LeWiz Ethernet MAC Controller

Core3 Test Bench Specification

LeWiz Communications, Inc.

"The Wizard of Internet Communications"

July 31, 2019
Revision 1.03

PO Box 9276
San Jose, CA 95157
www.lewiz.com
Email: support@lewiz.com

LeWiz can be contacted at: support@lewiz.com
or address: PO Box 9276, San Jose, CA 95157-9276
www.lewiz.com
Author: LeWiz Communications, Inc.

Change Log

Version	**Significant Changes**
1.00	Release version (pending review)
1.01 OSrc	Open source release
1.02	Update with Core2 info
1.03	Update with Core3 info

Table of Contents

NOTE: This document is intended for HW users using LeWiz MAC IP Core Test Bench.

1 Introduction

LeWiz's MAC IP core (LMAC) is designed for use in a wide range of applications from small IoT devices to high performance computing systems. The core is intended for use on FPGA or in SoC ASICs. Its PHY interface is designed to fit into a range of SerDes available in ASIC libraries or FPGA tools. The test bench is developed for simulation of the LMAC IP core. This document describes the test bench from the user's viewpoint. For the description of the LMAC IP core, please refer to *LeWiz Ethernet MAC Controller IP Core Specification*.

Three cores are planned so three test benches wuld be available to support the 3 cores (see table below). Core 1 test bench will be available first, followed by Core 2 test bench then Core 3 test bench.

Core Name	Speed	PHY Interfaces	Notes
Core1	10/100/1000Mbps	GMII	125MHz clock
Core2	10G/5G/2.5G/1Gbps	XGMII	156MHz clock (125MHz for 1Gbps)
Core3	100G/50G/40G/25G/10Gbps	X/XL/CGMII	390.6MHz clock (312.5MHz for 40G) (156MHz for 10G)

Release 1.03 of this document is intended to describe the Core3 test bench. Additional information about other cores will be added as they become available.

The test bench supports ModelSim simulator. The test bench suite includes the test bench code, test examples (test vector files, test scripts, test results), list of test examples, and this user manual. The test examples are intended to illustrate the main functionalities of the IP core (which is to transmit and receive packets). Each test has a test script which is self-contained so it's easy to run a test example. The test result of the test example shows the expected result of the specific test. The test vectors follow the format of the popular Wireshark tool.

The LMAC core can be used in ASIC or FPGA technologies such as Intel or Xilinx. Thus, the bench is developed to be technology independence. It does not contain any technology dependent item such as FIFO or memory elements from specific vendor or technology.

The test bench code contains behavioral code and is intended for simulation only and not intended for synthesis. An example of this model is the AXIS_MASTER model.

This specification describes the test bench, answers the questions that the user may have in running and using the test bench.

The reader is expected to be familiar with Ethernet, LMAC IP cores, networking and FPGA/ASIC development.

(Core1/Core2/Core3 has been released to Github. All applicable LMAC Core information are described in here as much as possible.)

1.1 Definitions, conventions and notations

- The word "connection" usually means TCP/IP connection.
- MAC = Media access controller (i.e. Ethernet controller chip in this case)
- B = in size, "B" means byte
- b = in size, "b" means bit
- lsb = least significant bit. Example: DATA[63:0], bit 0 is lsb, bit 63 is most significant bit (msb)
- API = Application programming interface
- CMD = command
- Reserved = unimplemented, undefined, or spare for future use.
- Optional = Optional are items that may or may not be supported or required.
- RX = Receive
- TX = Transmit
- Byte = 8 bit
- Word = 16 bit
- Double word = DWord = 32 bit
- Quad word = Qword = Qwd = 64 bit
- SW = Software
- HW = Hardware
- Beat = each data slot (64 bit quantity and attributes) per clock pulse on the bus transfer is referred to as a beat.
- SIP = Source IP address
- DIP = Destination IP address
- Sport = Source TCP (or UDP) port
- Dport = Destination TCP (or UDP) port
- NIC = Network interface card
- PLL = Phase lock loop
- LVDS = low-voltage differential signaling
- SerDes = Serializer, De-serializer converting serial analog data into parallel digital data or vice versa.
- Ethernet Packet = Each Ethernet Packet has a packet header and a packet payload. The normal packet size is <1518 bytes total including the header.
- Ethernet Frame = An Ethernet frame is formed by encapsulating framing information around the Ethernet packet so the Ethernet packet is contained within the Ethernet frame. See the diagrams below.
- Pulse = a single clock assertion for the signal.
- User Logic = Logic blocks designed by the user which interfaces to the LMAC.

- PHY = Physical layer. For this document, it means the interface logic/analog circuit interfacing from the LMAC to the physical layer of the Ethernet. The physical layer can be copper or fiber optic. If the physical layer is copper, it may require some logic to interface from the LMAC PHY interface to an external copper PHY chip such as those from Marvel. If the physical layer is optical, it may require a SerDes to interface from the LMAC XGMII (for example) to an external optical transceiver module.
- Promiscuous Mode = in Ethernet, Promiscuous is a mode where the LMAC accepts RX packets to any destination MAC address (even the destination MAC address does not match its own designated address). This mode may be used for debug or traffic capture purposes.
- SoF = Start of Frame
- EoF = End of Frame
- CRC = Cyclic redundancy check

2 Overview

Figure 2.1 shows various main functions inside LeWiz LMAC test bench.

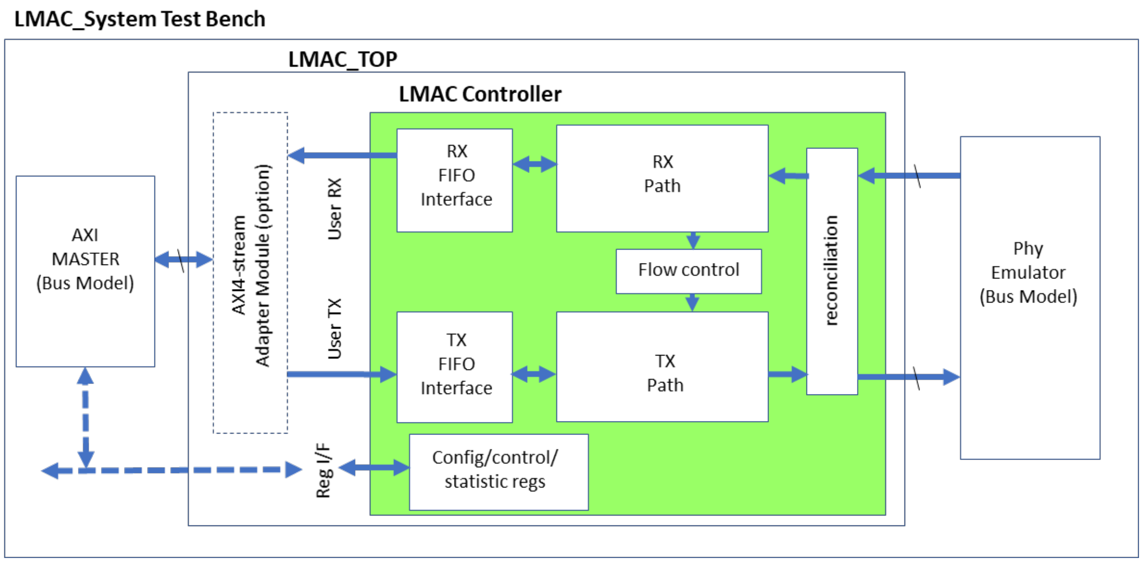


Figure 2.1: LMAC test bench block diagram (shown with GMII interface)

The bench contains the LMAC Core (LMAC Controller). It also integrates an optional AXI4-stream bus adapter module. This adapter module allows the LMAC controller to convert internal FIFO interface to AXI4-stream bus protocol interface. The bench also uses an AXI4-stream master (behavioral model) to generate traffic into the LMAC core. The AXI4-stream master also receives traffic from the LMAC core and has the ability to check the received data with user specified expected data. This gives the bench the ability to self-check and reports pass or fail conditions to the user. For Core1 and Core2, the AXI4-stream master is a 64-bit bus model and for Core3, the AXI4-stream master is a 256-bit bus model.

To emulate the PHY, the PHY emulator behavioral model is integrated into the bench. This emulator emulates the different PHY devices. For Core3, it emulates the PHY for 100G, 50G, 40G, 25G, and 10Gbps speeds. The PHY emulator also has the ability for loopback testing or network-generated traffic. In loopback testing, the packet transmitted by the TX-path will be looping back to the RX-path. The PHY emulator edits the fields in the packet to support this. For example, the source MAC address in the transmitted packet would become the destination MAC address for the RX side. In emulating network-generated traffic, the PHY emulator has the ability to accept user specified data patterns (or packets) contained in text file.

The entire bench is coded in Verilog and support ModelSim simulator.

As mentioned above, the bench requires test vector files from the user. This allows user to provide specific tests to the bench or instruct the bench to perform specific test or test conditions. The default path of these files is:
C:\LMAC3_INFO (on Windows platform path format)
Or
C:/LMAC3_INFO (in ModelSim path format. Note the use of "/")
The user must allow WRITE permission on this directory and ones below them.

(To be consistent and avoid manual code change in the test bench, user must create similar directory path on user's Windows system and put the test bench's code and files in this directory.)

3. Release Dependent Information

Each test bench release also contains a README files and a directory description file. These are release dependent and will not be described here.

This document only refers to the directory which are common for all releases.

Test examples are provided with each test bench release in C:\LMAC_INFO\TESTS directory.

The following uses examples to describe the formats of different file types used by the test bench.

4. Test Bench File Formats

The bench uses different file formats for performing different functions. The following describes the various files used by the bench. From these examples, the users should be able to construct their own if needed.

3.1. AXI4-Stream Master Data Files Format

Two data files are used by the AXI4-Stream master: *memory_wr_data.txt* and *memory_wr_ctrl.txt*. These files must be provided by user and can be created using a text editor. The AXI4-stream master uses the information in these 2 files to generate a transmit packet to the LMAC core and eventually the data goes out to the PHY emulator TX interface. See example below.

memory_wr_data.txt file contains the packet data in the Wireshark packet format (or big-endian network format). (The format for a TCP packet format is provided in Appendix A.)

Correspondingly, the *memory_wr_ctrl.txt* file contains control information about the corresponding packet in the *memory_wr_data.txt* file. Note that the "Starting Line" for the packet in the *memory_wr_ctrl.txt* file applies to the packet and not just the first qword. See example below.

These 2 files must be located at: (path name in Windows format)
C:\LMAC3_INFO\AXI_MASTER\ *memory_wr_data.txt*
C:\LMAC3_INFO\AXI_MASTER\ *memory_wr_ctrl.txt*

Note that the user must enable WRITE permission for this directory. The file names must be as above.

Below is a line-by-line description of the 2 files:
(The start of a packet is always on a qword boundary.)
(For ease of debug, it's recommended to have the same number of lines in both the *memory_wr_data.txt* file and the *memory_wr_ctrl.txt* file.)
("_" are inserted in hex numbers for easy reading here but should not appear in the actual files.)

Line #	**memory_wr_data.txt**	**memory_wr_ctrl.txt**	**Notes**
1	// comment example 1	//	Comment line
2	// IP, UDP	//	Comment line
3	// packet 1, 10 qwds	//	
4	1200_FFFF_FFFF_FFFF	6_000_09_00	Starting Line (see Note 1 below)
5	0045_0008_0900_0032	00	Packet info (or data)
6	1140_0000_0000_3C00	00	Packet info (or data)
7	0000_0000_0000_B27A	00	Packet info (or data)
8	2800_0104_0004_0000	00	Packet info (or data)
9	A8c0_0501_a8c0_7A9C	00	Packet info (or data)
10	0000_00BE_BAFE_CA01	00	Packet info (or data)
11	0000_0000_0000_0000	00	Packet info (or data)
12	0000_EFBE_ADDE_0000	00	Packet info (or data)
13	0000_EDD9_4BE1_0000	00	Only 6 bytes "EDD9_4BE1_0000" are data here
14	// end of pkt 1	//	
15	// ARP Reply packet	//	

16	// packet 2, 6 qwds	//	
17	602c_512b_49c9_1e00	2_000_05_00	Starting Line (see Note 2 below)
18	0100_0608_5bc8_b40c	00	Packet info (or data)
19	602c_0200_0406_0008	00	Packet info (or data)
20	5a07_a8c0_5bc8_b40c	00	Packet info (or data)
21	a8c0_512b_49c9_1e00	00	Packet info (or data)
22	0000_0000_0000_e707	00	Only 2 bytes "e707" are data
23	// end of pkt 2	//	
24			

NOTES:

1) Only 3 digits in the 32-bit value in the *memory_wr_ctrl.txt* file Starting Line are used. Others are don't care. Bit [15:8] is the number of qwords before the last qword. Bit [31:28] is the number of bytes in the last qword of the current packet. For packet 1: 6_000_09_00 means packet 1 has 9 data qwords followed by 6 data bytes
2) 2nd example, for packet 2 Starting Line:
   2_000_05_00 means packet 2 has 5 data qwords followed by 2 data bytes

AXI4-stream master also generates its own files in C:\LMAC3_INFO\AXI_MASTER directory for self-checking purposes. These will have different file names and must not be used or modified by the user.

3.2. ModelSim Test Script File Format

Script files contain ModelSim commands in a sequence required for testing a specific function in the bench. (Refer to ModelSim tool's user document for more information about the specific ModelSim command.)

To run the script:

1) Invoke the ModelSim simulator
2) At the command pane/window, type in:
   source path_to_script_file

Executing the script file causes a test to run in simulation.

Below is a line-by-line description of the typical scripts and commands used in the script. The example below shows how to transmit a packet in the test bench.

The script can force signals to set a test configuration. This allows flexibility in testing of different configurations using only script.

(A command line can be lengthy so in this document it's broken up in multiple lines to keep within the width of the page. The name of your specific script file may be different than the file name below.)

Line #	Script_file.txt	Notes
1	# comment a line	Comment line
2	vsim work.lmac_stream_tb	Causing the simulator to load the testbench
3	add wave -r /*	Track all the signals for waveform viewing
4	force -freeze sim:/LMAC_STREAM_TB/ fmac_speed 2'b01 0	Force a signal to a value (2'b01 = 1G speed, in this case)
5	force -freeze sim:/LMAC_STREAM_TB/phy_emulator_8b/ rx_pkt_gen_start_addr 64'd00 0	Initialize the starting address of the PHY emulator memory index to 64'd0.
6	run 200ns	Advance the simulation by 200nS
7	# Transmitting a packet # by pulsing the *gen_en_wr* signal	Note that in this case the clock on the AXI master is set at 250MHz or 4nS period.
8	force -freeze sim:/LMAC_STREAM_TB/gen_en_wr 1 0	Pull the control signal high
9	run 4ns	Keep high for 1 clock
10	force -freeze sim:/LMAC_STREAM_TB/gen_en_wr 0 0	Pull the control signal low to negate
11	run 10us	Run the simulation to see the packet come out at GMII interface level
12		

Note that the 1st *gen_en_wr* pulse, causes AXI master to default to its memory space address = 0.

The second *gen_en_wr* pulse, causes the AXI master to automatically calculate the next qword address for its next packet. The next packet qword address = total number of qwords contained in the previous packet(s).

Modelsim script files may contain *mem load* command to cause the content of memory in bus functional models to be loaded (as initialization vectors) from user specified text files (see example for PHY EMULATOR below) for testing purposes.

3.3. PHY Emulator – User Specified Test File Format

Two files are used by the PHY emulator. A 64-bit wide file contains the packet data. The packet data must be in network or big-endian format. A second file contains the corresponding control information for the packet. This is similar in concept to XGMII bus' RxD and RxC information. The packet data can be packets captured by the network tools such as Wireshark.

Below is a line-by-line dissection of the 2 files for a user specified packet as an example: (For illustration only)

(.mem are text files which can be edited by a text editor.)
(Comments can be inserted into the file with "//")
("_" are inserted for easy reading)

rx_pkt_gen_data.mem file has 64 bit per line (or 8 bytes)

rx_pkt_gen_ctrl.mem file has 8 bit per line. Each bit corresponds to the byte of the Data_file.mem. If the bit is 1, it means the corresponding data byte is a control information. 0 means the corresponding byte is a data byte.

Both .mem files are text files, created by the user using a text editor. Their path names must be: (Windows path format)
C:\LMAC_INFO\PHY_EMULATOR\ *rx_pkt_gen_data.mem*
C:\LMAC_INFO\ PHY_EMULATOR \ *rx_pkt_gen_ctrl.mem*

All the digits are in hex.

Line #	**rx_pkt_gen_data.mem**	**rx_pkt_gen_ctrl.mem**	**Notes**
1	// test example pkt 1	//	Comment line
2	0707_0707_0707_0707	FF	IDLE pattern
3	d555_5555_5555_55FB	01	Preamble pattern
4	1200_FFFF_FFFF_FFFF	00	Dest MAC is a broadcast
5	0045_0008_0900_0032	00	IP packet, Ethernet type
6	1140_0000_0000_3C00	00	UDP packet, length = 3C
7	0000_0000_0000_B27A	00	Packet info (or data)
8	2800_0104_0004_0000	00	Packet info (or data)
9	A8c0_0501_a8c0_7A9C	00	Packet info (or data)
10	0000_00BE_BAFE_CA01	00	Packet info (or data)
11	0000_0000_0000_0000	00	Packet info (or data)
12	0000_EFBE_ADDE_0000	00	Packet info (or data)
13	07FD_EDD9_4BE1_0000	C0	FD = EoF, and CRC field
14			

The memory in the PHY emulator model is deep (2048 qwords). It can contain multiple packets for use in a simulation session. This way different script files can share a data/control memory file pair.

5. Test Bench Loopback and Network Generated Mode

The test bench has 2 test modes for the PHY interface side: loopback mode and network-generated packet mode.

How to put the bench in loopback mode? If *rx_pkt_gen_sel* signal in the test bench (LMAC_STREAM_TB.v) equals 0, it puts the test bench in loopback mode. This causes the PHY emulator to modify the packets generated from the TX path to allow the LMAC

RX receive and process the packet. It mainly set the destination MAC address to the source MAC address before sending the packet into the RX path for processing.

If *rx_pkt_gen_sel* signal in the test bench (LMAC_STREAM_TB.v) equals 1, the test bench is in network packet generation mode where the PHY emulator acts as if packets are pumping in from the network.

6. Self-checking Capability

The AXI4-stream master has the ability to self-check a running test and prints out the results of pass or fail in the ModelSim transcript window.

In loopback mode, the self-checking logic of the AXI4-stream master compares the data it received from the RX path with the loopback data. In network-generated mode, it compares the data it received from the RX path with the data the user provided for the PHY emulator to generate.

If an error is found, it prints the error message and the conditions which caused the mismatch. The user can use this information to assist in the debugging of the design.

7. Other Components of the Test Bench

The test bench provides clock, reset, and configuration information for the LMAC core and other components.

Clocks are generated in the test bench main file: LMAC_STREAM_TB.v. The default clocks from Core3 is 390.62MHz clock rate (312.5MHz for 40Gbps speed and 156MHz for 10Gbps speed) except for the AXI-stream bus which operates at 2x the LMAC core clock rate.

Resets (both system level and PHY reset) are also timed and generated. Resets are negated after 100nS.

Unused signals are tied down or to a known condition in the test bench to avoid propagation of X or unknown conditions.

The bench also sets default values for the core. Some of these are:

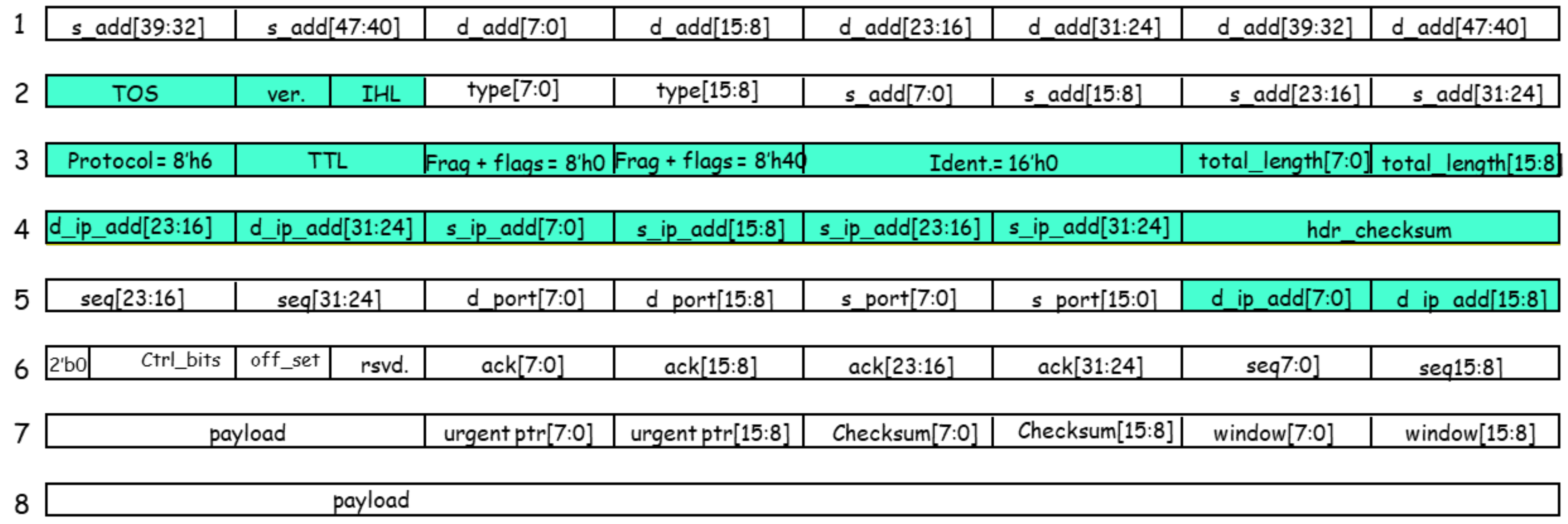
```
fmac_ctrl       =       32'h00000808;        //
fmac_ctrl1      =       32'h000005ee;        //pkt size = 1518 bytes
fmac_rxd_en     =       1'b1;                //enable LMAC RX path
mac_addr0       =       48'h001232FFFFFF;    //set a default MAC address
```

For Core3, the default LMAC speed is set to 100Gbps mode.
User can use the script to change these default values for user specific test.

(Blank Page)

APPENDIX A – Packet Format

Packet format: TCP packet format example



1	s_add[39:32]	s_add[47:40]	d_add[7:0]	d_add[15:8]	d_add[23:16]	d_add[31:24]	d_add[39:32]	d_add[47:40]
2	TOS	ver. IHL	type[7:0]	type[15:8]	s_add[7:0]	s_add[15:8]	s_add[23:16]	s_add[31:24]
3	Protocol = 8'h6	TTL	Frag + flags = 8'h0	Frag + flags = 8'h40	Ident.= 16'h0		total_length[7:0]	total_length[15:8]
4	d_ip_add[23:16]	d_ip_add[31:24]	s_ip_add[7:0]	s_ip_add[15:8]	s_ip_add[23:16]	s_ip_add[31:24]	hdr_checksum	
5	seq[23:16]	seq[31:24]	d_port[7:0]	d_port[15:8]	s_port[7:0]	s_port[15:0]	d_ip_add[7:0]	d_ip_add[15:8]
6	2'b0 Ctrl_bits	off_set rsvd.	ack[7:0]	ack[15:8]	ack[23:16]	ack[31:24]	seq7:0]	seq15:8]
7	payload		urgent ptr[7:0]	urgent ptr[15:8]	Checksum[7:0]	Checksum[15:8]	window[7:0]	window[15:8]
8	payload							

The above is a diagram of the TCP protocol packet. It has similar format as used in tools such as Wireshark
The bytes are in big endian format.
The number 1 to 8 on the very left side are quad-word indexes
The highlighted bytes in color belong to the IP layer
Above it is the Ethernet layer
Below the highlight is TCP layer

D_add = destination MAC address
S_add = source MAC address