LL10GEMAC IP reference design

Rev1.0 22-May-20

1 Introduction

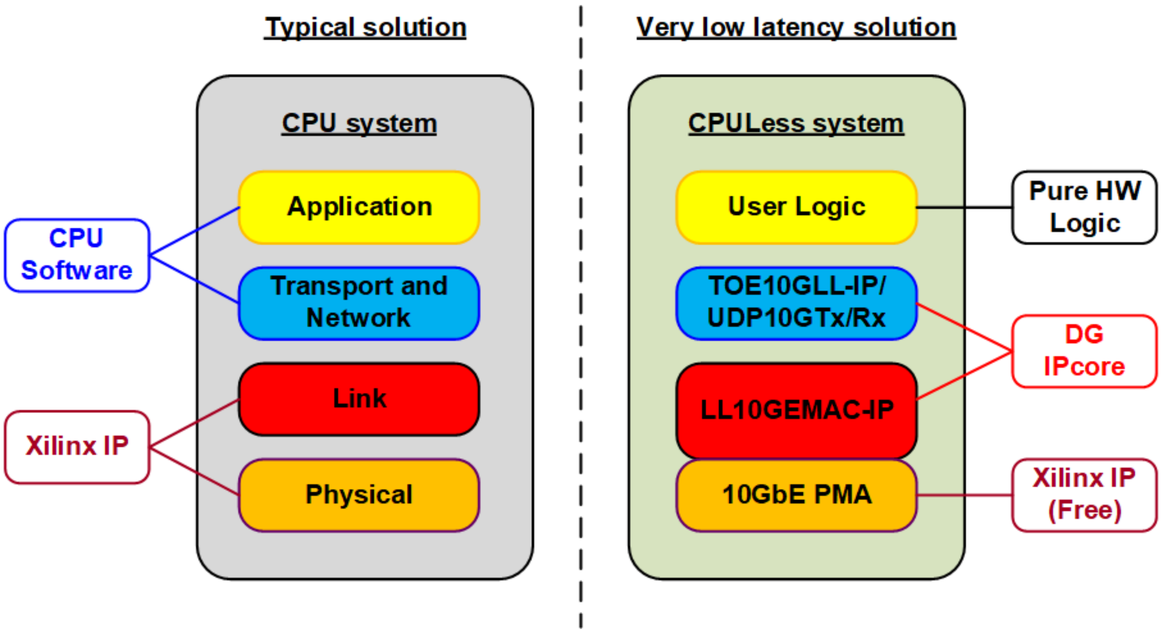


Figure 1-1 Low latency solution

The application, transport 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/25G Ethernet Subsystem which is Xilinx 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 must be designed instead. 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 Xilinx Transceiver.

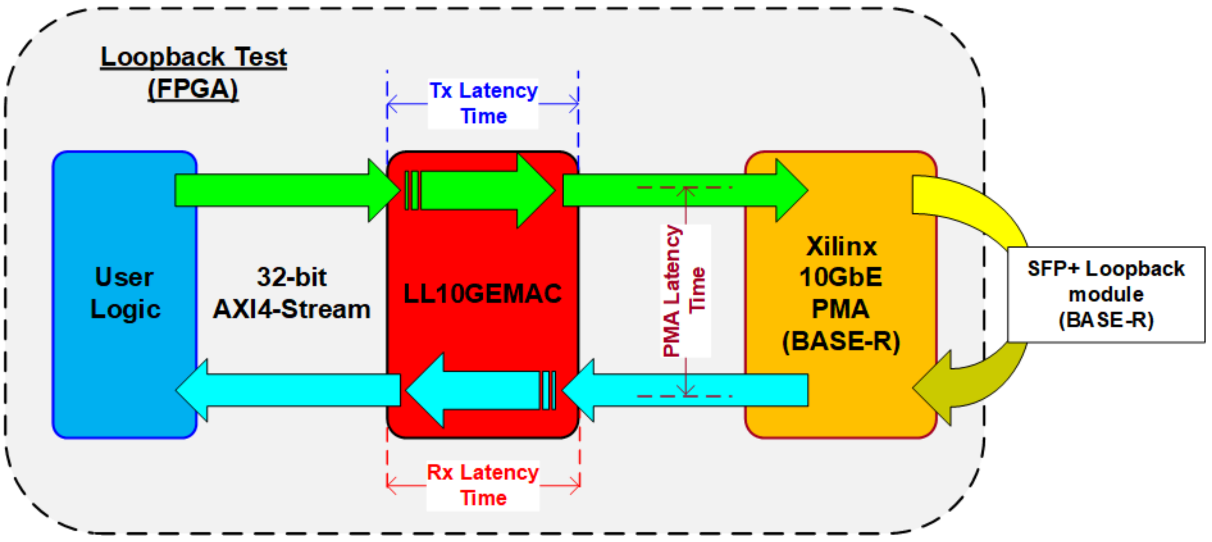


Figure 1-2 Loopback Test to check latency time

To check latency time of LL10GEMAC, the simple test logic can be designed as shown in Figure 1-2. SFP+ loopback module is inserted for transferring the packet from Tx interface to Rx interface. The user logic transfers the short packet and then verifies the received the packet to confirm the connection stability. Latency time can be measured by designing three counters, i.e. the counter for checking Tx latency time of LL10GEMAC, Rx latency time of LL10GEMAC, and PMA latency time of Xilinx transceiver and external SFP+ loopback cable.

CPU system is included for user interface by using Serial console. 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

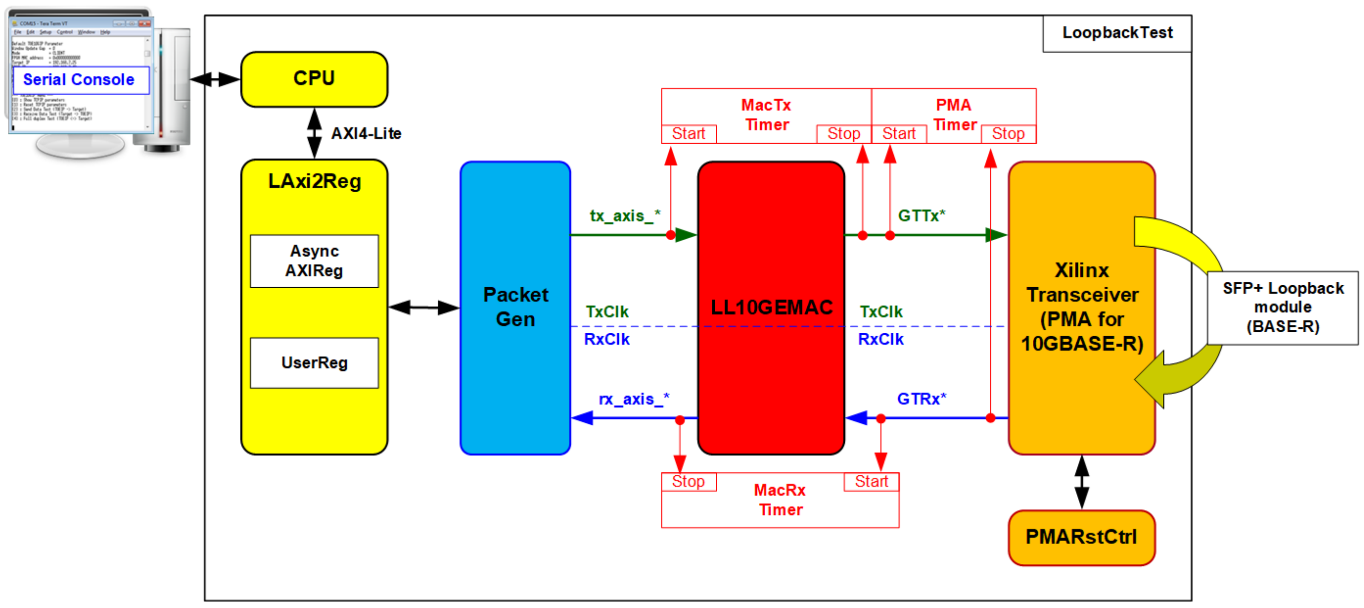


Figure 2-1 Loopback Test Block Diagram

CPU system is included for easy user interface, i.e. receiving the test parameters from the user and returning the test result for displaying via Serial console. CPU uses AXI4-Lite bus to interface with the hardware logic. LAxi2Reg 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. The packet is transferred from LL10GEMAC to Tx interface of Xilinx Transceiver, SFP+ Loopback module, and Rx interface of Xilinx Transceiver respectively. So, the same packet is decoded by LL10GEMAC and returned to PacketGen. The verification module in PacketGen can verify the received packet which must be the same as the transmitted packet to check the connection status. PMARstCtrl is designed to control reset sequence of Xilinx Transceiver.

The main objective of loopback test is to measure the latency time in LL10GEMAC IP and the low-level hardware containing Transceiver and SFP+ loopback module. Three timers are designed to capture latency time in Tx data path of LL10GEMAC (MacTx Timer), Rx data path of LL10GEMAC (MacRx Timer) and Xilinx Transceiver with external hardware (PMA Timer), as shown in Figure 2-1. The user can set the packet length, generated by PacketGen, by using Serial console. More details of each module are described as follows.

2.1 Xilinx Transceiver (PMA for 10GBASE-R)

PMA IP core for 10Gb Ethernet (BASE-R) can be generated by using Vivado IP catalog. In FPGA Transceivers Wizard, the user uses the following settings.

• Transceiver configuration preset	: GT-10GBASE-R
• Encoding/Decoding	: Sync. gearbox for 64B/66B
• Transmitter Buffer	: Bypass
• Receiver Buffer	: Bypass
• User/Internal data width	: 32

More details of Transceiver wizard such as Ultrascale model are described in the following link.

https://www.xilinx.com/products/intellectual-property/ultrascale_transceivers_wizard.html

2.2 LL10GEMAC

The IP core by DesignGateway implements low-latency EMAC and PCS logic for 10Gb Ethernet (BASE-R) standard. The user interface is 32-bit AXI4-stream bus. Please see more details from LL10GEMAC datasheet on our website.

2.3 PMARstCtrl

When the buffer inside Xilinx Transceiver is bypassed, the user logic must control reset signal to Tx and Rx buffer. The module is designed by state machine to run following step.
(1) Assert Tx reset to '1' for one clock cycle to the transceiver.
(2) Wait until Tx reset done = '1'.
(3) Finish Tx reset sequence and de-assert Tx reset output to start Tx operation.
(4) Assert Rx reset to the transceiver.
(5) Wait until Rx reset done = '1'.
(6) Finish Rx reset sequence and de-assert Rx reset output to start Rx operation.

2.4 PacketGen

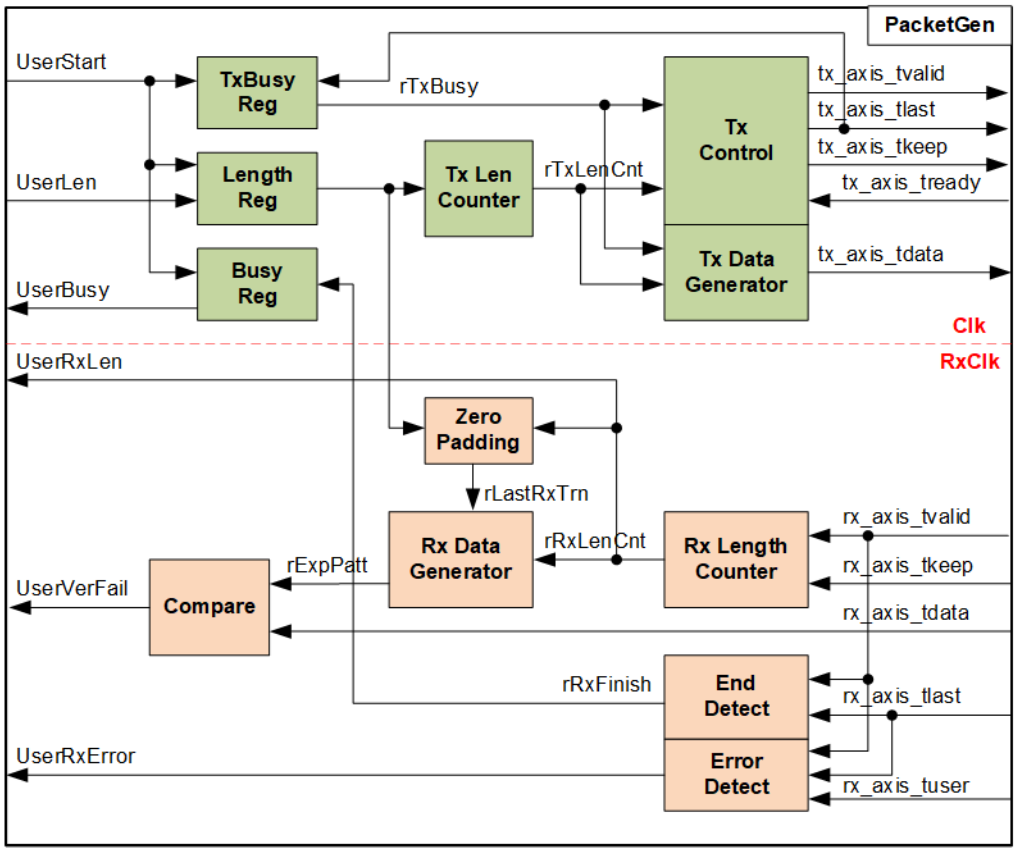


Figure 2-2 PacketGen module

PacketGen module is the test logic to send and receive one packet with LL10GEMAC-IP through AXI4-Stream interface. This module runs in two clock domains, i.e. Clk for transmitted operation and RxClk for received operation. As shown in Figure 2-2, the logic is split to two groups, i.e. Packet generator in the top side for generating one packet to AXI4-ST and Packet verification in the bottom side for verifying one packet from AXI4-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). Test pattern is 16-bit incremental data, created by the transmitted counter (rTxLenCnt). When packet size is not aligned to 32-bit, tx_axis_tkeep of the last data in the packet is asserted only some bits. At the same time, tx_axis_tlast is asserted to ‘1’ to finish the transmitted operation. TxBusy is asserted to ‘1’ after UserStart is received. After that, TxBusy changes from ‘1’ to ‘0’ when finishing transmitted operation (tx_axis_tlast=’1’). Tx Length Counter and Tx Data Generator are paused when tx_axis_tready is de-asserted to ‘0’. Transmitted data valid (tx_axis_tvalid) is always asserted to ‘1’ to send the packet via AXI4-ST until end of the transmitted packet.

UserBusy is designed for user monitoring PacketGen operation. When User asserts UserStart, UserBusy is asserted to ‘1’. UserBusy is de-asserted to ‘0’ after end of received packet is detected (rx_axis_tlast=’1’).

On the other hand, when the packet is returned from Tx AXI4-ST to Rx AXI4-ST via loopback cable, the packet is verified. The received data valid (rx_axis_tvalid) is applied to count the received packet size (rRxLenCnt). The counter output can be fed to the pattern generator (Rx Data Generator) to create the expected pattern for data verification. There is Zero-padding module for asserting rLastRxTrn to ‘1’ when the current received counter is equal to the packet size, set from user. After rLastRxTrn is asserted, the expect pattern changes to zero value to check zero-padding data in the packet. This feature is run when the packet size is less than 60 bytes. Fail flag (UserVerFail) is asserted to ‘1’ if the received data is not equal to the expected pattern. When the end of packet (rx_axis_tlast) 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, rx_axis_tuser is monitored. UserRxError is asserted to ‘1’ when rx_axis_tuser is equal to ‘1’.

2.5 Timer

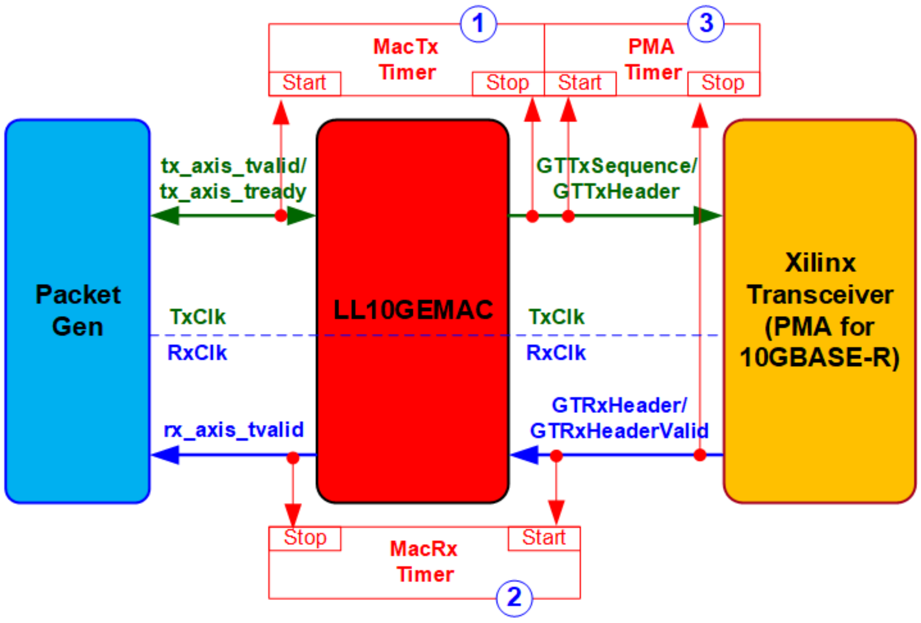


Figure 2-3 Timers in the reference design

There are three Timers in one test system to measure the latency time of LL10GEMAC and the latency time from Transceiver and hardware. All Timers are controlled by Start flag and Stop flag. In the test, one packet is transferred in the system. Start flag is asserted to '1' when the 1st data is found at the input of the measured module. Stop flag is asserted to '1' when the 1st 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, 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 LL10GEMAC IP. The timer starts when detecting the 1st data on Tx AXI4 Stream bus which is the interface between PacketGen and LL10GEMAC IP. The timer stops when detecting the 1st data on Tx Transceiver interface which is the interface with UltraScale Transceiver block.

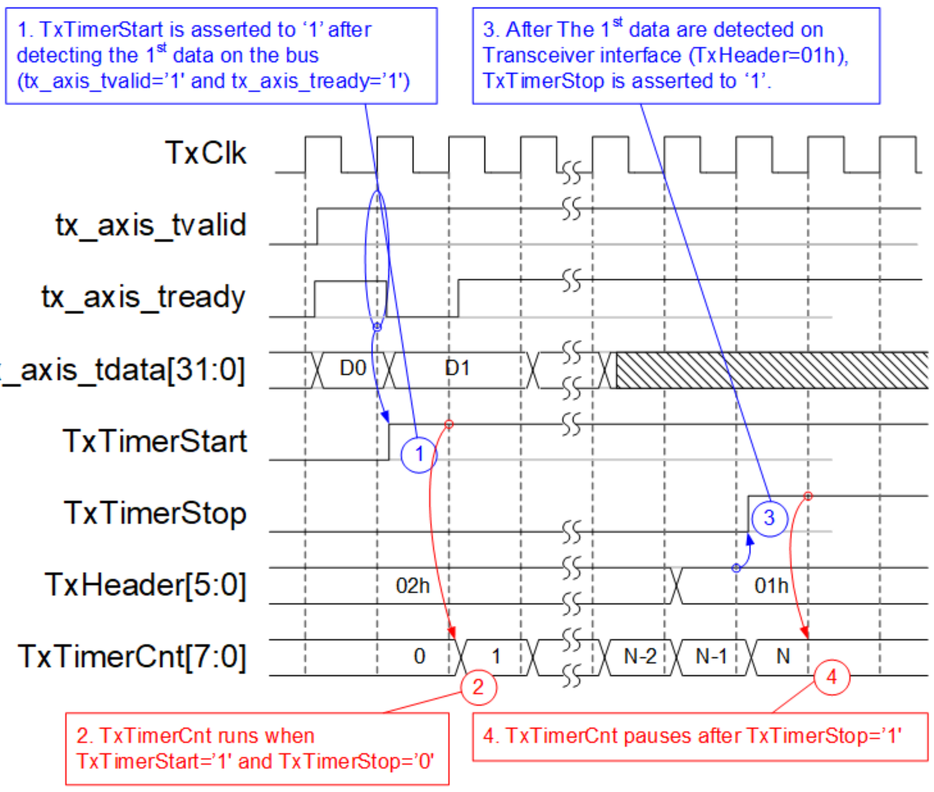

1. TxTimerStart is asserted to '1' after detecting the 1st data on the bus (tx_axis_tvalid='1' and tx_axis_tready='1')
3. After The 1st data are detected on Transceiver interface (TxHeader=01h), TxTimerStop is asserted to '1'.
TxClk
tx_axis_tvalid
tx_axis_tready
tx_axis_tdata[31:0]
D0
D1
TxTimerStart
TxTimerStop
TxHeader[5:0]
02h
01h
TxTimerCnt[7:0]
0
1
N-2
N-1
N
1
2
3
4
2. TxTimerCnt runs when TxTimerStart='1' and TxTimerStop='0'
4. TxTimerCnt pauses after TxTimerStop='1'

Figure 2-4 MacTxTimer timing diagram

Figure 2-4 shows the details to run MacTxTimer.

(1) The 1st data on AXI4 Stream is detected when tx_axis_tvalid='1' and tx_axis_tready='1'. After that, TxTimerStart is asserted to '1' to start Tx timer operation.

(2) The timer is increased every clock cycle until the stop flag is asserted.

(3) TxTimerStop is asserted to '1' when the 1st data on Transceiver interface is detected. The data is monitored via TxHeader which is equal to 01h for transferring the data, not header.

(4) Timer stops running and holds the value. The user can read the timer to check the latency time between the 1st data on AXI4 Stream and the 1st data on Transceiver interface.

2.5.2 MacRxTimer

The timer is designed to measure data latency time in Rx path of LL10GEMAC IP. The timer starts when the 1st data is detected on Rx Transceiver interface. The timer stops when the 1st data is detected on Rx AXI4 Stream bus, as shown in Figure 2-5.

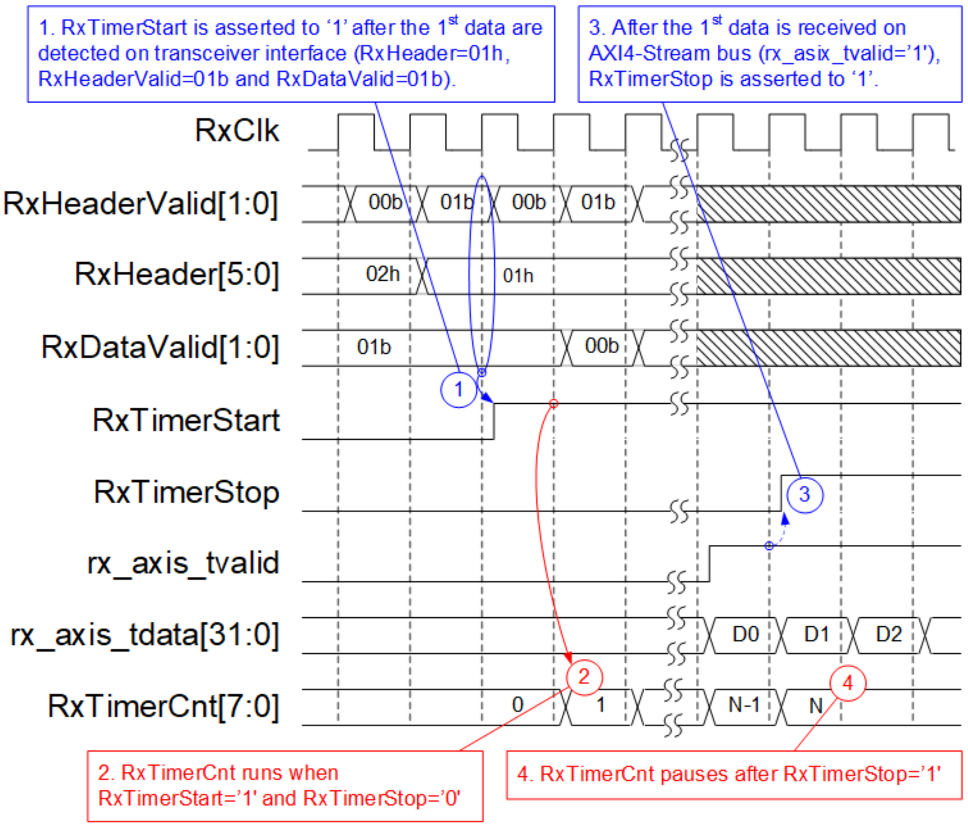
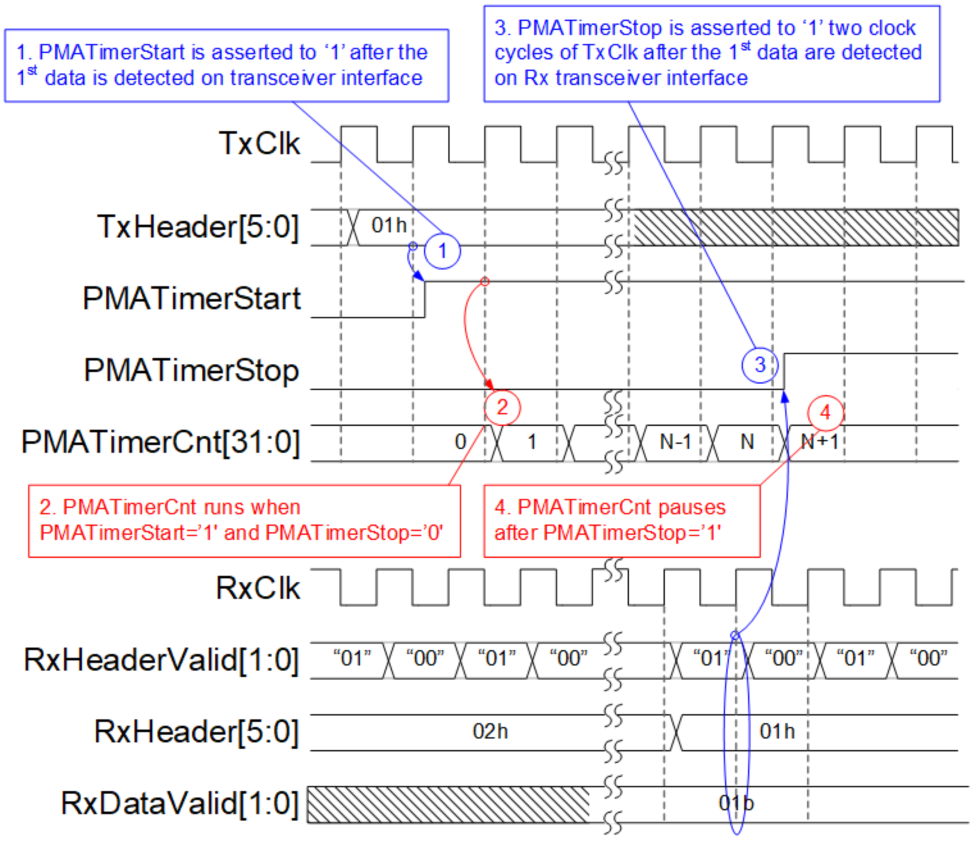

1. RxTimerStart is asserted to '1' after the 1st data are detected on transceiver interface (RxHeader=01h, RxHeaderValid=01b and RxDataValid=01b).
3. After the 1st data is received on AXI4-Stream bus (rx_asix_tvalid='1'), RxTimerStop is asserted to '1'.
RxClk
RxHeaderValid[1:0]
00b
01b
00b
01b
RxHeader[5:0]
02h
01h
RxDataValid[1:0]
01b
00b
RxTimerStart
RxTimerStop
rx_axis_tvalid
rx_axis_tdata[31:0]
D0
D1
D2
RxTimerCnt[7:0]
0
1
N-1
N
1
2
3
4
2. RxTimerCnt runs when RxTimerStart='1' and RxTimerStop='0'
4. RxTimerCnt pauses after RxTimerStop='1'

Figure 2-5 MacRxTimer timing diagram

(1) To assert Start flag of RxTimer, the logic scans the 1st data on Transceiver interface by detect RxHeaderValid=01b, RxHeader=01h, and RxDataValid=01b.
(2) After that, the timer is increased every clock cycle to count latency time.
(3) To assert Stop flag of RxTimer, the logic scans the 1st data valid on Rx AXI4 Stream bus by detecting rx_axis_tvalid='1'.
(4) After stopping the operation, RxTimer holds the value for the user reading latency time in Rx interface.

2.5.3 PMA Timer

The timer is designed to measure data latency time from Physical Medium Attachment layer (PMA) and external hardware which is connected by loopback cable. The timer starts when the 1st data is detected on Tx Transceiver interface. The timer stops when the 1st data is detected on Rx Transceiver interface. The condition for starting PMATimer is the same as the condition for stopping MACTxTimer. In the same way, the condition for stopping PMATimer is the same as the condition for starting MACRxTimer.



1. PMATimerStart is asserted to '1' after the 1st data is detected on transceiver interface
3. PMATimerStop is asserted to '1' two clock cycles of TxClk after the 1st data are detected on Rx transceiver interface
TxClk
TxHeader[5:0]
01h
PMATimerStart
PMATimerStop
PMATimerCnt[31:0]
0
1
N-1
N
N+1
2. PMATimerCnt runs when PMATimerStart='1' and PMATimerStop='0'
4. PMATimerCnt pauses after PMATimerStop='1'
RxClk
RxHeaderValid[1:0]
"01"
"00"
RxHeader[5:0]
02h
01h
RxDataValid[1:0]
01b

Figure 2-6 PMATimer timing diagram

(1) To assert Start flag of PMATimer, the same logic to assert Stop flag of MACTxTimer can be used.
(2) Similar to other times, PMA timer is increased every clock cycle to count latency time when Start='1' and Stop='0'.
(3) To assert Stop flag of PHYTimer, the logic scans the 1st data valid on Rx Transceiver interface, similar to start flag of MACRxtimer. This condition is run in RxClk but the PMATimer is run in TxClk. So, it needs to add asynchronous logic to forward Stop flag from RxClk to TxClk. It is estimated that one more clock cycle is increased to check the latency time from asynchronous circuit.
(4) After stopping the operation, PMATimer holds the value for the user reading latency time in PMA IP and external hardware.

2.6 CPU and Peripherals

32-bit AXI4-Lite bus is applied to be the bus interface for CPU accessing the peripherals such as Timer and UART. To control and monitor the test logic of LL10GEMAC, the test logic is connected to CPU as a peripheral on 32-bit AXI4-Lite bus. CPU assigns the different base address and the address range for each 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 AXI4-lite bus standard for writing and reading the register. LAxi2Reg module is designed to connect the CPU system as shown in Figure 2-7.

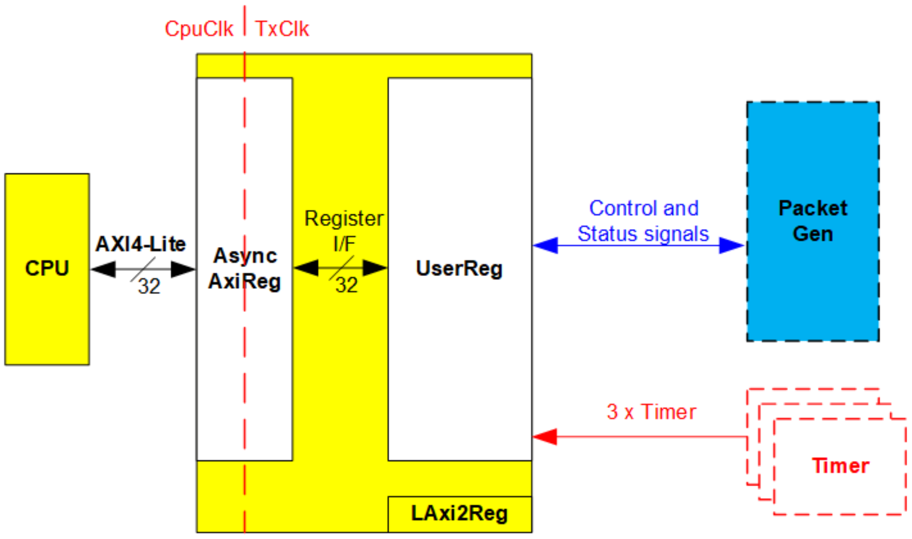


Figure 2-7 CPU and peripherals hardware

LAxi2Reg consists of AsyncAxiReg and UserReg. AsyncAxiReg is designed to convert the AXI4-Lite signals to be the simple register interface which has 32-bit data bus size (similar to AXI4-Lite data bus size). Moreover, AsyncAxiReg includes asynchronous logic to support clock crossing between CpuClk domain and TxClk domain.

UserReg includes the register file of the parameters and the status signals to control and monitor PacketGen and all Timers. More details of AsyncAxiReg and UserReg are described as follows.

2.6.1 AsyncAxiReg

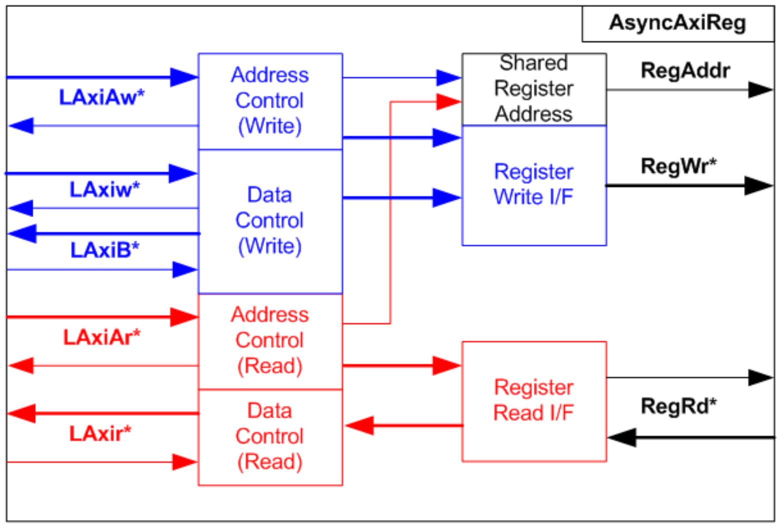


Figure 2-8 AsyncAxiReg Interface

The signal on AXI4-Lite bus interface can be split into five groups, i.e. LAxiAw* (Write address channel), LAxiw* (Write data channel), LAxiB* (Write response channel), LAxiAr* (Read address channel), and LAxir* (Read data channel). More details to build custom logic for AXI4-Lite bus is described in following document.
https://forums.xilinx.com/xlnx/attachments/xlnx/NewUser/34911/1/designing_a_custom_axi_slave_rev1.pdf

According to AXI4-lite standard, the write channel and the read channel are operated independently. Also, the control and data interface of each channel are run separately. So, the logic inside AsyncAxiReg to interface with AXI4-lite bus is split into four groups, i.e. Write control logic, Write data logic, Read control logic, and Read data logic as shown in the left side of Figure 2-8. Write control I/F and Write data I/F of AXI4-Lite bus are latched and transferred to be Write register interface with the shared register address. Besides, Read control I/F and Read data I/F of AXI4-Lite bus are latched and transferred to be Read register interface with the shared register address.

The simple register interface is designed to compatible with general RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RdReq and RdValid signal for controlling read latency. 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 2-9.

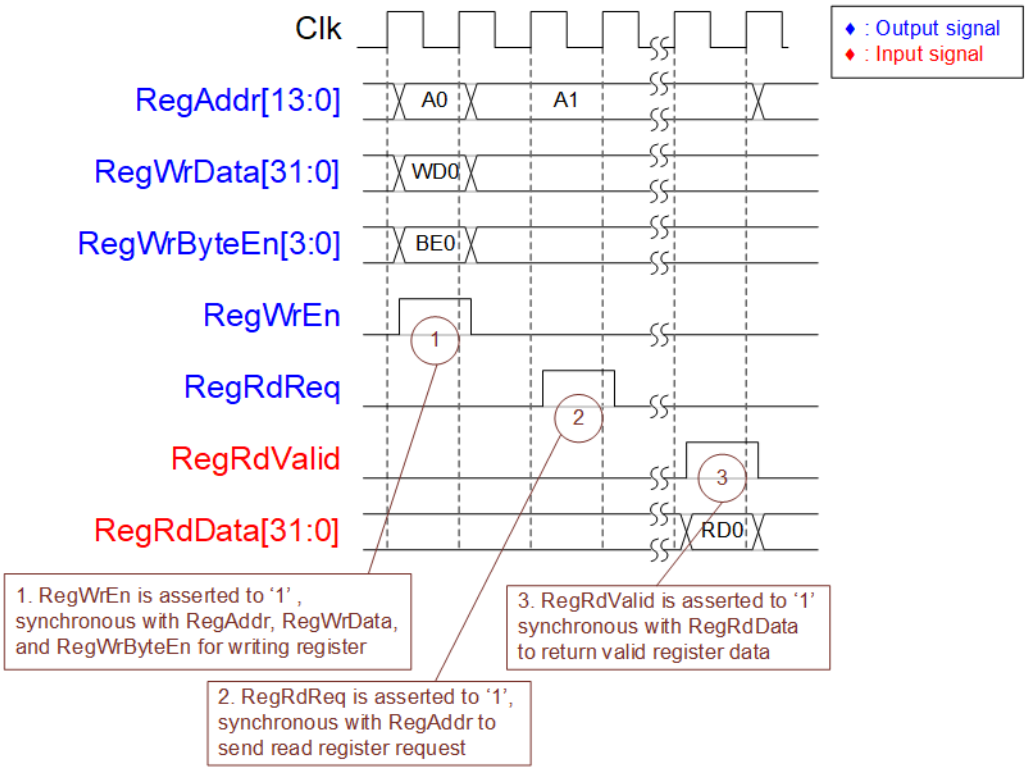

Clk
RegAddr[13:0]
RegWrData[31:0]
RegWrByteEn[3:0]
RegWrEn
RegRdReq
RegRdValid
RegRdData[31:0]
A0
A1
WD0
BE0
RD0
1
2
3
♦ : Output signal
♦ : Input signal
1. RegWrEn is asserted to '1', synchronous with RegAddr, RegWrData, and RegWrByteEn for writing register
2. RegRdReq is asserted to '1', synchronous with RegAddr to send read register request
3. RegRdValid is asserted to '1' synchronous with RegRdData to return valid register data

Figure 2-9 Register interface timing diagram

1) To write register, the timing diagram is similar to 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 bit for 4-byte data, i.e. bit[0] for RegWrData[7:0], bit[1] for RegWrData[15:8], and so on.
2) To read register, AsyncAxiReg 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, AsyncAxiReg forwards the read value to LAxir* interface.

2.6.2 UserReg

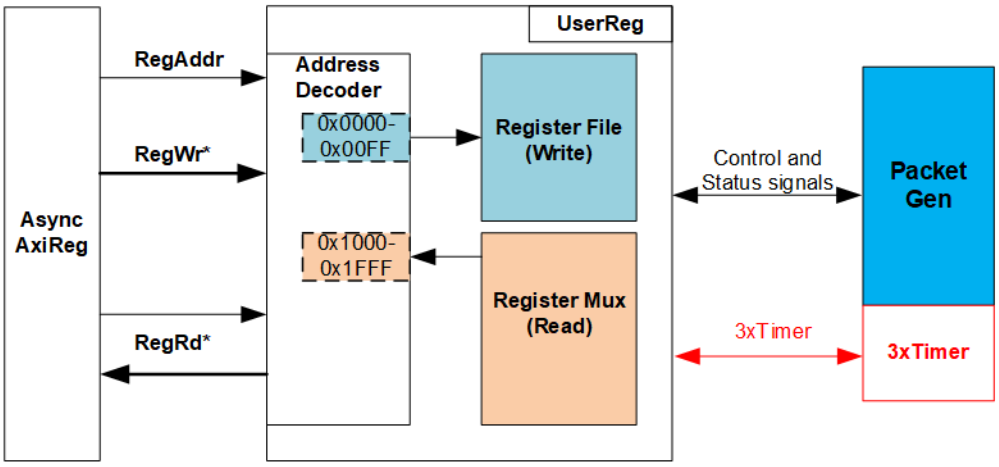


Figure 2-10 UserReg block diagram

The address range to map to UserReg is split into two areas, as shown in Figure 2-10

1) 0x0000 – 0x00FF: mapped to set control signals of the PacketGen module. This area is writing access only.
2) 0x1000 – 0x10FF: mapped to read status signals of PacketGen module and returned value of all Timers. This area is reading 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 cycles. RegRdValid is created by RegRdReq with asserting one D Flip-flop.

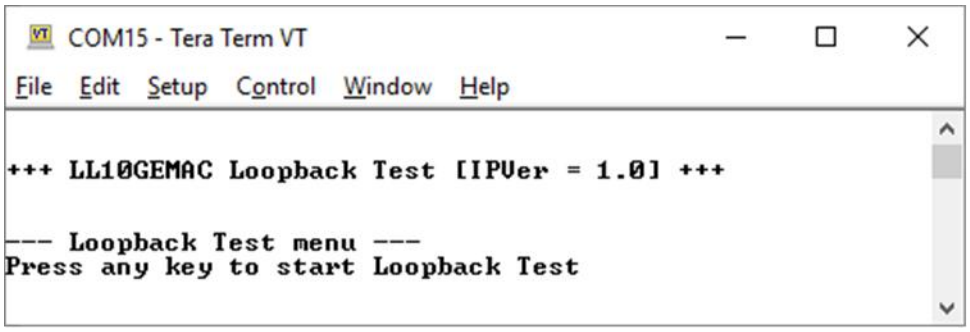
More details of the address mapping within UserReg module is shown in Table 2-1

Table 2-1 Register map Definition

Address Wr/Rd	Register Name (Label in the tenglpbacktest.c")	Description
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+0x1000 – BA+0x1FFF: Status signals (Read access only)		
BA+0x1000	User Status Reg (USRSTS_REG)	[0]: UserBusy signal of PacketGen 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, output of LL10GEMAC. [2]: Packet verification fail. '0': No error is found. '1': Received data is not correct. [3]: Error packet found in LL10GEMAC (rx_axis_tuser='1').
BA+0x1004	Received Length Reg RXLEN_REG	[15:0]: Received 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	MacTx Timer Reg (TXTIMER_REG)	[7:0]: Read value of TxTimer[7:0] to check latency time of Tx path of LL10GEMAC.
BA+0x1024	MacRx Timer Reg (RXTIMER_REG)	[7:0]: Read value of RxTimer[7:0] to check latency time of Rx path of LL10GEMAC.
BA+0x1028	PMA Timer Reg (PMATIMER_REG)	[31:0] Read value of PMATimer[31:0] to check latency of Transceiver and Loopback cable.
BA+0x1800	IPVersion Reg (IPVER_REG)	[31:0] IPVersion output from LL10GEMAC

3 CPU Firmware Sequence

After FPGA boot-up, Linkup status from Ethernet MAC (USRSTS_REG[1]) is polling. The CPU waits until LL10GEMAC-IP is linked-up. Next, welcome message and main menu are displayed on the console, as shown in Figure 3-1.



```
+++ LL10GEMAC Loopback Test [IPVer = 1.0] +++

--- Loopback Test menu ---
Press any key to start Loopback Test
```

Figure 3-1 Main Menu

3.1 Run Loopback test

The user inputs packet length parameters to start the loopback test. To run the test, test data is generated for sending to LL10GEMAC IP. After the packet is returned to the transceiver (PMA) via loopback cable, the verification module verifies the received data from LL10GEMAC. The details of the test sequence are described as follows.

1) Receive packet length transfer (byte) size from the user. The operation is cancelled when the input is invalid.
2) Set packet length to USRLEN_REG.
3) Start 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) CPU reads all timers (TXTIMER_REG, RXTIMER_REG, PMATIMER_REG) and displays latency time of each path on the console.

3.2 Function list in User application

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

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()	
Parameters	None
Return value	None
Description	Read TXTIMER_REG, RXTIMER_REG, and PHYTIMER_REG for displaying the latency time in ns unit.

void show_vererr(void)	
Parameters	None
Return value	None
Description	Read FAILNO_REG (error word address), EXPPAT_REG (expected value), and RDPAT_REG (read value) to display verification error details on the console

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	22-May-20	Initial version release