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

Core Facts

Provided with Core	
Documentation	Reference design manual Demo instruction manual
Design File Formats	Encrypted HDL
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference design manual
Additional Items	Demo on VCU118/KCU116
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- TCP/IP stack implementation
- Support IPv4 protocol
- Support one session per one TOE25G IP (Multisession can be implemented by using multiple TOE25G IPs)
- Support both Server and Client mode (Passive/Active open and close)
- Support Jumbo frame
- Transmit packet size aligned to 128-bit, bus size of transmitted data
- Total receive data size aligned to 128-bit, bus size of received data
- Transmit/Receive buffer size, adjustable for resource and performance balancing
- Simple data interface by 128-bit FIFO interface
- Simple control interface by 32-bit Register interface
- 64-bit AXI4 stream to interface for 10G/25G Ethernet MAC
- User clock frequency must be more than or equal to 195.3125 MHz for 25Gb Ethernet/156.25 MHz for 10Gb Ethernet
- Support 10G/25GbE by using 10G/25G Ethernet MAC and PCS
- Reference design available on VCU118/KCU116 board
- Not support data fragmentation feature
- Customized service for following features
  - Unaligned 128-bit data transferring
  - Buffer size extension by using Windows Scaling feature
  - Network parameter assignment by other methods

Table 1: Example Implementation Statistics for Ultrascale device

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB[1]	IOB	BRAMTile[2]	Design Tools
Virtex-Ultrascale+	XCVU9P-FLGA2104-2L-E	350	3896	4251	829	-	36	Vivado2019.1
Kintex UltraScale+	XCKU5P-FFVB676-2-E	350	3896	4252	864	-	36	Vivado2019.1

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

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

Applications

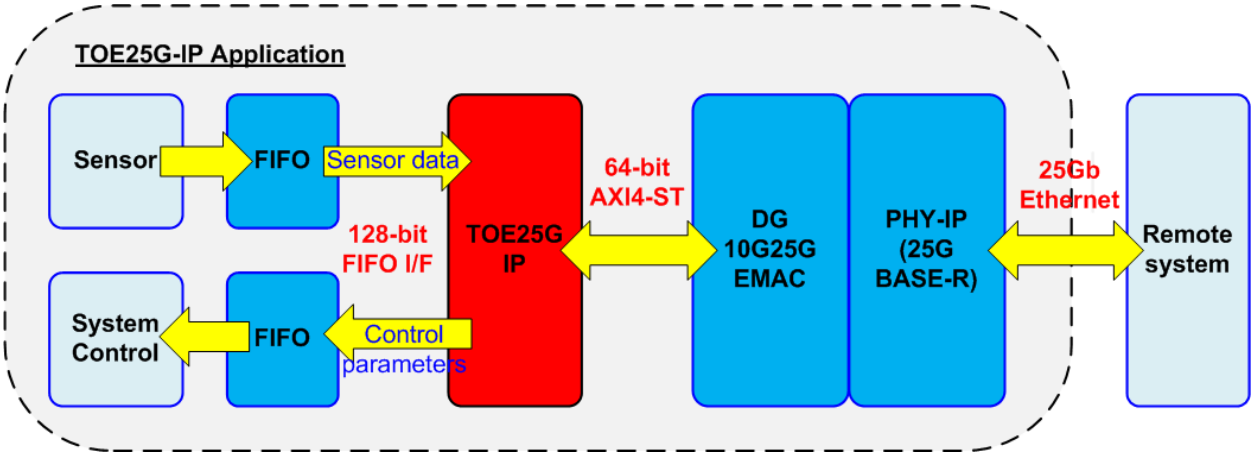

Figure 1: TOE25G IP Application

25Gb Ethernet is the communication channel which can transfer data at very high speed with remote controlling system. By using TCP/IP protocol for transferring via 25Gb Ethernet, the system can transfer very big data at very high speed rate with reliability. TOE25G IP is the IP which is integrated to the system for transferring data via 25Gb Ethernet without using CPU and external memory. So, the IP can fit with the application which needs to send or receive data at ultra high-speed rate based on 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 25Gb Ethernet by TOE25G IP. TOE25G IP is desigend to support full-duplex transfer in the same session, so Remote system can send the parameters for controlling the sensor monitoring system via 25Gb Ethernet.

General Description

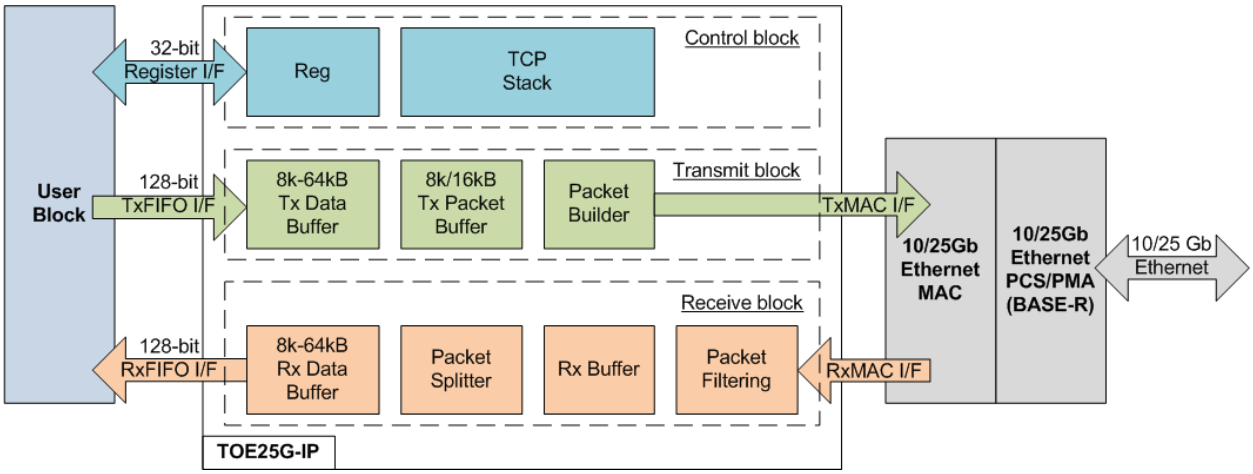

Figure 2: TOE25G IP Block Diagram

TOE25G IP core implements TCP/IP stack by hardware logic and connects with 10/25 Gb EMAC IP and PCS/PMA (BASE-R) module as the lower layer hardware. User interface of TOE25G IP consists of two interfaces, i.e. Register interface for control signals and FIFO interface for data signals.

Register interface has 5-bit address for accessing up to 32 registers, consisting of network parameters, command register and system parameters. The IP supports only one session, so the network parameters set by the user are fixed for assigning to TOE25G IP and the target device. After that, start IP initialization by de-asserting reset signal. Also, the reset process is necessary when some network parameters must be changed. The initialization process has three modes to get MAC address of the target device. After finishing the initialization process, the IP is ready for transferring data with the target device.

Following TCP/IP standard, the connection must be created by opening the port as the first step. The IP supports for both active open (the port opened by the IP) or passive open (the port opened by the target device). After that, data can be transferred from both sides. To send the data, the user sets total transfer size and packet size to the IP and then transfers the data via TxFIFO interface which is 128-bit data size. When the data is received from the target, the user reads the received data from the IP via RxFIFO interface. After finishing data transferring, the connection can be destroyed by using active close (the port closed by the IP) or passive close (the port closed by the target).

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, i.e. Tx data buffer and Tx packet buffer. In Rx path, one buffer is available, named 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

TOE25G IP core can be divided into three parts, i.e. control block, transmit block and receive block.

Control Block

- **Reg**

  All parameters of the IP are set via register interface which has 5-bit address signals and 32-bit data signals. Timing diagram of register interface is similar to single-port RAM interface. The address is shared for both write and read directions. The description of each register is defined as shown in Table 2.

Table 2: Register map Definition

RegAddr [4:0]	Reg Name	Dir	Bit	Description
00000b	RST	Wr /Rd	[0]	Reset IP. '0': No reset, '1': Reset. Default value is '1'. **After all network parameters are assigned, the user sets '0' to this register for loading parameter and starting system initialization. User must set this register to '1' and '0' respectively when some network parameters are changed. The network parameters controlled by RST register are SML, SMH, DML, DMH, DIP, SIP, DPN, SPN and SRV register.**
00001b	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, user needs to confirm that the system is in Idle status by checking busy signal de-asserted to '0'. Busy signal is read by bit[0] of CMD register or bit[0] of RegDataA1 signal.**
		Rd	[0]	System busy flag. '0': Idle, '1': IP is busy.
			[3:1]	Current IP status. "000": Send data, "001": Idle, "010": Active open, "011": Active close, "100": Receive data, "101": Initialization, "110": Passive open, "111": Passive close.
00010b	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.**
00011b	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.**
00100b	DIP	Wr /Rd	[31:0]	Define 32-bit target IP address. **To update this value, the IP must be reset by RST register.**
00101b	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.**
00110b	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.**
00111b	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.**
01000b	TDL	Wr	[31:0]	Total Tx data length in byte unit, but the length must be aligned to 16-byte (data bus size). Valid from 16-0xFFFFFFF0 (Bit[3:0] 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 (Busy='1'), the user can set the new value of 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	[31:0]	Remaining transfer length in byte unit which does not transmit.

RegAddr [4:0]	Reg Name	Dir	Bit	Description
01001b	TMO	Wr	[31:0]	Define timeout value for waiting Rx packet returned from the target. The counter is run under UserClk, so timer unit is equal to 1/UserClk. TimerInt is asserted to '1' when the packet is not received in time. Please see more details of TimerInt from Read value of TMO[7:0] register. This value is recommended 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
01010b	PKL	Wr /Rd	[15:0]	TCP data length of one Tx packet in byte unit, but the length must be aligned to 16-byte. Valid from 16-16000. Default value is 1456 byte which is the maximum size of non-jumbo frame that is aligned to 16-byte. Bit[3:0] 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 when the next command uses the same packet length.**
01011b	PSH	Wr /Rd	[2:0]	Sending mode when running Send data command. [0]-Disable to retransmit packet. '0': Generate the duplicate data packet for the last data packet in Send data command (default). '1': Disable the duplicate data packet. [1]-PSH flag in the transmitted packet. '0': PSH flag in TCP header of all transmitted packet is '0' (default). '1': PSH flag in TCP header of all transmitted packet is '1'.

RegAddr [4:0]	Reg Name	Dir	Bit	Description
01011b	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 from the window update packet lost. Data retransmission can activate the target device to regenerate the lost ACK 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.
01100b	WIN	Wr /Rd	[5:0]	Threshold value in 1Kbyte unit for sending window update packet. Default value is 0 (Not enable window update feature). The IP transmits the window update packet when the free space of the receive 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 the window size of the latest transmitted packet is equal to 2 Kbyte. After that, the user reads 1 Kbyte data from the IP. Free space of the receive buffer is updated from 2 Kbyte to be 3 Kbyte. The IP detects that the increased window 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.
01101b	ETL	Wr	[31:0]	Extended total Tx data length in byte unit. The size must be aligned to 16 byte. Bit[3:0] is ignored by the IP. User sets this register during 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 (0xFFFFFFF0 – read value of TDL). For example, the user sets TDL = 3.5 Gbyte in the first send data command. After the IP completes 2 Gbyte data (remaining size = 1.5 Gbyte), the user sets ETL register = 1.5 Gbyte.The total length is equal to 5 Gbyte (3.5 Gbyte of TDL + 1.5 Gbyte of ETL).
01110b	SRV	Wr /Rd	[1:0]	"00": 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. "01": Server mode. After RST register changes from '1' to '0', the IP waits 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'. "1x": Fixed MAC Mode. After RST register changes from '1' to '0', the IP updates the parameters and then de-asserts IP busy to '0'. Target MAC address is loaded by DML/DMH register. **Note: In Server mode, when RST register changes from '1' to '0', the target device needs to resend ARP request for TOE25G IP completing the IP initialization.**
01111b	VER	Rd	[31:0]	IP version
10000b	DML	Wr /Rd	[31:0]	Define 32-bit lower target MAC address (bit [31:0]) for this IP when SRV[1:0]="1x" (Fixed MAC). **To update this value, the IP must be reset by RST register.**
10001b	DMH	Wr /Rd	[15:0]	Define 16-bit upper traget MAC address (bit [47:32]) for this IP when SRV[1:0]="1x" (Fixed MAC). **To update this value, the IP must be reset by RST register.**

- **TCP Stack**

  TCP stack is the main controller of the IP for controlling the other modules in every process. The IP operation has two phases, i.e. IP initialization phase and data transferring phase.

  After RST register changes from ‘1’ to ‘0’, the initialization phase begins. There are three modes for running the initialization phase, set by SRV[1:0] register, i.e. Client mode, Server mode or Fixed MAC 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 TOE25G IP and the target device, it consists of three processes, i.e. opening the port, transferring data and closing the port. The IP supports to start the port opening or closing 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 as passive mode when TCP Stack detects SYN or FIN packet from Receive block. During port opening or closing process, 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 monitored to confirm the port status that 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 and Tx packet buffer. After the network parameters are read to build TCP header by Packet Builder, Transmit block sends TCP packet including the data from the user to the target device via Ethernet MAC. When 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 transferred 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 data lost 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	8kByte	Valid	Valid	Valid
10	16kByte	Valid	Valid	Valid
11	32kByte	Valid	No	Valid
12	64kByte	Valid	No	Valid

Transmit Block

There are two buffers in Transmit block, i.e. Tx data buffer and Tx packet buffer which the size can be adjusted by parameter assignment. Using bigger size may increase the transmit performance. The minimum size of Tx data buffer and Tx packet buffer is limited by the transmit packet size, set by PKL register. Data from Tx data buffer is split to be one packet size stored in Tx packet buffer. TCP header is prepared and the combined with TCP data stored in Tx packet buffer to build complete TCP packet. The data in Tx data buffer can be flushed after the target device returns ACK packet to confirm that the data is completely received. After finishing the send data command, the user can change the packet size and total data size for the new send data command by updating PKL and TDL register respectively.

- **Tx Data Buffer**

  This buffer size is set by “TxBufBitWidth” parameter of the IP. The valid value is 9-12 which is equal to the address size of 128-bit buffer, as shown in Table 3. The buffer size should be more than or equal to two times of Tx packet size, set by PKL register. This buffer stores the data from the user for preparing the transmit packet sent to the target device. Data is removed from the buffer when the target device confirms that the data is completely received. Consequently, when the buffer size is large, the IP can send many data to the target device without waiting ACK packet returned from the target to clear the buffer. If the ACK packet to clear the buffer is received and the user can fill the new data to the IP in time, the IP can send the new data to the target continuously without waiting the new data from the user. As a result, the system can achieve the best transmit performance on 25Gb Ethernet connection. Nevertheless, when the carrier and the networking interface have much latency time, all data in the Tx data buffer is completely transferred before the ACK packet to flush the buffer is received and the new data is received from the user. As a result, the transmit performance is reduced from the latency time.

  If total data from user is more than the value of TDL register, the data still remains 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 buffer size is set by “TxPacBitWidth” parameter of the IP. The valid value is 9-10 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> – 48).

- **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 TCP header. After that, all TCP header is 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, TOE25G 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), TOE25G IP cannot reorder the data and detect as data lost condition. After that, the data recovery process is run by generating duplicate 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 verified by this module to validate the packet. The packet is valid when the following conditions are met.

(1) Network parameters are matched to the 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 9-12 for 8Kbyte – 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 TOE25G 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 is full and then the target device cannot send more data to the IP. So, 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 sending or receiving data with the IP.

10G/25G Ethernet MAC

Ethernet MAC implements the MAC layer for 10/25Gb Ethernet. The user interface to connect with TOE25G IP is 64-bit AXI4 stream. TOE25G IP can directly connect with DG 10G25GEMAC IP, but the additional logic with small FIFO must be designed when connecting with Xilinx 10/25G EMAC IP.

Design Gateway provides 10G25GEMAC IP which can be applied with 10G/25G Ethernet solution. The resource can be optimized with less latency time. More details of DG 10G25GEMAC IP Core are described in following website.

https://dgway.com/products/IP/10GEMAC-IP/dg_10g25gemacip_data_sheet_xilinx_en.pdf

Xilinx provides 10/25G Ethernet Subsystem (Ethernet MAC and Ethernet PCS/PMA) with many features, described in the following website.

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

10/25G Ethernet PCS/PMA (10/25GBASE-R)

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

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

Core I/O Signals

Descriptions of all parameters and I/O signals are provided in Table 4 - Table 6. The EMAC interface is 64-bit AXI4 stream interface.

Table 4: Core Parameters

Name	Value	Description
TxBufBitWidth	9-12	Setting Tx data buffer size. The value is the address bus size of this buffer.
TxPacBitWidth	9-10	Setting Tx packet buffer size. The value is the address bus size of this buffer.
RxBufBitWidth	9-12	Setting Rx data buffer size. The value is the address bus size of this buffer.

Table 5: User I/O Signals (Synchronous to Clk)

Signal	Dir	Description
Common Interface Signal		
RstB	In	Reset IP core. Active Low.
Clk	In	User clock for running TOE25G IP. The frequency must be more than or equal to 195.3125 MHz for 25Gb Ethernet.
User Interface		
RegAddr[4:0]	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=00001b)
RegDataA8[31:0]	Out	32-bit read value of TDL register (RegAddr=01000b)
RegDataA9[31:0]	Out	32-bit read value of TMO register (RegAddr=01001b)
Tx Data Buffer Interface		
TCPTxFfFlush	Out	Tx data buffer within the IP is reset. Assert 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[127:0]	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. Assert to '1' when the connection is opened.
TCPRxFfRdCnt[11:0]	Out	Data counter of Rx data buffer to show the number of received data in 128-bit unit.
TCPRxFfLastRdCnt[3:0]	Out	Remaining byte of the last data in Rx data buffer when total received data in the buffer is not aligned to 16-byte unit. User cannot read the data until all 16-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	Asserted to '1' to read data from Rx data buffer.
TCPRxFfRdData[127:0]	Out	Data output from Rx data buffer. Valid in the next clock cycle after TCPRxFfRdEn is asserted to '1'.

Table 6: EMAC I/O Signals (Synchronous to MacClk)

Signal	Dir	Description
MacClk	In	Receive clock from EMAC core which is equal to 390.625MHz for 25Gb Ethernet.
tx_axis_tdata[63:0]	Out	Transmitted data. Valid when tx_axis_tvalid='1'.
tx_axis_tkeep[7:0]	Out	Transmitted data byte enable. Valid when tx_axis_tvalid='1'.
tx_axis_tvalid	Out	Valid signal of transmitted data.
tx_axis_tlast	Out	Control signal to indicate the final word in the frame. Valid when tx_axis_tvalid='1'.
tx_axis_tuser	Out	Control signal to indicate an error condition. This signal is always '0'.
tx_axis_tready	In	Handshaking signal. Asserted to '1' when tx_axis_tdata has been accepted. This signal must not be de-asserted to '0' when a packet is transmitting.
rx_axis_tdata[63:0]	In	Received data. Valid when rx_axis_tvalid='1'
rx_axis_tvalid	In	Valid signal of received data. rx_axis_tvalid must be asserted to '1' continuously for transferring a packet.
rx_axis_tlast	In	Control signal to indicate the final word in the frame Valid when rx_axis_tvalid='1'.
rx_axis_tuser	In	Control signal asserted at the end of received frame (rx_axis_tvalid='1' and rx_axis_tlast='1') to indicate that the frame has CRC error.'0': normal packet, '1': error packet.
rx_axis_tready	Out	Handshaking signal. Asserted to '1' when rx_axis_tdata has been accepted. rx_axis_tready is de-asserted to '0' for 2 clock cycles to be the gap size between each received packet.

Timing Diagram

IP Initialization

The initialization process begins after user changes RST register from ‘1’ to ‘0’. TOE25G IP can run in three modes, set by SRV register, i.e. Client mode (SRV=”00”), Server mode (SRV=”01”), and Fixed MAC mode (SRV=”1x”). The details of each mode are shown in the following timing diagram.

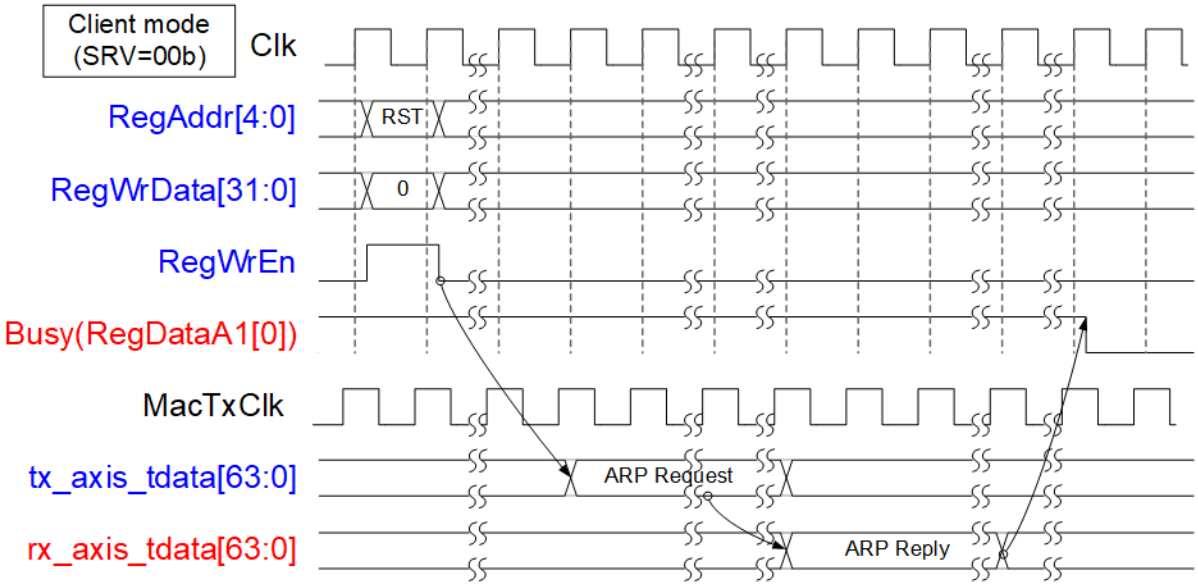

Figure 3: IP Initialization in Client mode

As shown in Figure 3, in Client mode TOE25G IP sends ARP request and waits ARP reply returned from the target device. Target MAC address is extracted from ARP reply packet. After finishing, Busy signal is de-asserted to ‘0’.

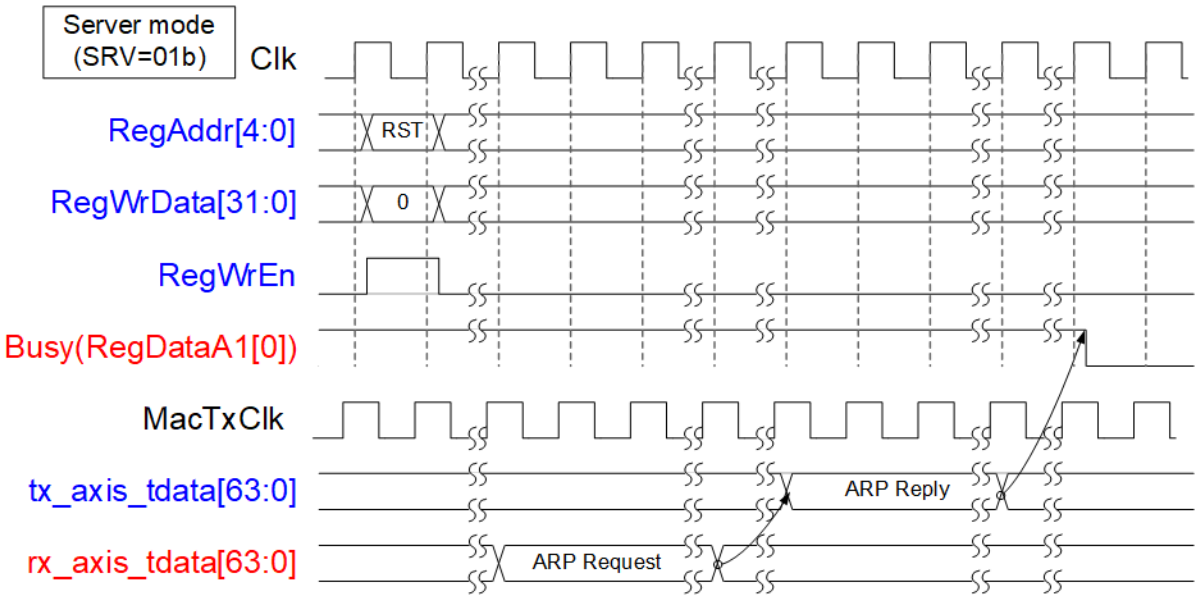

Figure 4: IP Initialization in Server mode

As shown in Figure 4, after reset process in Server mode, TOE25G IP waits ARP request sent by the target device. After that, TOE25G IP returns ARP reply to the target. Target MAC address is extracted from ARP request packet. Finally, Busy signal is de-asserted to '0'.

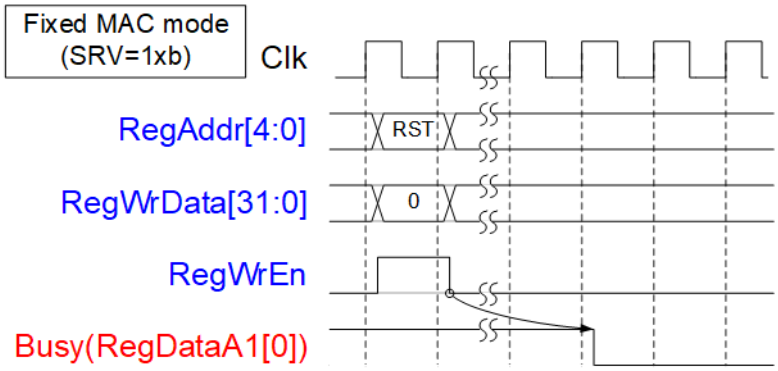

Figure 5: IP Initialization in Fixed mode

As shown in Figure 5, after reset process in Fixed MAC mode, TOE25G IP updates all parameters from the registers. Target MAC address is loaded from DML and DMH register. After finishing, 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 6, 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.

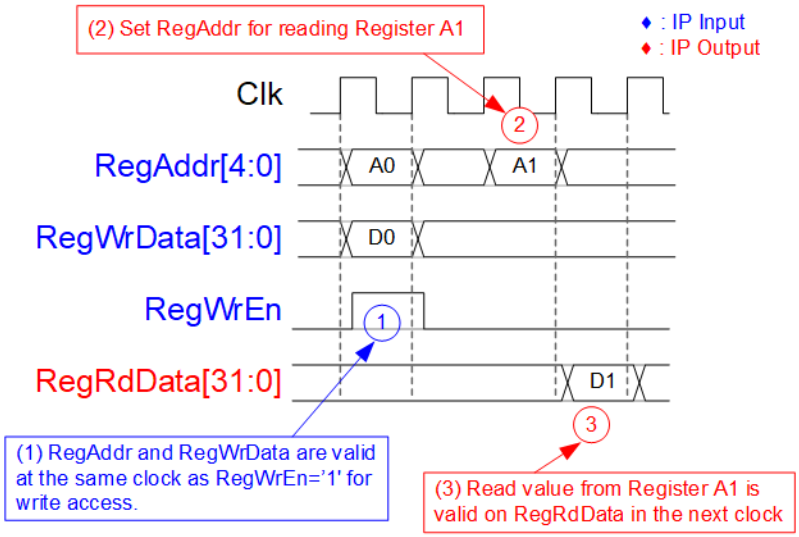

Figure 6: Register interface timing diagram

As shown in Figure 7, 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.

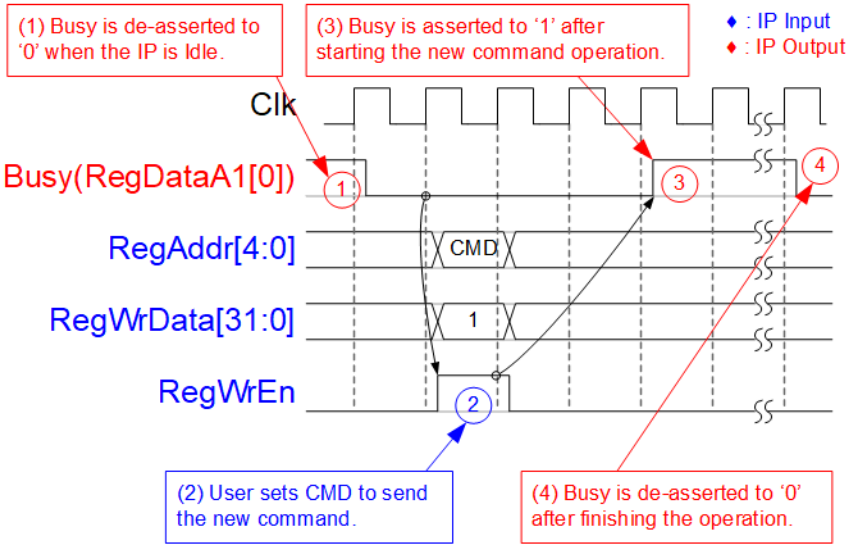

Figure 7: CMD register timing diagram

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 8.

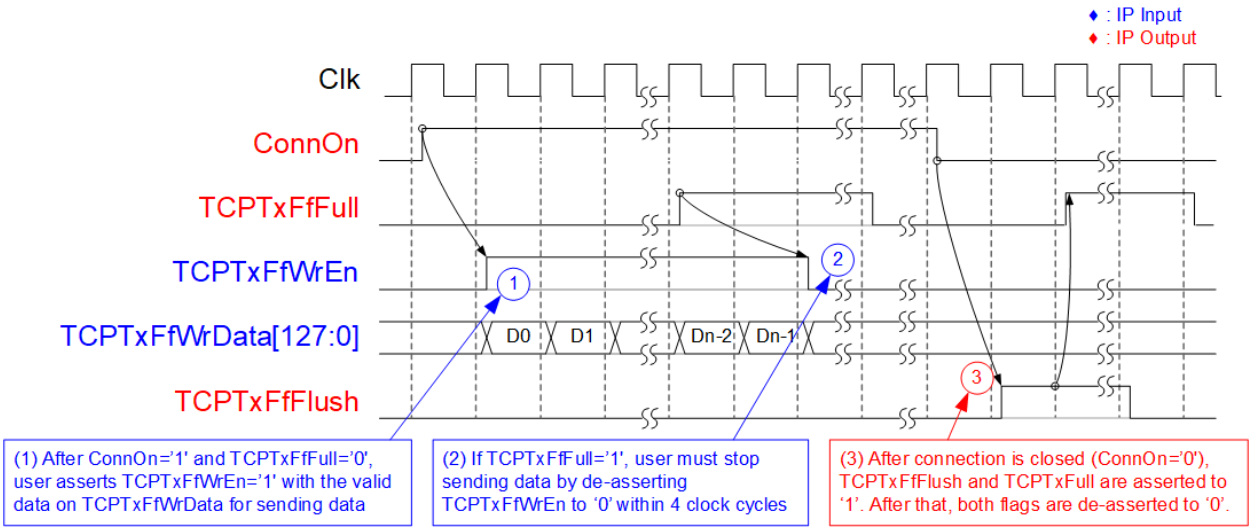

Figure 8: Tx FIFO interface timing diagram

(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 TOE25G 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 9.

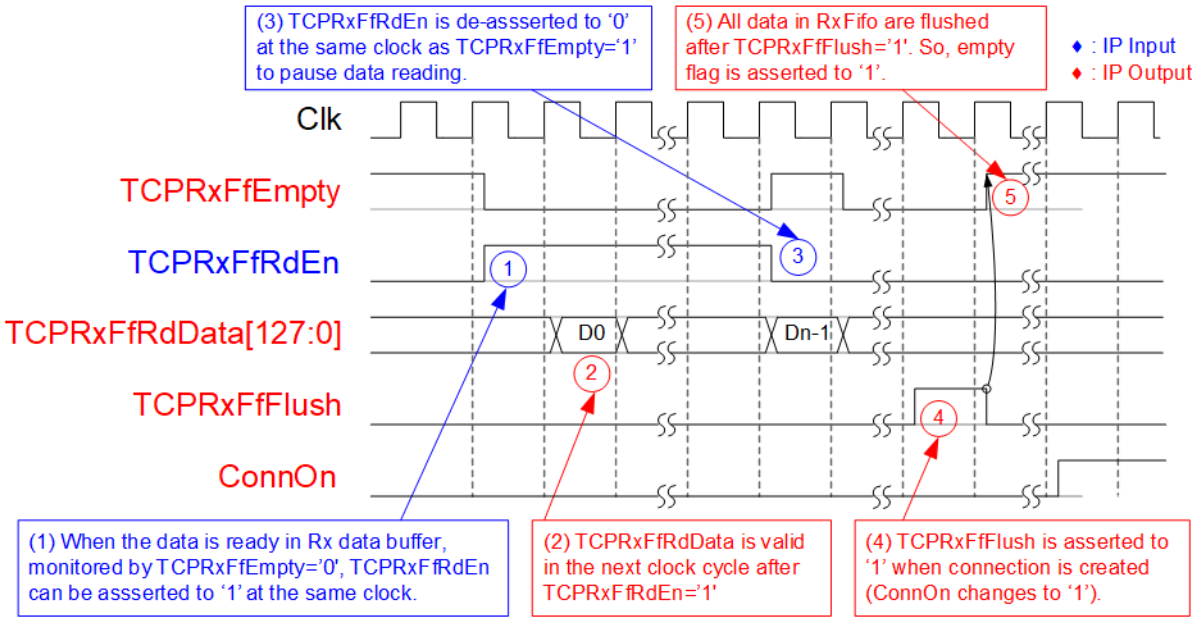

Figure 9: Rx FIFO interface timing diagram by using Empty flag

(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 TCPRxFfRdEn='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'.

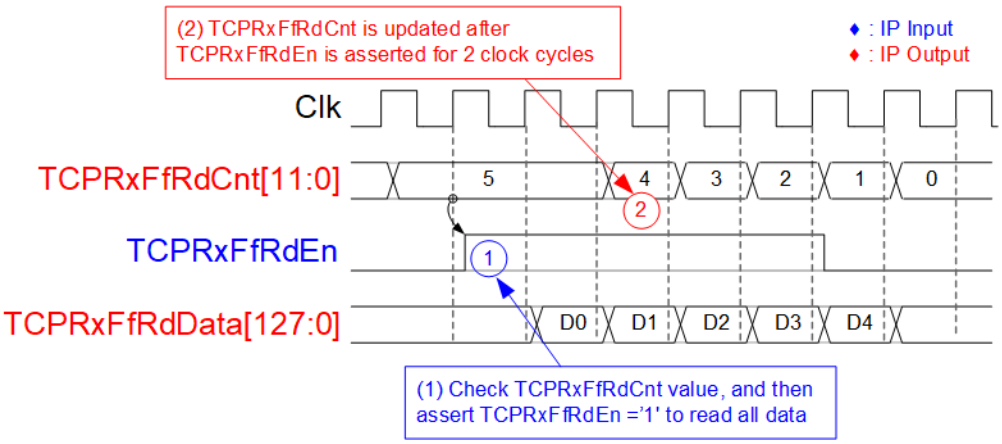

Figure 10: Rx FIFO interface timing diagram by using read counter

If user logic reads data as burst mode, TOE25G IP has read counter signal to show the total data stored in Rx FIFO interface as 128-bit unit. For example, Figure 10 shows five data available in Rx data buffer. So, user can assert TCPRxFfRdEn to '1' for 5 clock cycles to read all data from Rx data buffer. The latency time between read counter (TCPRxFfRdCnt) and read enable (TCPRxFfRdEn) is 2 clock cycles.

EMAC Interface

EMAC interface of TOE25G IP is designed by using 64-bit AXI4-stream interface. The limitation is that TOE25G IP cannot pause data transmission when the packet does not end. So, tx_axis_tready must be asserted to '1' during transmitting a packet. tx_axis_tready can be de-asserted to '0' after the last data in the packet is transferred, as shown in Figure 11.

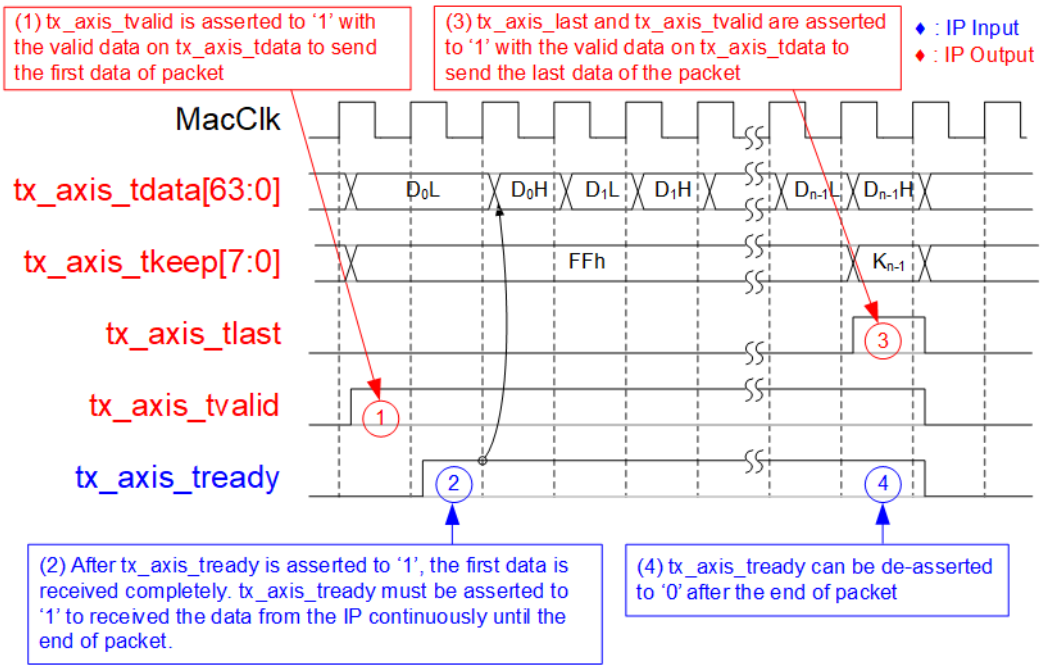

Figure 11: Transmit EMAC interface timing diagram

(1) TOE25G IP asserts tx_axis_tvalid with the first data of the packet. All signals are latched until tx_axis_tready is asserted to '1' to accept the first data.
(2) After the first data is accepted by EMAC, tx_axis_tready must be asserted to '1' to accept all remaining data in the packet from TOE25G IP until end of packet. The IP sends all data of one packet continuously.
(3) tx_axis_tlast and tx_axis_tvalid are asserted to '1' when the last data of the packet is transmitted.
(4) After the end of the packet, tx_axis_tready can be asserted to '0' to pause the next packet transmission.

Similar to Transmit EMAC interface, the data of one packet must be received 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 12.

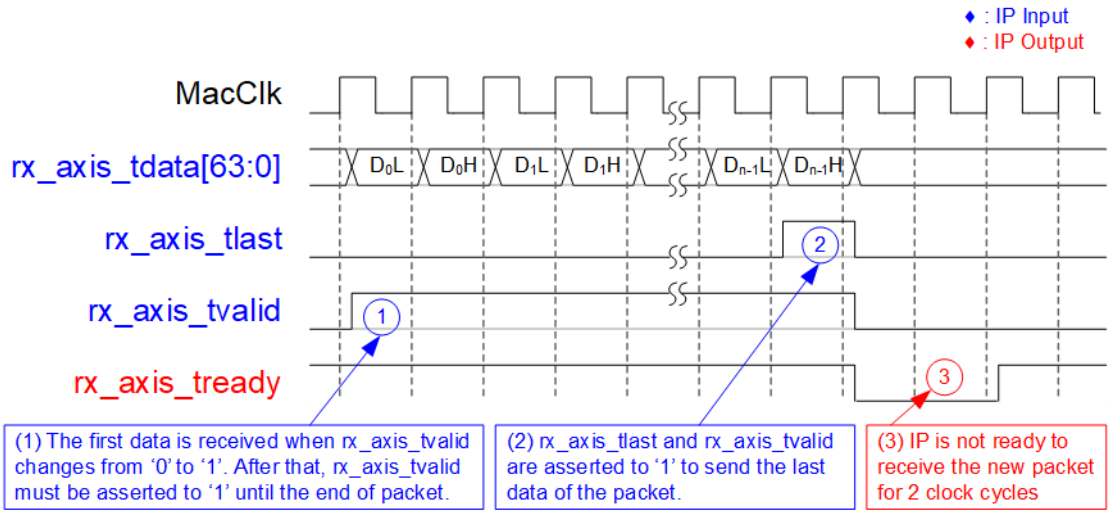

Figure 12: Receive EMAC interface timing diagram

(1) TOE25G IP detects start of the receive frame when rx_axis_tvalid changes from '0' to '1' and the first data is valid on rx_axis_tdata. After that, rx_axis_tready is asserted to '1' to accept all data until the end of the packet. rx_axis_tvalid must be asserted to '1' for sending the data of one packet continuously.

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

(3) After that, TOE25G IP de-asserts rx_axis_tready for 2 clock cycles to complete the packet post processing. So, EMAC must support to pause the data packet transmission for 2 clock cycles.

As the limitation of Tx and Rx EMAC interface that TOE25G IP must transfer the data of one packet continuously. However, Xilinx EMAC IP cannot guarantee that the data for Tx and Rx interface is transferred continuously for one packet. As a result, TOE25G IP does not support to directly connect with Xilinx EMAC IP (10/25G Ethernet Subsystem IP). The buffer within the additional logic is required to store the data when Xilinx EMAC is not ready for transferring the data in each packet. The reference design of TOE25G IP by using Xilinx EMAC IP can be requested.

The default reference design of TOE25G IP uses DG 10G25GEMAC IP which can directly connect without the additional logic.

Example usage

Client mode (SRV[1:0]=”00”)

The example step to set register for transferring data in Client mode is shown as follows.

1) Set RST register=’1’ to reset the IP.
2) Set SML/SMH for MAC address, DIP/SIP for IP address and DPN/SPN for port number.
   Note: DPN is optional setting 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 from TOE25G 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 TOE25G 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 TOE25G 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 TOE25G 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 TOE25G IP firstly).

Server mode (SRV[1:0]=”01”)

Comparing to Client mode which MAC address is decoded from ARP reply packet after TOE25G IP sends ARP request packet, Server mode decodes MAC address from ARP request packet. The process for transferring data is the same as Client mode. The example step of Server mode is shown as follows.

1) Set RST register=’1’ to reset the IP.
2) Set SML/SMH for MAC address, DIP/SIP for IP address and DPN/SPN for port number.
3) Set RST register=’0’ to start the IP initialization process by waiting 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

Fixed MAC mode (SRV[1:0]=”1x”)

In Fixed MAC mode, MAC Address of the target device is loaded by DML and DMH register. The process for transferring the data is similar to Client and Server mode. The example steps of Fixed MAC mode are shown as follows

1) Set RST register=’1’ to reset the IP.
2) Set SML/SMH for MAC address of TOE25G IP, DML/DMH for MAC address of the target device, DIP/SIP for IP address and DPN/SPN for port number.
3) Set RST register=’0’ to start the IP initialization process. After finishing the initialization, busy signal is de-asserted to ‘0’.
4) Remaining steps are similar to step 4 – 6 of Client mode

Verification Methods

The TOE25G IP Core functionality was verified by simulation and also proved on real board design by using VCU118/KCU116 evaluation board.

Recommended Design Experience

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

Ordering Information

This product is available directly from Design Gateway Co., Ltd. Please contact Design 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	Aug-5-2020	New release
1.1	Aug-24-2020	Rename IP from TenGEMAC to 10G25GEMAC
1.2	Sep-15-2020	Correct ARP figure and add KCU116 board