LL10GEMAC-IP reference design

LL10GEMAC-IP reference design
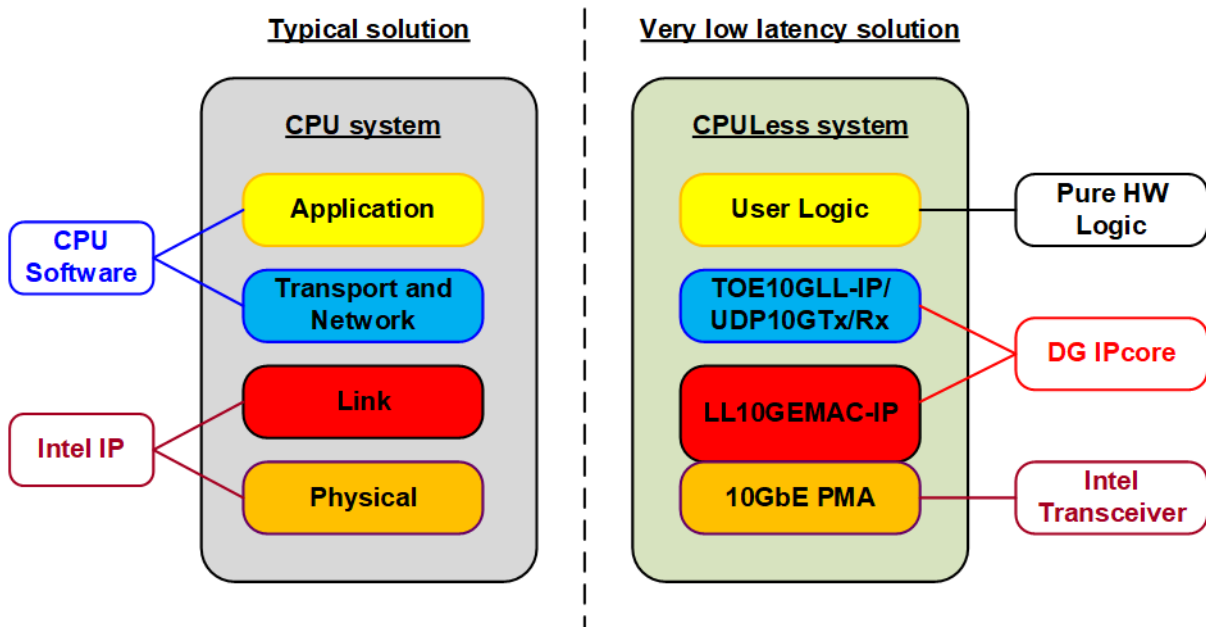
Rev1.0   30-Jun-23

1   Introduction

Figure 1-1 Low latency solution

The application layer, transport layer, and network layer of Ethernet system in FPGA are mostly implemented by the CPU software for system flexibility. The Link and Physical layer can be designed by using 10G EMAC and 10G BASE-R PHY which are Intel IP core. In some applications which are very sensitive about data latency such as HFT (High Frequency Trading), using CPU system shows too much latency time because of software-hardware handling process.

To achieve the lowest latency time, pure hardwired logic is purposed. As shown in the right side of Figure 1-1, the low-level protocol is designed by using LL10GEMAC-IP operating with 10GbE PMA. Moreover, the high-level protocols such as TCP/IP and UDP/IP can be implemented by pure-hardwired logic such as TOE10GLL-IP, UDP10GTx-IP, and UDP10GRx-IP. By using all hardwired logic solution, user can design simple logic for transferring the data via 10Gb Ethernet system with achieving very low latency time.

LL10GEMAC-IP consists of EMAC and PCS logic (the top part of Physical layer) while PMA logic (the low part of Physical layer) is implemented by using Intel Transceiver.
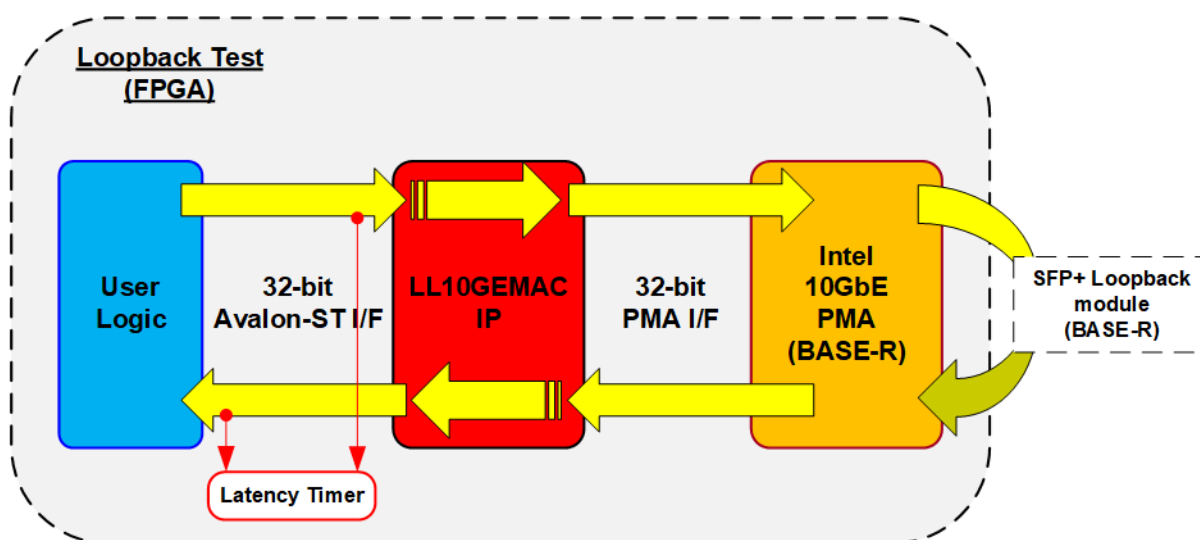
Figure 1-2 Loopback Test to check latency time

To check latency time of LL10GEMAC-IP, the simple test logic can be designed as shown in Figure 1-2. SFP+ loopback module can be inserted for transferring the packet from Tx interface to Rx interface. Also, the internal loopback inside the transceiver can be applied when SFP+ loopback module is not available.

To run the test, the user logic transfers the small packet and then verifies the received packet to confirm the connection stability. Latency time can be measured by designing the counter which starts counting at the Avalon-stream interface of LL10GEMAC-IP from the first data of Tx path to the first data of Rx path. Therefore, latency time from Intel Transceiver and SFP+ loopback module is included.

CPU system is included for user interface by using NiosII command shell. The user can start the test operation and see the result from the test on the console. More details of the demo are described as follows.
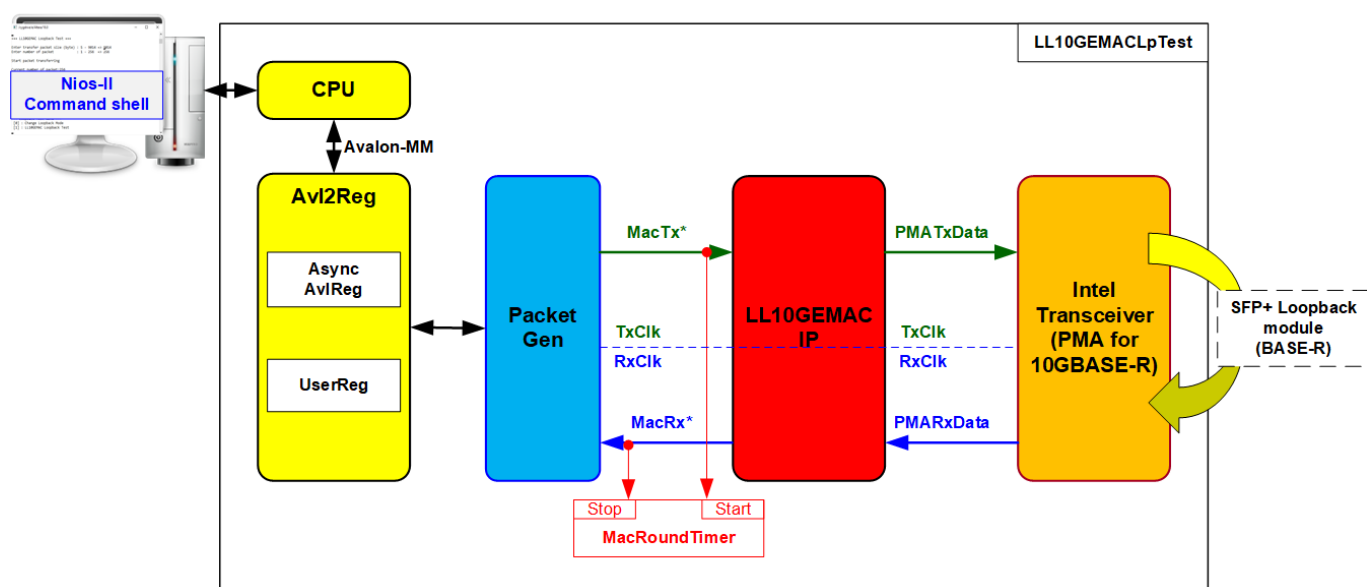
2   Hardware overview

Figure 2-1 Loopback Test Block Diagram

CPU system is included for easy user interface. The user sets the test parameters and checks the test result on the console which communicates by using JTAG UART. CPU uses Avalon-MM bus to interface with the hardware logic. Avl2Reg is the interface module to connect control and status signals of the hardware for CPU setting and monitoring.

The loopback system uses PacketGen which is the test logic to generate small Ethernet packet to LL10GEMAC-IP. The packet is transferred from LL10GEMAC-IP to Tx interface of Intel Transceiver. The data stream can be loopback via SFP+ loopback module or internal loopback inside the transceiver. After that, the packet is returned from the transceiver via Rx interface. LL10GEMAC-IP decodes the packet and returned to PacketGen for verifying the data. The received packet must be similar to the transmitted packet if the connection is stable.

The main objective of loopback test is to measure the latency time in LL10GEMAC-IP and Transceiver. MacRoundTimer is designed to capture round-trip latency time for transferring the packet from Tx interface of LL10GEMAC-IP to Rx interface of LL10GEMAC-IP, as shown in Figure 2-1. The user can use NiosII command shell to set the packet length and the number of packets which is generated by PacketGen. The number of packets is set to run the test many times to get many results, controlled by CPU firmware. After that, CPU calculates to find the minimum value, the maximum value, and the average value of round-trip time for displaying on the console. More details of each module are described as follows.

2.1   10GBASE-R PMA

Intel transceiver can be configured to be 10GBASE-R PMA. The user interface is 32-bit data running at 322.265625 MHz. In the demo, the user can run the demo by using internal loopback inside the transceiver or external loopback via SFP+ loopback module.

The configuration of 10GBASE-R PMA by using Transceiver Native PHY wizard is as follows.
- Transceiver configuration rules   : PCS Direct
- Data rate   : 10312.5 Mbps
- PCS Direct interface width   : 32
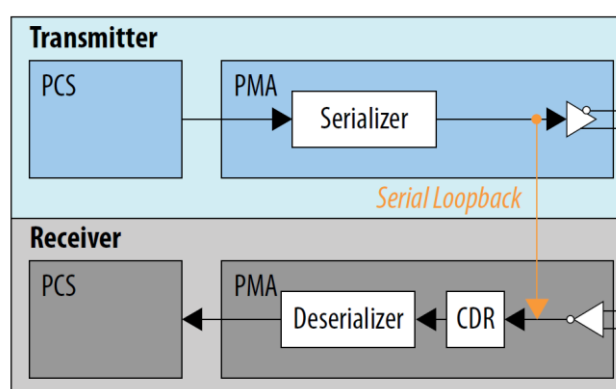- Enable rx_seriallpbken port (to run internal loopback mode)

Figure 2-2 Serial loopback mode in Intel transceiver

Figure 2-2 shows the definition for the internal loopback path from Transceiver PHY user guide which can be downloaded from following link.
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-10/ug_arria10_xcvr_phy.pdf

2.2   LL10GEMAC-IP

The IP core by DesignGateway implements low-latency EMAC and PCS logic for 10Gb Ethernet (BASE-R) standard. The user interface is 32-bit Avalon-stream bus. Please see more details from LL10GEMAC-IP datasheet on our website.
https://dgway.com/products/IP/Lowlatency-IP/dg_ll10gemacip_data_sheet_intel.pdf
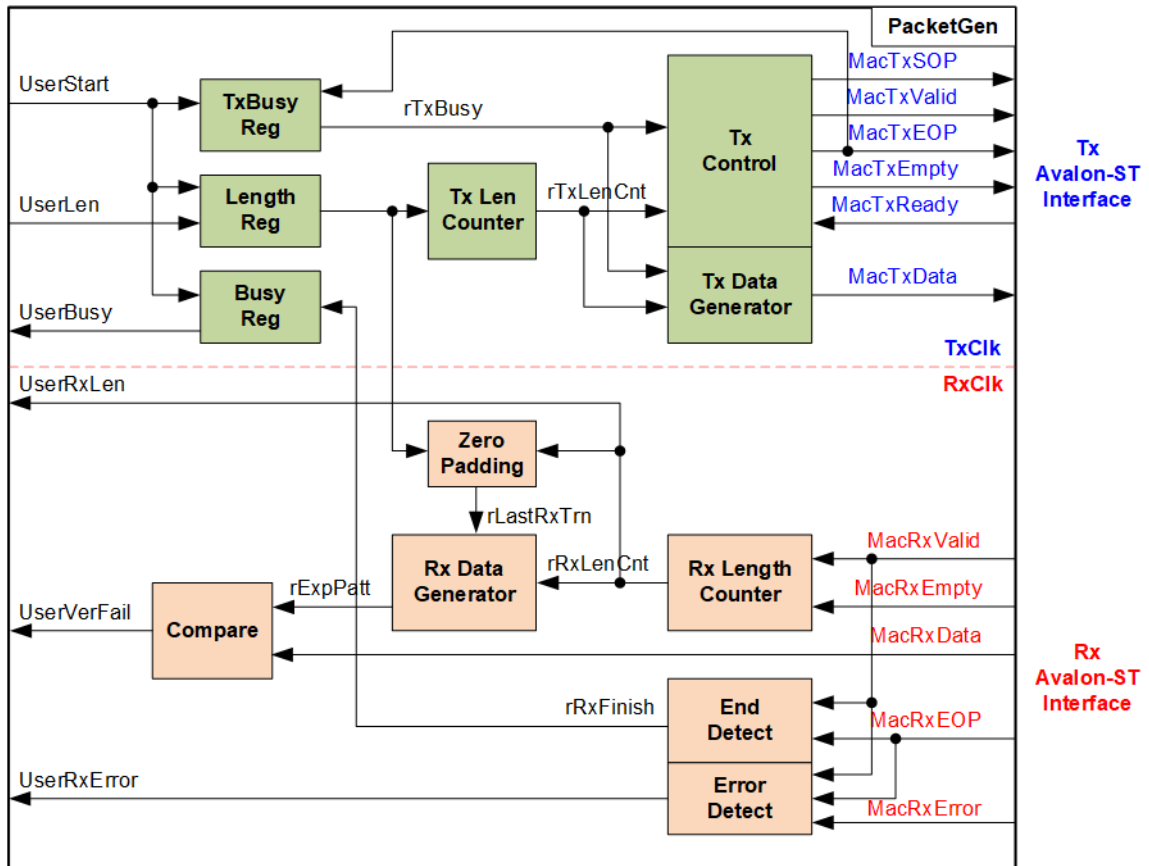
2.3 PacketGen

Figure 2-3 PacketGen module

PacketGen module is the test logic to send and receive one packet with LL10GEMAC-IP through Avalon-Stream interface. This module runs in two clock domains - TxClk for transmit operation and RxClk for receive operation. As shown in Figure 2-3, the logic is split to two groups, i.e., Packet generator in the top side for generating one packet to Avalon-ST and Packet verification in the bottom side for verifying one packet from Avalon-ST.

After receiving start pulse from the user (UserStart), this module generates one test packet which has packet length equal to packet size parameter (UserLen). The start signal of the packet (MacTxSOP) is asserted when the first data is transmitted. Test pattern is 16-bit incremental data, created by the transmit counter (rTxLenCnt). When packet size is not aligned to 32-bit, MacTxEmpty of the last data in the packet is varied from 00b to 11b, depending on the number of unused bytes. At the same time, MacTxEOP is asserted to '1' to finish the transmit operation. TxBusy is asserted to '1' after UserStart is asserted. While TxBusy changes from '1' to '0' after finishing transmit operation (MacTxEOP='1'). Tx Length Counter and Tx Data Generator are paused when MacTxReady is de-asserted to '0'. Transmitted data valid (MacTxValid) is always asserted to '1' to send the packet via Tx Avalon-ST until the end of the transmitted packet.

UserBusy is designed for user to monitor PacketGen operation. When User asserts UserStart, UserBusy is asserted to '1'. UserBusy is de-asserted to '0' after finishing the loopback operation by receiving end of the packet (MacRxEOP='1').

On the other hand, when the packet is loop-back to Rx Avalon-ST, the packet is verified. The received data valid (MacRxValid) is applied to count the numbers of received data (rRxLenCnt). The counter output can be fed to the pattern generator (Rx Data Generator) to create the expected pattern for data verification. Zero-padding module is run when the packet size is less than 60 bytes. After the current receive counter is equal to the packet size, set by user, rLastRxTrn will be asserted to '1' to fill the expected pattern with zero value until the 60-byte packet length is reached. Fail flag (UserVerFail) is asserted to '1' if the received data is not equal to the expected pattern. When the end of packet (MacRxEOP) is asserted to '1', Finish flag (rRxFinish) is asserted to '1' for de-asserting UserBusy to '0'. Finish flag is auto-cleared after UserBusy is de-asserted to '0'. When the end of packet is received, MacRxError is monitored. UserRxError is asserted to '1' when MacRxError is equal to '1'.
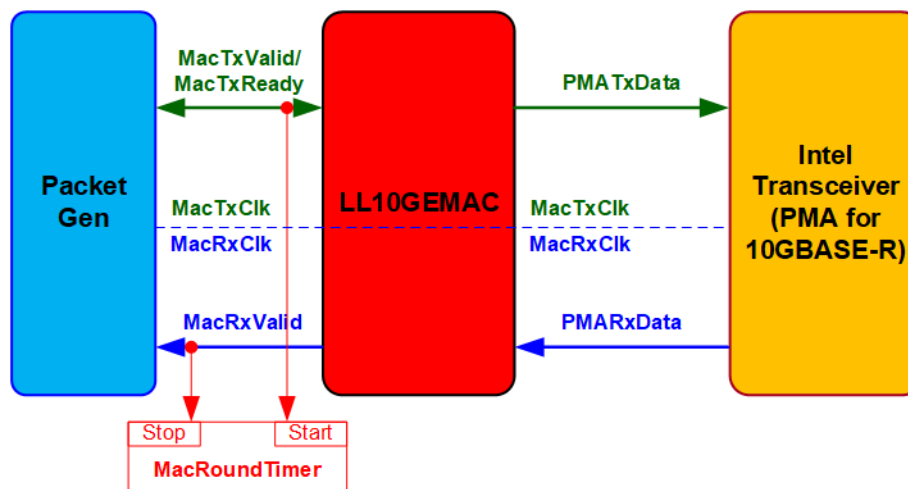
2.4   Timer

Figure 2-4 Timer in the reference design

A timer named MacRoundTimer in the test system is created to count the round-trip latency time measured from Tx data path LL10GEMAC-IP to Rx data path of LL10GEMAC-IP (Round-trip latency). Therefore, latency time is the sum of the latency inside LL10GEMAC and Intel transceiver. MacRoundTimer are controlled by Enable flag. In the test, one packet is transferred in the system. Enable flag is asserted to '1' when the first data is found at the input of the measured module. It is de-asserted to '0' when the first data is found at the output of the measured module. The timer latches the value to return to CPU after the test operation is finished. The timer and Enable flag are reset when the user starts the new test loop. More details of the timer are described as follows.
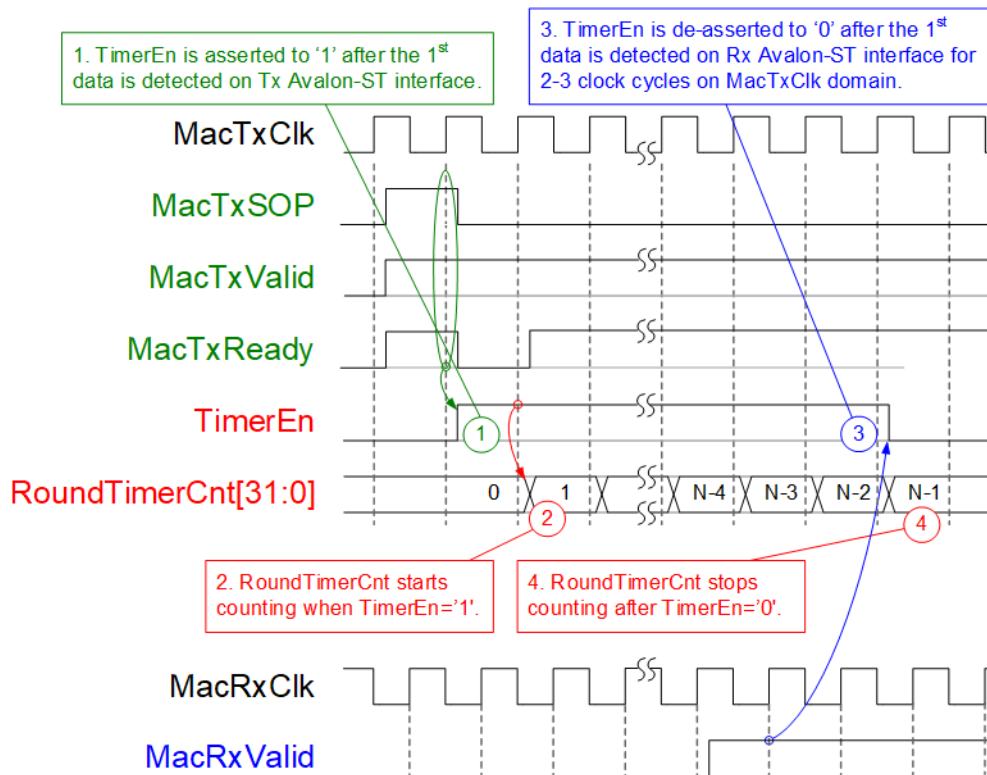
Figure 2-5 MacRoundTimer timing diagram

(1) The first data on Tx Avalon-ST interface is detected when MacTxValid='1', MacTxSOP='1', and MacTxReady='1'. After that, TimerEn is asserted to '1' to start the timer operation.

(2) When TimerEn='1', the timer is incremented every clock cycle to count the latency time.

(3) TimerEn is de-asserted to '0' when the first data on Rx Avalon-ST interface is detected, monitored by MacRxValid='1'. This condition is run in MacRxClk domain but MacRoundTimer is run in MacTxClk domain. Therefore, asynchronous logic is added to forward Stop flag from MacRxClk to MacTxClk. Asynchronous logic is designed by adding two Flip-Flops on MacTxClk domain. Therefore, the latency time measured by the timer is increased about 2-3 clock cycles, depending on the phase shift from MacRxClk to MacTxClk. In the demo system, the timer value is subtracted by two in CPU firmware to remove latency time from asynchronous logic by using minimum value.

(4) Timer stops running and holds the value. The user can read the timer to check round-trip latency time.

2.5 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 test 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 one peripheral at a time.

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 supporting CPU writing and reading. Avl2Reg module is designed to connect the CPU system as shown in Figure 2-6.
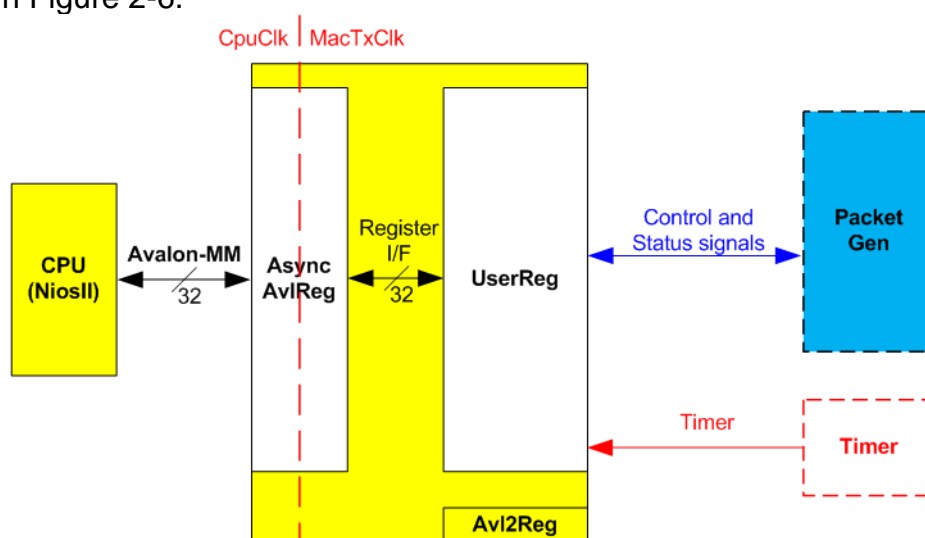
Figure 2-6 CPU and peripherals hardware

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). Also, 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 to control and monitor PacketGen and the Timer. More details of AsyncAvlReg and UserReg are described as follows.

2.5.1 AsyncAvlReg

Figure 2-7 AsyncAxiReg Interface

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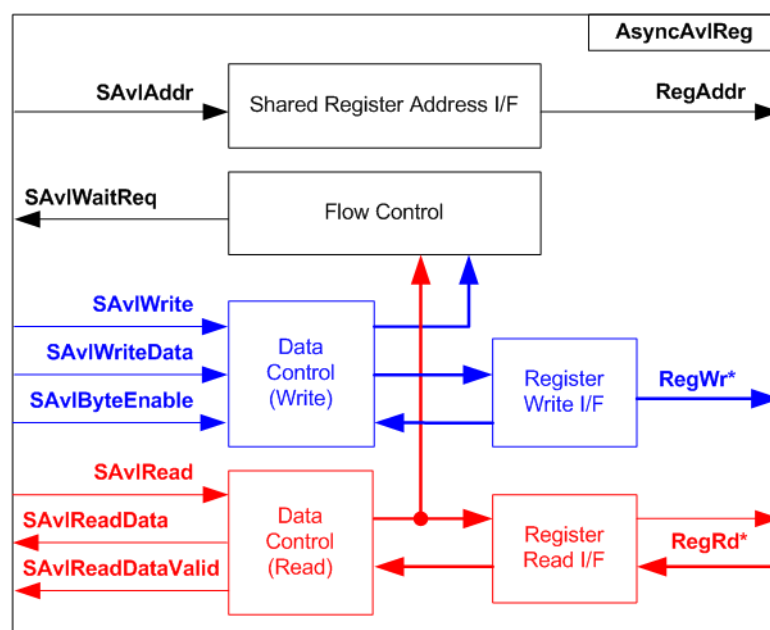https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

According to Avalon-MM specification, 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 controls SAvlWaitReq to hold the next request from Avalon-MM interface if the current request does not finish. Write control and Write data I/F of Avalon-MM bus are latched and transferred to be Write register interface with clock-crossing registers. Similarly, Read control I/F are latched and transferred to be Read register interface with clock-crossing registers. After that, the returned data from Register Read I/F is transferred to Avalon-MM bus by using clock-crossing registers. Address I/F of Avalon-MM is latched and transferred to Address register interface as well.

The simple register interface is compatible with single-port RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RdReq and RdValid signals for controlling read latency time. 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 2-8
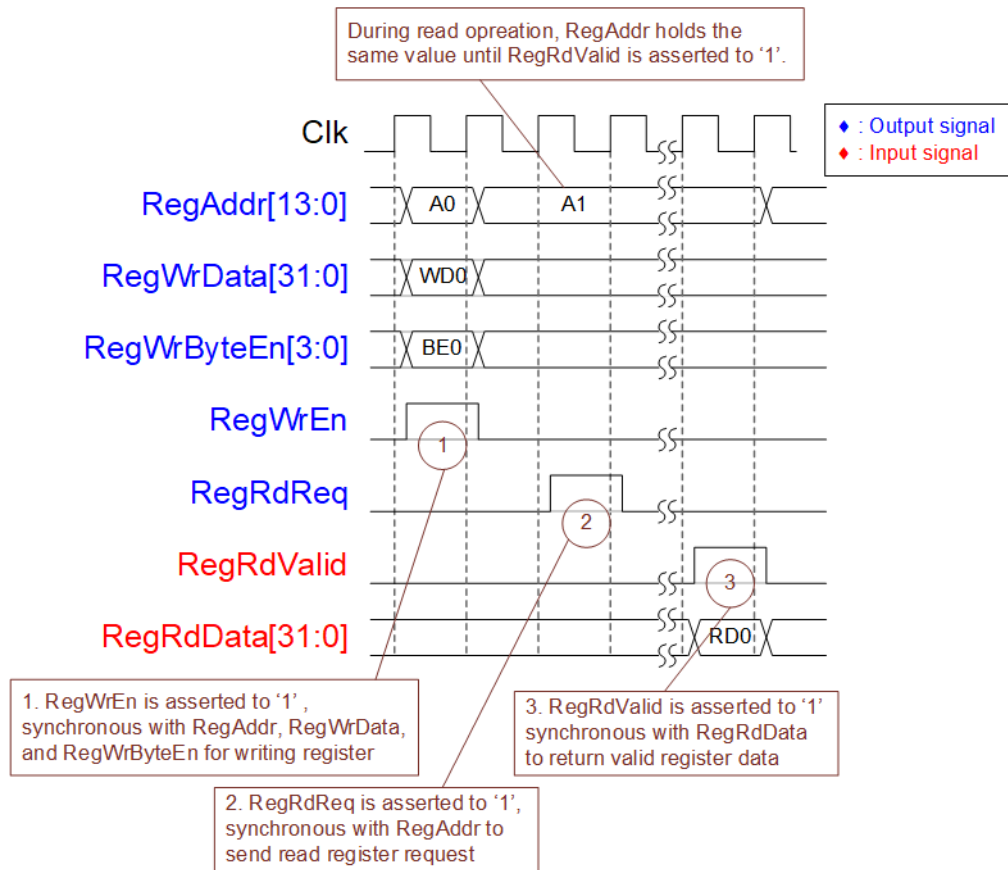
During read opreation, RegAddr holds the same value until RegRdValid is asserted to '1'.

◆ : Output signal
◆ : Input signal

1. RegWrEn is asserted to '1' , synchronous with RegAddr, RegWrData, and RegWrByteEn for writing register

2. RegRdReq is asserted to '1', synchronous with RegAddr to send read register request

3. RegRdValid is asserted to '1' synchronous with RegRdData to return valid register data

Figure 2-8 Register interface timing diagram

1) To write register, the timing diagram is similar to single-port 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. Bit[0], [1], [2], and [3] are equal to '1' when RegWrData[7:0], [15:8], [23:16], and [31:24] are valid respectively.

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. During read operation, the address value (RegAddr) does not change the value until RegRdValid is asserted to '1'. So, the address can be used for selecting the returned data by using multiple layers of multiplexer.

3) The read data is returned on RegRdData bus by the slave with asserting RegRdValid to '1'. After that, AsyncAvlReg forwards the read data to SAvlRead interface.
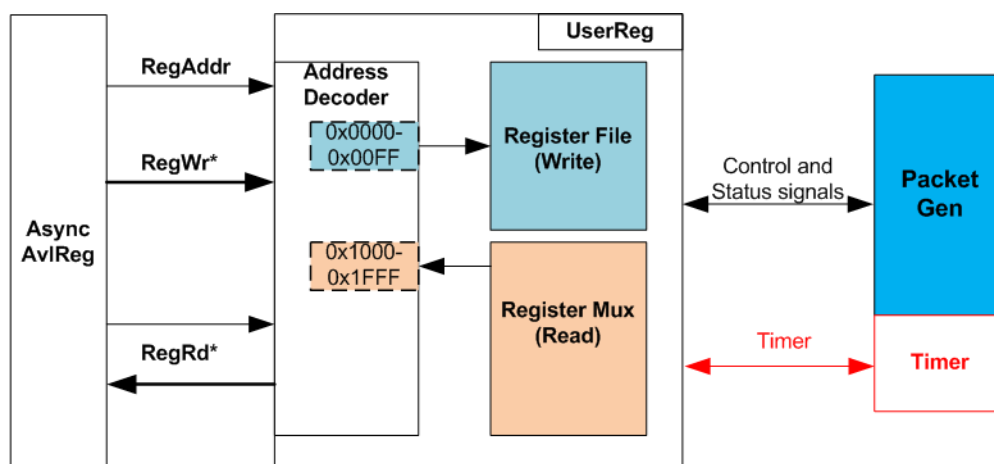
2.5.2 UserReg

Figure 2-9 UserReg block diagram

The address range to map to UserReg is split into two areas, as shown in Figure 2-9

1) 0x0000 – 0x00FF: mapped to set control signals of the PacketGen module. This area is write-access only.
2) 0x1000 – 0x10FF: mapped to read status signals of PacketGen module and returned value of the Timer. This area is read-access only.

Address decoder decodes the upper bit of RegAddr for selecting the active hardware. The register file inside UserReg is 32-bit size, so 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, one multiplexer is designed. Register Mux is the data multiplexer to select the read data for returning to CPU, so the latency of read data is equal to one clock cycle. RegRdValid is created by RegRdReq with asserting one D Flip-flop.

More details of the address mapping within UserReg module are shown in Table 2-1

Table 2-1 Register map Definition

Address	Register Name	Description
Wr/Rd	(Label in the ll10gemaclptest.c")	
BA+0x0000 – BA+0x00FF: Control Signal of PacketGen (Write-access only)		
BA+0x0000	User Command Reg (USRCMD_REG)	[0]: Start test operation. Set '1' to start the test. This signal is auto-cleared after the system begins the operation.
BA+0x0004	User Length Reg (USRLEN_REG)	[15:0]: Tx Packet size in byte unit. Valid from 5-9014 byte. When the packet size is less than 60-byte, zero-padding is filled by EMAC.
BA+0x0008	EMAC Reset Reg (EMACRST_REG)	[0]: Active-high reset signal for LL10GMAC IP.
BA+0x000C	Loopback Reg (LPBACK_REG)	[0]: Set loopback mode. 0: External (Loopback by connecting SFP+ loopback module) 1: Internal (Loopback by using Transceiver)
BA+0x1000 – BA+0x1FFF: Status signals (Read-access only)		
BA+0x1000	User Status Reg (USRSTS_REG)	[0]: Asserted when PacketGen is processed. Assert to '1' after USRCMD_REG[0] is set to '1'. De-assert to '0' after PacketGen finishes Tx and Rx transmission. [1]: Ethernet Linkup status, mapped to Linkup signal of LL10GEMAC-IP. [2]: Packet verification fail. '0': No error is found. '1': Received data is not correct. [3]: Asserted when LL10GEMAC-IP detects the error by asserting MacRxError. De-asserted to '0' when the new operation is started by setting USRCMD_REG[0]='1'.
BA+0x1004	Receive Length Reg (RXLEN_REG)	[15:0]: Receive packet size in byte unit. This value is equal to USRLEN_REG when USRLEN_REG is not less than 60-byte. Otherwise, RXLEN_REG is equal to 60-byte because zero-padding is included.
BA+0x1020	Timer Reg (TIMER_REG)	[31:0]: Read value of MacRoundTimer[31:0] to check round-trip latency time of LL10GEMAC-IP with Transceiver.
BA+0x1800	IPVersion Reg (IPVER_REG)	[31:0]: IPVersion output from LL10GEMAC-IP

3 CPU Firmware Sequence

After FPGA boot-up, LL10GEMAC-IP is initialized by setting loopback mode to be External mode or Internal mode. To run External loopback mode, it needs to connect SFP+ loopback module on the board. Otherwise, it does not need to use the loopback module. After that, reset signal is asserted and the IP starts initialization process. During initialization process, Linkup status from Ethernet MAC (USRSTS_REG[1]) is polling. The CPU waits until LL10GEMAC-IP is linked-up. Finally, main menu is displayed on the console, as shown in Figure 3-1.

```
+++ LL10GEMAC Loopback Test [IPVer = 2.0] +++

Input Loopback Mode : [0]External [1]Internal => 0

Start Reset
Reset Complete

Ethernet Link Up

--- Loopback Test menu ---
 [0] : Change Loopback Mode
 [1] : LL10GEMAC Loopback Test
```

Figure 3-1 Main Menu

There are two menus – Change loopback mode and Run loopback test. The first menu is designed to switch the loopback mode to external or internal mode. The details to run loopback test are described as follows.

3.1   Run Loopback test

The user inputs packet length and the number of packets to start the loopback test. After that, the test data is generated for sending to LL10GEMAC-IP. After the packet is loop-back returned to the transceiver by internal or external mode, the received packet from LL10GEMAC-IP is verified and latency time is measured. The details of the test sequence are described as follows.

1) Receive packet length transfer (byte) size and the number of packets from the user. The operation is cancelled when the input is invalid.
2) Set packet length to USRLEN_REG.
3) Start the test operation by setting USRCMD_REG[0]='1'.
4) The CPU waits until the operation is finished by monitoring busy flag (USRSTS_REG[0]) which changes from '1' to '0'.
5) Check error flag in the test (USRSTS_REG[3:2]). If some errors are found, error message is displayed.
6) Check receive length (RXLEN_REG) and display the error message if the read value is not equal to the expected length. Typically, the expected length is equal to the packet length, set by the user. The expected length is equal to 60 bytes if the packet length is less than 60 bytes which is the value including zero-padding.
7) Decrease total number of packets. If remained value is not equal to 0, repeat step 3) – 6). Before going to step 3), CPU calculates the minimum value, the maximum value, and the average value of round-trip latency time.
8) CPU displays the result - the minimum time, the maximum time, and the average time on the console.

3.2 Function list in User application

This topic describes the function list to run LL10GEMAC-IP loopback test.

void init_emac(void)	
Parameters	None
Return value	None
Description	Receive loopback mode from the user and set to LPBACK_REG. Next, start reset operation by asserting and de-asserting reset signal, controlled by EMACRST_REG. Finally, calling wait_ethlink function to wait until the ethernet link up.

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 3.1

void show_result(unsigned int av_ltc, unsigned int min_ltc, unsigned int max_ltc)	
Parameters	av_ltc: average latency time in clock cycle unit min_ltc: minimum latency time in clock cycle unit max_ltc: maximum latency time in clock cycle unit
Return value	None
Description	Convert the unit from clock cycle to be ns unit and display the results, i.e., the minimum latency time, the maximum latency time, and the average latency time.

void show_vererr(void)	
Parameters	None
Return value	None
Description	Read USRSTS_REG[2] (verify failed) and USRSTS_REG[3] (error status from LL10GEMAC-IP). Display the error message following the error flag.

void wait_ethlink(void)	
Parameters	None
Return value	None
Description	Read link-up status (USRSTS_REG[1]) and wait until the connection is linked up.

4 Revision History

Revision	Date	Description
1.0	26-Mar-21	Initial version release