Ethernet hardware encapsulator

(xilinx 7 series fpga project)

version 2.1

Vladimir Efimov

https://www.linkedin.com/in/vladimir-efimov

(Dec.2016)

1. PROJECT VERSIONS AND DOCUMENTATION

Version number consists of two numbers presented as [release].[revision].

Any new major feature normally is being implemented in next release of the project. Any minor changes, corrections in the code or documentation as well as bug fixes are reflected in the next revision number of the same release.

All documentation is incremental and relies on the documentation produced for previous releases. In order to access most complete and bug free documentation please rely on the documentation with highest revision number.

In order to implement the current release of the project read all documentation starting from version 1.[highest_rev_number] and go through all incremental changes implemented in all intermediate releases.

2. CHANGES IN THE CURRENT RELEASE OF THE PROJECT

a. Additional module “L23temac2fifo” has been designed and added into “dataplane”. “L23temac2fifo” generates “tuser” signal in case of overflow of FIFO buffer on receive end.

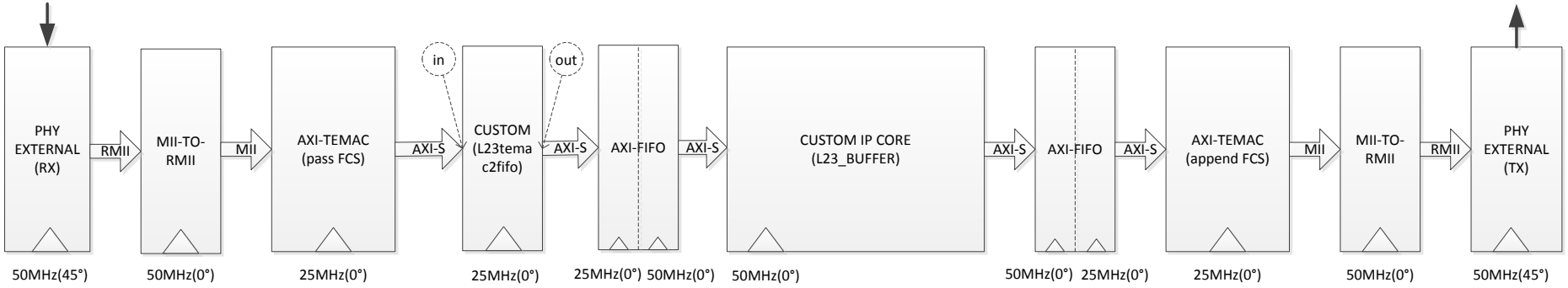
b. Extra functionality has been added to “Ethernet Encapsulator” IP core - “L23_buffer”. Prior sending buffered ethernet frames out, “L23_buffer” IP core prepends/encapsulates them with a header stored in the “header RAM”.

c. Communication interface has been introduced between “dataplane” and “control unit” that allows management of “dataplane” registers and programming “header RAM” via software running on the “control unit”.

d. Demonstration management software has been developed for the “control unit”.

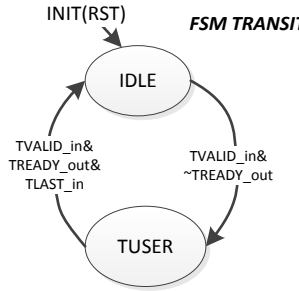
3. INTRUDUCTION OF “L23temac2fifo” IP CORE INTO DATAPLANE

In the current release there is ability to stall “L23_buffer” from “control plane”. When stalled, “L23_buffer” will keep receiving frames but will not send them out until all buffer memory is consumed. That will ultimately cause overflow of receive AXI-FIFO. Because there is no buffering within the receive logic of TEMAC [1] (its AXI-stream master port lacks input “tready” signal) the need for custom “L23temac2fifo” block that will detect AXI-FIFO overflow condition and signal it by means of “tuser” becomes obvious.



Custom “L23temac2fifo” core is injected between TEMAC receive AXI interface (AXI stream master) and receive AXI-FIFO buffer in the “dataplane”.

“L23temac2fifo” IMPLEMENTATION



FSM TRANSITIONS:

SIGNAL ASSIGNS:

TVALID_out = TVALID_in
TREADY_in = TRADY_out
TLAST_out = TLAST_in
TDATA_out = TDATA_in
TUSER_out = (((FSM_STATE == TUSER) && TVALID_in && TRADY_out && TLAST_in) || (TUSER_in))

“L23temac2fifo” FSM transits into “TUSER” state if “AXI-FIFO” is full i.e. (TVALID_in & ~TREADY_out). Then at the end of the received ethernet frame (indicated by high “TLAST_in” signal) “TUSER_out” signal is asserted by FSM and FSM returns to initial “IDLE” state.

4. “L23_buffer” IP CORE DESIGN

In current release “L23_buffer” has been converted from being the plane buffer to the block that performs encapsulation of the buffered frames prior sending buffered frames out via AXI-stream master interface (L23o).

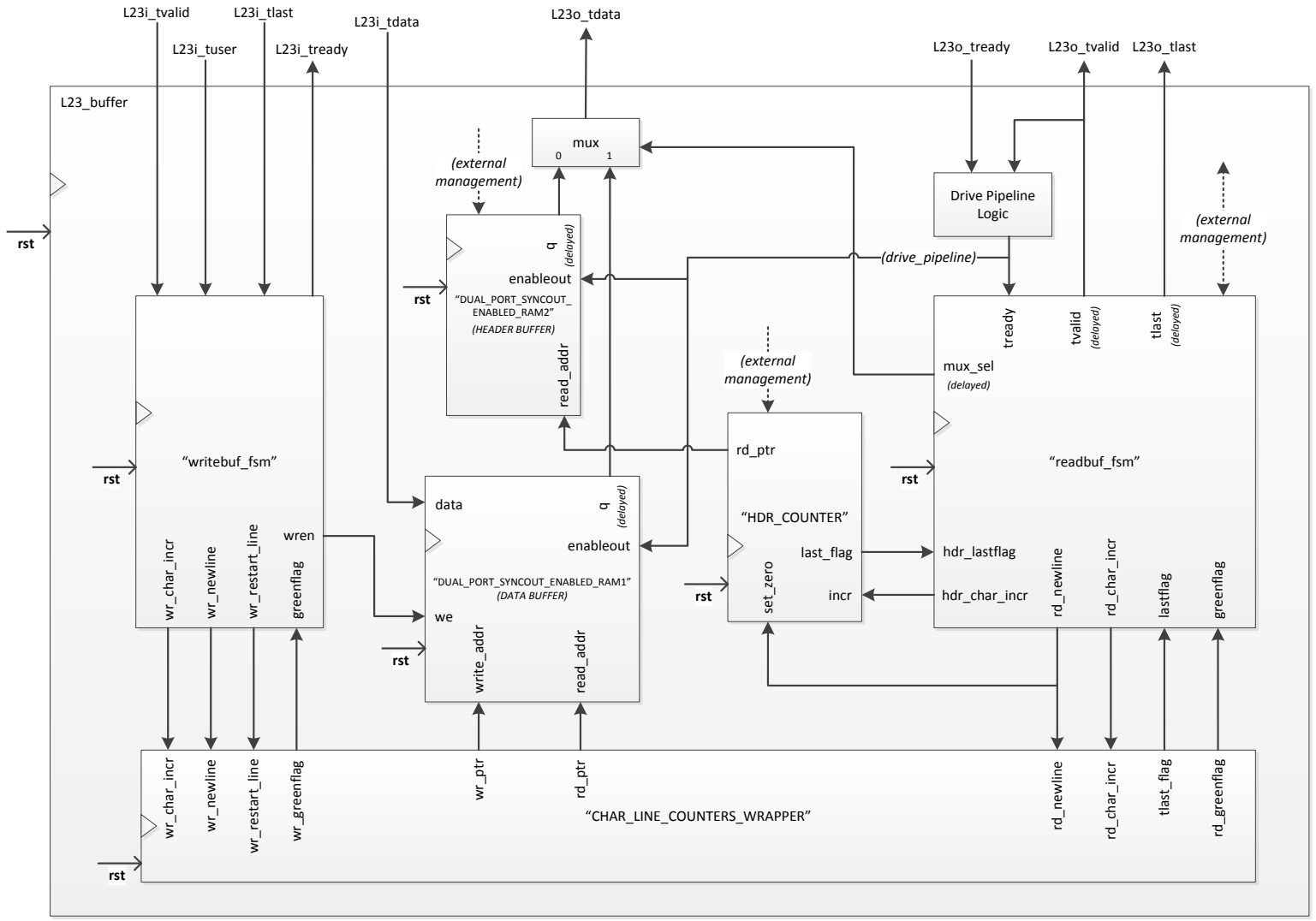
Three additional components have been introduced into “L23_buffer”:

1. header storage buffer - “HEADER BUFFER”,
2. header counter - “HDR_COUNTER”,
3. output multiplexer - “mux”.

The “SIMPLE DUAL PORT BLOCK RAM” that was the only memory used to store the frames in the previous release remains unchanged but called the “data buffer” now. Neither “CHAR_LINE_COUNTERS_WRAPPER” nor “writebuf_fsm” have been changed. Read FSM “readbuf_fsm” has been redesigned in order to send header of the frame from “HEADER BUFFER” followed by the frame body stored in the “data buffer”.

The source of output data is selected by output MUX, controlled by read FSM. Content of “DATA BUFFER” dynamically changes as data passes through the block whereas the content of “HEADER BUFFER” remains fixed. The length of the header (number of bytes from “HEADRE BUFFER” that has to be sent prior a “body” of a frame stored in “DATA_BUFFER”) is also fixed and controlled by “HDR_COUNTER”. Note that “fixed” doesn’t mean that the value is “hardcoded”. It can be reprogrammed by “control unit” via “external management” interfaces of “HDR_COUNTER” and “HEADER BUFFER” but the data passing through the “dataplane” has no effect on its content.

“L23_buffer” module



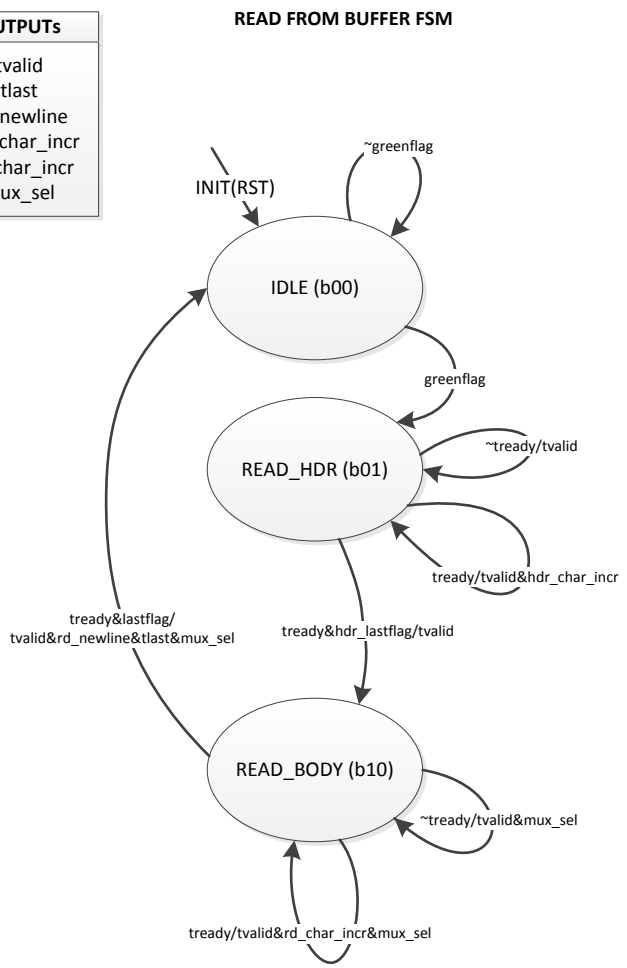
Notes: Drive_pipeline = ~Stall_Pipeline; Stall_Pipeline = (tvalid & ~tready);
Pipeline - one clock delayed outputs from syncronous read BRAM and “readbuf_fsm”.

4.2 “READBUF_FSM” MODULES

Read FSM has been modified in this release as it has to read data from two memory blocks “HEADER” and “DATA” buffers instead of single memory block. To control pointer of “HEADER” memory and select source of the output data two output signals “hdr_char_incr” and “mux_sel” has been added to Read FSM. Also “hdr_lastflag” input signal has been added in addition to “lastflag” that indicate whether the byte being read from the “HEADER” and “DATA” memory is the last byte. As was explained earlier (see the “pipeline” design in the documentation of previous release) the read FSM can be designed as if all memory blocks in the design has asynchronous read.

The Read FSM depicted on the right is implemented in “readbuf_fsm” module. Within that module FSM is complemented with additional delay logic that makes sideband signals aligned with delayed output from RAMs and “external management” signals which are used to stall the FSM in “IDLE” state or to obtain current state of the FSM.

INPUTs	OUTPUTs
greenflag hdr_lastflag lastflag tready	tvalid tlast rd_newline hdr_char_incr rd_char_incr mux_sel



5. Management of “dataplane” from “control unit”

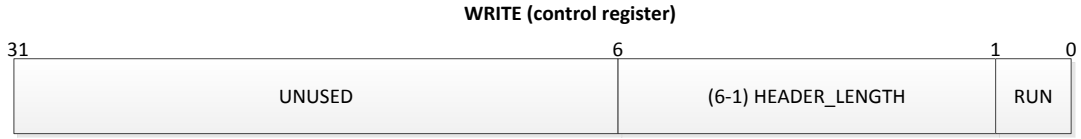
Although the “dataplane” is comprised of number of IP cores the only “manageable” IP core which behaviour can be managed from the “control plane” is “L23_buffer”. In “L23_buffer” IP core there are two DUAL PORT RAM blocks. “DATA BUFFER” DUAL PORT RAM block has read and write ports and both are used for processing (temporary storing) of “dataplane” traffic. The only read port of “HEADER”DUAL PORT RAM is used for processing (header supplying) of “dataplane” traffic whereas write port is used for programming the “HEADER” RAM with “fixed” header content and therefore marked as “management” on “L23_buffer” module diagram depicted earlier. This “management” write port is connected to the “AXI BRAM controller” IP block that converts conventional BRAM memory interface to “AXI-Lite” interface that allows connectivity with the “control plane”.

In addition to the “HEADER” RAM write port there are two other management ports within “L23_buffer” for “HDR_COUNTER” and “readbuf_fsm”. These two management ports are connected to “AXI GPIO” IP block that has “AXI-Lite” interface in order to communicate with the “control plane”.

The “control plane” can control the behaviour of the “L23_buffer” by reading and modifying registers of “AXI GPIO” block as well as reprograming content of the “HEADER” RAM using the following addresses:

“Control/Status” register

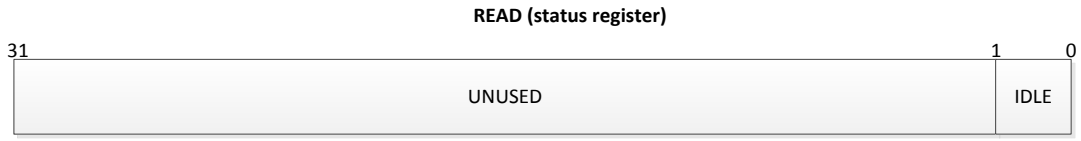
Write address: 0x40000000



RUN (1bit): Run/stop “readbuf_fsm”, (1-running, 0-stopped);

HEADER_LENGTH (6bits): “Real_header_length” = “configured_header_length” + 1;

Read address: 0x40000000



IDLE (1bit): Idle/running “readbuf_fsm”, (1-idle, 0-running);

“HEADER” RAM

Write Base address: 0xc2000000 (access word aligned): “HEADER” content;

XMD Debug Example (prepend packets with “0xa0,0xa1,0xa2,0xa3” header)

XMD% mwr 0x40000000 0x0 /*Stall Read FSM*/

XMD% mrd 0x40000000 /*Check Read FSM status*/

40000000: 00000001 /*1 means that Read FSM status is IDLE*/

XMD% mwr 0xc2000000 0xa0 /*Write 1-st byte value ‘0xa0’ in “HEADER_RAM”*/

XMD% mwr 0xc2000004 0xa1 /*Write 2-nd byte value ‘0xa1’ in “HEADER_RAM”*/

XMD% mwr 0xc2000008 0xa2 /*Write 3-rd byte value ‘0xa2’ in “HEADER_RAM”*/

XMD% mwr 0xc200000c 0xa3 /*Write 4-th byte value ‘0xa3’ in “HEADER_RAM”*/

XMD% mwr 0x40000000 0x7 /*Write “header_length” value to make HEADER_LENGTH = 4 and run Read FSM*/

6. Demonstartion software for “control unit”

To troubleshoot communication between “dataplane” and “control unit” XMD has been used. In order to have independently working system that has no reliance on Xilinx tools, management software has also been developed. This software allows the user to interact with the system via rs232 port and depending on the user input some pre-defined procedures are being executed such as stall of the “dataplane”, request “dataplane” status or reprograming of “HEADER” RAM. This software is mainly intended for demonstration purposes but can also be used as basic example from which more complex software or operating system drivers can be derived.

7. “L23_buffer” IP CORE VERIFICATION

The “SrcArray” contains input data which is supplied to the input of the “L23_buffer” core. Input data is driven by “tvalid” signal randomly generated by the testbench.

//L23mgmt_refvalue = 6'b000010; / Real_header_length=3; value is supplied by testbench to “L23_buffer” DUT*/*

//L23mgmt_data = {8'b11000000, 8'b10000000, 8'b01000000}; / “c0, 80, 40” – testbench programs memory of “L23_buffer” DUT with these values */*

//SrcArray bits:[9]-tuser, [8]-tlast, [7:0]-tdata;
reg [9:0] SrcArray [0:30] =

//1st packet will be accepted
(10'h011,10'h012,10'h013,10'h014,10'h015,10'h016,10'h117,

//2nd packet will be accepted
10'h021,10'h022,10'h023,10'h024,10'h025,10'h026,10'h027,10'h128,

//3rd packet will be discarded because the last byte of the packet has active “tuser” flag – 10’h337
10'h031,10'h032,10'h033,10'h034,10'h035,10'h036,10'h337,

//4th packet will be accepted
10'h041,10'h042,10'h043,10'h044,10'h045,10'h046,10'h047,10'h048,10'h149);

Testbench output:

```
# run 5000ns
@170 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:11; L23i_tlast:0; L23i_tuser:0;
@190 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:12; L23i_tlast:0; L23i_tuser:0;
@210 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:13; L23i_tlast:0; L23i_tuser:0;
@230 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:14; L23i_tlast:0; L23i_tuser:0;
@270 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:15; L23i_tlast:0; L23i_tuser:0;
@290 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:16; L23i_tlast:0; L23i_tuser:0;
@310 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:17; L23i_tlast:1; L23i_tuser:0;
@390 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:40; L23o_tlast:0;
@410 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:80; L23o_tlast:0;
@430 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:21; L23i_tlast:0; L23i_tuser:0;
@430 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:c0; L23o_tlast:0;
@450 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:22; L23i_tlast:0; L23i_tuser:0;
@490 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:11; L23o_tlast:0;
@510 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:12; L23o_tlast:0;
@530 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:23; L23i_tlast:0; L23i_tuser:0;
@530 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:13; L23o_tlast:0;
@550 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:24; L23i_tlast:0; L23i_tuser:0;
@550 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:14; L23o_tlast:0;
@590 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:25; L23i_tlast:0; L23i_tuser:0;
@590 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:15; L23o_tlast:0;
@610 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:26; L23i_tlast:0; L23i_tuser:0;
@670 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:27; L23i_tlast:0; L23i_tuser:0;
@690 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:16; L23o_tlast:0;
@710 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:28; L23i_tlast:1; L23i_tuser:0;
@770 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:31; L23i_tlast:0; L23i_tuser:0;
@770 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:17; L23o_tlast:1;
@790 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:32; L23i_tlast:0; L23i_tuser:0;
@790 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:40; L23o_tlast:0;
@810 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:33; L23i_tlast:0; L23i_tuser:0;
@830 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:80; L23o_tlast:0;
@850 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:34; L23i_tlast:0; L23i_tuser:0;
@850 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:c0; L23o_tlast:0;
@890 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:35; L23i_tlast:0; L23i_tuser:0;
@890 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:21; L23o_tlast:0;
@910 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:22; L23o_tlast:0;
@930 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:36; L23i_tlast:0; L23i_tuser:0;
@950 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:37; L23i_tlast:1; L23i_tuser:1;
@950 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:23; L23o_tlast:0;
@990 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:41; L23i_tlast:0; L23i_tuser:0;
@990 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:24; L23o_tlast:0;
@1050 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:42; L23i_tlast:0; L23i_tuser:0;
@1050 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:25; L23o_tlast:0;
@1090 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:43; L23i_tlast:0; L23i_tuser:0;
@1090 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:26; L23o_tlast:0;
@1110 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:44; L23i_tlast:0; L23i_tuser:0;
@1110 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:27; L23o_tlast:0;
@1130 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:45; L23i_tlast:0; L23i_tuser:0;
@1150 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:46; L23i_tlast:0; L23i_tuser:0;
@1150 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:28; L23o_tlast:1;
@1170 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:47; L23i_tlast:0; L23i_tuser:0;
@1250 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:48; L23i_tlast:0; L23i_tuser:0;
@1270 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:49; L23i_tlast:1; L23i_tuser:0;
@1310 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1350 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1350 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:40; L23o_tlast:0;
@1370 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:80; L23o_tlast:0;
@1390 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1390 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:c0; L23o_tlast:0;
@1410 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1450 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1490 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1490 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:41; L23o_tlast:0;
@1530 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1570 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1590 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:42; L23o_tlast:0;
@1610 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:43; L23o_tlast:0;
@1630 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1630 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:44; L23o_tlast:0;
@1650 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1650 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:45; L23o_tlast:0;
@1690 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:46; L23o_tlast:0;
@1730 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1730 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:47; L23o_tlast:0;
@1770 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:48; L23o_tlast:0;
@1790 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1830 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1830 OUTPUT: L23o_tvalid:1; L23o_tready:1; L23o_tdata:49; L23o_tlast:1;
@1910 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1950 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@1970 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@2030 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@2090 INPUT: L23i_tvalid:1; L23i_tready:1; L23i_tdata:xx; L23i_tlast:x; L23i_tuser:x;
@2110 Finished posedge process loop
@2120 Finished negedge process loop
```

We can see packets 1,2 and 4 at the output of the “L23_buffer” with prepended 3 bytes header (0x40; 0x80; 0xc0) that has been written into “header buffer” by the testbench prior streaming simulation. Packet 3 has been discarded because of the “tlast+tuser” bit. Output data is “throttled” by “tready” signal randomly generated by the testbench.

REFERENCES

[1] Tri-Mode Ethernet MAC v9.0, LogiCORE IP Product Guide, Vivado Design Suite, PG051 October 5, 201, p87 “Normal Frame Reception”.