

TOE10G IP loopback reference design

Rev1.0 13-Aug-19

1 Introduction

The loopback demo is designed to measure the data latency for both sending and receiving direction of DG TenGEMAC IP. Two sets of TOE10G-IP, TenGEMAC IP, and PCS/PMA IP are connected together to run as loopback, as shown in Figure 1-1.

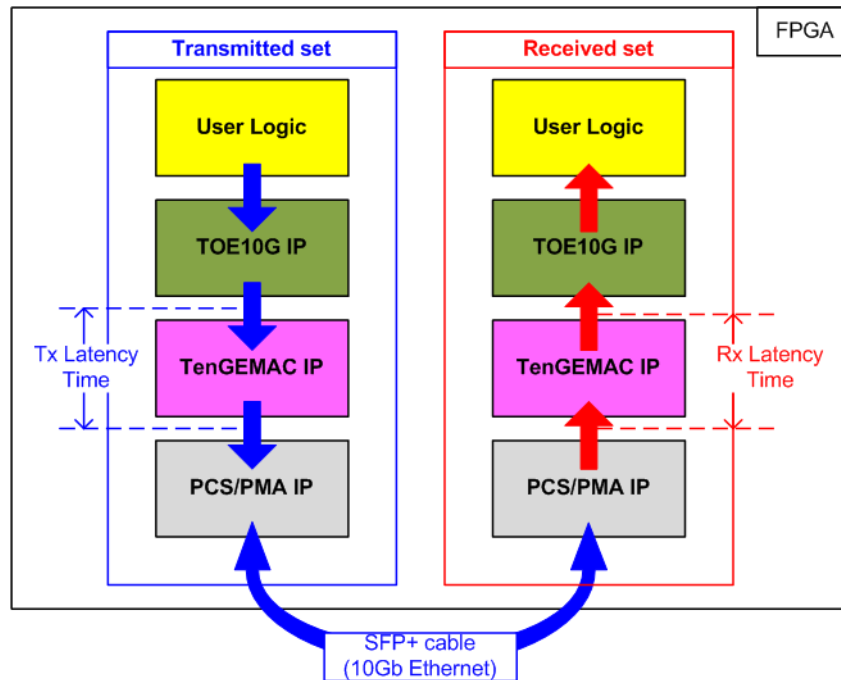


Figure 1-1 Loopback test system

The transmitted set in the left side transmits test data from the user logic to PHY while the received set in the right side receives and verifies test data by the user logic. The data from the transmitted set is forwarded to the received set by connecting the SFP+ cable between two SFP+ connectors to run as loopback cable.

Timer is designed within the hardware for precision measurement. Figure 1-1 uses two timers to measure the latency time in TX data path and Rx data path. After running the test, CPU reads the timer and displays the result on the console through JTAG UART. CPU and JTAG UART are applied to be the user interface for receiving the user input and displaying the test result.

More details of the demo are described as follows.

2 Hardware overview

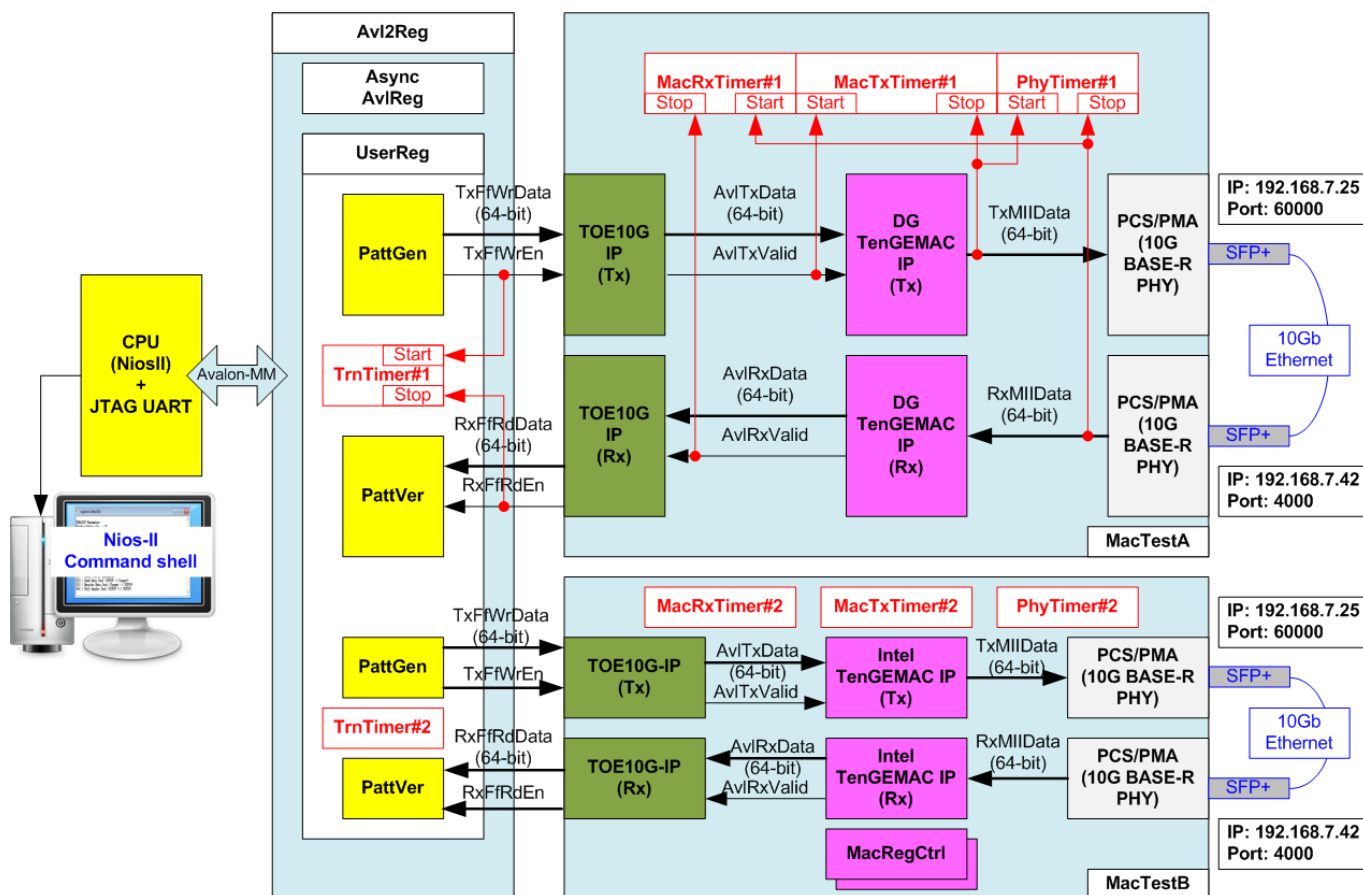


Figure 2-1 Demo Block Diagram

To measure the latency time in data path of EMAC IP system, the data loopback hardware is designed as shown in Figure 2-1. The test parameters are set by the user from JTAG UART. User can set three parameters, i.e. packet size, total size in one loop, and the number of loops to test. CPU sets all parameters from the user to UserReg module through Avalon-MM bus. UserReg is the main controller to control the run time of each sub module. The test data is generated by PattGen and sent to the TOE10G IP (Tx). The TOE10G IP encapsulates the packet to create TCP packet. The TCP packet is sent to TenGEMAC IP to create Ethernet packet. Finally, the Ethernet packet is forwarded to PCS/PMA block to send 10Gb Ethernet data to SFP+ connector. SFP+ cable is connected between two SFP+ connectors on FPGA board as a loopback cable. So, 10Gb Ethernet data are returned to the PCS/PMA block in the other channel.

Another set of TOE10G IP, TenGEMAC IP, and PCS/PMA is applied to decode 10Gb Ethernet data from the loopback cable to be the test data for verification in PattVer module. So, the operation of the receiver side is reversed from the sender side. 10Gb Ethernet data is decoded by TenGEMAC IP for creating Ethernet packet. TOE10G IP decodes the TCP data from the received Ethernet packet.

The demo is designed to compare the latency time of two TenGEMAC IPs, i.e. DG TenGEMAC IP and Intel TenGEMAC IP. So, two blocks of loopback test are included. The first one is MacTestA and another one is MacTestB. The latency time is measured by using four timers.

- (1) TrnTimer: Count total time usage for transferring data from PattGen to PattVer in each loop.
- (2) MacTxTimer: Count data latency time in Tx path of TenGEMAC IP.
- (3) PhyTimer: Count data latency time from the input of Tx PCS/PMA to the output of Rx PCS/PMA.
- (4) MacRxTimer: Count data latency time in Rx path of TenGEMAC IP.

After finishing the operation in each loop, the CPU reads the value from all timers to calculate the latency time of each data path. The user can run many loops to find total run time for long run test. More details of each module are described as follows.

2.1 10 Gb BASE-R PHY

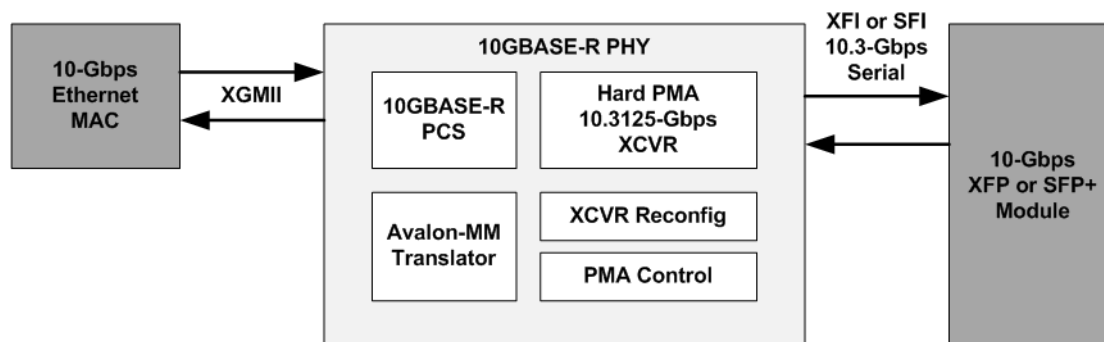


Figure 2-2 10G BASE-R PHY

10G BASE-R PHY is the Intel FPGA IP core to connect with XFP or SFP+ optical module for 10Gb Ethernet network solution. The user interface to connect with 10G Ethernet MAC is XGMII interface running at 156.25 Mbps. More details of 10G BASE-R PHY are described in the following link.

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

2.2 10G Ethernet MAC

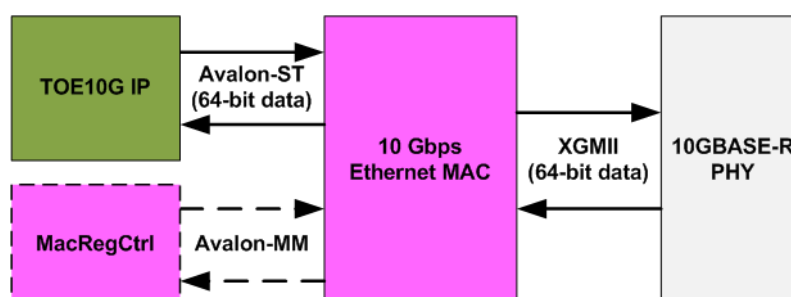


Figure 2-3 10G Ethernet MAC

Ethernet MAC implements the MAC layer of 10 Gb Ethernet. The user interface is 64-bit Avalon-ST interface. XGMII interface is applied to connect with 10 Gb BASE-R PHY. In the demo, two TenGEMAC IPs are connected to compare the latency time of data path.

The first one is TenGEMAC IP from Design Gateway which is described in more details from following link.

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

The second one is TenGEMAC IP from Intel FPGA which can read more details from following link.

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

TenGEMAC IP from Intel FPGA has Avalon-MM interface for register configuration while TenGEMAC IP from Design Gateway does not have this port. So, MacRegCtrl module to configure the register of TenGEMAC IP must be included when using Intel TenGEMAC IP.

2.3 MACRegCtrl

This module is designed to configure parameters of Intel TenGEMAC IP and monitor EMAC status through Avalon-MM bus. The logic is simply designed by using state machine. This module runs once after system powers up for initializing Ethernet MAC.

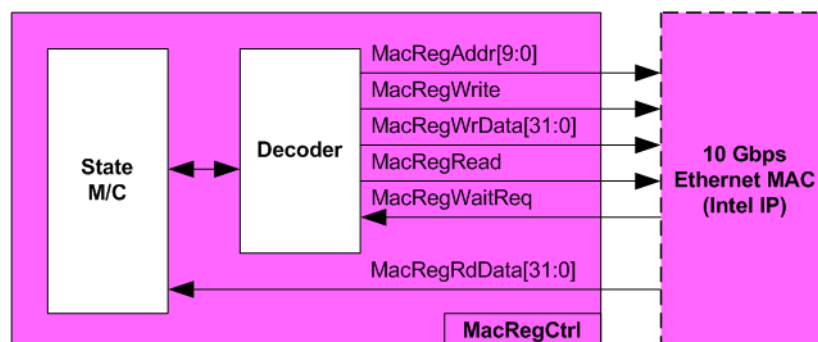


Figure 2-4 MACRegCtrl block diagram

The sequence of initialization sequence is below.

- 1) Disable transmit and receive path of EMAC.
- 2) Wait until transmit and receive path are idle.
- 3) Set receive module to remove CRC and padding.
- 4) Disable pause frame transmission.
- 5) Enable transmit and receive path of EMAC.

2.4 TOE10G IP

TOE10G IP implements TCP/IP stack and offload engine. User interface consists of control signals and data signals. Control and status signals are accessed through register interface. Data signals are accessed through FIFO interface. More details are described in datasheet.
https://dgway.com/products/IP/TOE10G-IP/dg_toe10gip_data_sheet_altera_en.pdf

2.5 Timer

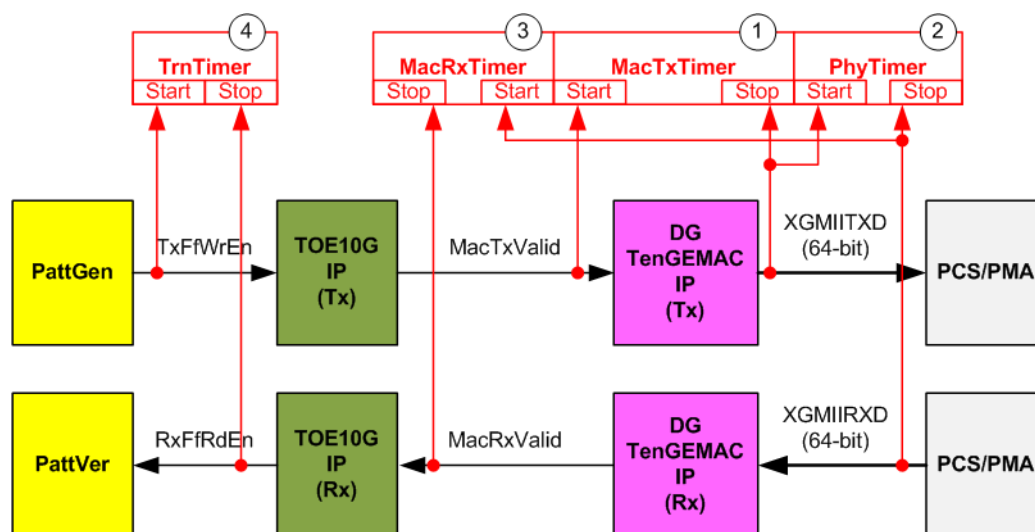


Figure 2-5 Timers in the reference design

There are four Timers in one test system to measure the latency time of EMAC, the latency time from Physical layer and hardware, and the total time usage for running one test loop.

All Timers are controlled by Start flag and Stop flag. The timer is enabled when Start flag = '1' and Stop flag = '0'. When Stop flag is asserted to '1', timer is stopped. CPU can read the latency time from each timer after all operations are completed. The timer, Start flag, and Stop flag are reset when the user starts the new test loop. More details of each timer are described as follows.

2.5.1 MacTxTimer

The timer is designed to measure latency time in Tx data path of TenGEMAC IP. The timer starts when detecting the 1st data on Tx Avalon-ST bus which is interface between TOE10G-IP and TenGEMAC IP. The timer stops when detecting the 1st data on Tx XGMII interface which is the interface with PCS/PMA block. The 1st data on Tx XGMII interface is always available in the next clock after the preamble and SFD code, so the logic compares the data with the preamble and SFD to stop the timer.

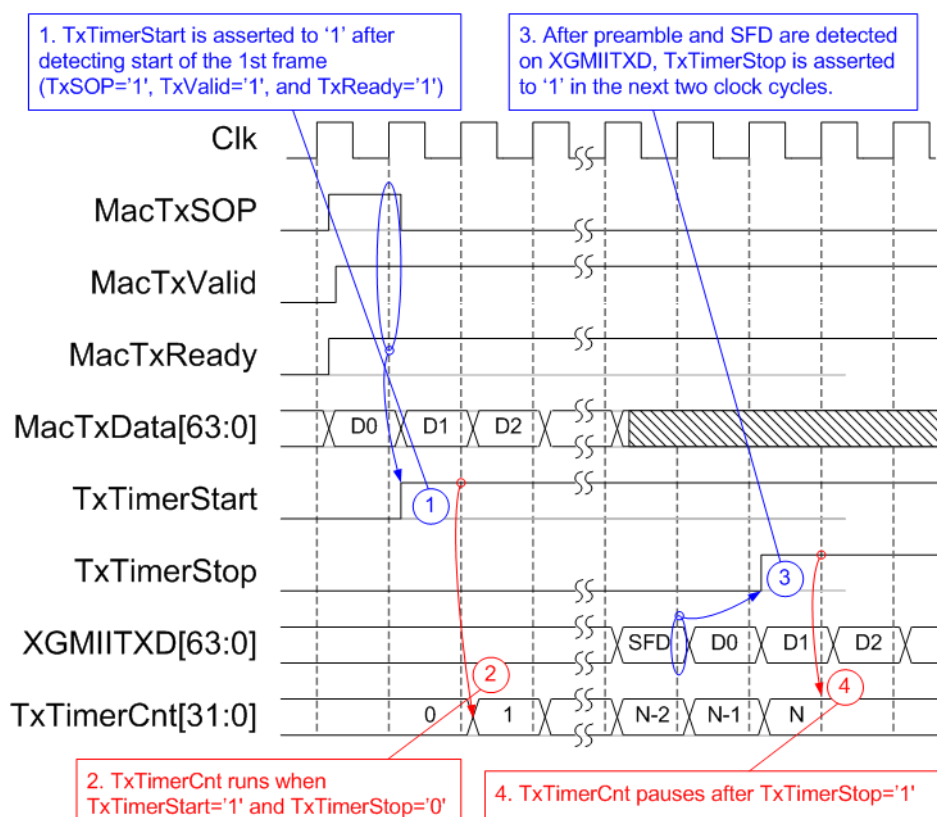


Figure 2-6 MacTxTimer timing diagram

Figure 2-6 shows the details to run MacTxTimer. The 1st data on Avalon-ST bus is detected when MacTxSOP='1', MacTxValid='1', and MacTxReady='1'. The 1st data on XGMII interface is detected by comparing the data with 7-byte preamble and SFD code. TxTimer is run when Start='1' and Stop='0'. After Stop is asserted to '1', Timer stops running and the user can read the timer to check the latency time between the 1st data on Avalon-ST and the 1st data on XGMII interface.

2.5.2 PhyTimer

The timer is designed to measure data latency time from Physical layer and external hardware for loopback connection. The timer starts when detecting the 1st data on Tx XGMII interface. The timer stops when detecting the 1st data on Rx XGMII interface. The condition to start PhyTimer is the same as the condition to stop MacTxTimer. For Rx XGMII interface, two data formats of the 1st data could be received. First, 64-bit of the 1st data is aligned to 64-bit XGMII_RXD signal. Second, 32-bit of the 1st data, 3-byte preamble, and 1-byte SFD code are received at the same clock. In case of the 2nd condition, the data must be rearranged by EMAC to align 64-bit data bus, as shown in Figure 2-7.

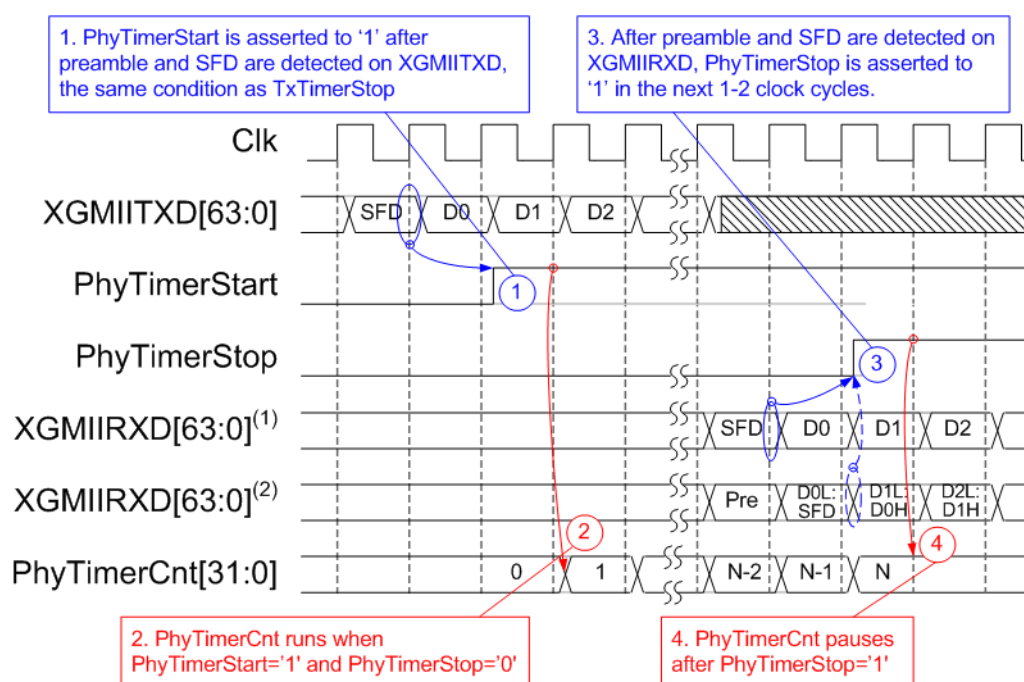


Figure 2-7 PhyTimer timing diagram

To assert Start flag of PhyTimer, it can use the same logic to assert Stop flag of TxTimer. To assert Stop flag of PhyTimer, the logic must detect by two conditions, following the received data format. PhyTimerStop is asserted to '1' after the 1st data is received which may be 64-bit or 32-bit. Similar to TxTimer, PhyTimer is run when PhyTimerStart='1' and PhyTimerStop='0'. The value is latched when Stop flag is asserted to '1'.

2.5.3 MacRxTimer

The timer is designed to measure data latency time in Rx path of TenGEMAC IP. The timer starts when the 1st data is detected on Rx XGMII interface. The timer stops when the 1st data is detected on Rx Avalon-ST interface. The condition to start MacRxTimer is the same as the condition to stop PhyTimer.

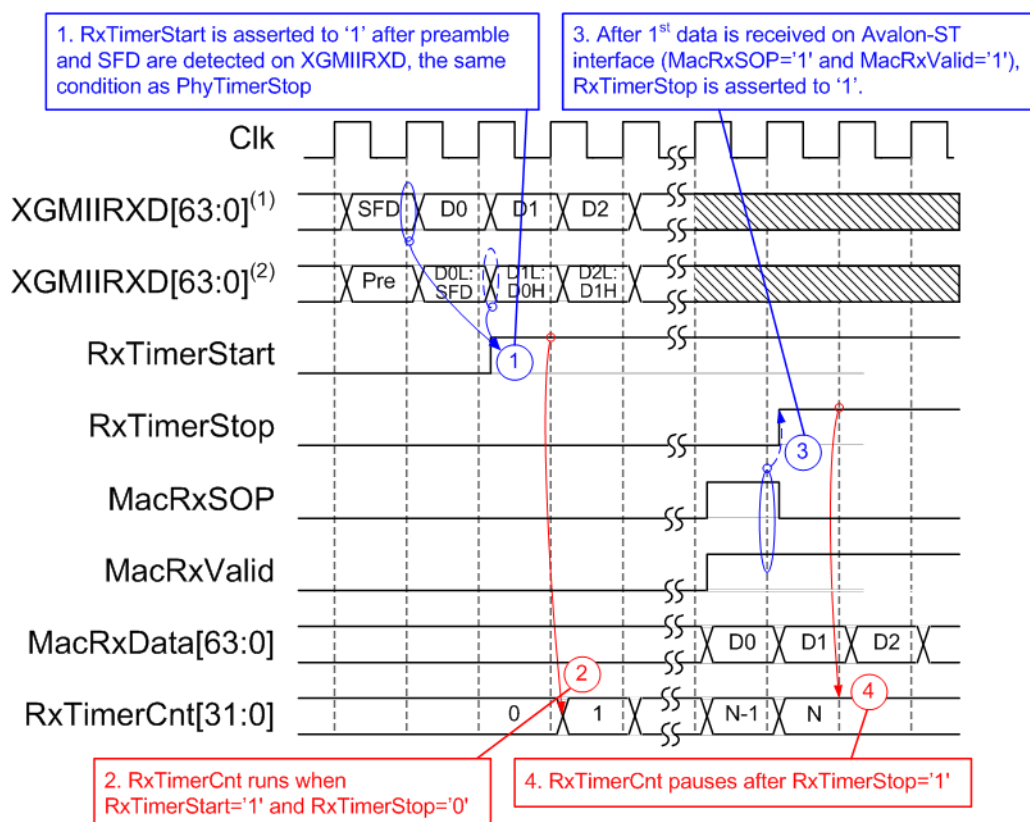


Figure 2-8 RxTimer timing diagram

To assert Start flag of RxTimer, the same logic to assert Stop flag of PhyTimer can be used. To assert Stop flag of RxTimer, the logic scans the 1st data valid on Rx Avalon-ST by detecting MacRxSOP='1' and MacRxValid='1'. Similar to other Timers, RxTimer runs when RxTimerStart='1' and RxTimerStop='0'. The value does not change when Stop flag is asserted to '1'.

2.5.4 TrnTimer

The timer is designed to measure total run time in the system. The timer starts when the 1st data is sent from user logic to TOE10G-IP. The timer stops when user logic receives the last data from TOE10G-IP.

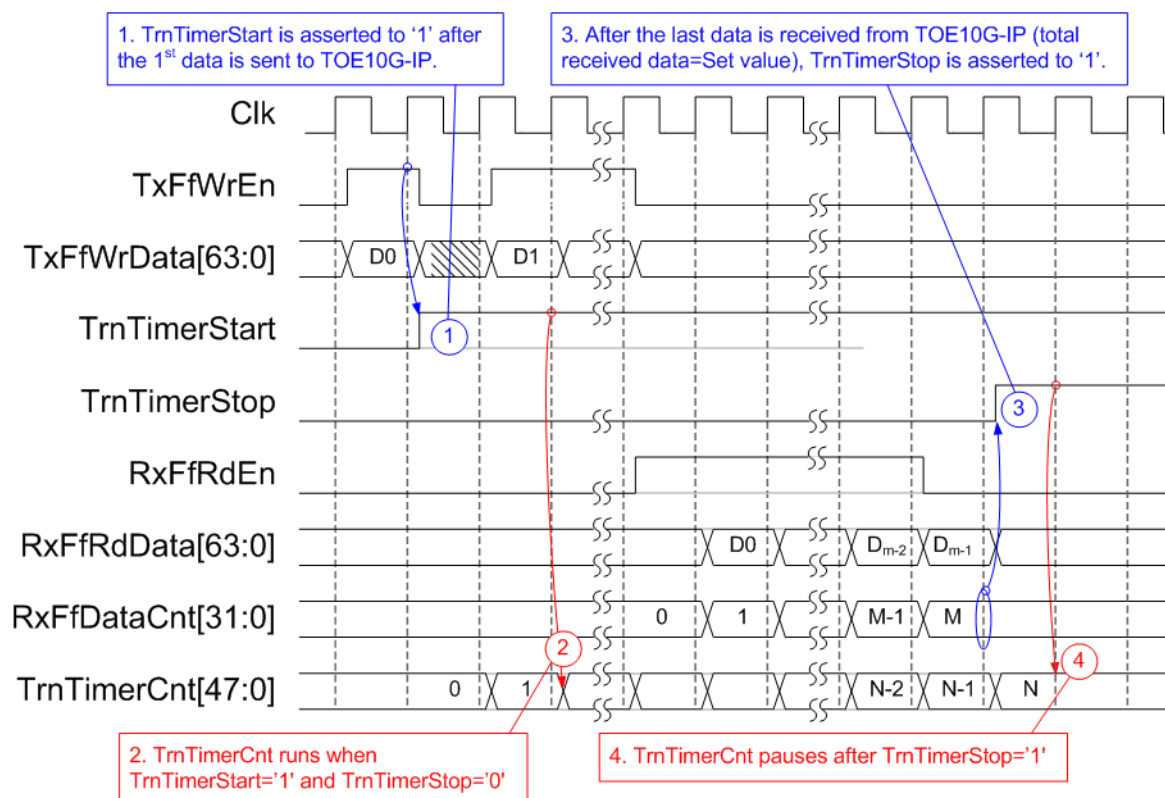


Figure 2-9 TrnTimer timing diagram

Assume that total data size is equal to M. Start flag is asserted to '1' after TxFfWrEn is asserted to '1' to send the 1st data (D0). After that, the 1st data is returned to the received path. The user reads the 1st data by asserting RxFfRdEn to '1'. When reading data from the FIFO, the data counter (RxFfDataCnt) increases following RxFfRdEn. Stop flag is asserted to '1' when the last data is read, monitored by checking RxFfDataCnt=Total transfer size. TrnTimer runs when TrnTimerStart='1' and TrnTimerStop='0'. The value does not change when Stop flag is asserted to '1'.

2.6 CPU and Peripherals

32-bit Avalon-MM is applied to be the bus interface for the CPU accessing the peripherals such as Timer and JTAG UART. To control and monitor the loopback system, the control and status signals are connected to register for CPU access as a peripheral through 32-bit Avalon-MM bus. CPU assigns the different base address and the address range to each peripheral for accessing the internal registers of the peripheral.

In the reference design, the CPU system is built with one additional peripheral to access the test logic. The base address and the range for accessing the test logic are defined in the CPU system. So, the hardware logic must be designed to support Avalon-MM bus standard for writing and reading the register. Avl2Reg module is designed to connect the CPU system as shown in Figure 2-10.

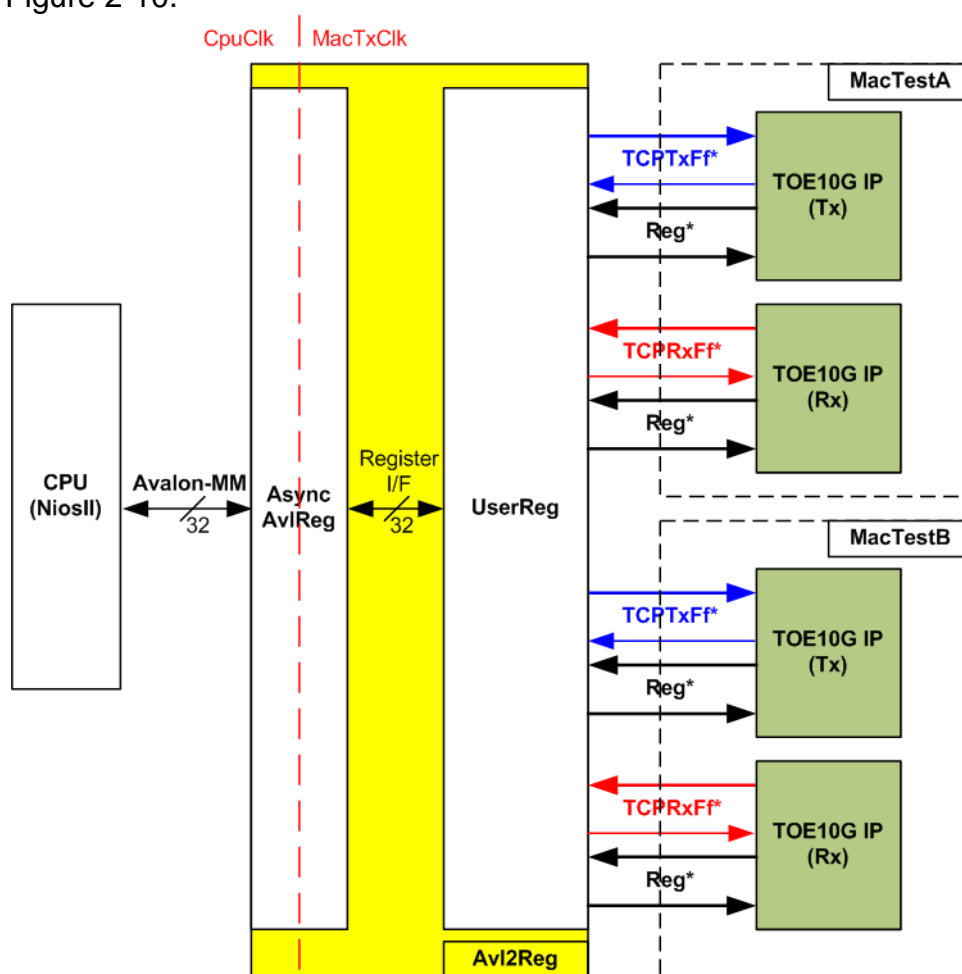


Figure 2-10 Avl2Reg block diagram

Avl2Reg consists of AsyncAvlReg and UserReg. AsyncAvlReg is designed to convert the Avalon-MM signals to be the simple register interface which has 32-bit data bus size (similar to Avalon-MM data bus size). Otherwise, AsyncAvlReg includes asynchronous logic to support clock crossing between CpuClk domain and MacTxClk domain.

UserReg includes the register file of the parameters and the status signals. Also, data interface and control interface of all TOE10G IP are connected to UserReg. More details of AsyncAvlReg and UserReg are described as follows.

2.6.1 AsyncAvlReg

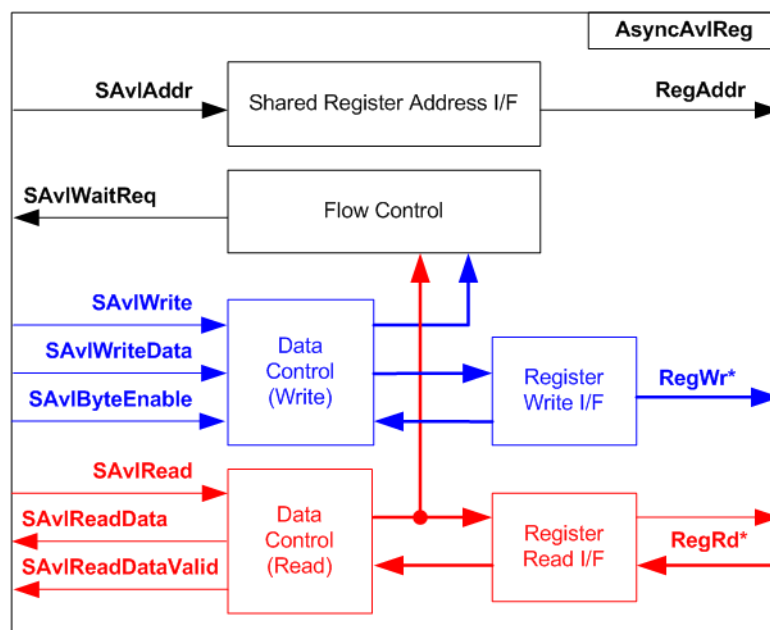


Figure 2-11 AsyncAvlReg Interface

The signals on Avalon-MM bus interface can be split into three groups, i.e. Write channel (blue color), Read channel (red color) and Shared control channel (black color). More details of Avalon-MM interface specification are described in following document.

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

According to Avalon-MM specification, only one command (write or read) can be operated at a time. The logics inside AsyncAvlReg are split into three groups, i.e. Write control logic, Read control logic, and Flow control logic. Flow control logic to control SAvlWaitReq is applied to hold the next request from Avalon-MM interface while the current request is operating. Write control I/F and Write data I/F of Avalon-MM bus are latched and transferred as Write register. On the other hand, Read control I/F and Read data I/F of Avalon-MM bus are latched and transferred as Read register. Address I/F of Avalon-MM is latched and transferred to Address register interface as well.

The simple register interface is designed to be compatible to general RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RdReq signal. The address of register interface is shared for write and read transaction. So, user cannot write and read the register at the same time. The timing diagram of the register interface is shown in Figure 2-12

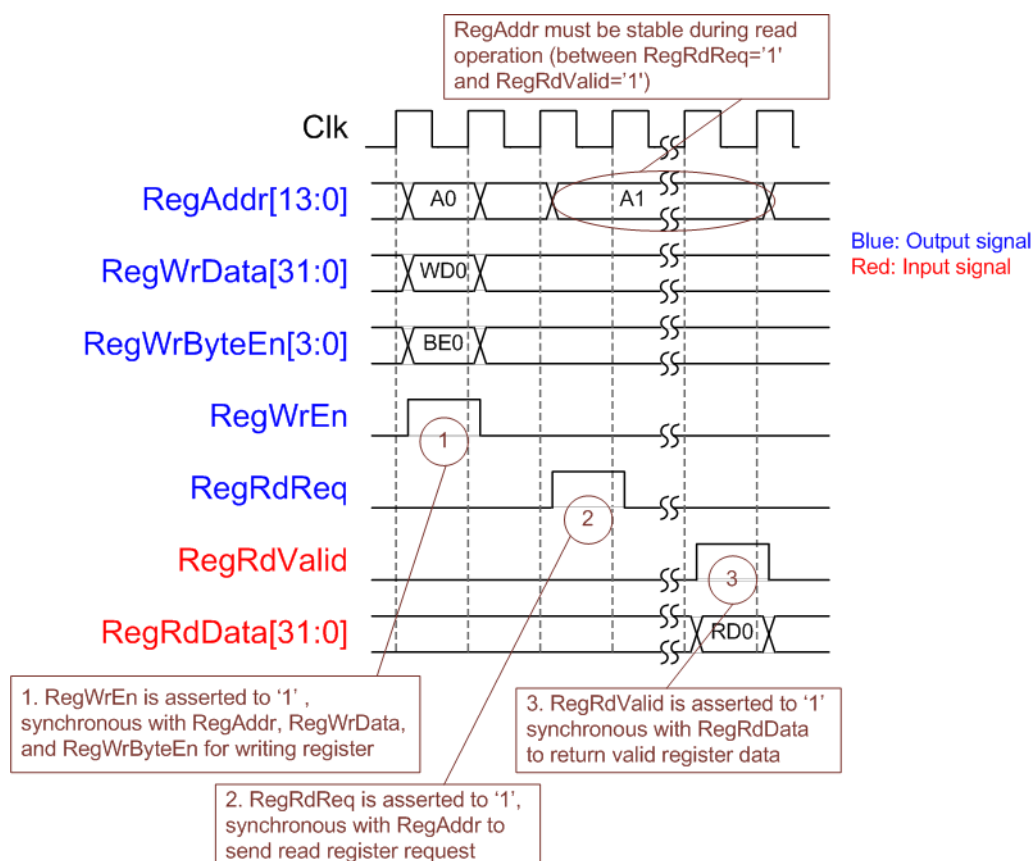


Figure 2-12 Register interface timing diagram

- 1) To write register, the timing diagram is the same as a general RAM interface. RegWrEn is asserted to '1' with the valid signal of RegAddr (Register address in 32-bit unit), RegWrData (write data of the register), and RegWrByteEn (the write byte enable). Byte enable has four bits to be the byte data valid, i.e. bit[0] for RegWrData[7:0], bit[1] for RegWrData[15:8], and so on.
- 2) To read register, AsyncAvlReg asserts RegRdReq to '1' with the valid value of RegAddr. 32-bit data must be returned after receiving the read request. The slave must monitor RegRdReq signal to start the read transaction.
- 3) The read data is returned on RegRdData bus by the slave with asserting RegRdValid to '1'. After that, AsyncAvlReg forwards the read value to SAvlRead interface.

2.6.2 UserReg

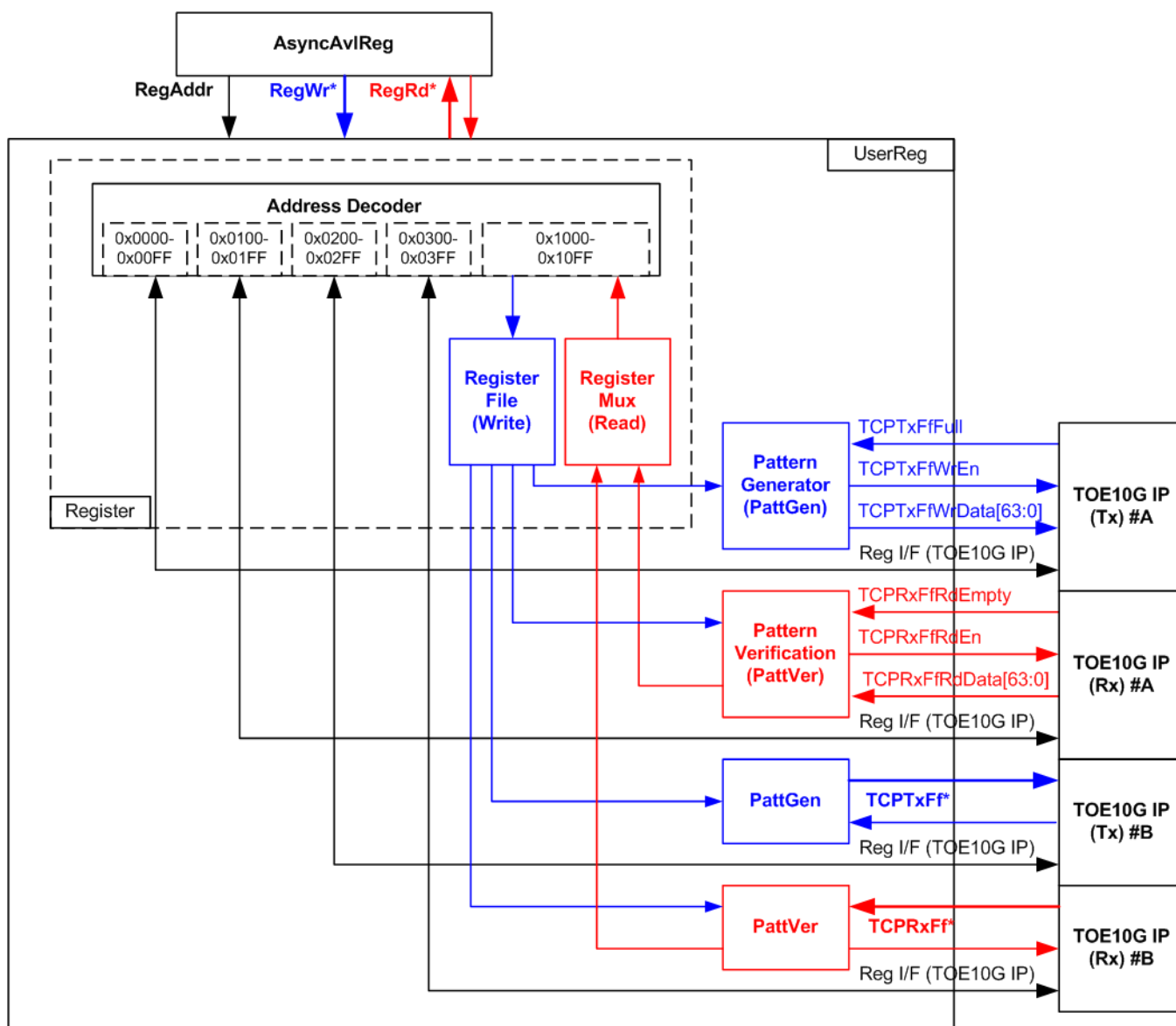


Figure 2-13 UserReg block diagram

The logic inside UserReg has three operations, i.e. Register, Pattern generator (PattGen), and Pattern verification (PattVer). Register block decodes the address requested from AsyncAvlReg and then selects the active register for write or read transaction. Pattern generator block is designed to send 64-bit test data to TOE10G IP following FIFO interface standard. Pattern verification block is designed to read and verify 64-bit data from TOE10G IP following FIFO interface standard. More details of each block are described as follows.

Register Block

The address range to map to UserReg is split into five areas.

0x0000 – 0x00FF: Register of TOE10G IP (Tx) #A

0x0100 – 0x01FF: Register of TOE10G IP (Rx) #A

0x0200 – 0x02FF: Register of TOE10G IP (Tx) #B

0x0300 – 0x03FF: Register of TOE10G IP (Rx) #B

0x1000 – 0x10FF: UserReg area

Address decoder decodes the upper bit of RegAddr for selecting the active hardware. The register file of UserReg is 32-bit bus size and Write byte enable (RegWrByteEn) is not used. To set the parameters in the hardware, the CPU must use 32-bit pointer to force 32-bit valid value of the write data.

To read register, two multiplexers are designed to select the read data within each address area. The lower bit of RegAddr is applied in each Register area. Next, the address decoder uses the upper bit to select the read data from each area for returning to CPU. Totally, the latency of read data is equal to two clock cycles, so RegRdValid is created by RegRdValid with asserting two D Flip-flops. More details of the address mapping within UserReg module is shown in Table 2-1

Table 2-1 Register map Definition

Address Wr/Rd	Register Name (Label in the tenglptest.c")	Description
BA+0x0000 – BA+0x00FF: TOE10G IP Register Area (TX) #A More details of each register are described in TOE10G IP datasheet.		
BA+0x0000	IP_RST_REG	Mapped to RST register within TOE10G IP
BA+0x0004	IP_CMD_REG	Mapped to CMD register within TOE10G IP
BA+0x0008	IP_SML_REG	Mapped to SML register within TOE10G IP
BA+0x000C	IP_SMH_REG	Mapped to SMH register within TOE10G IP
BA+0x0010	IP_DIP_REG	Mapped to DIP register within TOE10G IP
BA+0x0014	IP_SIP_REG	Mapped to SIP register within TOE10G IP
BA+0x0018	IP_DPN_REG	Mapped to DPN register within TOE10G IP
BA+0x001C	IP_SPN_REG	Mapped to SPN register within TOE10G IP
BA+0x0020	IP_TDL_REG	Mapped to TDL register within TOE10G IP
BA+0x0024	IP_TMO_REG	Mapped to TMO register within TOE10G IP
BA+0x0028	IP_PKL_REG	Mapped to PKL register within TOE10G IP
BA+0x002C	IP_PSH_REG	Mapped to PSH register within TOE10G IP
BA+0x0030	IP_WIN_REG	Mapped to WIN register within TOE10G IP
BA+0x0034	IP_ETL_REG	Mapped to ETL register within TOE10G IP
BA+0x0038	IP_SRV_REG	Mapped to SRV register within TOE10G IP
BA+0x0100 – BA+0x01FF: TOE10G IP Register Area (RX) #A This mapping address is the same as BA+0x0000 – BA+0x00FF but adding gap address by 0x0100		
BA+0x0200 – BA+0x02FF: TOE10G IP Register Area (TX) #B This mapping address is the same as BA+0x0000 – BA+0x00FF but adding gap address by 0x0200		
BA+0x0300 – BA+0x03FF: TOE10G IP Register Area (RX) #B This mapping address is the same as BA+0x0000 – BA+0x00FF but adding gap address by 0x0300		

Address Wr/Rd	Register Name (Label in the tenglptest.c")	Description
BA+0x1000 – BA+0x10FF: UserReg control/status		
BA+0x1000 Wr/Rd	User command (USER_CMD_REG)	<p>Wr</p> <p>[0] – Start Transmitting. Set '0' to start transmitting. [1] – Data Verification enable ('0': Disable data verification, '1': Enable data verification)</p> <p>Rd</p> <p>[0] – Tx Busy for MacTest#A ('0': Idle, '1': Tx module is busy) [1] – Data verification error for MacTest#A ('0': Normal, '1': Error) This bit is auto-cleared when user starts new operation or reset. [2] – Tx Busy for MacTest#B ('0': Idle, '1': Tx module is busy) [3] – Data verification error for MacTest#B ('0': Normal, '1': Error) This bit is auto-cleared when user starts new operation or reset. [4] – Mapped to ConnOn signal of TOE10G IP#A (TX) [5] – Mapped to ConnOn signal of TOE10G IP#A (RX) [6] – Mapped to ConnOn signal of TOE10G IP#B (TX) [7] – Mapped to ConnOn signal of TOE10G IP#B (RX)</p>
BA+0x1004 Wr/Rd	User reset (USER_RST_REG)	<p>Wr</p> <p>[0] – Reset signal. Set '1' to reset the logic. This bit is auto-cleared to '0'. [8] – Set '1' to clear TimerInt latched value for TOE10G IP#A (TX) [9] – Set '1' to clear TimerInt latched value for TOE10G IP#A (RX) [10] – Set '1' to clear TimerInt latched value for TOE10G IP#B (TX) [11] – Set '1' to clear TimerInt latched value for TOE10G IP #B (RX)</p> <p>Rd</p> <p>[8] – Latched value of TimerInt output from IP for TOE10G IP#A (TX) ('0': Normal, '1': TimerInt='1' is detected) This flag can be cleared by system reset condition or setting USER_RST_REG[8]='1'. [9] – Latched value of TimerInt output from IP for TOE10G IP#A (RX) ('0': Normal, '1': TimerInt='1' is detected) This flag can be cleared by system reset condition or setting USER_RST_REG[9]='1'. [10] – Latched value of TimerInt output from IP for TOE10G IP#B (TX) ('0': Normal, '1': TimerInt='1' is detected) This flag can be cleared by system reset condition or setting USER_RST_REG[10]='1'. [11] – Latched value of TimerInt output from IP for TOE10G IP#B (RX) ('0': Normal, '1': TimerInt='1' is detected) This flag can be cleared by system reset condition or setting USER_RST_REG[11]='1'. [16] – Ethernet Linkup status #A (TX) ('0': Link down, '1': Link up) [17] – Ethernet Linkup status #A (RX) ('0': Link down, '1': Link up) [18] – Ethernet Linkup status #B (TX) ('0': Link down, '1': Link up) [19] – Ethernet Linkup status #B (RX) ('0': Link down, '1': Link up)</p>

Address Wr/Rd	Register Name (Label in the tenglbacktest.c")	Description
BA+0x1020 Wr/Rd	Total transmit length #A (USER_TXLEN_A_REG)	Wr [31:0] – Total transmitted size in Qword unit (64-bit) for MacTest#A. Valid from 1-0xFFFFFFFF. Rd [31:0] – Current transmitted size in Qword unit (64-bit) for MacTest#A. The value is cleared to 0 when USER_CMD_REG is written by user.
BA+0x1024 Rd	FIFO status #A (USER_FFSTS_A_REG)	Rd [2:0]: Mapped to TCPRxFfLastRdCnt signal of TOE10G IP#A (RX) [15:3]: Mapped to TCPRxFfRdCnt signal of TOE10G IP#A (RX) [24]: Mapped to TCPTxFfFull signal of TOE10G IP#A (RX)
BA+0x1028 Rd	Total receive length #A (USER_RXLEN_A_REG)	Rd [31:0] – Current received size in Qword unit (64-bit) for MacTest#A. The value is cleared to 0 when USER_CMD_REG is written by user.
BA+0x102C Rd	Transfer timer (low) #A (USER_TTIML_A_REG)	Rd [31:0] – Read value of TrnTimer[31:0] #A to check total time usage for transferring loopback data of MacTest#A.
BA+0x1030 Rd	Transfer timer (high) #A (USER_TTIMH_A_REG)	Rd [15:0] – Read value of TrnTimer[47:32] #A to check total time usage for transferring loopback data of MacTest#A.
BA+0x1034 Rd	MAC TX timer #A (USER_MTTIM_A_REG)	Rd [31:0] – Read value of MacTxTimer[31:0] #A to check latency time in Tx data path of TenGEMAC #A
BA+0x1038 Rd	MAC RX timer #A (USER_MRTIM_A_REG)	Rd [31:0] – Read value of MacRxTimer[31:0] #A to check latency time in Rx data path of TenGEMAC #A
BA+0x103C Rd	Physical timer #A (USER_PYTIM_A_REG)	Rd [31:0] – Read value of PhyTimer[31:0] #A to check latency time in Tx PCS/PMA #A, external hardware, and Rx PCS/PMA #A.
BA+0x1040 Wr/Rd	Total transmit length #B (USER_TXLEN_B_REG)	Wr [31:0] – Total transmitted size in Qword unit (64-bit) for MacTest#B. Valid from 1-0xFFFFFFFF. Rd [31:0] – Current transmitted size in Qword unit (64-bit) for MacTest#B. The value is cleared to 0 when USER_CMD_REG is written by user.
BA+0x1044 Rd	FIFO status #B (USER_FFSTS_B_REG)	Rd [2:0]: Mapped to TCPRxFfLastRdCnt signal of TOE10G IP#B (RX) [15:3]: Mapped to TCPRxFfRdCnt signal of TOE10G IP#B (RX) [24]: Mapped to TCPTxFfFull signal of TOE10G IP#B (RX)
BA+0x1048 Rd	Total receive length #B (USER_RXLEN_B_REG)	Rd [31:0] – Current received size in Qword unit (64-bit) for MacTest#B. The value is cleared to 0 when USER_CMD_REG is written by user.
BA+0x104C Rd	Transfer timer (low) #B (USER_TTIML_B_REG)	Rd [31:0] – Read value of TrnTimer[31:0] #B to check total time usage for transferring loopback data of MacTest#B.
BA+0x1050 Rd	Transfer timer (high) #B (USER_TTIMH_B_REG)	Rd [15:0] – Read value of TrnTimer[47:32] #B to check total time usage for transferring loopback data of MacTest#B.
BA+0x1054 Rd	MAC TX timer #B (USER_MTTIM_B_REG)	Rd [31:0] – Read value of MacTxTimer[31:0] #B to check latency time in Tx data path of TenGEMAC #B
BA+0x1058 Rd	MAC RX timer #B (USER_MRTIM_B_REG)	Rd [31:0] – Read value of MacRxTimer[31:0] #B to check latency time in Rx data path of TenGEMAC #B
BA+0x105C Rd	Physical timer #B (USER_TTIML_B_REG)	Rd [31:0] – Read value of PhyTimer[31:0] #B to check latency time in Tx PCS/PMA #B, external hardware, and Rx PCS/PMA #B.

Pattern Generator

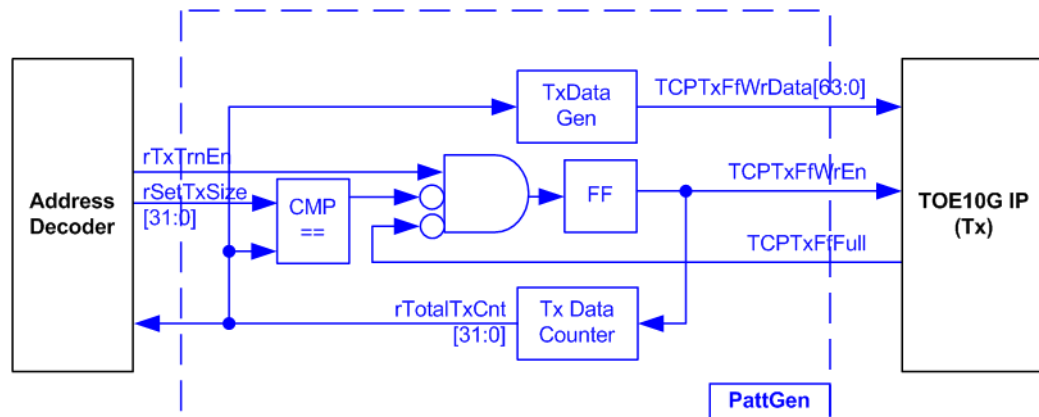


Figure 2-14 PattGen block

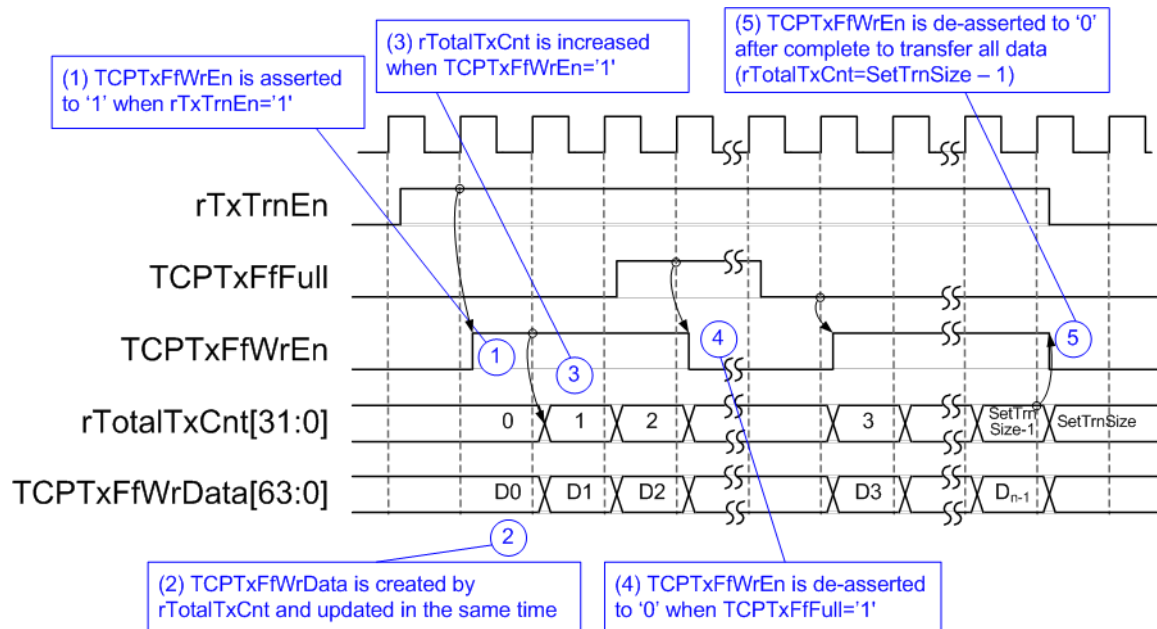


Figure 2-15 PattGen Timing diagram

PattGen is designed to generate test data to TOE10G IP. rTxTrnEn is asserted to '1' when USER_CMD_REG[0] is set to '0'. When rTxTrnEn is '1', TCPTxFfWrEn is controlled by TCPTxFfFull. TCPTxFfWrEn is de-asserted to '0' when TCPTxFfFull is '1'. rTotalTxCnt is the data counter to check total data sent to TOE10G IP. rTotalTxCnt is also used to generate 32-bit incremental data to TCPTxFfWrData signal. rTxTrnEn is de-asserted to '0' when finishing transferring total data (total data is set by rSetTxSize).

Pattern Verification

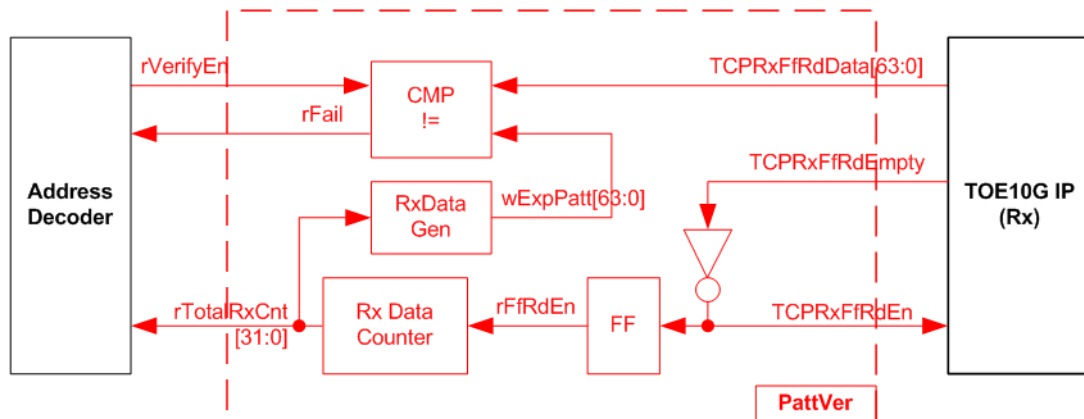


Figure 2-16 PattVer block

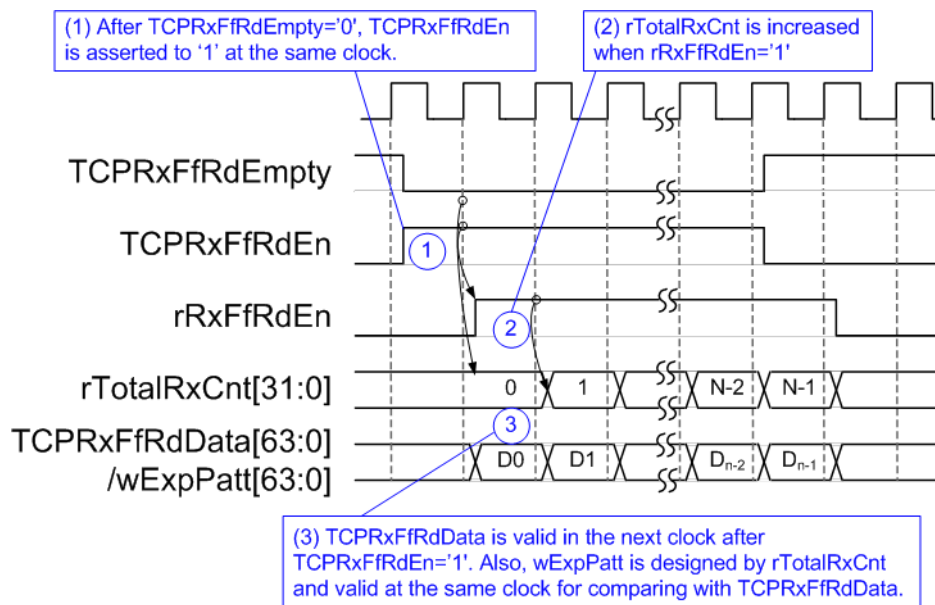


Figure 2-17 PattVer Timing diagram

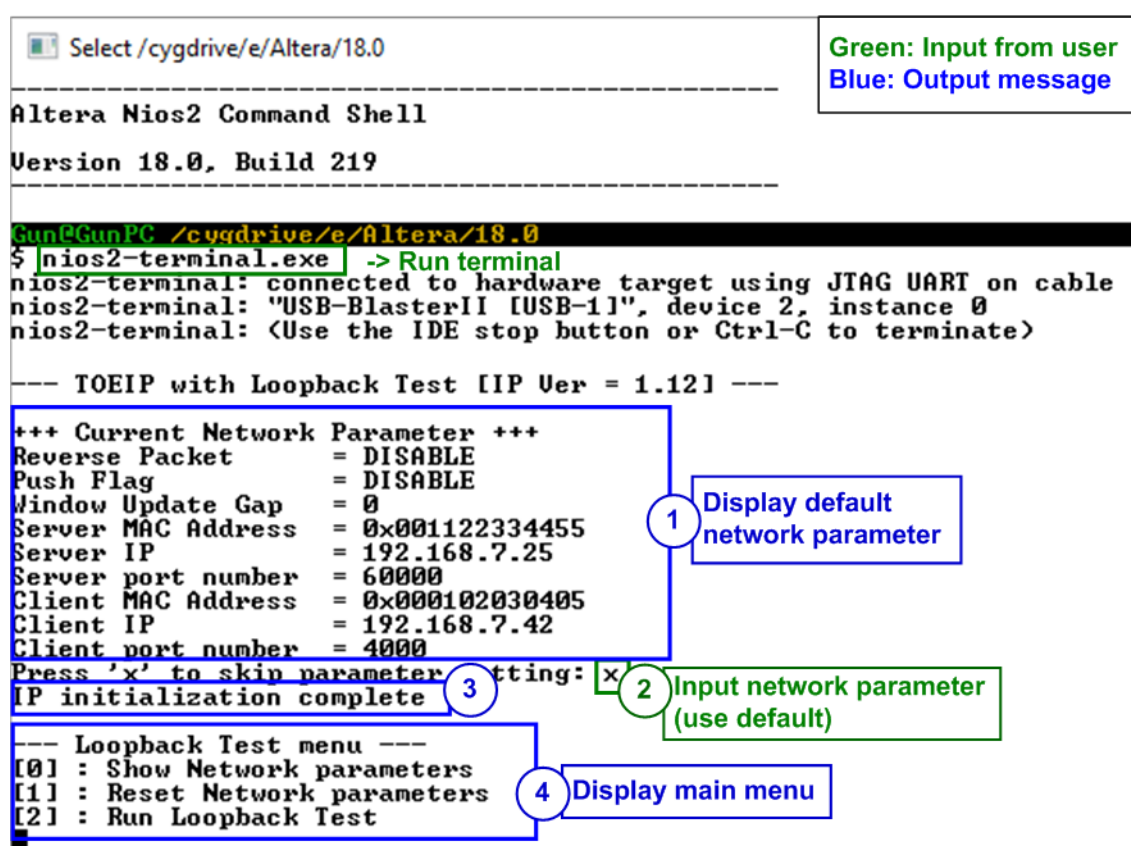
PattVer is designed to read test data from TOE10G IP with or without data verification, depending on rVerifyEn flag. When rVerifyEn is set to '1', data comparison is enabled to compare read data (TCPRxFfRdData) to the expected pattern (wExpPatt). If data verification is failed, rFail will be asserted to '1'. TCPRxFfRdEn is designed by using NOT logic of TCPRxFfRdEmpty. TCPRxFfRdData is valid for data comparison in the next clock. rRx FfRdEn which is one clock latency of TCPRxFfRdEn is applied to be counter enable of rTotalRxCnt to count total transfer size. rTotalRxCnt is used to generate wExpPatt.

3 CPU Firmware Sequence

After FPGA boot-up, 10G Ethernet link-up of four ports (USER_RST_REG[19:16]) is polling. The CPU waits until all ports are linked-up. Next, welcome message and default network parameter are displayed on the console.

User can select to change network parameter of client and server. The demo sets Tx TOE10G-IP to run as server while Rx TOE10G-IP is set to run as client mode. Two test environments (MacTest#A and MacTest#B) use the same network parameters.

After finishing parameter setting, all TOE10G-IPs start the initialization sequence. The client TOE10G IP sends ARP request to get the MAC address from the server TOE10G-IP. After that, server TOE10G IP returns ARP reply to complete initialization process. Figure 3-1 shows the example of the initialization sequence after system boot-up.



```

Select /cygdrive/e/Altera/18.0

Altera Nios2 Command Shell
Version 18.0, Build 219

Gun@GunPC /cygdrive/e/Altera/18.0
$ nios2-terminal.exe -> Run terminal
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-BlasterII [USB-1]", device 2, instance 0
nios2-terminal: <Use the IDE stop button or Ctrl-C to terminate>

--- TOEIP with Loopback Test [IP Ver = 1.12] ---

+++ Current Network Parameter +++
Reverse Packet      = DISABLE
Push Flag          = DISABLE
Window Update Gap   = 0
Server MAC Address  = 0x001122334455
Server IP           = 192.168.7.25
Server port number  = 60000
Client MAC Address  = 0x000102030405
Client IP           = 192.168.7.42
Client port number  = 4000
Press 'x' to skip parameter setting: x
IP initialization complete

--- Loopback Test menu ---
[0] : Show Network parameters
[1] : Reset Network parameters
[2] : Run Loopback Test
  
```

Green: Input from user
Blue: Output message

1 Display default network parameter

2 Input network parameter (use default)

3 IP initialization complete

4 Display main menu

Figure 3-1 Example of initialization sequence in loopback test on NiosII terminal

The initialization sequence in Figure 3-1 is shown as follows.

- 1) CPU waits until all Ethernet ports are linked up (USER_RST_REG[19:16]="1111"). The default parameters are displayed.
- 2) User inputs 'x' for using default parameters or inputs other keys to change some parameters. More details for changing some parameters are described in Reset IP menu (topic 3.2).
- 3) CPU waits until all TOE10G IPs finish the initialization sequence (IP_CMD_REG[0] TX/RX #A/#B ='0').
- 4) Main menu is displayed. There are three test operations for user selection. More details of each menu are described as follows.

3.1 Show parameters

This menu is used to show current parameters of TOE10G IP, server/client MAC address, server/client IP address, server/client port, and sending mode. The sequence of display parameters is as follows.

- 1) Read network parameter from each variable in firmware.
- 2) Print out each variable.

3.2 Reset IP

This menu is applied to change TOE10G IP parameters i.e. server/client MAC address, server/client IP address, server/client port, and sending mode. After setting updated parameter to TOE10G IP register, the CPU resets all IPs to re-initialize by using new parameters. Finally, the CPU monitors busy flag until the initialization process is finished. The sequence to reset IP is as follows.

- 1) Display current parameter value to the console.
- 2) Receive new input parameters from user and check input value whether valid or not. If the input is invalid, the old value will be used instead.
- 3) Force reset to TOE10G IP by setting IP_RST_REG[0]='1'.
- 4) Set all parameters to TOE10G IP register such as IP_SML_REG and IP_DIP_REG.
Four IPs are set in following sequence.
 - a. Set Tx TOE10G-IP in MacTest#A by using server parameters.
 - b. Set Tx TOE10G-IP in MacTest#B by using server parameters.
 - c. Set Rx TOE10G-IP in MacTest#A by using client parameters.
 - d. Set Rx TOE10G-IP in MacTest#B by using client parameters.
- 5) De-assert IP reset by setting IP_RST_REG[0]='0'.
- 6) Clear PattGen and PattVer logic by sending reset to user logic (USER_RST_REG[0]='1').
- 7) Monitor IP busy flag (IP_CMD_REG[0]) of all channels until the flag changes to '0'. The initialization sequence of all IPs is finished.

3.3 Run Loopback test

The user inputs three parameters to start the loopback test, i.e. total transfer (byte) size, packet (byte) size, and the number of loops. The operation will be cancelled if some inputs are invalid. During the test, 32-bit incremental data is generated to be the test data for sending to Tx TOE10G IP. The verification module also generates 32-bit incremental data to verify the received data from Rx TOE10G IP. Data from Tx TOE10G IP is transferred to 10G EMAC (Tx), PCS/PMA (Tx), SFP+ transceiver, SFP+ DAC cable as loopback cable consequently. The data from loopback cable is returned to Rx modules, starting from SFP+ transceiver, PCS/PMA (Rx), 10G EMAC (Rx), and Rx TOE10G IP. For long run test, the number of loops to run the test is controlled by the CPU. CPU repeats the test loop until total run loop is equal to the set value. The sequence of this test is as follows.

- 1) Receive total transfer (byte) size, packet (byte) size, and the number of loops from the user. All inputs must be valid. The operation will be cancelled when some inputs are invalid.
- 2) Set total transfer size and packet size to IP_TDL_REG and IP_PKL_REG of Tx TOE10G IP #A and #B. Also, the CPU sets total transfer size to user logic (USER_TXLEN_A/B_REG).
- 3) Set open port command to IP_CMD_REG of Tx TOE10G IP#A and #B. After that, CPU waits until all connection status of TOE10G IP are ON.
- 4) Set write command to IP_CMD_REG of Tx TOE10G IP#A and #B.
- 5) Set reset flag to clear the user logic (USER_RST_REG[0]='1') and clear interrupt flag (USER_RST_REG[11:8]=0xF). After that, set user command register to start data pattern generator and enable data verification (USER_CMD_REG[1:0]=2). The data from Test pattern generator is transferred to Tx TOE10G IP #A and #B.
- 6) The CPU waits until the operation is finished. For Tx TOE10G IP, CPU checks complete status by monitoring IP busy flag (IP_CMD_REG[0]) de-asserting to '0'. For Rx TOE10G IP, CPU checks complete status by checking total received length from user logic (USER_RXLEN_A/B_REG=total transfer size). During running, CPU reads the current transfer size from USER_TX/RXLEN_A_REG and displays on the console every second.
- 7) Check data verification flag (USER_CMD_REG[1] and USER_CMD_REG[3]). If some errors are found, error message will be displayed.
- 8) Repeat operation from step 2) – 7) following the number of loops.
- 9) Set close port command to IP_CMD_REG of Tx TOE10G IP#A and #B. After that, CPU waits until all connection status of TOE10G IP are OFF.
- 10) CPU calculates total performance and shows test result on the console.

3.4 Function list in User application

This topic describes the function list to run TOE10G IP loopback test.

void exec_port(unsigned int port_ctl)	
Parameters	port_ctl: 1-Open port, 0-Close port
Return value	None
Description	Write IP_CMD_REG of Tx TOE10GIP to open or close connection. After that, call read_conon function to monitor connection status until it changes from ON to OFF or OFF to ON, depending on port_ctl mode.

void init_param(void)	
Parameters	None
Return value	None
Description	Initialize by setting network and configure parameters to register of all TOE10G IPs. This function calls init_ip to write TOE10GIP register by using network and configure parameters that are declared in global variable. The sequence for initialization in this function is following topic 3.2 in step 3) – 7)

void init_ip(unsigned int baddr, unsigned int srv_mode)	
Parameters	baddr: base address of TOE10G IP register srv_mode: Setting initialization mode (0: client, 1: server)
Return value	None
Description	Initialize TOE10G IP by writing TOE10G IP register which uses baddr as base address to select the IP. The list of network and configure parameters is stored in global variable. srv_mode is used to select server or client parameter set from global variable.

int input_param(void)	
Parameters	None
Return value	0: Valid input, -1: Invalid input
Description	Receive network and configure parameters from user, i.e. server/client MAC address, server/client IP address, server/client port number, sending mode, and windows update threshold. If the input is valid, the parameters will be updated. Otherwise, same values are used. After receiving all parameters, the current value of each parameter will be displayed.

int loopback_test(void)	
Parameters	None
Return value	0: The operation is successful -1: Receive invalid input or error is found
Description	Run Loopback test following description in topic 3.3

Unsigned int read_conon(void)	
Parameters	None
Return value	0: Some connections are OFF, 1: All connections are ON.
Description	Read value from USER_CMD_CONNON register. Return 0 when some connections are not ON.

void show_cursize(unsigned long long prevrd_trn)	
Parameters	prevrd_trn: total transfer data of all loops, except the latest loop, in 8-byte unit.
Return value	None
Description	Read current transfer size of MacTest#A from USER_TXLEN_A_REG and USER_RXLEN_A_REG. After that, calculate the sum of prevrd_trn and the size in the latest loop from USER_TXLEN_A_REG and USER_RXLEN_A_REG. Display the result in Byte, KByte, or MByte unit on the console.

void show_interrupt(void)	
Parameters	None
Return value	None
Description	Display descriptions of timer interrupt reading from IP_TMO_REG of all TOE10G IPs.

void show_param(void)	
Parameters	None
Return value	None
Description	Display the current value of the network and configure parameters setting to TOE10G IP, i.e. server/client MAC address, server/client IP address, server/client port number, sending mode, and windows update threshold.

void show_result(unsigned long long totrd_size_A, unsigned long long totrd_time_A, unsigned long long totrd_size_B, unsigned long long totrd_time_B)	
Parameters	totrd_size_A: Total data of all loops from MacTest#A in 8-byte unit. totrd_time_A: Total run time of MacTest#A in 6.4 ns unit. totrd_size_B: Total data of all loops from MacTest#B in 8-byte unit. totrd_time_A: Total run time of MacTest#B in 6.4 ns unit.
Return value	None
Description	Display total transfer size for all loops from the input parameters. After that, calculate total time usage and the performance in MB/s unit for displaying on the console. Also, the latency time from all timer is read, i.e. USER_MTTIM_A/B_REG, USER_MRTIM_A/B_REG, and USER_PYTIM_A/B_REG for displaying in ns unit.

void wait_ethlink(void)	
Parameters	None
Return value	None
Description	Read Ethernet link-up status of four ports (USER_RST_REG[19:16]) and wait until all ports are linked up (USER_RST_REG[19:16]="1111").

4 Revision History

Revision	Date	Description
1.0	13-Aug-19	Initial version release