LL10GEMAC with AAT Demo reference design

Rev1.0 7-Dec-21

1 Introduction

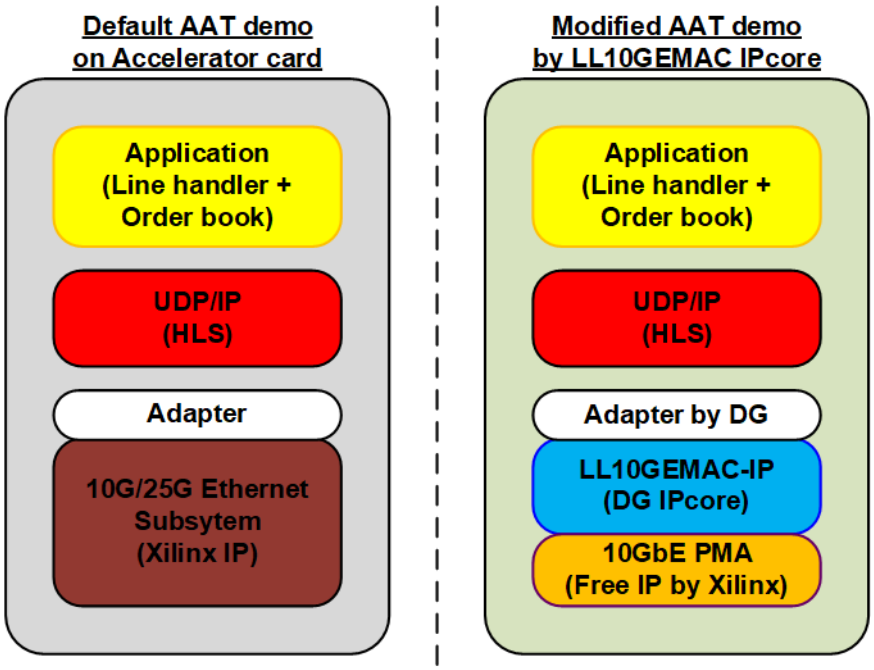

Figure 1-1 The comparison of modified AAT demo and default AAT demo

There is Accelerated Algorithmic Trading (AAT) demo provided by Xilinx to run on Accelerator card for the trading application which needs low latency solution. AAT system is developed to reduce the CPU utilization and the latency of market data processing system. More details of AAT demo can be requested from following link.
https://www.xilinx.com/applications/data-center/financial-technology/accelerated-algorithmic-trading.html

The default AAT demo uses 10G/25G Ethernet Subsystem (Xilinx IP core), configured for low-latency application. The user interface of Ethernet Subsystem is 32-bit data interface at 312.5 MHz. The Ethernet kernel interface is 64-bit interface at the application clock which the frequency is configured by the application system. Therefore, the adapter logic must be designed to interface 10G/25G Ethernet Subsystem with the application.

Similarly, LL10GEMAC-IP which is Design Gateway IP core uses 32-bit interface at 322.266 MHz. The adapter logic is also designed by HDL to transfer packet between the application and LL10GEMAC-IP in both transfer directions with low latency time. By using LL10GEMAC-IP and the new adapter, the modified AAT demo achieves the lower latency time than the default AAT demo. The PMA logic for 10GBASE-R to connect with LL10GEMAC-IP is free provided by Xilinx.

The applications of AAT demo such as UDP/IP, line handler, and order book are coded by HLS. Therefore, it is flexible for user modifying to compatible to each market data standard. The details about system overview, hardware components, and interface are described as follows.

2 Hardware overview

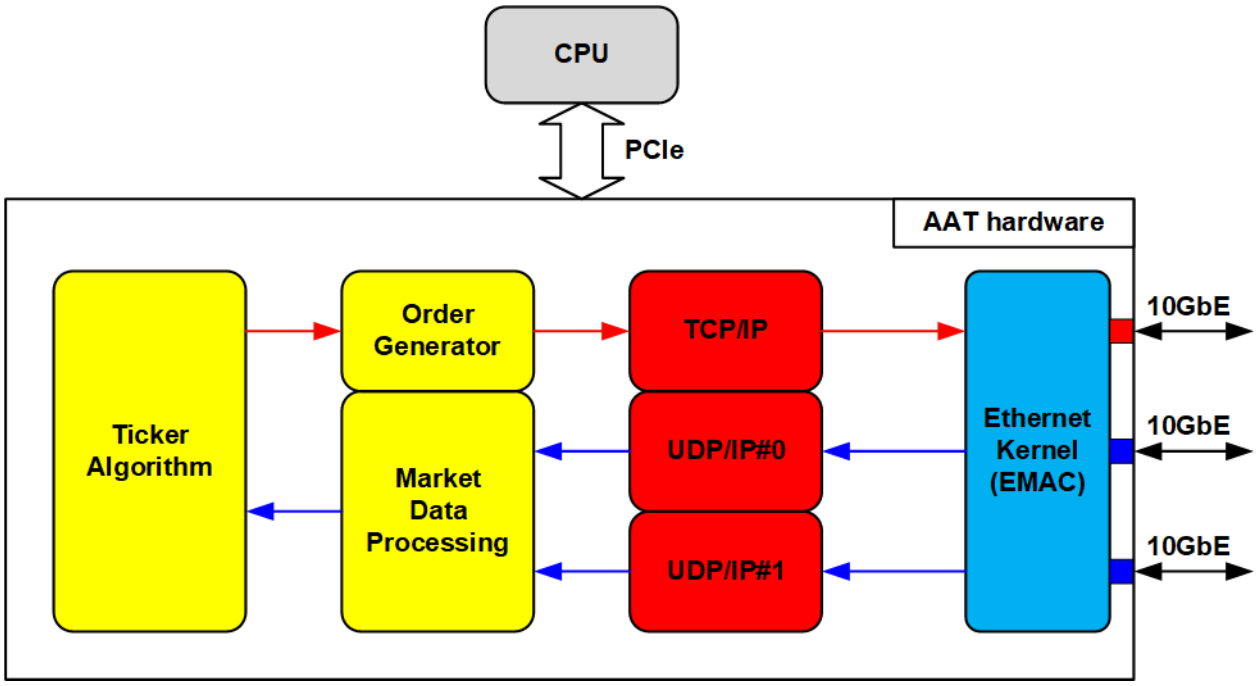

Figure 2-1 AAT hardware block diagram

As shown in Figure 1-1, AAT hardware implemented on Accelerator card consists of many blocks. Two 10Gb Ethernet channels are applied to receive market data by using UDP/IP protocols. Thus, the output of two EMACs are connected to UDP/IP#0 and #1 for decoding UDP/IP packet. After processing Market data and ticker is found, the order is generated by Order Generator. The order packet uses TCP/IP protocol which is reliable protocol, so TCP/IP block is applied to create order packet. The TCP/IP packet is transferred by using the third 10Gb Ethernet channel.

This reference design modifies Ethernet kernel by replacing LL10GEMAC-IP while the other modules are not modified. The software that is run on CPU is the default application. Thus, this document describes only the modification part of AAT demo which is Ethernet kernel hardware.

3 Ethernet Kernel

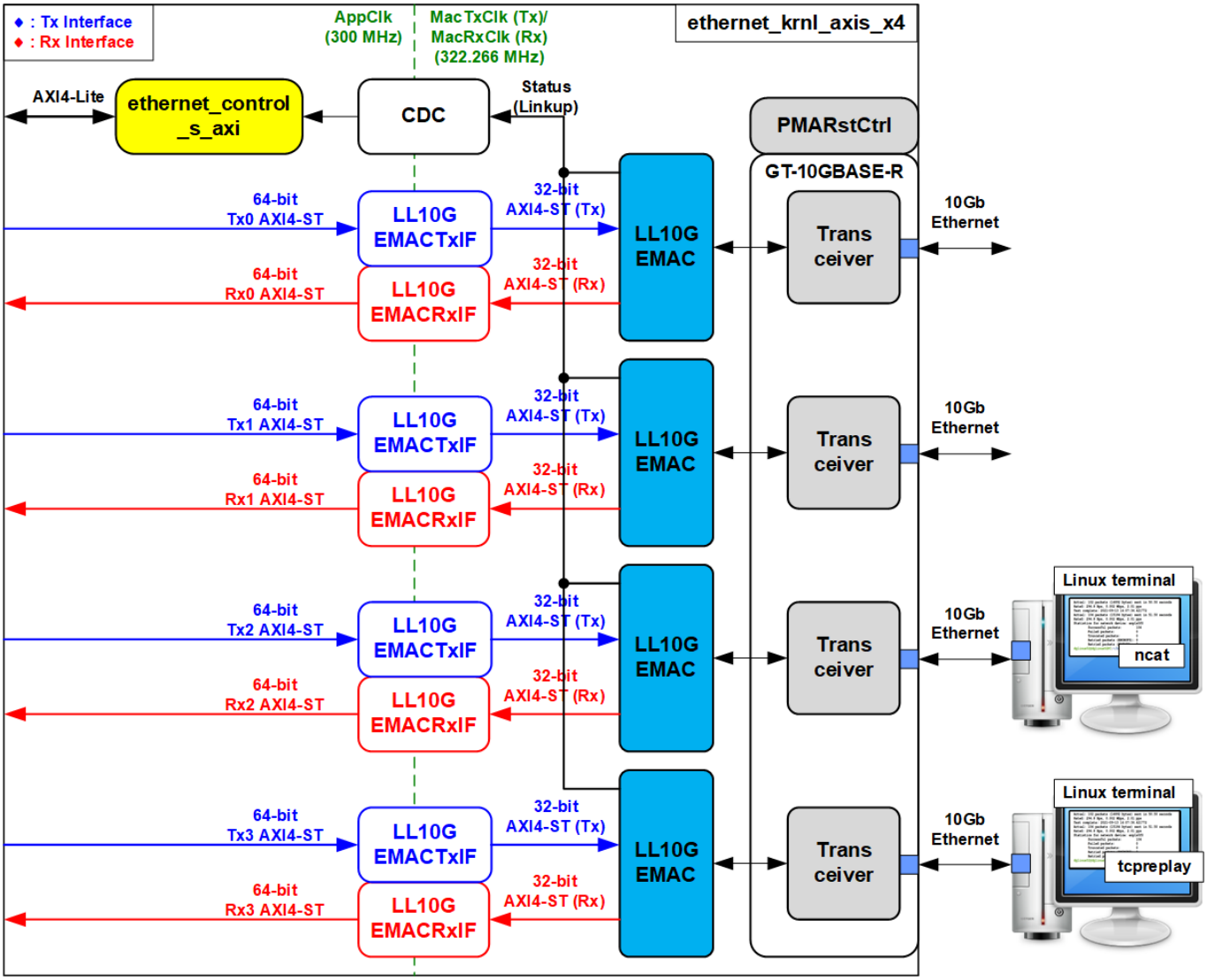

Figure 3-1 Ethernet kernel block diagram

Ethernet kernel is modified from Ethernet hardware kernel of the Accelerated Algorithmic Trading system by integrating LL10GEMAC-IP and modifying LL10GEMACTxIF/LL10GEMACRxIF to reduce latency time. To achieve the Vitis kernel interface requirements, AXI4-Lite is applied to be the control and status interface while AXI4-ST is applied to be data interface. The Accelerator card uses QSFP28 connector which can be split to four channels of 10Gb Ethernet for network connection. Thus, Ethernet kernel integrates four sets of EMAC interface, LL10GEMAC-IP, and Transceiver (configured to 10GBASE-R PMA). Kernel interface must run in AppClk domain which can set the frequency by the tools. While user interface of Transceiver for 10GBASE-R must be run in MacTxClk for Tx interface and MacRxClk for Rx interface. Therefore, CDC (Clock domain crossing) must be integrated to interface the signals that are run in different clock domain.

Xilinx demo provides ethernet_control_s_axi module to convert AXI4-Lite interface to be internal signals for write/read access. It is mapped to the control/status signals of the other modules. If the signals are run in different clock domain, CDC is applied. The status signal returned by LL10GEMAC is Linkup status of all Ethernet connections.

Data interface of Ethernet kernel has 4 interfaces but using the same interface type, 64-bit AXI4-Stream standard interface. The transmitting and receiving AXI4-stream interface are connected to LL10GEMACTxIF and LL10GEMACRxIF, respectively. The LL10GEMACTxIF module is data adapter from MacTxClk to AppClk and downsize data width from 64 bits to 32 bits. The LL10GEMACRxIF module is the data adapter from AppClk to MacRxClk and upsize data width from 32 bits to 64 bits. The LL10GEMACTxIF and LL10GEMACRxIF are connected with 10G Ethernet MAC controller (LL10GEMAC-IP) by using 32-bit AXI4 stream interface (AXI4-ST) in Tx interface and Rx interface, respectively.

LL10GEMAC-IP, provided by Design Gateway, implements the Ethernet MAC layer and PCS layer with low latency time. Tx interface and Rx interface of AXI4-ST are run in different clock domain, MacTxClk and MacRxClk respectively. LL10GEMAC-IP needs to run with Xilinx Transceiver which is configured to be PMA module for 10GBASE-R interface. PMARstCtrl is designed to control reset sequence of Xilinx Transceiver.

To run AAT demo, at least two 10Gb Ethernet connections are applied. First connection is connected with TestPC that runs tcpreplay command to transfer market data via UDP/IP protocol on Linux OS. Second connection is connected with TestPC that runs ncat command to listen TCP port for trading order via TCP/IP protocol on Linux OS.

The ethernet_control_s_axi module, the peripheral of this kernel, the overview of the AAT system, the host-software shell, and the test procedure of the AAT system can be requested from Xilinx AAT Site.
https://www.xilinx.com/applications/data-center/financial-technology/accelerated-algorithmic-trading.html

3.1 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 : Raw
- Transmitter Buffer : Bypass
- Receiver Buffer : Bypass
- User/Internal data width : 32

Four channels are enabled in AAT demo for four Ethernet connections.
The example of Transceiver wizard in Ultrascale model is described in the following link.
https://www.xilinx.com/products/intellectual-property/ultrascale_transceivers_wizard.html

3.2 LL10GEMAC

The IP core by Design Gateway 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-IP datasheet on our website.
https://dgway.com/products/IP/Lowlatency-IP/dg_ll10gemacip_data_sheet_xilinx_en.pdf

3.3 PMARstCtrl

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

3.4 LL10GEMACTxIF

This module is AXI4-stream data adapter to interface with transmitting path of LL10EMAC in which AXI4-stream crossing clock domains and 64-to-32-bit data width conversion are performed. This module is designed for achieving low-latency performance. Thus, there are some limitations for using this module listed below.

- MacTxClk frequency over UserClk frequency ratio must be less than two.
- If UserTxValid is dropped during the transmission, the latency is increased. The error is asserted to EMAC to cancel the packet transmission and then the logic starts packet retransmission.

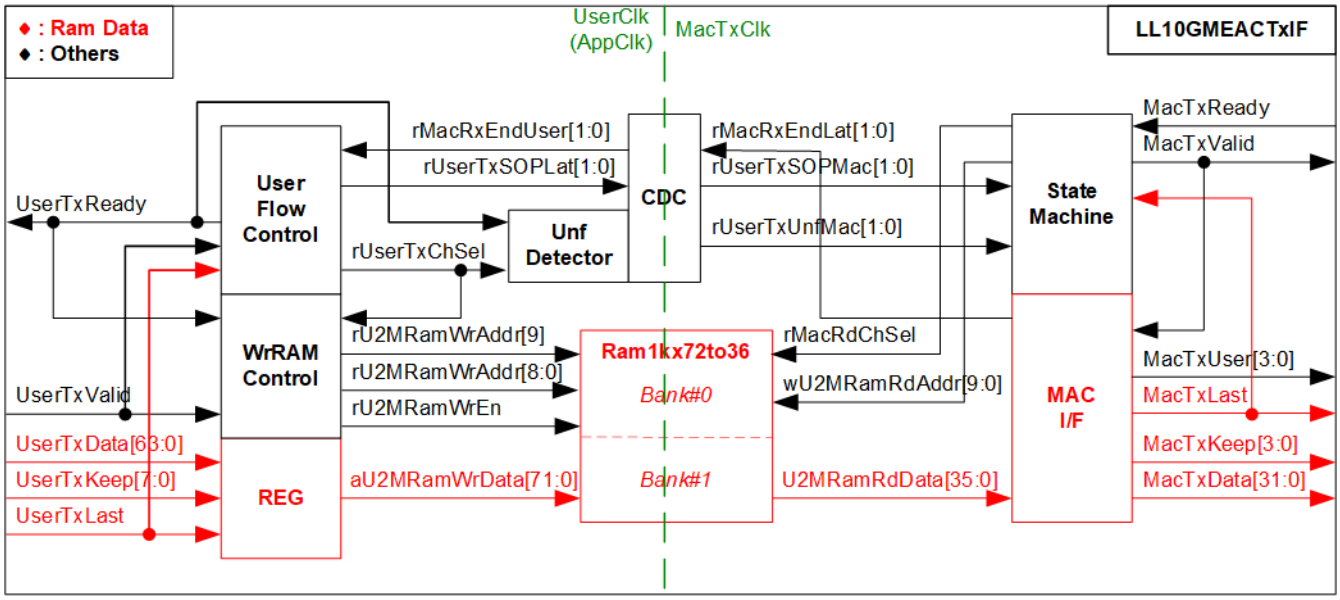

Figure 3-2 LL10GEMACTxIF Logic diagram

As shown in Figure 3-2, there is Ram1k72to36 to store data stream from User interface to EMAC. The Write controller (the logic to write RAM) is run on UserClk while the Read controller (the logic to read RAM) is run on MacTxClk. Also, CDC (clock domain crossing) is included to transfer the flow control signals across clock domain, i.e., Start-of-packet on User I/F, Underflow flag on User I/F, and End-of-packet on Mac I/F.

Ram1kx72to36

The Ram1kx72to36 module is block memory which is configured for simple dual-port block RAM type with asymmetric data width feature. This memory is used to store 72-bit data width in UserClk domain and cross to 36-bit data width in MacTxClk domain. 36-bit data consists of 32-bit data, 2-bit encoded keep, 1-bit last flag, and 1-bit reserved signal. The Ram1kx72to36 module is divided into 2 banks to store the AXI4-Stream data from user. The first bank (ch#0) has a memory address at 0 to 511 and the second bank (ch#1) has a memory address at 512 to 1023. One bank is designed to store one packet, so the maximum packet length is equal bank size, 4096 bytes (512 x 64-bit). The packet length can be extended by extending RAM depth.

Write controller

User flow controller asserts UserTxReady to accept the new packet transmission when the next RAM bank is free. WrRAM Controller generates Write address to store one packet data to the same bank during transferring packet to RAM. After transferring final data of a packet, MSB of the address is inverted to switch the active bank. Besides, User flow controller also asserts Start-of-frame flag (rUserTxSOPLat) when the first data of the new packet is received. This signal is forwarded to the Read controller via CDC to start packet transmission. After the Read controller transfers the final data of packet from each bank, End-of-frame flag (rMacRxEndLat) will be asserted. User flow control uses End-of-frame flag to free the RAM.

There is the limitation of EMAC that a packet cannot pause transmission before transferring the final data. Therefore, data in RAM must be always ready for reading and the Write controller needs to write data to RAM without pausing until end of packet. Unf detector is designed to assert Underflow flag (rUserTxUnfLat) when UserTxValid is de-asserted to '0' before end of packet. This signal is also fed to the Read controller via CDC to cancel the packet transmission.

Read controller

State machine controls data flow to forward the read data from RAM to EMAC. MacTxValid is asserted to '1' when State machine detects Start-of-frame flag of the new packet. The read address is created by State machine to read RAM data which consists of 32-bit data, keep data, and last flag. MAC I/F includes the decoder to convert 2-bit encoded keep signal to be 4-bit signal. If Underflow flag is asserted before transmitting the final data, MacTxLast and MacTxUser will be asserted to cancel the current packet transmission. After that, State machine starts packet retransmission by reading the first data of the same bank. When the final data is transmitted successfully, State machine will switch the RAM bank for the next packet transmission.

Figure 3-3 - Figure 3-4 shows timing diagram of the Write Controller in normal condition and underflow condition. The following details are available for timing diagram description.

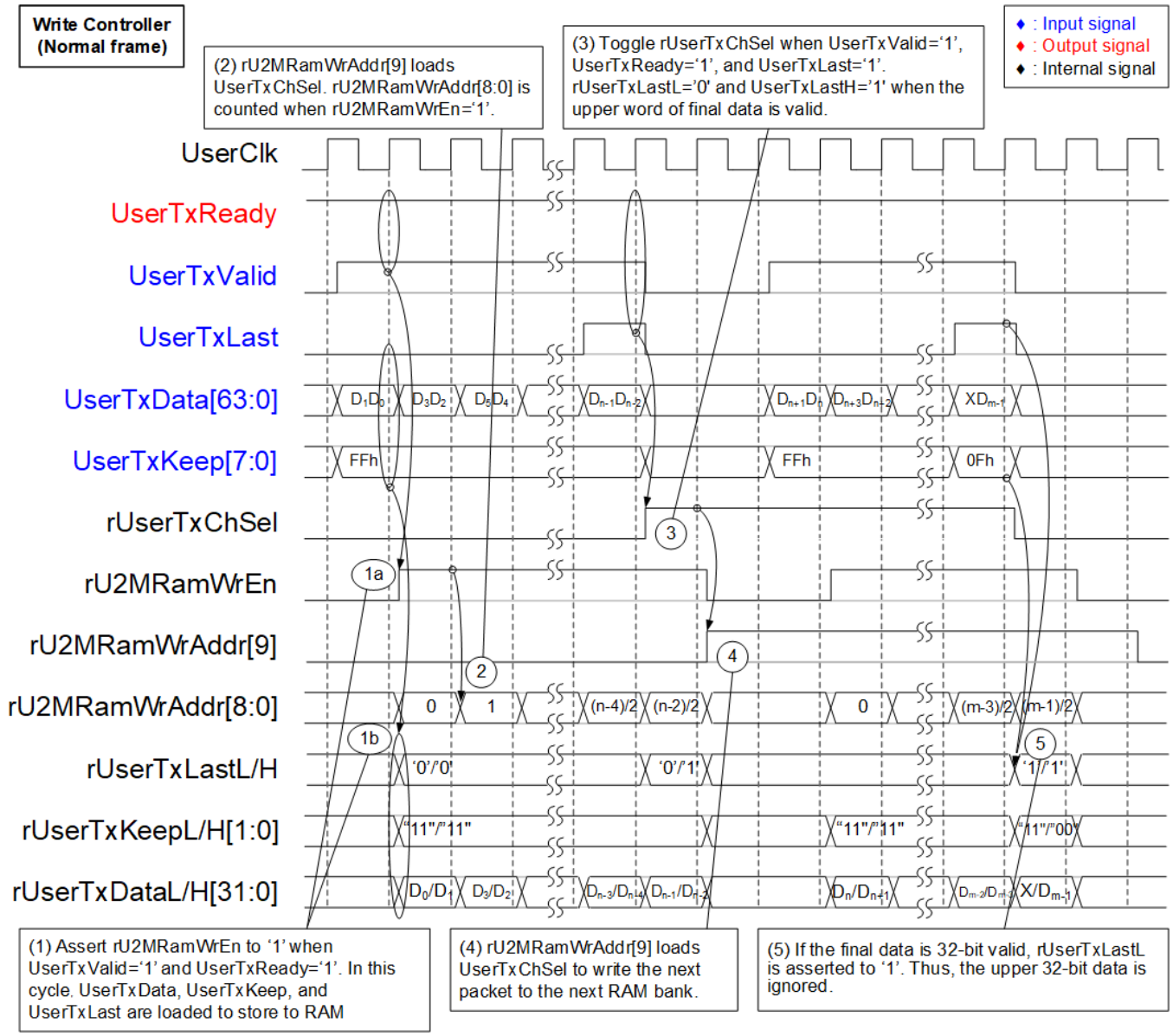

Figure 3-3 Write controller of LL10GEMACTxIF Timing diagram in normal condition

(1) The 64-bit AXI4-Stream data input is accepted by asserting UserTxReady to '1' when UserTxValid='1'. After that, the write enable signal (rU2MRamWrEn) is asserted to '1'. At the same time, User data (UserTxData), User byte enable (UserTxKeep), and User last flag (UserTxLast) are loaded to rUserTxDataL/H, rUserTxKeepL/H, and rUserTxLastL/H. To reduce data width of RAM, 8-bit UserTxKeep is encoded to be 2-bit UserTxKeepL/H by following mapping.

01h->00b/00b, 03h->00b/01b, 07h->00b/10b, 0Fh->00b/11b,
1Fh->00b/11b,3Fh->01b/11b, 7Fh->10b/11b, FFh->11b/11b.

UserTxKeepL/H is equal to "00","01","10", or "11" when the lower/upper 32-bit data is valid for 0-1 byte, 2 bytes, 3 bytes, or 4 bytes. 72-bit write data to RAM is mapped to store user input by following assignment.

Bit[31:0]: rUserTxDataL, Bit[33:32]: rUserTxKeepL, Bit[34]: rUserTxLastL, Bit[35]:RSV,
Bit[67:36]: rUserTxDataH, Bit[69:68]: rUserTxKeepH, Bit[70]: rUserTxLastH, Bit[71]:RSV.

MSB of Write RAM address (rU2MRamWrAddr[9]) is applied to select the active RAM bank. This value loads the value from rUserTxChSel which shows the active bank. While the remained Write RAM address is reset to 0 to store the first data at the first address of the active bank.

(2) rU2MRamWrAddr[8:0] is counted when the write enable (rU2MRamWrEn) signal is asserted to '1' to store the next data at the next RAM address.
(3) When the final data of packet is received (UserTxValid='1', UserTxLast='1', and UserTxReady='1'), the bank indicator (rUserTxChSel) is toggled for storing the next packet to the next bank. To store the final data to RAM, rUserTxLastH is always asserted to '1' when UserTxLast='1'. rUserTxLastL needs to check UserTxLast and UserTxKeep value. If the upper word is valid (UserTxKeep[7:4] is not equal to 0), rUserTxLastL is de-asserted to '0' to indicate that the final data is placed on the upper 32-bit data.
(4) MSB of Write RAM address (rU2MRamWrAddr[9]) updates the value to write data to the next bank. After that, the next packet is stored to the next bank until end of packet.
(5) When the upper 32-bit data is not valid (UserTxKeep[7:4]='0'), rUserTxLastL is asserted to '1' to indicate that the final data is placed on the lower 32-bit data.

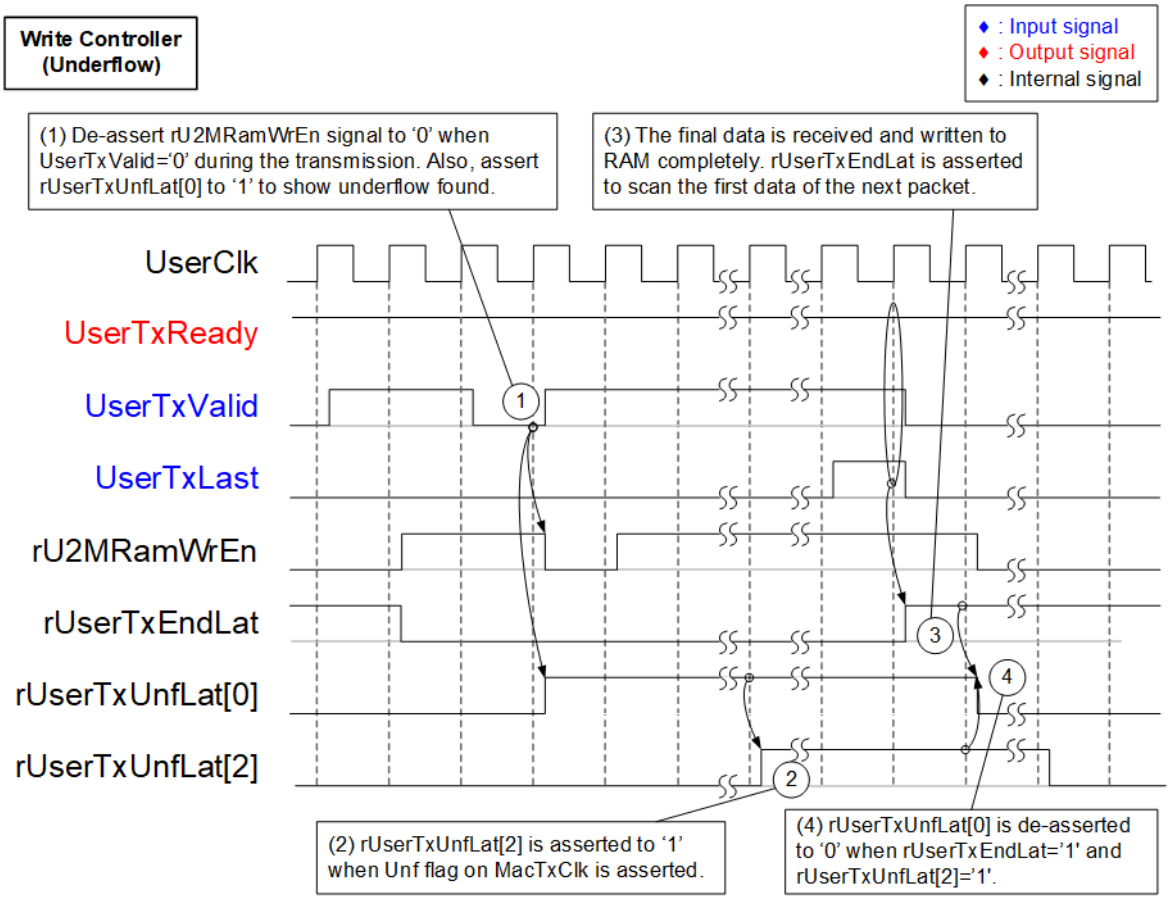

Figure 3-4 Write controller of LL10GEMACTxIF Timing diagram in underflow condition

LL10GEMACTxIF asserts Underflow flag which is internal signal to cancel the current packet transmission when UserTxValid is de-asserted to '0' before end of packet. After that, the packet needs to retransmit. Figure 3-4 shows the process to assert Underflow flag on Write controller and forward to Read controller to cancel the packet transmission. While the retransmission is operated on the Read controller which describes in more details later.

(1) When UserTxValid is de-asserted to '0' before sending the final data (UserTxEndLat='0'), rUserTxUnfLat[0] is asserted to '1'. Similar to normal condition, the write enable signal (rU2MRamWrEn) is asserted to '1' to store the User data when the data is accepted (UserTxValid='1' and UserTxReady='1'). Therefore, all data in the packet is stored to RAM completely without the effect from UserTxValid de-asserted.
(2) Underflow flag (rUserTxUnfLat[0]) is forwarded to CDC (Clock domain crossing) to assert Underflow flag in MacTxClk domain. After that, the Read controller cancels the current packet transmission and then starts packet retransmission. rUserTxUnfLat[2] is asserted to '1' when the Read controller cancels the packet transmission.
(3) rUserTxEndLat is asserted to '1' after the final data of packet is received (UserTxLast='1', UserTxValid='1', and UserTxReady='1').
(4) Underflow flag (rUserTxUnfLat[0]) is de-asserted to '0' when the final data is received (rUserTxEndLat='1') and the Read controller cancels the packet transmission (rUserTxUnfLat[2]='1').

Figure 3-5 shows the timing diagram of the State Machine which is the Read Controller to transfer in normal condition. The following details are available for timing diagram description.

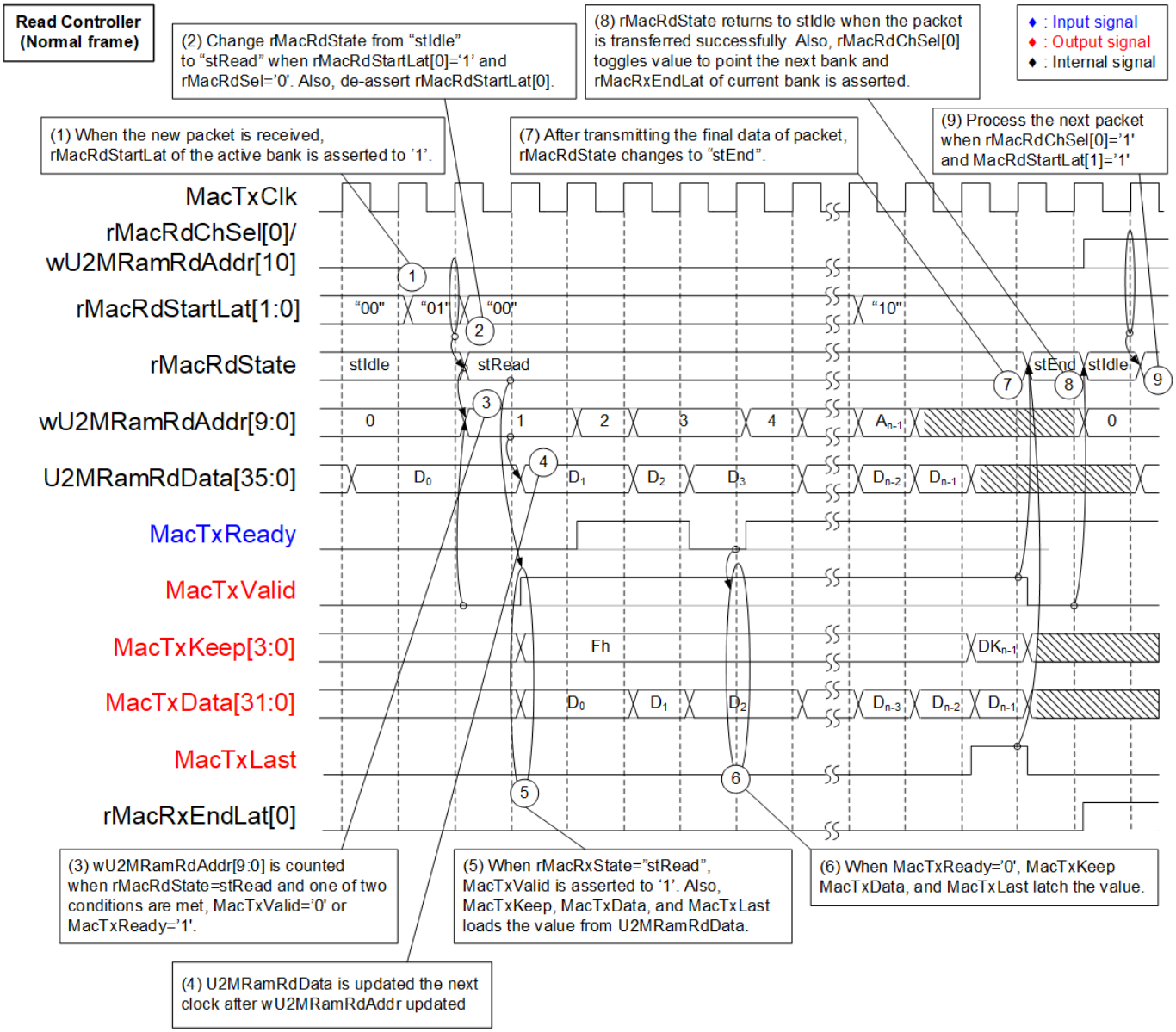

Figure 3-5 Read controller of LL10GEMACTxIF Timing diagram in normal condition

(1) Start flag (rMacRdStartLat), asserted by the Write controller and fed to CDC to apply in Read controller, is asserted to '1' when the new packet is received. After that, the Read controller starts processing.
(2) If the start flag of the current active bank is asserted (rMacRdStartLat[0]='1' when rMacRdChSel[0]='0'), the state changes from Idle state ("stIdle") to Read state ("stRead"). *Note: rMacRdChSel[0] shows the current active bank. '0'-Bank#0 and '1'-Bank#1.*
(3) wU2MRamRdAddr[9:0] is always initialized with 0. The address is only incremented when the main state is in Read state (rMacRdState=stRead) and either it is the first time loading data (MacTxValid='0') or EMAC is receiving AXI4-Stream output (MacTxReady= '1'). This signal is designed by combination logic, so the value is updated at the same cycle that MacTxValid='0' or MacRxReady='1'. The MSB of the read address (wUMRamRdAddr[10]) has the same value as the channel indicator (rMacRdChSel) for selecting one of two areas in memory.
(4) The read data output from RAM, U2MRamRdData, is updated after changing the read address (wUMRamRdAddr[10:0]).
(5) MacTxValid signal is asserted to '1' when the main state (rMacRdState) is in Read state ("stRead"). At the same time, 36-bit read data from RAM (U2MRamRdData) is loaded to be the output to MAC interface.
Bit[31:0]: MacTxData, Bit[33:32]: Decoded to be MacTxKeep, Bit[34]: MacTxLast
The decoded equation: 00b->0001b, 01b->0011b, 10b->0111b, and 11b->1111b.
(6) When the current data is not accepted by EMAC (MacTxReady='0' and MacTxValid='1'), MacTxKeep, MacTxData, and MacTxLast latch its own value.
(7) After the final data is sent to EMAC (MacTxValid='1' and MacTxLast='1'), the state changes to stEnd.
(8) In stEnd, the state waits until MacTxValid is de-asserted to '0' to confirm the final data is completely accepted. After that, the state changes to "stIdle" to wait for the next RAM bank transferring. In this cycle, the active bank (rMacRdChSel[0]) is toggled and End transfer flag (rMacRxEndLat) is also asserted. End flag is fed to the Write controller to free the RAM for storing the new received packet from user.
(9) Return to step (1) to wait until rMacRdStartLat of the active bank is asserted. The active bank is switched to the new bank (rMacRdChSel[0]='1').

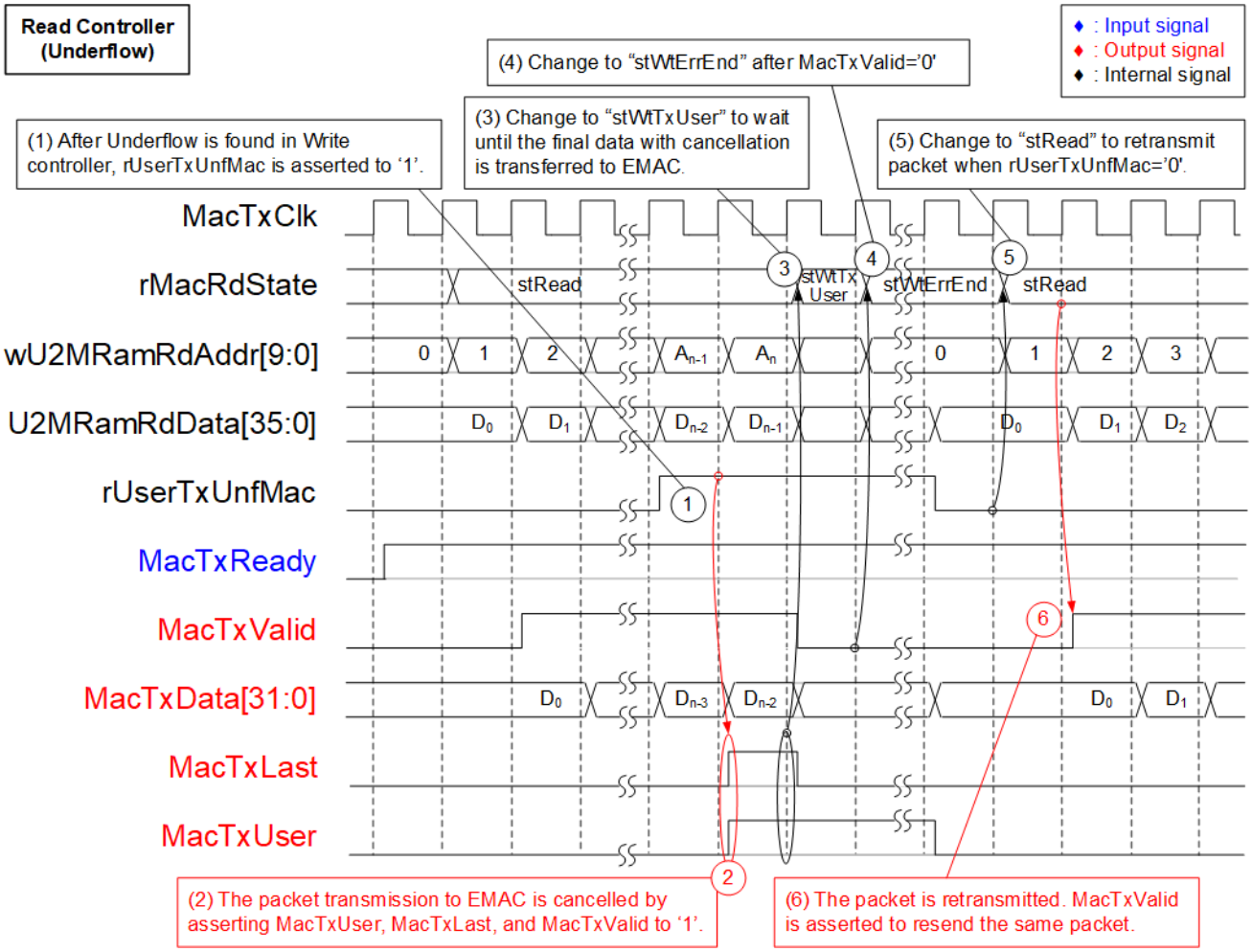

Figure 3-6 Read controller of LL10GEMACTxIF Timing diagram in underflow condition

(1) Underflow flag in MacTxClk domain (rUserTxUnfMac) is asserted to '1' when the Write controller detects the user de-asserts UserTxValid to '0' before sending the final data of packet.
(2) If rUserTxUnfMac is asserted to '1' when the main state is "stRead", the packet transmission to EMAC will be stopped by asserting MacTxValid, MacTxLast, and MacTxUser to '1'. When MacTxUser is asserted at the end of packet, EMAC finishes the packet transmission with error condition to the target. The target will ignore this packet.
(3) The state machine changes to "stWtTxUser" to wait until the final data is sent to EMAC successfully.
(4) The state machine confirms successful status by detecting MacTxValid de-asserted to '0'. After that, the state changes to stWtErrEnd.
(5) The state machine waits until the underflow flag (rUserTxUnfMac) is de-asserted to '0'. After that, the state changes to "stRead" to start packet retransmission.
(6) MacTxValid is asserted to '1' and the same packet is retransmitted. The first data (D0) is sent on MacTxData again.

3.5 LL10GEMACRxIF

This module is AXI4-Stream data adapter to interface with receiving path of LL10GEMAC in which AXI4-stream crossing clock domains and 32-to-64-bit data width conversion are performed. This module includes AXI4-ST 32bto64b Converter, Error Detection, AsyncFWFT32x69, and AXI4-ST Read Controller.

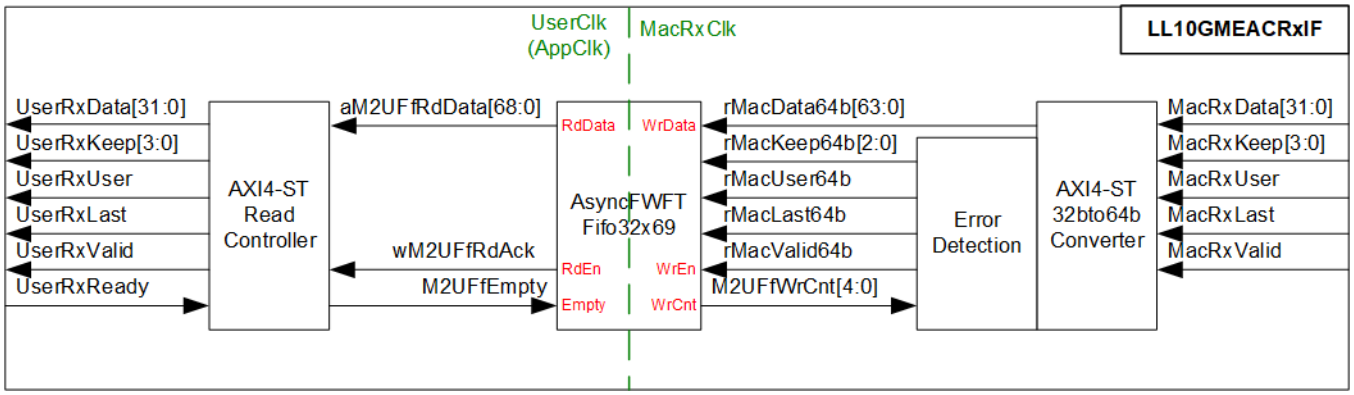

Figure 3-7 LL10GEMACRxIF Logic diagram

The AXI4-ST32bto64b Converter converts 32-bit data to 64-bit data and write data into AsyncFWFT32x69. Error Detection is designed to cancel write operation by two conditions. First, the received packet size is more than the threshold value, 9014 bytes. Second, the free area in FIFO is less than threshold value, 8 data. When the error is found, Error Detection asserts rMacUser64b, rMacLast64b, and rMacValid64b to '1' to finish the current packet transmission with error status.

The AsyncFWFT32x69 module is asynchronous FWFT FIFO used to store data in MacRxClk domain and cross to UserClk domain. 69-bit data is applied to store 64-bit data, 3-bit keep, 1-bit error flag, and 1-bit last flag. 3-bit keep is encoded value to show the valid byte of each data, 1-8 byte. The AXI4-Stream Read Controller reads data from AsyncFWFT32x69 and forwards AXI4-Stream data to user. The data is read when FIFO is not empty. Also, user must be ready to receive data or the sent data is the first data of packet.

Figure 3-8 shows the timing diagram to write FIFO inside LL10GEMACRxIF. The following details are available for timing diagram description.

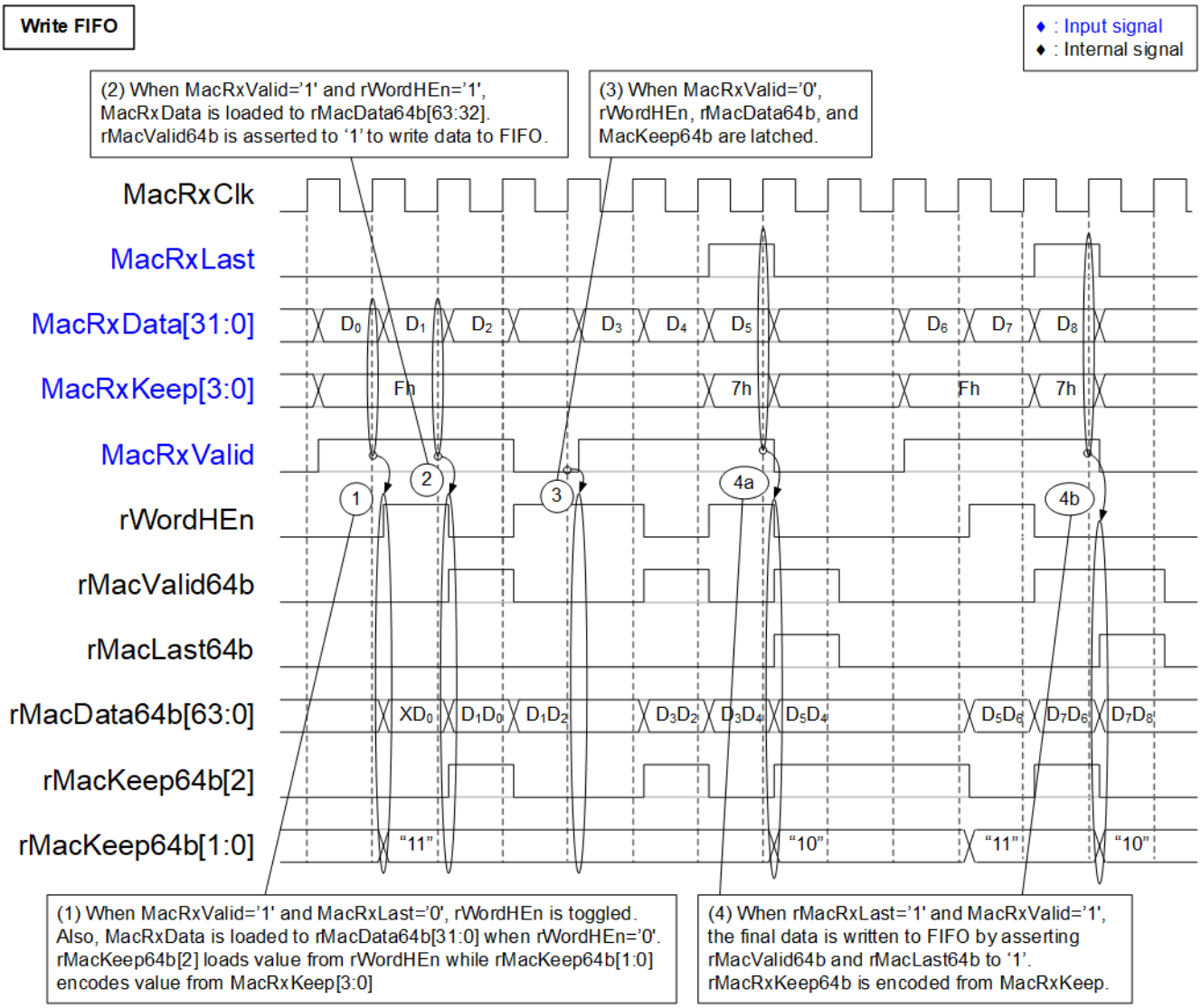

Figure 3-8 Write FIFO logic of LL10GEMACRxIF Timing diagram

(1) The rWordHEn signal is initialized with '0'. rWordHEn is asserted to '1' if receiving 32-bit lower data (MacRxValid='1' and rWordHEn='0'). Otherwise, rWordHEn is de-asserted to '0' to receive 32-bit upper data (MacRxValid='1' and rWordHEn='1').

The 8-bit keep signal used to indicate byte enabled is encoded into 3-bit register, called rMacKeep64b[2:0]. MacRxKeep[3:0] is able to determine byte enabled in 32-bit data only, so it is encoded to be rMacKeep64b[1:0] by using the mapping table: 0001b->00b, 0011b->01b, 0111b->10b, 1111b->11b. rMacKeep64b[2] is designed by loading rWordHEn='1' to show the 32-bit upper data is valid.

The 32-bit upper data (rMacData64b[64:32]) and the 32-bit lower data (rMacData64b[31:0]) are loaded from MacRxData[31:0], controlled by rWordHEn. The lower word is loaded when rWordHEn='0'. Otherwise, the upper word is loaded.

(2) At the same time as the 32-bit upper data (rMacData64b[64:32]) loaded, 64-bit data (rMacData64b), 3-bit keep (rMacKeep64b), 1-bit error flag (rMacUser64b), and 1-bit last flag (rMacLast64b) are written to FIFO by asserting rMacValid64b to '1'.
(3) When EMAC pauses data transmission by de-asserting MacRxValid to '0', rWordHEn, rMacData64b, and rMacKeep64b latch the same value.
(4) When end of packet is transferred (MacRxValid='1' and MacRxLast='1'), MacRxKeep[3:0] might not be all one. In Figure 3-8, MacRxKeep[3:0]=0111b, so rMacKeep64b[1:0] is encoded to be 10b. Also, rMacLast64b is asserted to '1'. There are two different conditions to store the final data.
   a) MacRxLast='1', MacRxValid='1', and rWordHEn='1'. In this condition, the upper data (rMacData64b[63:32]) is valid and rMacKeep64b[2] is asserted to '1'.
   b) MacRxLast='1', MacRxValid='1', and rWordHEn='0'. In this condition, only the lower data (rMacData64b[31:0]) is valid and rMacKeep64b[2] is de-asserted to '0'. rWordHEn is still equal to '0' after receiving the final data.

Figure 3-9 shows the timing diagram of the AXI4-Stream Read Controller and the following details are available for timing diagram description.

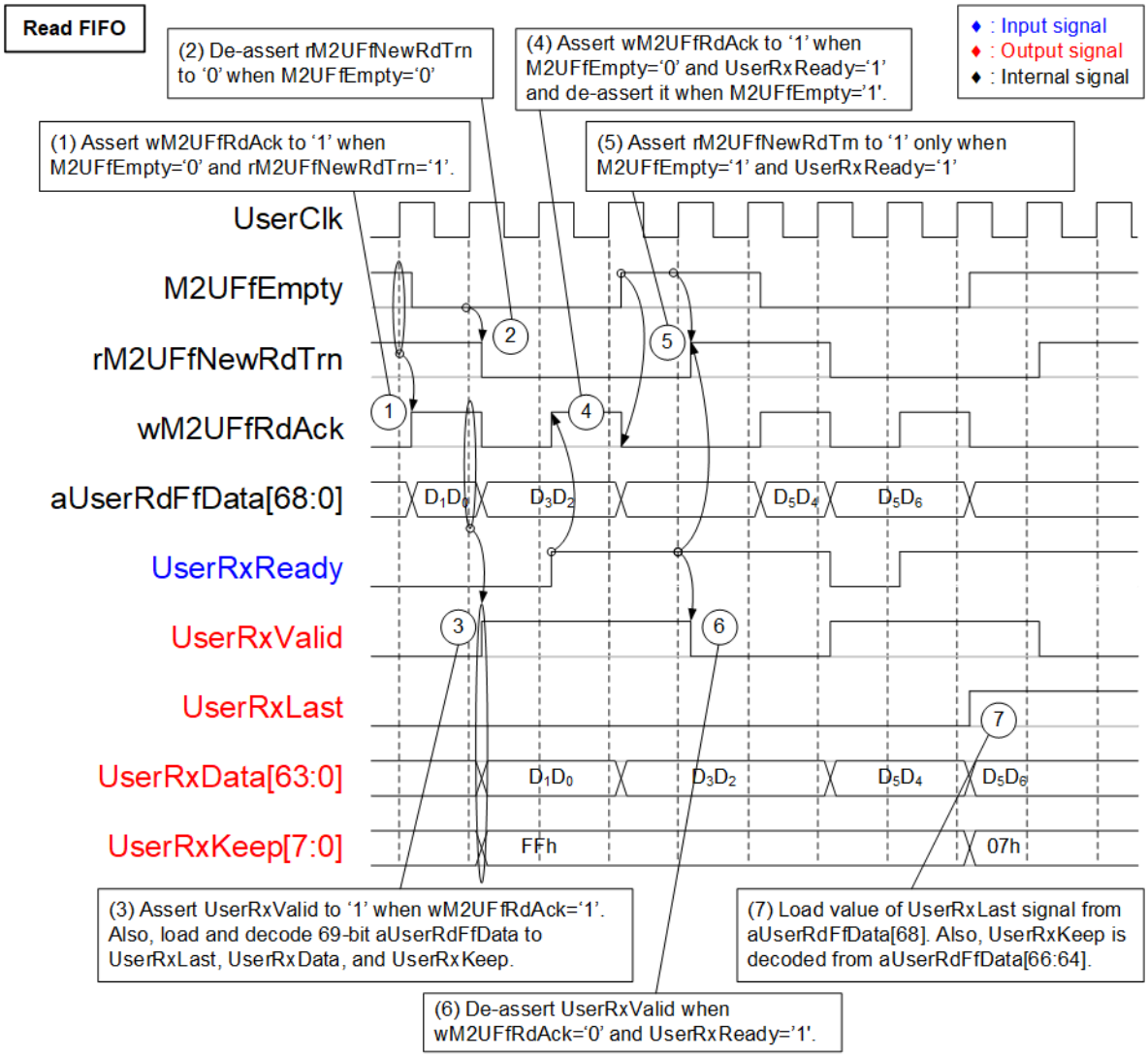

Figure 3-9 LL10GEMACRxIF Read Controller Timing diagram

(1) After the data is written to FIFO, M2UFfEmpty will be de-asserted to '0'. FIFO is FWFT type, so the read FIFO data (aUserRdFfData) is valid at the same time as FIFO read enable (wM2UFfRdAck) asserted. wM2UFfRdAck is immediately asserted to '1' when the FIFO is not empty (M2UFfEmpty='0') and there is no remained latched data in the Read controller (rM2UFfNewRdTrn='1'). After that, the data is transferred to user.
(2) De-assert rM2UFfNewRdTrn to pause reading data from FIFO when there is remained latch data, checked by M2UFfEmpty='0'. After that, wM2UFfRdAck is controlled by UserRxReady.
(3) After the read FIFO enable signal (wM2UFfRdAck) is asserted to '1', the AXI4-Stream valid output (UserRxValid) is asserted to '1'. Also, AXI4-Stream data (UserRxData[63:0]) and last flag (UserRxLast) are loaded from aUserRdFfData[63:0] and aUserRdFfData[68], respectively. aUserRdFfData[66:64] is decoded to create the AXI4-Stream keep (UserRxKeep[7:0]). The decode table of keep signal is shown as follows.
000b -> 01h, 001b -> 03h, 010b -> 07h, 011b -> 0Fh,
100b -> 1Fh, 101b -> 3Fh, 110b -> 7Fh, and 111b -> FFh.
(4) When rM2UFfNewRdTrn='0' and M2UFfEmpty='0', wM2UFfRdAck is controlled by UserRxReady. It is asserted to '1' to read the next data when UserRxReady is asserted to '1' to accept the current data.
(5) When the latch data is transferred to user completely (UserRxReady='1') and there is no remained data in FIFO (M2UFfEmpty='1'), rM2UFfNewRdTrn is re-asserted to '1'. Now the status is similar to step (1) to wait for the new received data in FIFO.
(6) At the same time, UserRxValid is de-asserted to '0' to pause data transmission.
(7) Last flag (UserRxLast) is asserted to '1' when aUserRdFfData[68] is equal to '1'. UserRxKeep when sending the last data may be equal to 01h, 03h, 07h, …, FFh to show valid byte of the final data.

4 Software

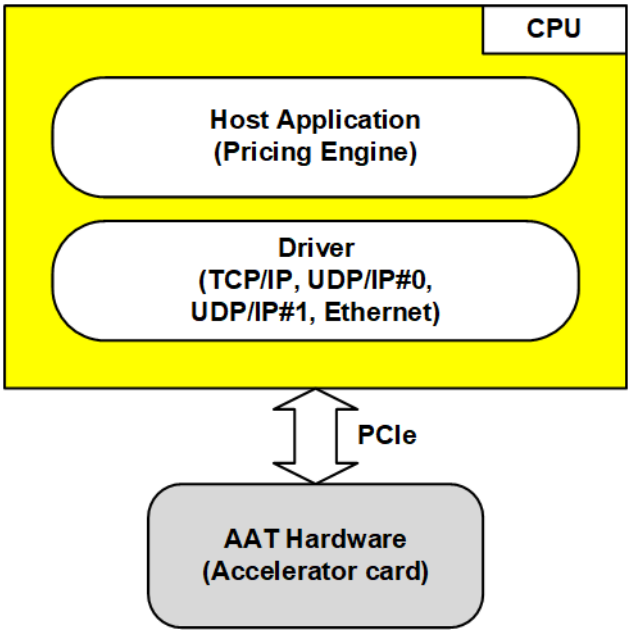

Figure 4-1 Software on AAT demo

AAT demo provides the software source code that can be compiled for running on the processor system that also integrates Accelerator card. There is no modification on AAT software in this reference design. Thus, all source code and document can be requested from Xilinx AAT Site.

The application is designed to show the status of the hardware during processing the market data. Each hardware kernel has its own status which can be displayed on the console. For example, the result from orderbook is Bid/Ask price that is decoded from the received market data as shown in. Figure 4-2.

```
>> orderbook readdata
+-----------------------------------------------------------------------------+
|        Symbol Index =  10          ||    Timestamp = 0x0067015FB0F5C700     |
+------------------------------------++-----------------------------------+
|                BID                 ||                ASK                |
+------------+------------+----------++----------+------------+----------+
|   Count    |  Quantity  |   Price  ||  Price   |  Quantity  |  Count   |
+------------+------------+----------++----------+------------+----------+
|          9 |        830 | 10000.00 || 10200.00 |        140 |        5 |
|          8 |         50 |  9900.00 || 10250.00 |        770 |        3 |
|          6 |        930 |  9800.00 || 10300.00 |        790 |        7 |
|          4 |        860 |  9700.00 || 10350.00 |        910 |        1 |
|          6 |        190 |  9600.00 || 10400.00 |        120 |        6 |
+------------+------------+----------++----------+------------+----------+

>>
```

Figure 4-2 The example of Orderbook status

5 Revision History

Revision	Date	Description
1.0	7-Dec-21	Initial version release