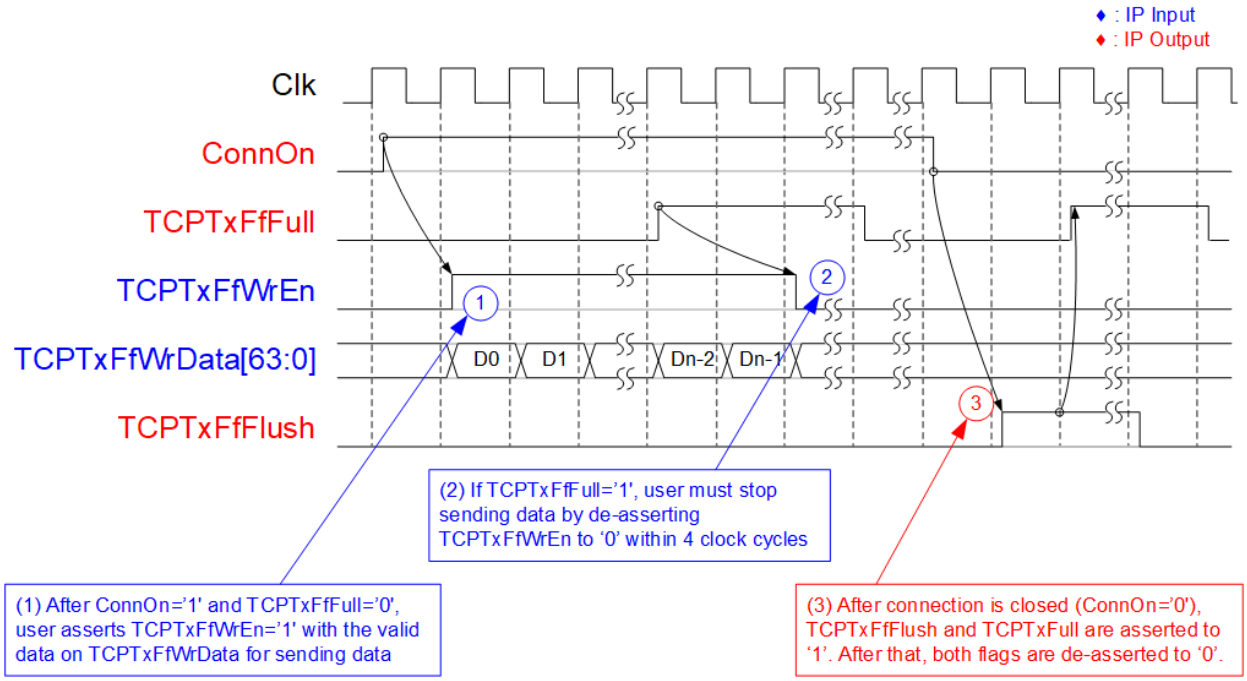

Figure 7: Tx FIFO interface timing diagram

(1) Before sending data, user needs to confirm that full flag (TCPTxFfFull) is not asserted to '1' and ConnOn must be equal to '1'. After that, TCPTxFfWrEn can be asserted to '1' with valid value of TCPTxFfWrData.

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

(3) After finishing transferring all data, the port can be closed by TOE10G IP (active) or the target device (passive). After the port is closed, the following situations are found.
   a) ConnOn changes from '1' to '0'.
   b) TCPTxFfFlush is asserted to '1' to flush all data inside TxFIFO.
   c) TCPTxFfFull is asserted to '1' to pause data sent by the user during closing the connection.

Rx FIFO Interface

After the received data is stored in Rx data buffer, the user can read the data from Rx data buffer by using Rx FIFO interface. Empty flag is monitored to check data available status and then asserts read enable signal to read the data, similar to read interface of general FIFO, as shown in Figure 8.

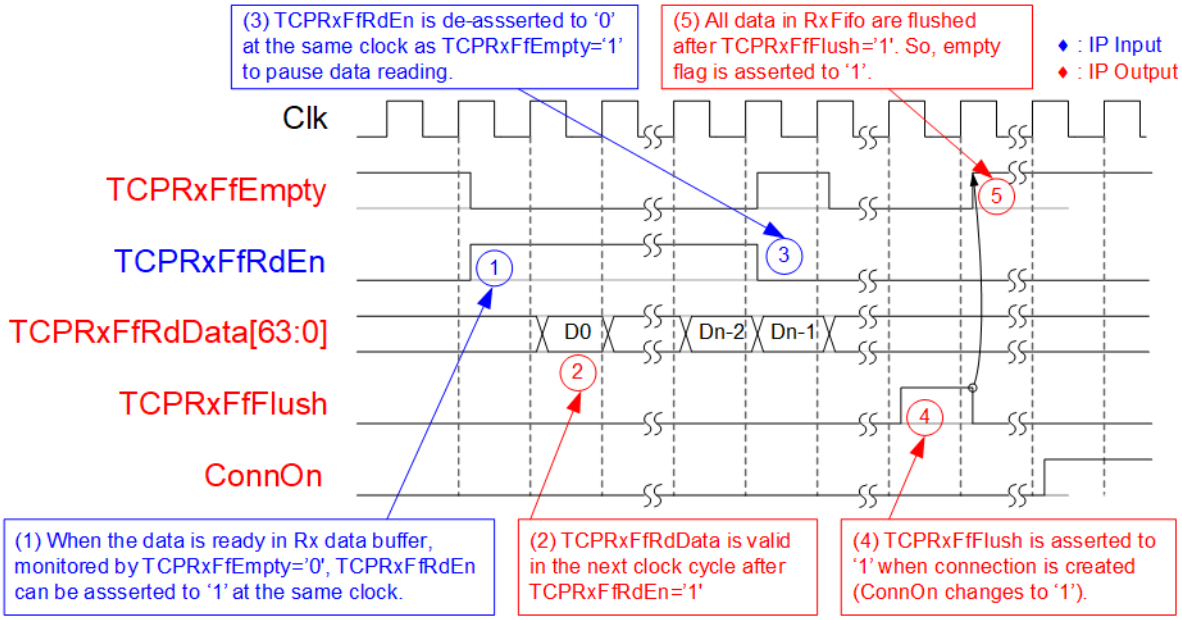

Figure 8: Rx FIFO interface timing diagram by Empty flag

(1) TCPRxFfEmpty is monitored to check data available status. When data is ready (TCPRxFfEmpty='0'), TCPRxFfRdEn can be asserted to '1' to read data from Rx data buffer.
(2) TCPRxFfRdData is valid in the next clock cycle.
(3) Reading data must be immediately paused by de-asserting TCPRRxFfRdEn='0' when TCPRxFfEmpty='1'.
(4) User must read all data from Rx data buffer before the connection is new created. All data in Rx data buffer is flushed and TCPRxFfFlush is asserted to '1' when the new connection is created. After finishing new connection created, ConnOn changes from '0' to '1'.
(5) After finishing Flush operation, TCPRxFfEmpty is asserted to '1'.

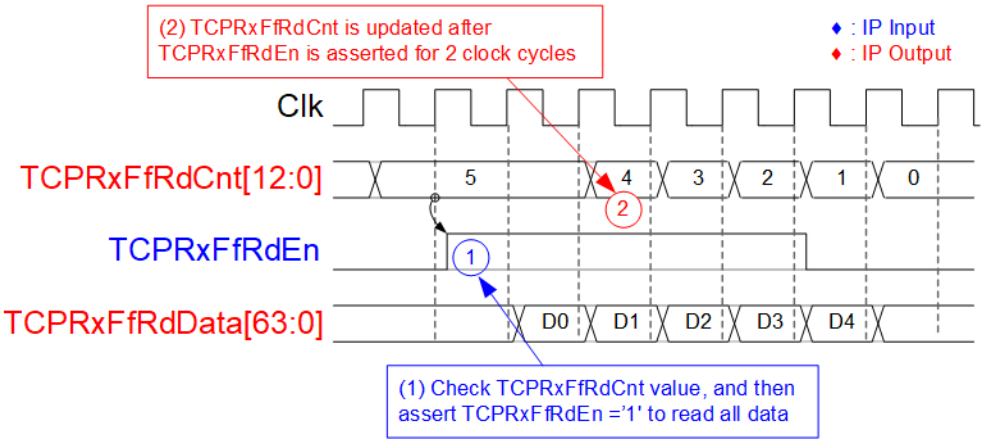

Figure 9: Rx FIFO interface timing diagram by using read counter

If user logic reads data as burst mode, TOE10G IP has read counter signal to show the total number of data stored in Rx FIFO interface as 64-bit unit. For example, Figure 9 shows five data available in Rx data buffer. Therefore, user can assert TCPRxFfRdEn to '1' for 5 clock cycles to read all data from Rx data buffer. The latency time to update read counter (TCPRxFfRdCnt) after asserting read enable (TCPRxFfRdEn) is 2 clock cycles.

EMAC Interface

EMAC interface of TOE10G IP is designed by using 64-bit Avalon-stream interface. The limitation is that TOE10G IP cannot pause data transmission when the packet does not end. So, MacTxReady must be asserted to '1' during transmitting a packet. MacTxReady can be de-asserted to '0' after the last data in the packet is transferred, as shown in Figure 10.

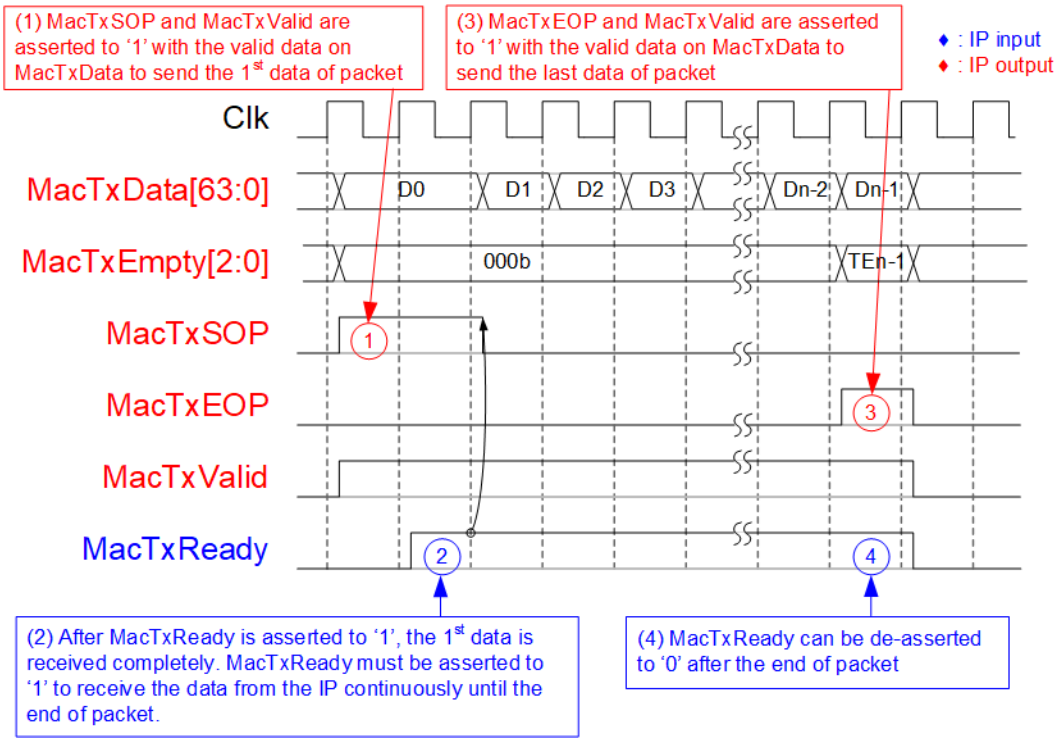

Figure 10: Transmit EMAC Interface timing diagram

(1) TOE10G IP asserts MacTxSOP and MacTxValid with the first data of the packet. All signals are latched until MacTxReady is asserted to '1' to accept the first data.
(2) After the first data is accepted, MacTxReady must be asserted to '1' to accept all remaining data in the packet from TOE10G IP until end of packet. The IP sends all data of one packet continuously.
(3) MacTxEOP and MacTxValid are asserted to '1' when the last data of the packet is transmitted.
(4) After the end of packet, MacTxReady can be asserted to '0' to pause the next packet transmission.

Similar to Transmit EMAC interface, the data of one packet must be transferred continuously in Receive EMAC interface. Valid signal must be asserted to '1' from the start of the packet to the end of the packet, as shown in Figure 11.

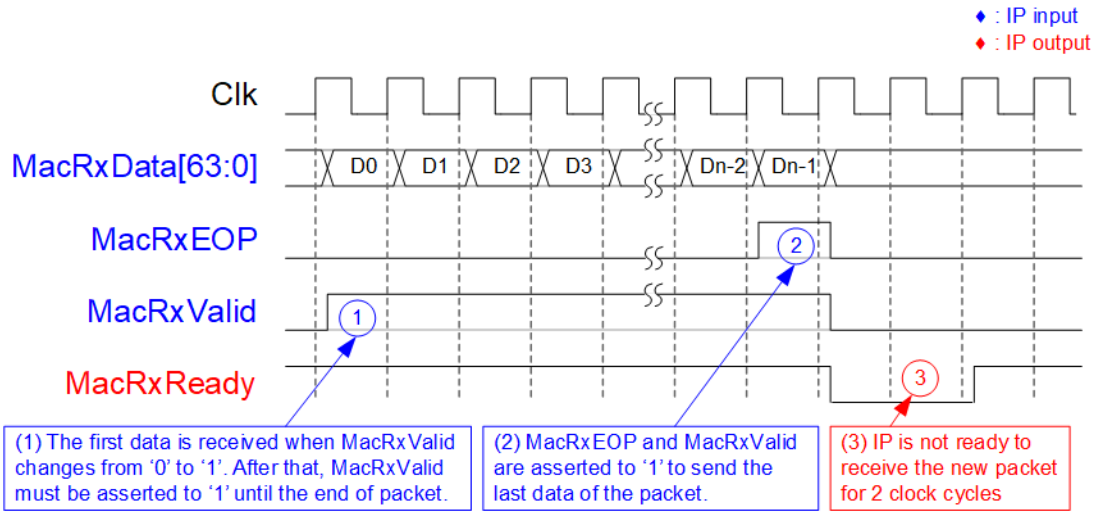

Figure 11: Receive EMAC Interface timing diagram

(1) TOE10G IP detects start of the receive frame when MacRxValid changes from '0' to '1' and the first data is valid on MacRxData. After that, MacRxReady is asserted to '1' to accept all data until the end of the packet. MacRxValid must be asserted to '1' for sending the data of one packet continuously.

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

(3) After that, TOE10G IP de-asserts MacRxReady for 2 clock cycles to complete the packet post processing. So, EMAC must support to pause the data packet transmission after the end of packet for 2 clock cycles.
Note: Typically, EMAC has at least two-clock cycle gap size between each received packet for receiving Ethernet FCS, EFD, and IFG.

Example usage

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

The example steps to set register for transferring data in Client mode are 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.
   Note: DPN is optional setting which is applied when the port is opened by IP (Active open).
3) Set RST register='0' to start the IP initialization process by sending ARP request packet to get Target MAC address from ARP reply packet. Busy signal is de-asserted to '0' after finishing 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 TOE10G IP). After that, wait until Busy flag is de-asserted to '0'.
   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 transmit length) and PKL register (packet size). Next, set CMD register = "Send Data" to start data transmission. The user sends the data to TOE10G IP via TxFIFO interface before or after setting CMD register. When the command is finished, busy flag is de-asserted 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 can be destroyed by two modes.
   a. Active close: Set CMD register = "Close connection" to close the connection (FIN packet is firstly sent by TOE10G IP). After that, wait until Busy flag is de-asserted to '0'.
   b. Passive close: Wait until "ConnOn" signal = '0' (FIN packet is sent from the target to TOE10G IP firstly).

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

Comparing to Client mode which MAC address is decoded from ARP reply packet after TOE10G IP sends ARP request packet, Server mode decodes MAC address from ARP request packet. The process for transferring data is similar to Client mode. The example steps for running in Server mode are 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 start the IP initialization process by waiting for ARP request packet to get Target MAC address. Next, the IP creates ARP reply packet returned to the target device. After finishing the initialization, busy signal is de-asserted to '0'.
4) Remaining steps are similar to step 4 – 6 of Client mode

PKL and TDL setting in Send command

When running Send command, the IP can run in two modes. First is when TDL is equal to N times of PKL. Second is when TDL is not equal to N times of PKL. More details of each mode are described as follows.

TDL = N times of PKL

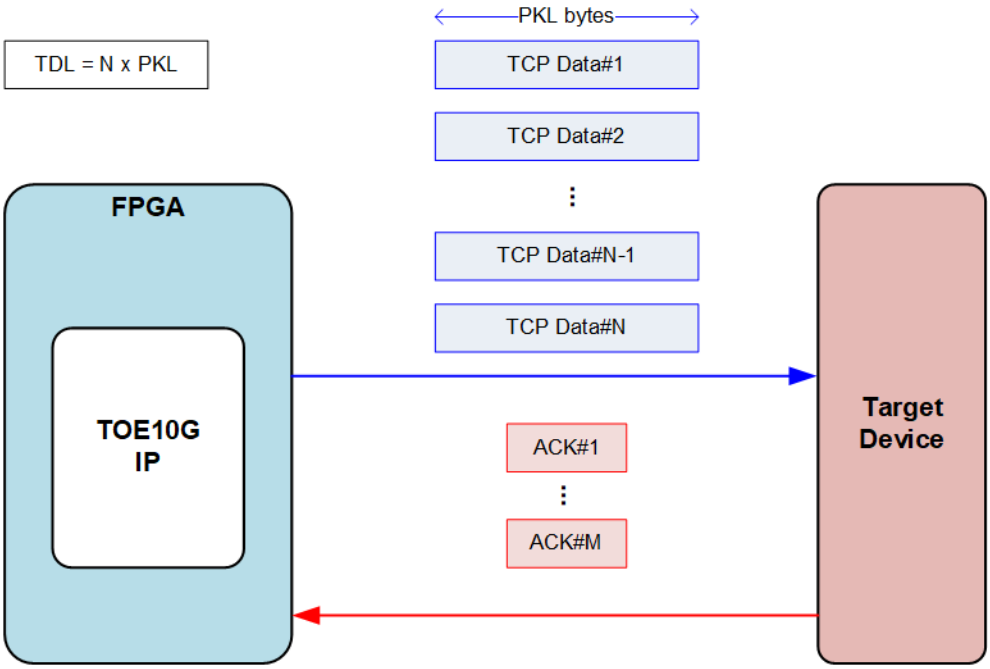

Figure 12: TCP packet when TDL = N times of PKL

When TDL value is equal to N times of PKL value, the data from user is split to N packets and forwarded to the target device, as shown in Figure 12. If the target device returns ACK packet to be response for every TCP packet, there are N ACK packets in the network system. To improve network performance, the several ACK packets are combined to be one packet. This technique is called TCP delayed ACK. Therefore, the numbers of ACK packet returned from the target device may be less than the numbers of data packet from TOE10G IP when running Send command.

PSH[0] set value is not effect for this condition. The last data packet (TCP Data#N) is sent only one time.

TDL = N times of PKL + Residue

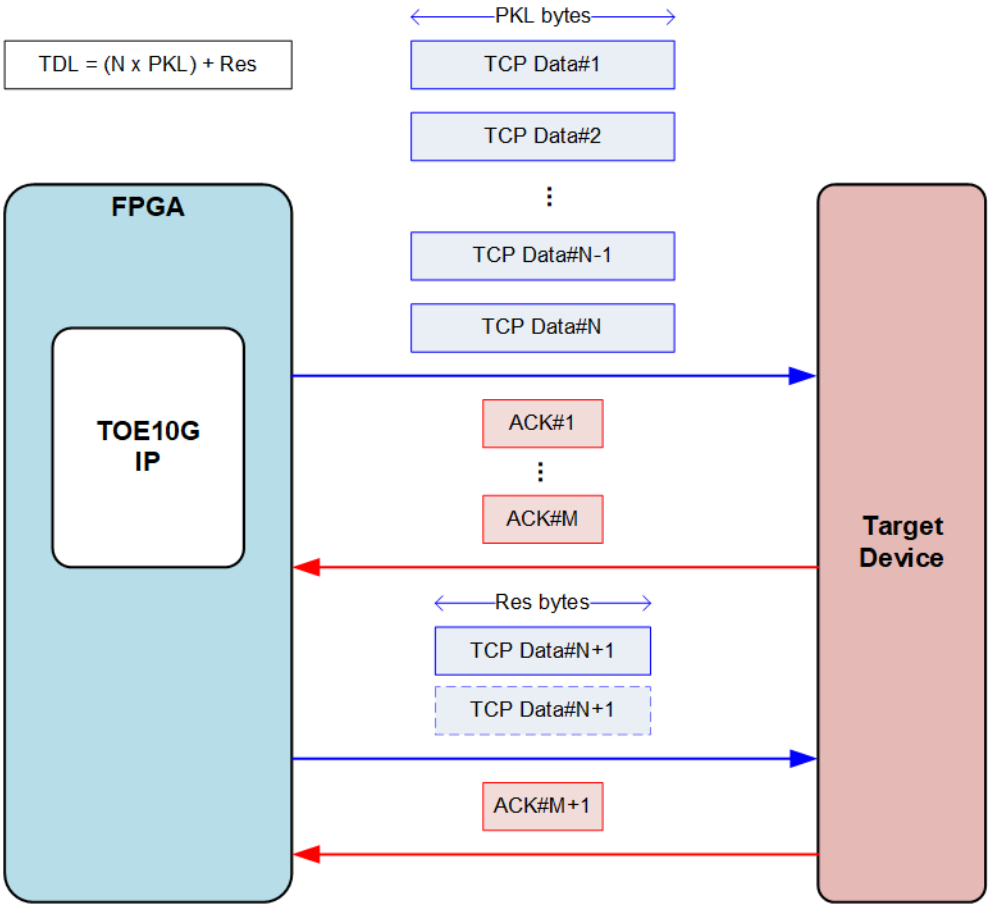

Figure 13: TCP packet when TDL = (N times of PKL) + Residue

When TDL value is not equal to N times of PKL value, the data sent to the target device is split to N packets of PKL-byte data and the last packet which has Res-byte data. As shown in Figure 13, the first step is similar to the condition that TDL is equal to N times of PKL. The IP needs to receive the ACK packet from the target device to confirm all N-packet is received completely. After that, the last packet which consists of the residue byte data is sent to the target device. If PSH[0] register is equal to '0' (default value), the residue packet is sent two times. Otherwise, the last packet is one time sent. The send command is finished when ACK from the target is returned to confirm the last packet is received.

Note: If target device is run on some OSs which enables delayed ACK feature, ACK#M packet, returned to confirm that TCP Data#N packet is accepted, may be arrived so late by timeout condition in some conditions. Therefore, the target device needs to disable delayed ACK feature or the TDL value should be aligned to PKL value in the system that is rather sensitive to this latency time.

Verification Methods

The TOE10G IP Core functionality was verified by simulation and also proved on real board design by using Arria10 SoC/Arria10 GX/Cyclone10 GX/Stratix10 GX/Stratix10 MX development 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 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
1.0	18-May-2016	New release
1.1	10-Aug-2017	Add SRV register and support Arria10 GX board
1.2	30-May-2019	Add PSH[2] and TMO[7] to support for sending reverse data packet
1.3	15-Aug-2019	Add support Cyclone10GX board and DG EMAC IP
1.4	19-Jun-2020	Update Application details
1.5	26-Aug-2020	Support S10GX device
1.6	2-Oct-2020	Update Figure3 and Figure4
1.7	4-Mar-2021	Support Stratix10 MX
1.8	23-Sep-2021	Add PKL/TDL setting in Send command topic