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

Core Facts

Provided with Core	
Documentation	User Guide, Design Guide
Design File Formats	Encrypted HDL
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
- Simple control interface by 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

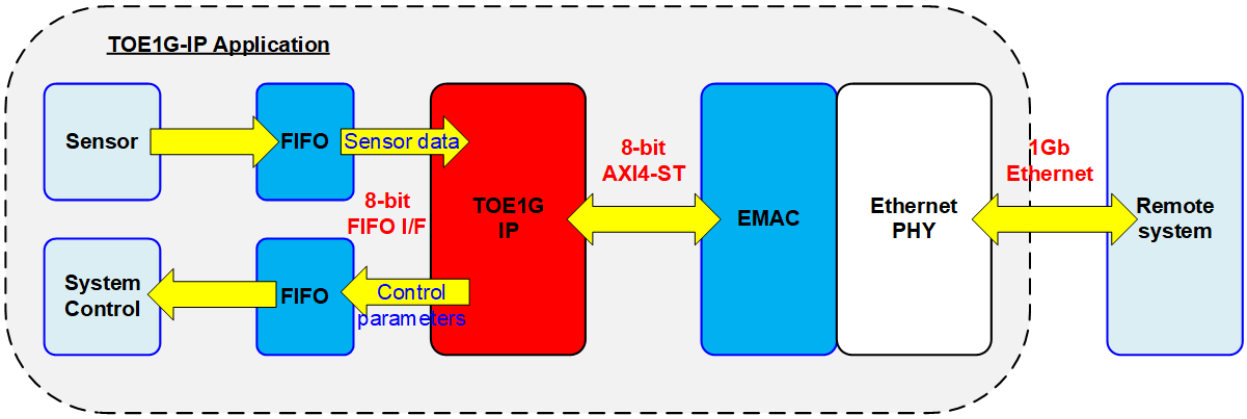

Figure 1: TOE1G IP Application

1 Gb Ethernet is the well-known communication channel for transferring data with remote controlling system. By using TCP/IP protocol for transferring data via 1Gb Ethernet, the system can transfer data with reliability. TOE1G IP is the IP which is integrated to the system for transfering data via 1Gb Ethernet without using CPU and external memory. So, the IP can fit with the application which needs to send or receive 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 1Gb Ethernet by TOE1G IP. TOE1G IP is desigend to support full-duplex transfer in the same session, so Remote system can send the paramter for controlling the sensor monitoring system via 1Gb Ethernet. In our website, we also have FTP server demo by using TOE1G IP. Please contact us for more details about FTP server demo.

TOE1G IP is designed for transferring data at the highest speed, so the latency time is much from internal pipeline register and buffer. For low-latency application such as FinTech, it is recommended to use our low-latency IP instead. Please see more details of low-latency IP from our website.

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

General Description

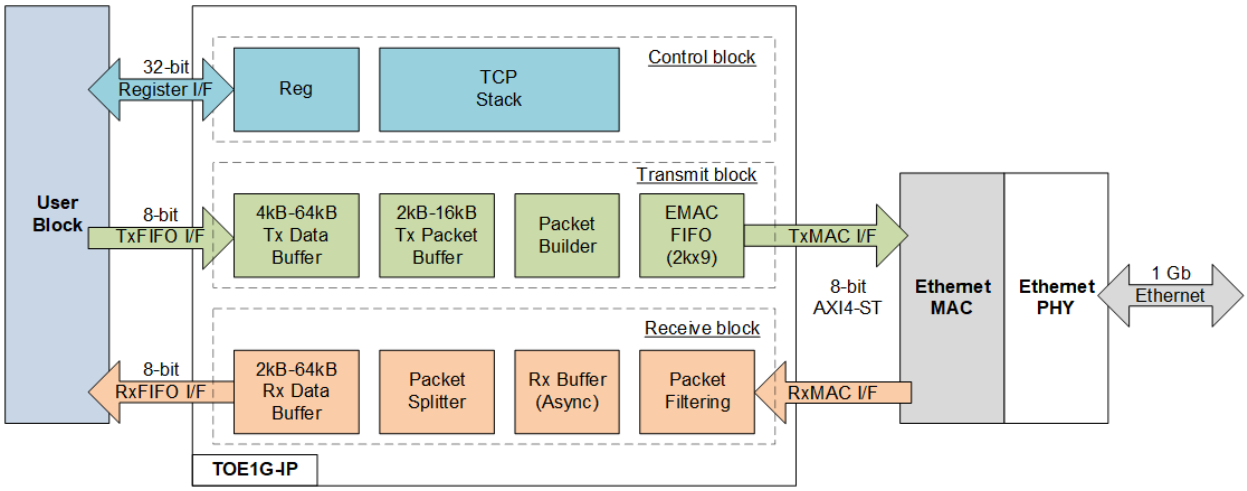

Figure 2: TOE1G IP Block Diagram

TOE1G IP core implements TCP/IP stack by hardwire logic and connects with EMAC IP and external PHY module as the lower layer hardware. User interface of TOE1G IP consists of two interfaces, i.e. Register interface for control signals and FIFO interface for data signals.

Register interface has 4-bit address for accessing up to 16 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 TOE1G 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 two 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 8-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

TOE1G-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 4-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 [3:0]	Reg Name	Dir	Bit	Description
0000b	RST	Wr /Rd	[0]	Reset IP. '0': No reset, '1': Reset. Default value is '1'. **After all parameters are assigned, the user sets '0' to this register for loading parameters and starting system initialization. User must set this register to '1' and '0' respectively when some network paramters are changed. 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, 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 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. Valid range is 1-0xFFFFFFFF. **User needs to set this register before setting CMD register = Send data (00b). The IP loads TDL register 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		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 125 MHz, so timer unit is equal to 8 ns. 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 setting PSH[2] 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 one 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='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.**
1011b	PSH	Wr /Rd	[2:0]	Sending mode when running Send data command. [0]-Disable to retransmit packet. '0': Generate the duplicated data packet for the last data packet in Send data command (default). '1': Disable the duplicated data packet. [1]-PSH flag in the transmit 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 [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 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.
1100b	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.
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 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 TOE1G IP completing the IP initialization.**
1111b	VER	Rd	[31:0]	IP version

- **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 two modes for running the initialization phase, set by SRV[0] register, i.e. Client mode or 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 TOE1G 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
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

There are two adjustable 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 then 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 12-16 which is equal to the address size of 8-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 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 1Gb Ethernet connection. Nevertheless, when the carrier and the networking interface have much latency time, all data in the Tx data buffer will be completely transferred before the ACK packet to flush the buffer is received and the new data is filled by 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 size is set by “TxPacBitWidth” parameter of the IP. The valid value is 11-14 and the description of the parameter is shown in Table 3. This buffer 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> – 4).

- **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.

- **EMacFIFO**

EMacFIFO is 2k x 9-bit FIFO to support flow control of transmit data 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, TOE1G 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), TOE1G 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. Also, the buffer is designed as asynchronous buffer to convert clock domain of received data stream from receive clock to be transmit clock.

- **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 11-16 for 2Kbyte – 64Kbyte buffer size. Rx data buffer size is applied to be the window size of the transmit packet. When Rx data buffer is big enough, the target device can send many data to TOE1G 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.

Ethernet MAC

The reference design of the IP uses TEMAC core from Xilinx. TOE1G IP can directly connect to Xilinx EMAC IP. More details of the IP are provided in following website.

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='1' in Write process.
RegWrData[31:0]	In	Register write data bus. Valid when RegWrEn='1'.
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 ('1' : connection is opened, '0': connection is closed)
TimerInt	Out	Timer interrupt. Assert to high for 1 clock cycle when timeout is detected. More details of Interrupt status could be checked from TMO[7:0] register.
Busy	Out	IP busy status ('0': IP is idle, '1': 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. Assert to '1' for 1 clock cycle 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[7:0]	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 new connection is opened.
TCPRxFfRdCnt[15:0]	Out	Data counter of Rx data buffer to show the number of received data in byte unit.
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[7:0]	Out	Data output from Rx data buffer. Valid in the next clock cycle after TCPRxFfRdEn is asserted to '1'.

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='1'.
MacTxLast	Out	Control signal to indicate the final byte in a frame.Valid when MacTxValid='1'.
MacTxUser	Out	Control signal to indicate an error condition. This signal is always '0'.
MacTxReady	In	Handshaking signal. Asserted to '1' 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 '1' continuously for transferring a packet.
MacRxData[7:0]	In	Received data bus from EMAC core. Valid when MacRxValid='1'.
MacRxLast	In	Control signal to indicate the final byte in the frame. Valid when MacRxValid='1'.
MacRxUser	In	Control signal asserted at the end of received frame to indicate that the frame has an error. '0': normal packet, '1': error packet. Valid when MacRxValid='1' and MacRxLast='1'.
MaxRxReady	Out	Handshaking signal. Asserted to '1' when MacRxData has been accepted. MacRxReady is de-asserted to '0' for 3 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’. TOE1G IP can run in two modes, set by SRV[0] register, i.e. Client mode (SRV[0]=’0’) and Server mode (SRV[0]=’1’). The details of each mode are shown in the following timing diagram.

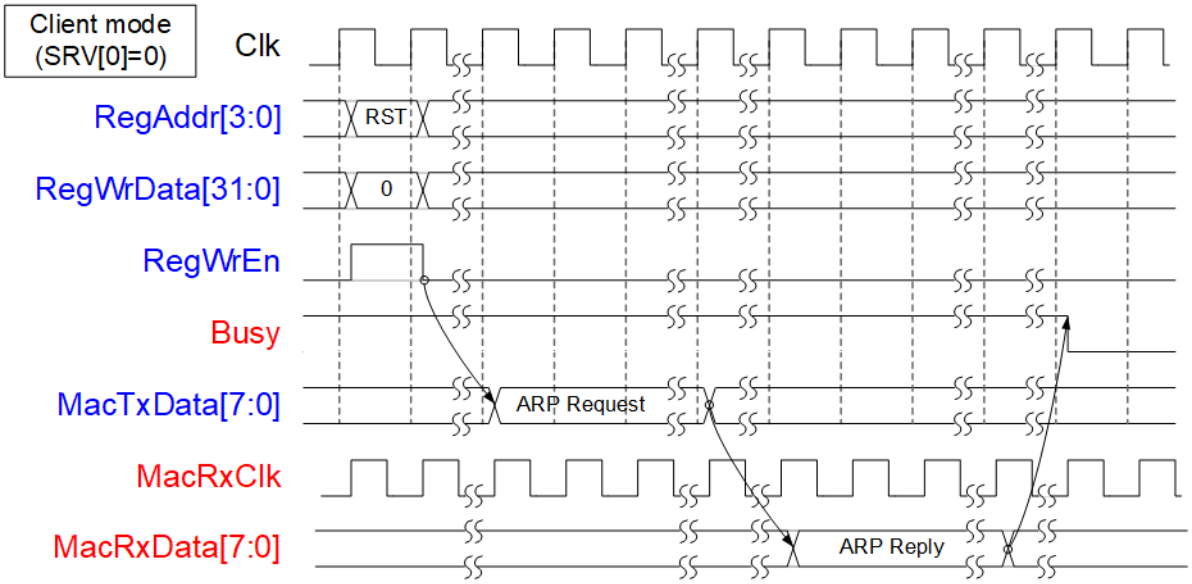

Figure 3: IP Initialization in Client mode

As shown in Figure 3, in Client mode TOE1G 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’.

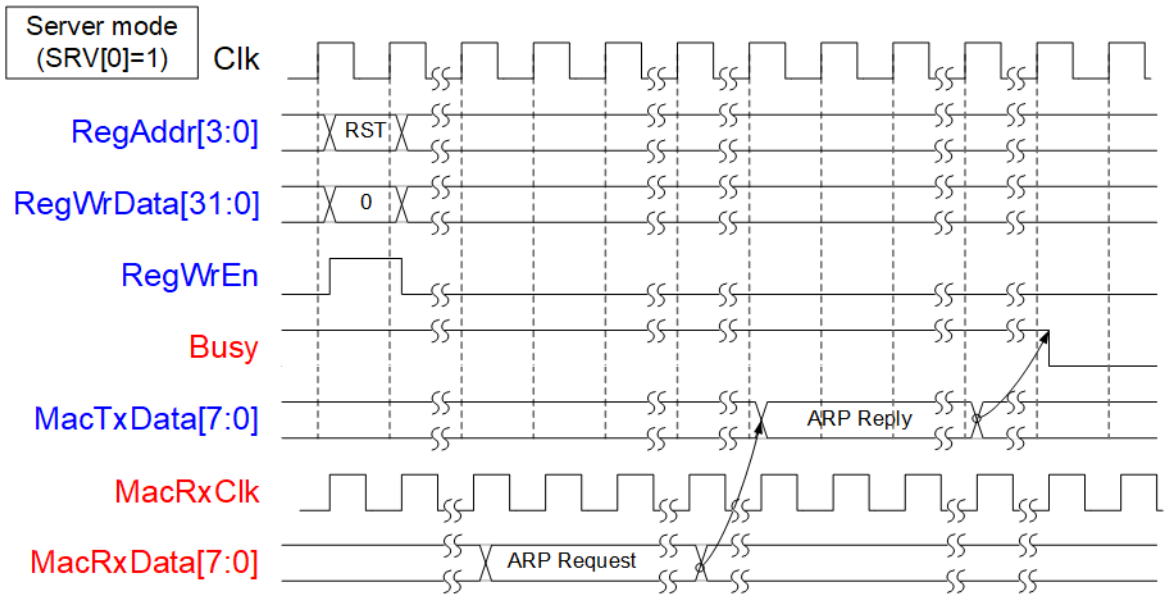

Figure 4: IP Initialization in Server mode

As shown in Figure 4, after reset process in Server mode, TOE1G IP waits ARP request sent by the target device. After that, TOE1G 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.

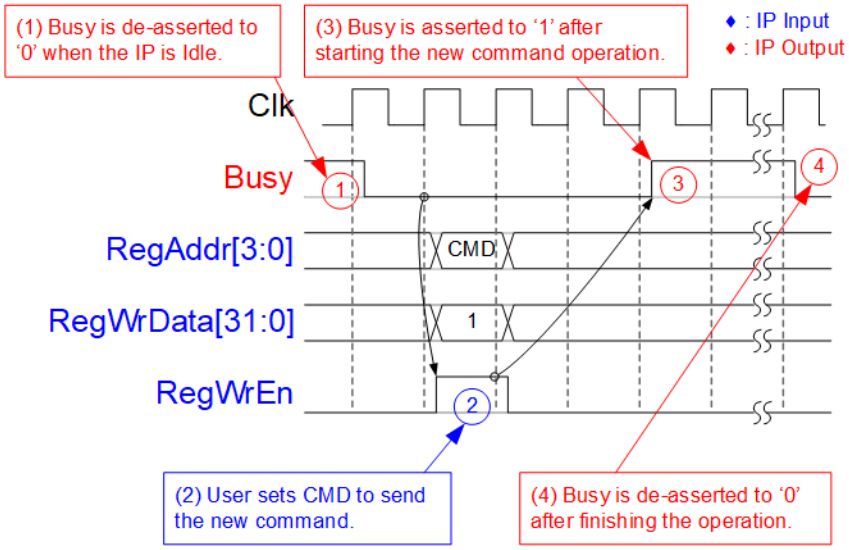

Figure 5: Register interface timing diagram

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.

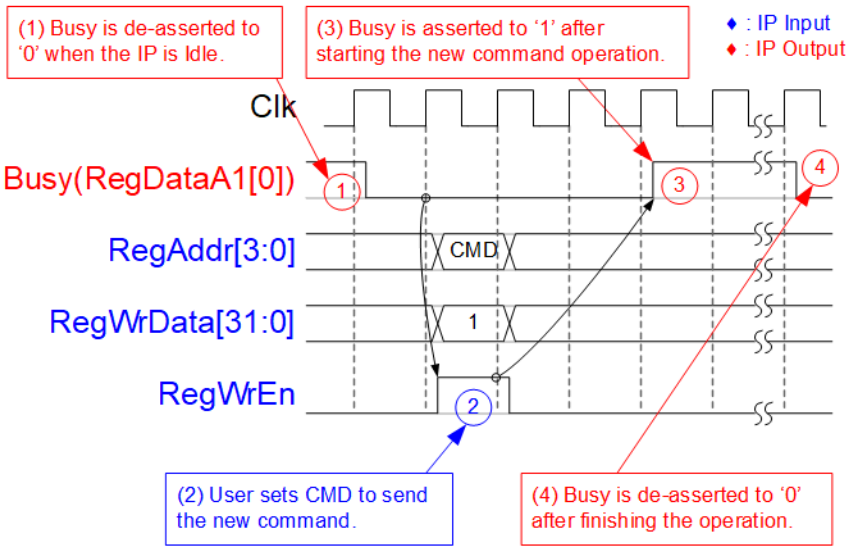

Figure 6: CMD register timing diagram

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.

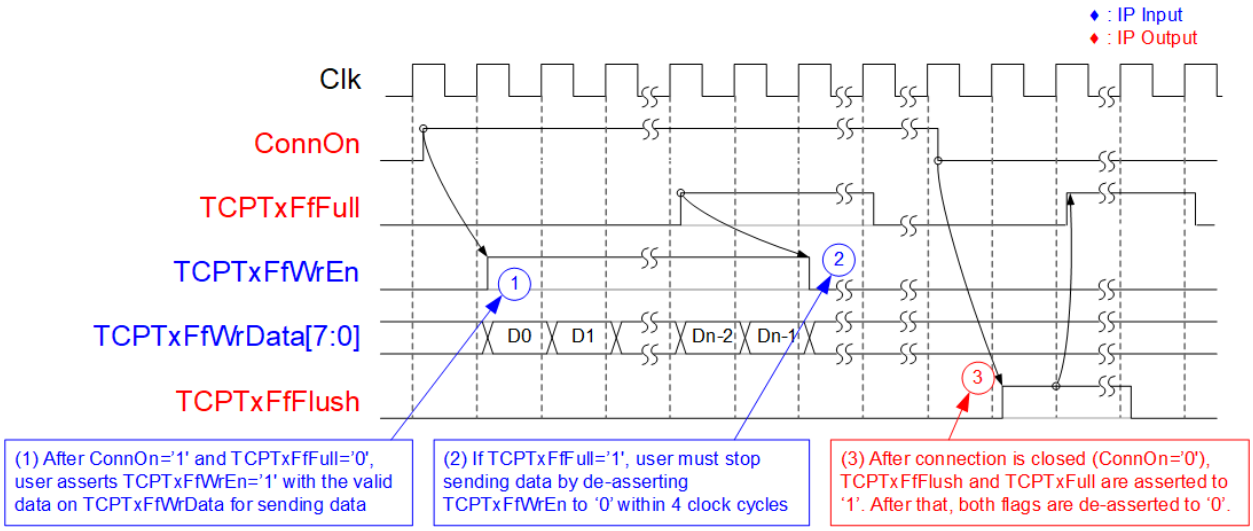

Figure 7: Tx FIFO interface timing diagram

(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 TOE1G 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.

In normal case, TCPTxFfFull flag can be negated from '1' to '0' when RST register changes from '1' to '0' to start IP initialization. During reset process, full flag is asserted to '1' to block data input from user. TCPTxFull is de-asserted following RST register.

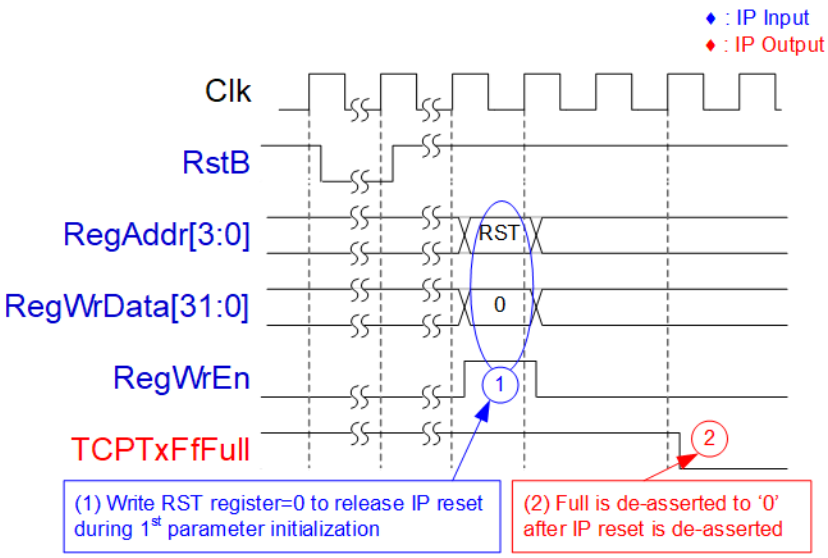

Figure 8: TCPTxFfFull de-asserted from IP reset

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.

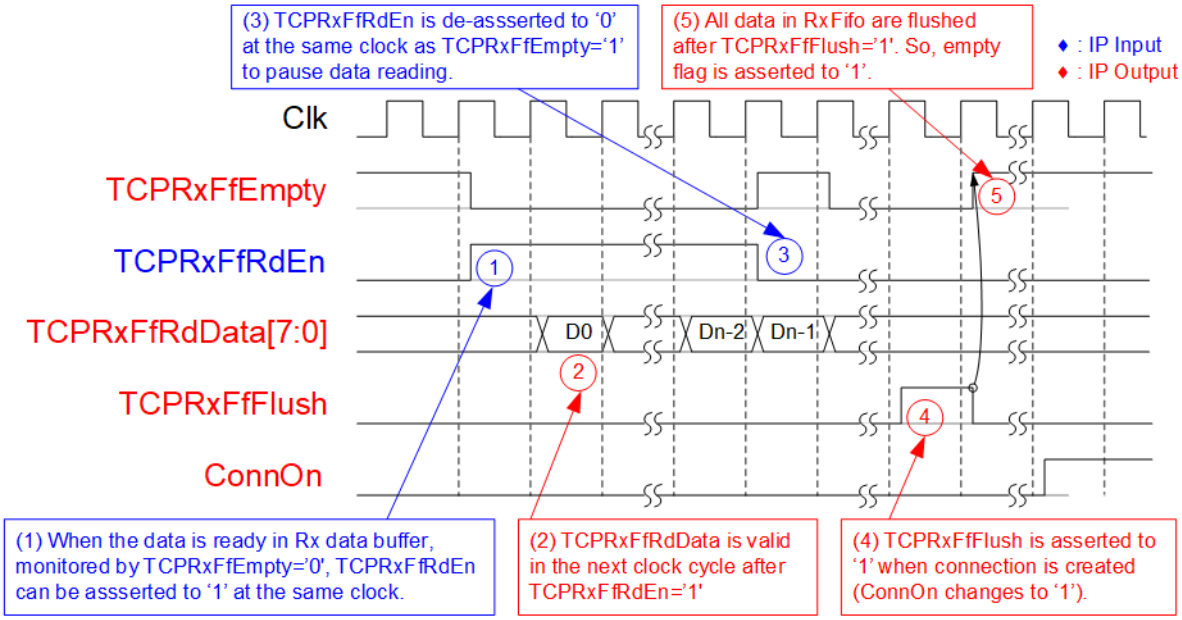

Figure 9: Rx FIFO interface timing diagram by Empty flag

(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'.

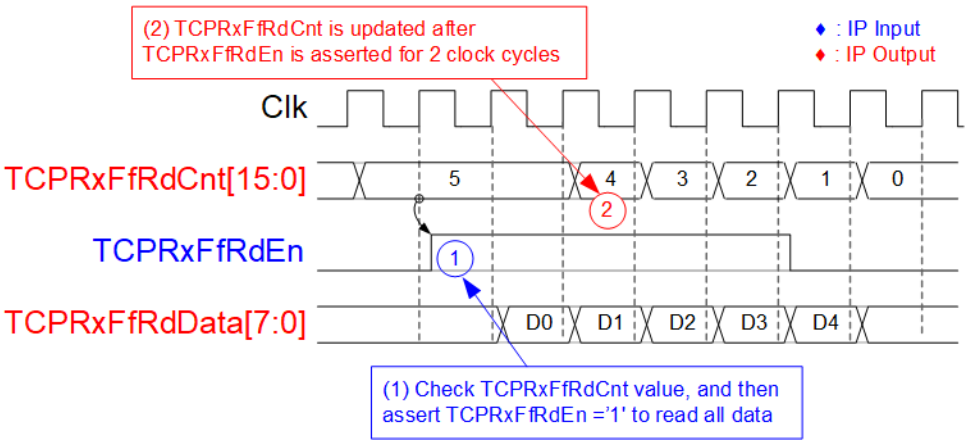

Figure 10: Rx FIFO interface timing diagram by using read counter

If user logic reads data as burst mode, TOE1G IP has read counter signal to show the total data stored in Rx FIFO interface as byte unit. For example, in Figure 10 there are 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 from read enable (TCPRxFfRdEn) to read counter (TCPRxFfRdCnt) is 2 clock cycles.

EMAC Interface

For EMAC interface, timing diagram is compatible to Xilinx TEMAC IP core. Figure 11 shows the example to transmit one packet from TOE1G IP to EMAC.

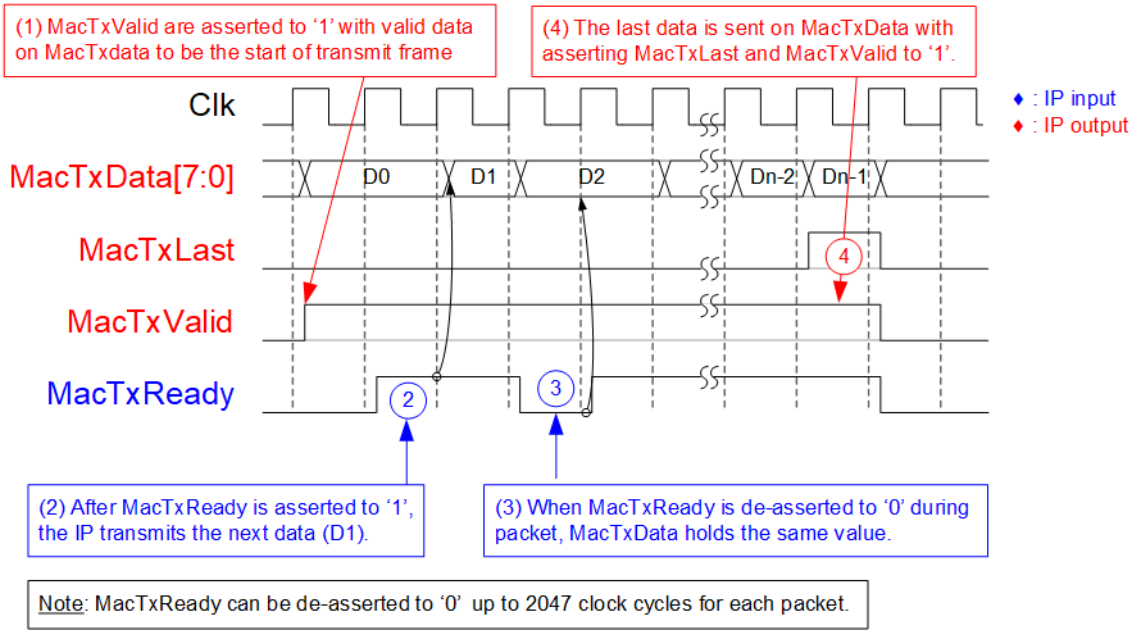

Figure 11: Transmit EMAC Interface

(1) TOE1G IP asserts MacTxValid with the first data of the packet on MacTxData. All signals are latched until MacTxReady is asserted to '1' to send the acknowledgement for the 1st data.

(2) TOE1G IP sends the 2nd data after MacTxReady is asserted to '1'.

(3) From Xilinx EMAC behavior, MacTxReady may be de-asserted to '0' during packet transmission. MacTxData holds the same value until MacTxReady is re-asserted to '1'.

Note: EMacFIFO inside TOE1G IP is designed to store Tx data stream from Transmit Block when EMAC is not ready. As EMacFIFO depth is 2047, TOE1G IP supports MacTxReady de-asserted to '0' less than 2047 cycles for one packet transmission.

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

For Receive 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. Timing diagram of Receive EMAC interface has two formats. First is timing diagram when connecting with Xilinx EMAC IP, as shown in Figure 12. Second is timing diagram when connecting with other modules by using AXI4-ST interface, as shown in Figure 13.

TOE1G-IP needs to run post processing after finishing each received packet, so it needs to have 3-clock cycle gap size between each packet transmission. Xilinx EMAC IP always de-asserts valid signal to pause data transmission in Receive EMAC interface for 4 clock cycles before asserting end-of-packet. So, MacRxReady, output from TOE1G IP, is always asserted to '1' when connecting with Xilinx EMAC IP.

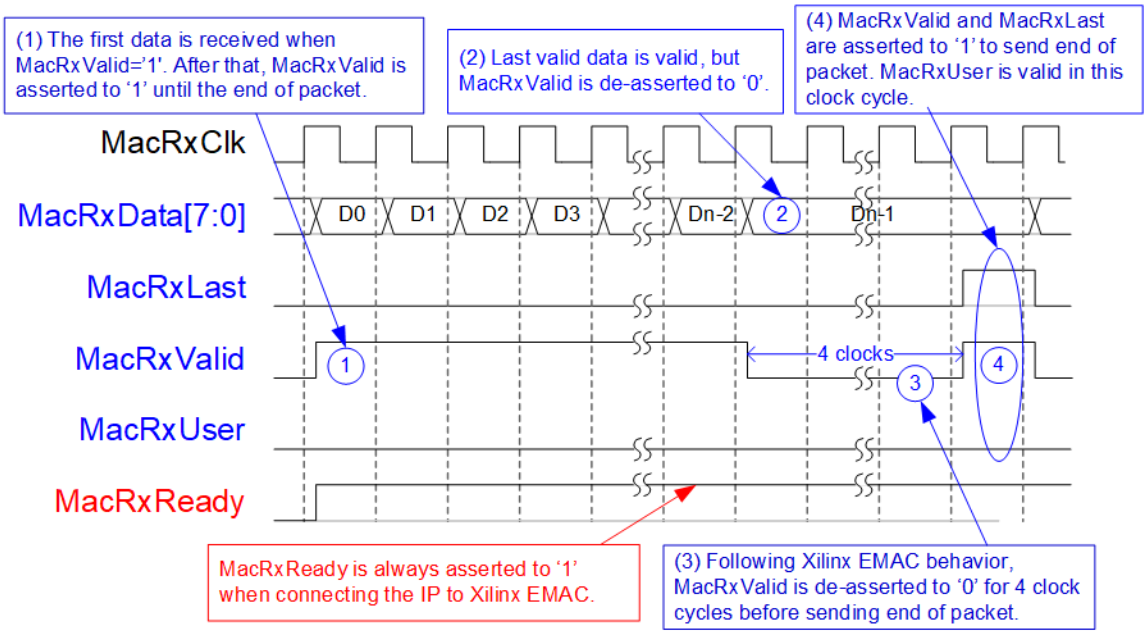

Figure 12: Receive EMAC Interface when connecting with Xilinx EMAC

(1) UDP1G IP detects the start of received frame when MacRxValid changes from '0' to '1'. At the same time, the 1st received data is read from MacRxData to start packet processing by TOE1G IP. MacRxReady is always asserted to '1' until end of the packet.
(2) The last data must be valid on MacRxData for TOE1G IP reading although the control signal to show the last data still not be transmitted from EMAC. TOE1G IP needs to read data of one packet continuously.
(3) EMAC de-asserts MacRxValid for 4 clock cycles to receive and validate 32-bit CRC of Ethernet packet.
(4) MacRxLast and MacRxValid are asserted to '1' to send the end of packet. TOE1G IP reads MacRxUser to validate if the packet has the good CRC status. The IP ignores the packet when MacRxUser is asserted.

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

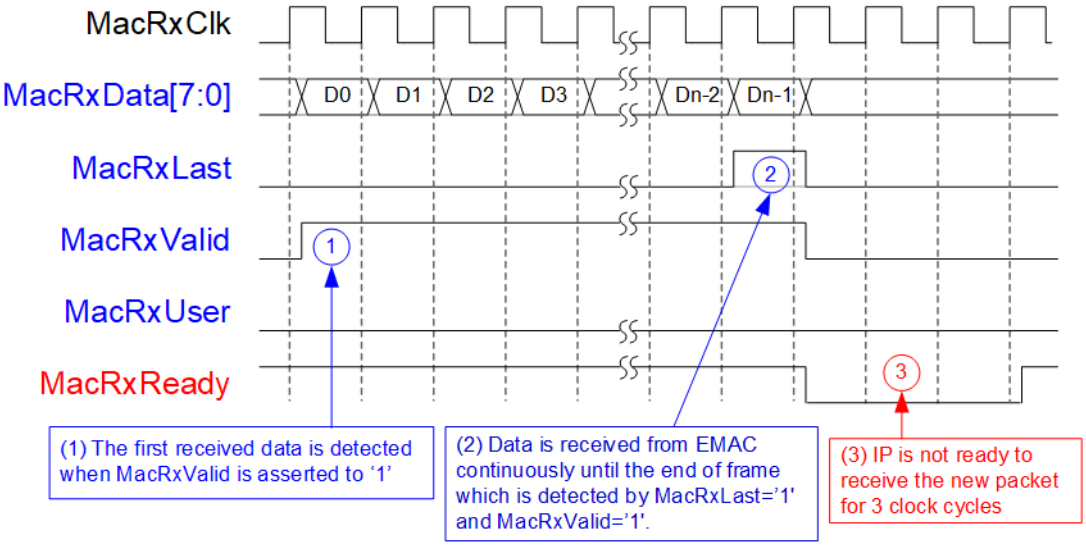

Figure 13: Receive EMAC Interface when connecting through other modules

(1) TOE1G IP detects start of the received frame when MacRxValid changes from '0' to '1'. In this time, the first data is valid on MacRxData. In this condition, MacRxReady is asserted to '1' to accept all data until the end of the packet. Also, MacRxValid must be asserted to '1' for transmitting the data of one packet continuously.

(2) The end of the packet is detected when MacRxLast='1' and MacRxValid='1'. 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]='0')

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 by TOE1G 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 TOE1G 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 TOE1G 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 TOE1G 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 TOE1G IP firstly).

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

Comparing to Client mode which MAC address is decoded from ARP reply packet after TOE1G 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

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
1.0	03-Dec-2012	New release
1.1	11-Dec-2012	Update IP port and buffer size description
1.2	19-Nov-2013	Update Figure1
1.3	28-Nov-2013	Add AC701 Support
1.4	13-Mar-2014	Update port name and add more description for Transmit operation
1.5	24-Apr-2014	Add VC707 Support
2.0	07-Aug-2014	Update IP to support full-duplex
2.1	20-Nov-2014	Add TCPTxFfFull condition
2.2	19-Jan-2015	Support Zynq device and add PSH register
2.3	24-Dec-2015	Add Busy status and register readback
2.4	2-Sep-2016	IP core renamed from TOE2-IP to TOE1G-IP
2.5	14-Nov-2016	Add Zynq Mini-ITX support and add WIN register
2.6	11-May-2017	Add EMACIF block to support new version of Xilinx EMAC
2.7	1-Aug-2018	Add SRV register and MacRxReady
2.8	30-Jul-2020	Add PSH[2] register and TMO description
2.9	2-Oct-2020	Update company info