Design Gateway Co.,Ltd

54 BB Building 14th Fl., Room No.1402
Sukhumvit 21 Rd. (Asoke), Klongtoey-Nua,
Wattana, Bangkok 10110
Phone: 66(0)2-261-2277
Fax: 66(0)2-261-2290
E-mail: ip-sales@design-gateway.com
URL: www.design-gateway.com

Core Facts

Provided with Core	
Documentation	User Guide, Design Guide
Design File Formats	Encrypted HDL
Verification	Test Bench, Simulation Library
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on KCU105
Simulation Tool Used	
ModelSim	
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- 10 Gbps Ethernet
- 64-bit XGMII interface running at 156.25 MHz
- Same clock domain for transmit and receive XGMII interface
- AXI4-stream protocol for connecting with the user logic (TOE10G IP or UDP10G IP)
- Low transmit and receive latencies (3 clock cycles for transmit side, 7 clock cycles fo receive side)
- FCS inserting (CRC-32) to the transmit packet
- FCS checking (CRC-32) for the received packet
- Not append the zero padding in the transmit packet
- Not remove the zero padding from the received packet
- Support the 1st byte data on byte0 (bit[7:0]) or byte4 (bit[39:32]) of receive XGMII interface

Table 1: Example Implementation Statistics for 7-Series device

Family	Example Device	Fmax (MHz)	Slice Regs	Slice LUTs	Slices[1]	IOB	BRAMTile	Design Tools
Kintex-7	XC7K325TFFG900-2	156.25	1107	1885	547	-	-	Vivado2017.4
Zynq-7000	XC7Z045FFG900-2	156.25	1107	1885	543	-	-	Vivado2017.4
Virtex-7	XC7VX485TFFG1761-2	156.25	1107	1886	557	-	-	Vivado2017.4

Table 2: Example Implementation Statistics for Ultrascale device

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB[1]	IOB	BRAMTile	Design Tools
Kintex-Ultrascale	XCKU040FFVA1156-2E	156.25	1072	1873	326	-	-	Vivado2017.4
Zynq-Ultrascale+	XCZU9EG-FFVB1156-2-I	156.25	1072	1873	333	-	-	Vivado2017.4
Virtex-Ultrascale+	XCVU6P-FLGA2104-2L	156.25	1072	1880	311	-	-	Vivado2017.4

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

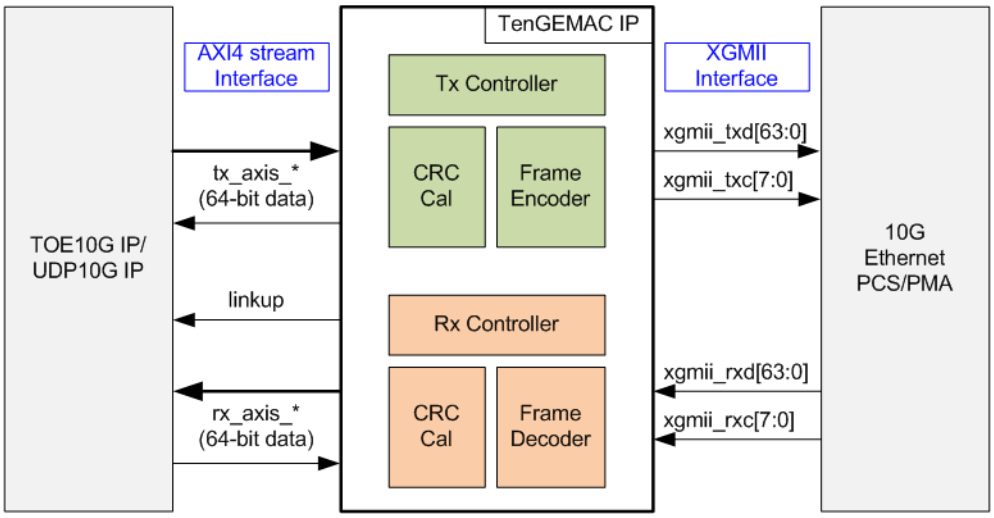

Figure 1: TenGEMAC IP Block Diagram

Applications

By using DG Network IP suite (TOE10G IP/UDP10G IP and TenGEMACIP), the system implemented on FPGA can transfer Ethernet packet via 10Gb Ethernet following TCP/IP protocol or UDP/IP protocol with good performance, low data latency, and small resource utilization. It is the solution to implement Ethernet streaming and monitoring system without CPU and external memory usage.

General Description

TenGEMAC IP implements the MAC layer for 10Gb Ethernet solution. Before transmitting the packet from AXI4 stream interface (TOE10G IP or UDP10G IP) to XGMII interface (10G Ethernet PCS/PMA), the preamble, SFD, and FCS are appended as the packet header by TenGEMAC IP. After end of packet transmission, the interframe gap is appended as the footer to be the gap size for each packet.

On the other hand, the received packet from XGMII interface (10G Ethernet PCS/PMA) is verified by TenGEMAC IP. The preamble, SFD, and FCS are removed from the received packet before forwarding to AXI4 stream interface. If the FCS is not correct, the IP will assert the error signal to AXI4 stream interface.

Data input stream between start of frame and end of frame on AXI4 stream interface must be valid every clock because TenGEMAC IP does not include Tx buffer to pause data transmission to 10G Ethernet PHY (PCS/PMA). Similar to transmit path, the received packet from 10G Ethernet PHY is forwarded to AXI4 stream interface continuously because there is no internal buffer of received path in TenGEMAC IP.

TenGEMAC IP does not support zero padding function, so the user logic on AXI4 stream interface must append the padding by itself for the small packet which is less than 60 byte size. TenGEMAC IP supports to receive the packet from XGMII interface when the 1st byte is placed on byte0 (xgmii_rxd[7:0]) and byte 4 (xgmii_rxd[39:32]) only. Otherwise, the received packet is ignored.

Functional Description

TenGEMAC IP is designed to support data transmission in both directions at the same time. So, the logic to transmit data and receive data is run independently.

Transmit Block

- **CRC Cal**

This module is designed to calculate 32 bit CRC from 64-bit data, input from AXI4 stream interface. The polynomial of CRC-32 to create FCS following IEEE802.3 standard is as follows.

$P(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X^{1} + 1$

FCS is appended to the packet as the footer after finishing forwarding all data.

- **Frame Encoder**

This module inserts the header and the footer to the transmit packet from AXI4 stream interface. 7-byte preamble and 1-byte SFD (Start of frame delimiter) are appended as the header of the packet while 4-byte FCS and 1-byte EFD (End of frame delimiter) are appended as the footer of the packet. After finishing one frame, 12-Idle cycle is inserted to be IFG (interframe gap) between two packets.

- **Tx Controller**

The controller monitors the control signals on AXI stream interface. When new frame is found, it will assert the control signal to the Frame encoder for inserting the header and the footer into the frame. After that, the controller deasserts ready signal on AXI stream interface for 3 clock cycles to finish FCS, EFD, and interframe gap insertion process.

Received Block

- **CRC Cal**

This module is the same module as CRC Cal in Transmit Block. The FCS output from this module is applied to verify the received packet. The error will be asserted on Rx AXI4 stream interface if the FCS in the received packet is not correct.

- **Frame Decoder**

The data and control signals from XGMII interface are decoded by this module to check link up status, start of frame, and end of frame. The data type output from the decoder is forwarded to Rx controller to validate the data sequence in the packet. The received packet removing the header and the footer is rearranged to align the 1st byte data of the packet on byte0 only. So, the 1st byte data on AXI4 stream interface is always on rx_axis_tdata[7:0].

- **Rx Controller**

The controller checks SFD and FCS in the received packet. The controller asserts the data valid to AXI4 stream interface when the packet header is correct. The error is asserted by the controller when the FCS of the packet is not correct.

TOE10G IP/UDP10G IP

TOE10G IP implements TCP/IP stack and the offload engine while UDP10G IP implements UDP/IP stack and the offload engine. More details of the IP are described in the datasheet.

https://dgway.com/products/IP/TOE10G-IP/dg_toe10gip_data_sheet_xilinx_en.pdf

https://dgway.com/products/IP/UDP10G-IP/dg_udp10gip_data_sheet_xilinx_en.pdf

10 Gigabit Ethernet PCS/PMA (10GBASE-R)

10 Gigabit Ethernet PCS/PMA (10GBASE-R) IP core is no charge Xilinx IP core. XGMII is the interface with EMAC. It implements 10.3125 Gbps PHY which provides a direct connection to SFP+ optical module. For more details, please download the IP datasheet from the following link.

https://www.xilinx.com/products/intellectual-property/10gbase-r.html

Core I/O Signals

Descriptions of all signal I/Os are provided in Table 2.

Table 2: Core I/O Signals

Signal	Dir	Description
RstB	In	Reset IP core. Active Low.
Clk	In	156.25 MHz fixed clock frequency input, synchronous to XGMII interface.
Linkup	Out	Assert to '1' when Rx XGMII sends idle code (PHY initialization is finished).
AXI4 stream interface		
tx_axis_tdata[63:0]	In	Transmitted data to AXI4 stream interface. Valid when tx_axis_tvalid is asserted to '1'.
tx_axis_tkeep[7:0]	In	Byte enable of 64-bit tx_axi_tdata. One bit is asserted to '1' when each byte is valid. Bit[0] for tdata[7:0], [1] for tdata[15:8], and so on. This signal must be equal to 0xFF to enable every data in the packet, except the last word which the data may be valid only some bytes. The byte enable of the last byte has 8 values, i.e. 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, and 0xFF when last data is valid for 1 byte – 8 byte. The signal is valid when tx_axis_tvalid is asserted to '1'.
tx_axis_tvalid	In	Transmitted data valid signal. Assert to '1' when tx_axis_tdata/tkeep are valid.
tx_axis_tlast	In	Assert to '1' to indicate the final word in the frame. Valid when tx_axis_tvalid is asserted to '1'.
tx_axis_tready	Out	Handshaking signal. Asserted to '1' when tx_axis_tdata has been accepted. This signal is not de-asserted to '0' when the packet is transmitting.
rx_axis_tdata[63:0]	Out	Received data. Valid when rx_axis_tvalid is asserted to '1'.
rx_axis_tkeep[7:0]	Out	Received data byte enable. One bit is asserted to '1' when each byte is valid. Bit[0] for tdata[7:0], [1] for tdata[15:8], and so on. The signal is valid when rx_axis_tvalid is asserted to '1'.
rx_axis_tvalid	Out	Received data valid signal. Assert to '1' when rx_axis_tdata/tkeep are valid.
rx_axis_tlast	Out	Assert to '1' to indicate the final word in the frame. Valid when rx_axis_tvalid is asserted to '1'.
rx_axis_tuser	Out	Asserted at the end of the frame to indicate that the frame has an error. '1': normal packet, '0': error packet. Valid when rx_axis_tlast and rx_axis_tvalid are asserted to '1'.
XGMII interface		
xgmii_txd[63:0]	Out	Transmit data to PHY
xgmii_txc[7:0]	Out	Transmit control to PHY
xgmii_rxd[63:0]	In	Received data from PHY
xgmii_rxc[7:0]	In	Received control from PHY

Timing Diagram

IP Initialization

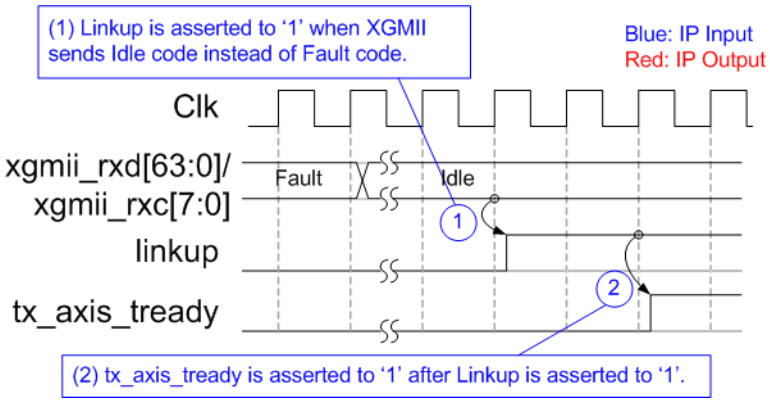

Figure 2: Linkup status

After finishing reset sequence, the IP monitors the received data from XGMII interface. When PHY is not ready, Fault code is returned from XGMII interface. After PHY finishes the initialization, XGMII sends Idle code instead of Fault code. After that, the IP changes linkup to '1' and then tx_axis_tready is asserted to '1'. As a result, the IP is ready to transfer the packet between AXI4 stream interface and XGMII interface.

Transmit interface

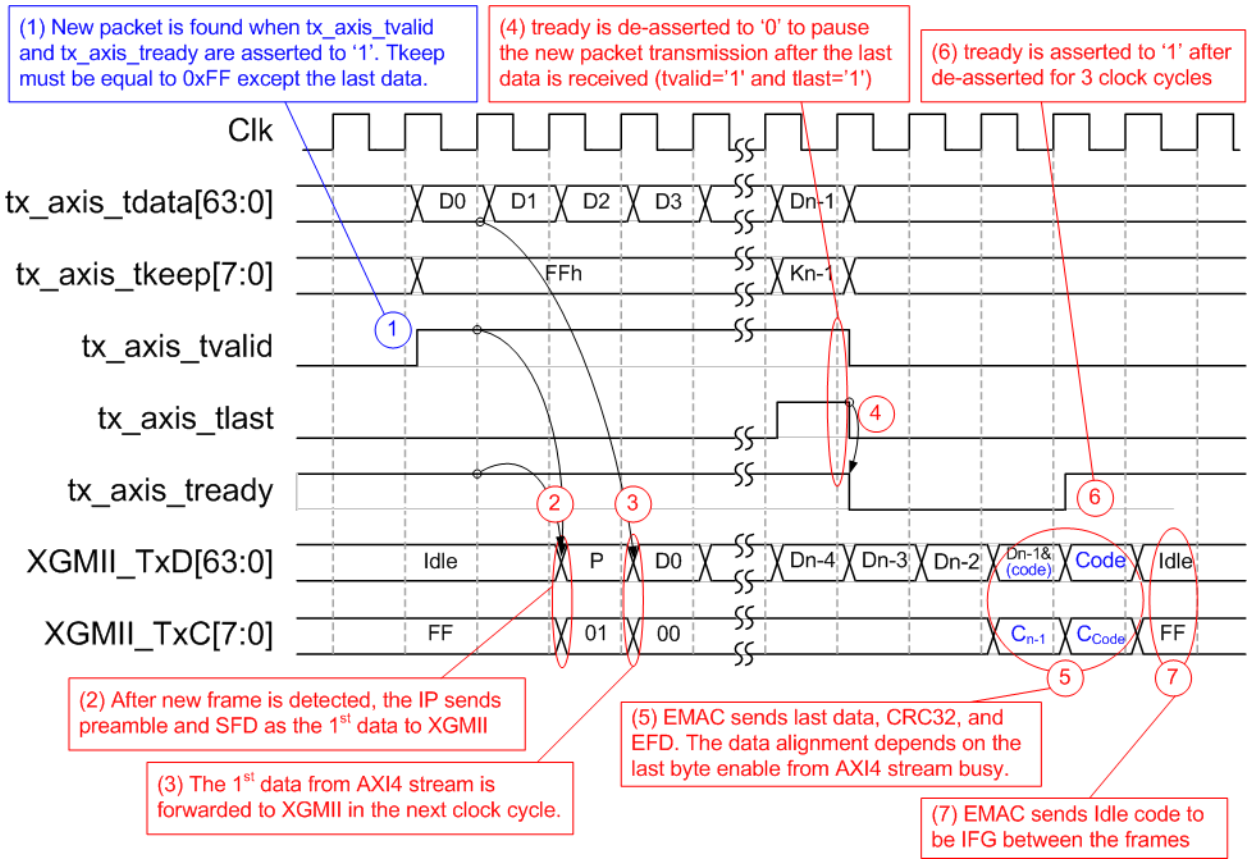

Figure 3: Transmit packet timing diagram

The example to send the data stream from AXI4 stream to XGMII is shown in Figure 3.

(1) The new packet is found when tx_axis_tvalid is asserted to '1' with the IP ready status (tx_axis_tready='1'). Tkeep input from all data must be equal to 0xFF because all 64-byte data from AXI4 stream must be valid, except the last data. The last byte enable depends on the number of valid bytes of the last data on AXI4 stream interface.

(2) The IP starts the operation to calculate FCS (CRC-32) and appends 7-byte preamble and SFD to the packet as the 1st data.

(3) The IP forwards the 1st data from AXI4 stream to XGMII in the 2nd clock. TxC value during data forwarding is always equal to 0x00, except the last data which depends on tkeep input at the end of frame.

(4) After the last data is found (detected by tvalid='1' and tlast='1'), the IP ready (tx_axis_tready) is de-asserted to '0' for 3 clock cycles. During the gap size, the IP inserts 4-byte FCS, EFD, and 12-byte Idle for IFG. The gap size is also applied to compensate the time usage for inserting 8-byte header (7-byte preamble and SFD).

(5) The last data from AXI4 stream is forwarded to XGMII. If the last data is not aligned to 8, the IP will append the footer (4-byte FCS, EFD, and 12-byte Idle) to the remaining byte. The remaining footer is transmitted to XGMII in the next clock.

(6) The IP asserts tready to '1' after de-asserting for 3 clock cycles.

(7) The IP sends at least 12-byte Idle code and waits the new packet.

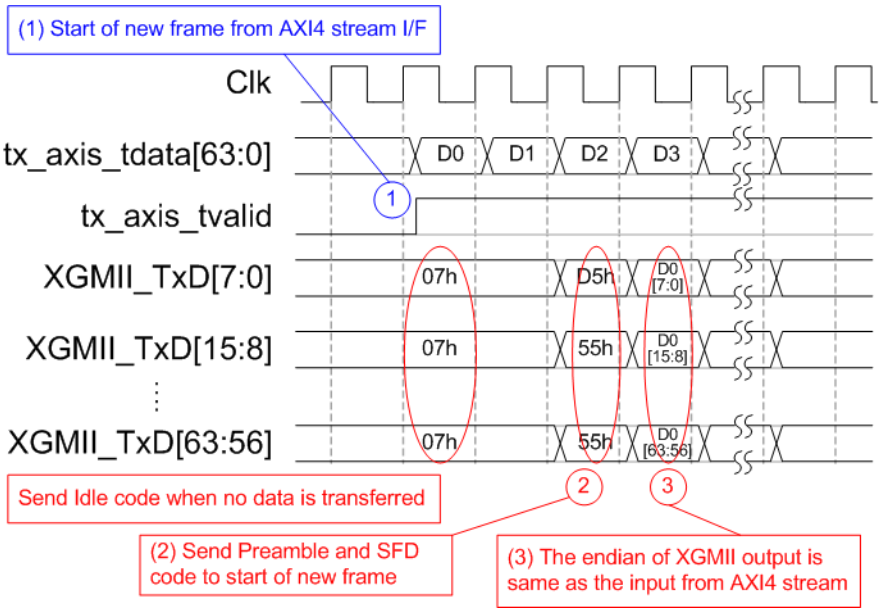

Figure 4: Transmit packet endian

Figure 4 shows more details of the endian relation between the data input from AXI4 stream interface and the data output to XGMII interface. The endian of the data does not change. After sending Idle code, Preamble, and SFD, byte0 of the 1st data (D0[7:0]) is placed on XGMII_TxD[7:0], byte1 on XGMII_TxD[15:8], and so on.

Received Interface

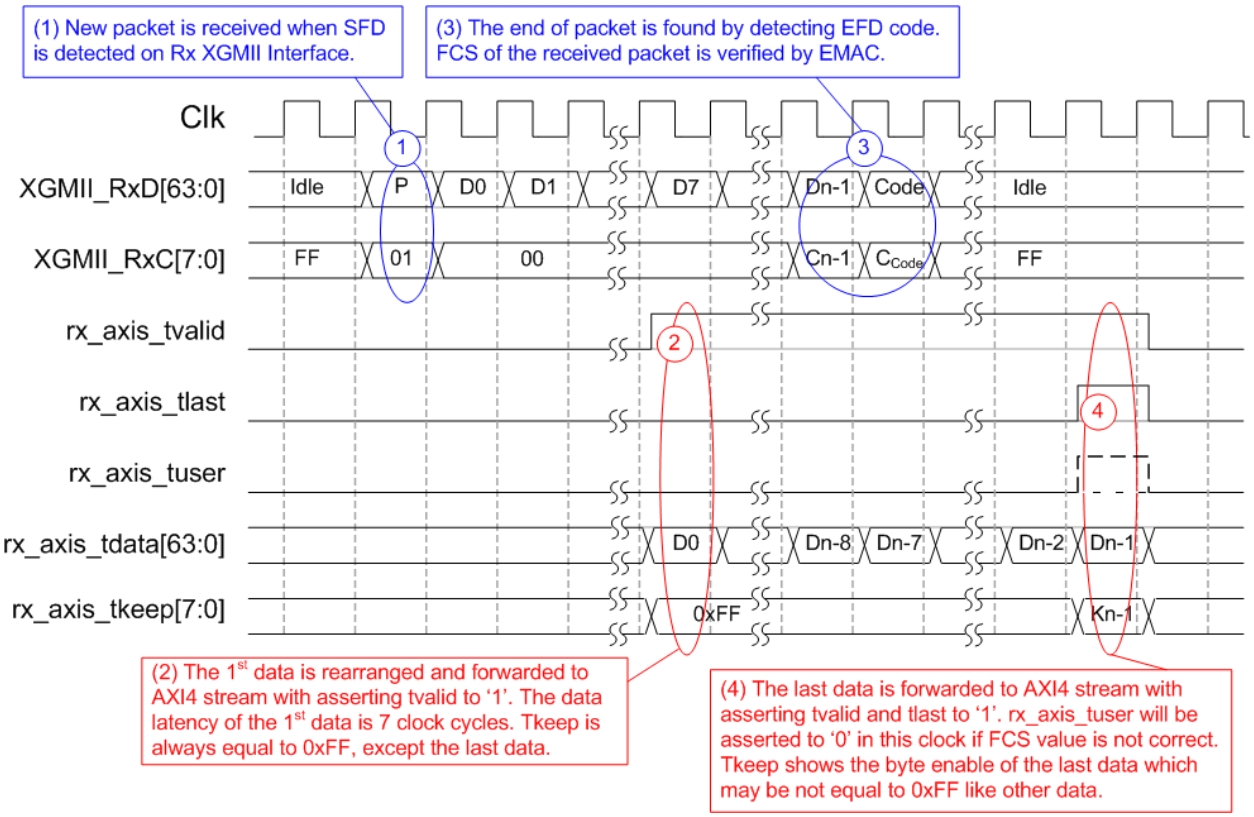

Figure 5: Received packet timing diagram when the 1st data is on byte0

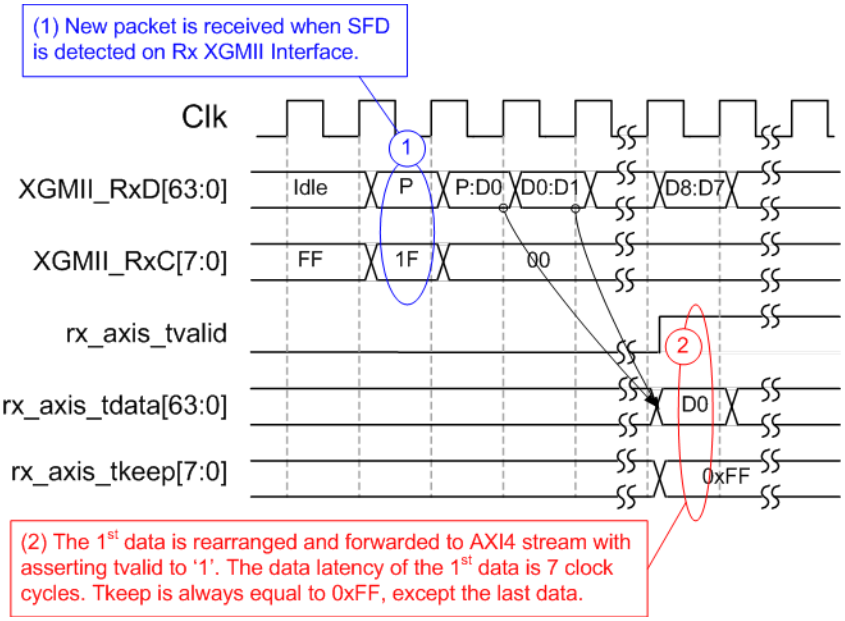

Figure 6: Received packet timing diagram when the 1st data is on byte4

The examples to receive the data stream from XGMII to AXI4 stream are shown in Figure 5 and Figure 6. Two examples are displayed to show the different format of the 1st byte data which may be placed on byte0 or byte4. Though the 1st data is different, the latency of the output (measured from the 1st data) in both examples are same (7 clock cycles). The details of timing diagram is as follows.

(1) The IP detects SFD code on XGMII interface. The IP rearranges the received packet following the position of the SFD code. The 1st data is placed on the data bus in the next clock which may be valid for 32-bit or 64-bit.
(2) After receiving the 1st data for 7 clock cycles, the IP sends the 1st data which is reformatted to place the 1st byte on byte0 only. Tkeep is always equal to 0xFF, except the last data which depends on the number of valid bytes.
(3) The IP detects EFD code on XGMII interface. FCS is loaded to compare with the calculated FCS value.
(4) After that, the IP asserts rx_axis_tlast and tvalid to '1' with the last data on tdata bus. In this clock, rx_axis_tuser will be asserted to '1' if the received FCS is equal to the calculated FCS. Otherwise, tuser is de-asserted to '0'.

After the IP sends the last data, rx_axis_tvalid is de-asserted to '0' at least 2 clock cycles. The gap size of tvalid is found because preamble, SFD, EFD, and IFG on XGMII interface are removed by EMAC IP.

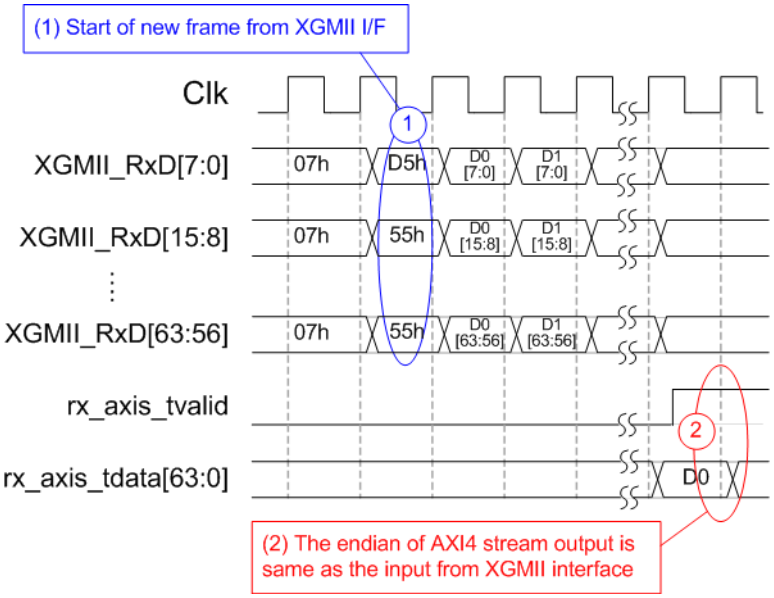

Figure 7: Received packet endian

Similar to Tx path, the endian of the data input from XGMII interface to the data output of AXI4 stream interface does not change.

Verification Methods

The TenGEMAC IP Core functionality was verified by simulation and also proved on real board design by using KCU105 evaluation board.

Recommended Design Experience

User must be familiar with HDL design methodology to integrate this IP into the design.

Ordering Information

This product is available directly from Design Gateway Co., Ltd. Please contact Design Gatway Co., Ltd. for pricing and additional information about this product using the contact information on the front page of this datasheet.

Revision History

Revision	Date	Description
1.0	Jul-3-2019	New release