Strmá 3001/11B
616 00 Brno
Czech Republic

UDP/IPv4 for 10 G Ethernet

Reference manual

Strmá 3001/11B
616 00 Brno
Czech Republic

Document Information:

Document content : Reference Manual
Project Title : UDP/IPv4 for 10 G Ethernet
Author : Marek Kváš
Version : 1.0
Date : 5 May 2017

Document Revision History:

Version	Date	Author	Comments
1.0	2017-04-26	MK	Document created
1.0	2017-05-05	MK	Document finalized

Strmá 3001/11B
616 00 Brno
Czech Republic

IP core revision history:

Version	Date	Description
1.0	2017-04-26	Core released for public use.

Strmá 3001/11B
616 00 Brno
Czech Republic

Contents

1 Overview

The User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite. As such, it is the standard part of network stack implementations available on probably all platforms equipped with Ethernet interface.

The UDP protocol uses simple connectionless transmission model with minimal protocol overhead. It uses no handshaking mechanism and so it provides no means to detect neither loss of packets nor ordering change nor duplicates.

Based on specific requirements of each particular application, loss detection and ordering/duplication detection are often added as a part of user protocol of an application layer. User can balance protocol overhead to its needs.

This core, UDP/IPv4 for 10 G Ethernet, implements mandatory parts of UDP, IPv4 and Ethernet (MAC) protocols. It is minimal implementation of complete RFC compliant UDP/IP stack.

It doesn't implement supporting protocols as Address Resolution Protocol (ARP – translating IP addresses to MAC addresses), Dynamic Host Configuration Protocol (DHCP – often use to assign IP addresses dynamically) or Internet Control Message Protocol (ICMP – services like ping). Services that are commonly provided by these protocols must be replaced by user defined mechanisms.

The main field for deployment of this core are high-speed data transfer applications mainly in controlled environment of local networks or simple peer-to-peer connections, but it is not limited to it.

The core is aimed to be used for 10 G Ethernet in both optic and metallic version. It is also ready to be used with PHYs that support up to six speeds – 10 Gbps, 5 Gbps, 2.5 Gbps, 1 Gbps, 100 Mbps, 10 Mbps.

2 Key Features and limitations

- UDP protocol based communication at full bandwidth at 10 Gbps speed without lost packets
  - Maximum UDP datagram size given by user-defined synthesized FIFO sizes.
  - UDP check-sum not used (its optional in RFC).
- IP version 4
  - Broadcast supported (both local and subnet mask based).
  - IP fragmentation not supported.
  - IP options not supported
- XGMII interface (64 bit at 156.25 MHz) for connection to lower layers (e.g. Xilinx RXAUI core).
- Both RX and TX user interfaces are buffered – user doesn't have to process/provide one word in each clock cycle and doesn't need to know datagram size in advance.

3 Function description

Internal structure of the UDP/IPv4 is depicted in the Figure 1. The core consists of two paths – transmission(TX) and receive(RX) path. Except for the clock, reset and HOST SETTINGS interface, these paths are independent of each other.

The HOST SETTINGS interface consists of MAC address, IP address and subnet mask. These information are used as source addresses in the TX path (in MAC and IPv4 headers) and for filtering of incoming packets in the RX path.

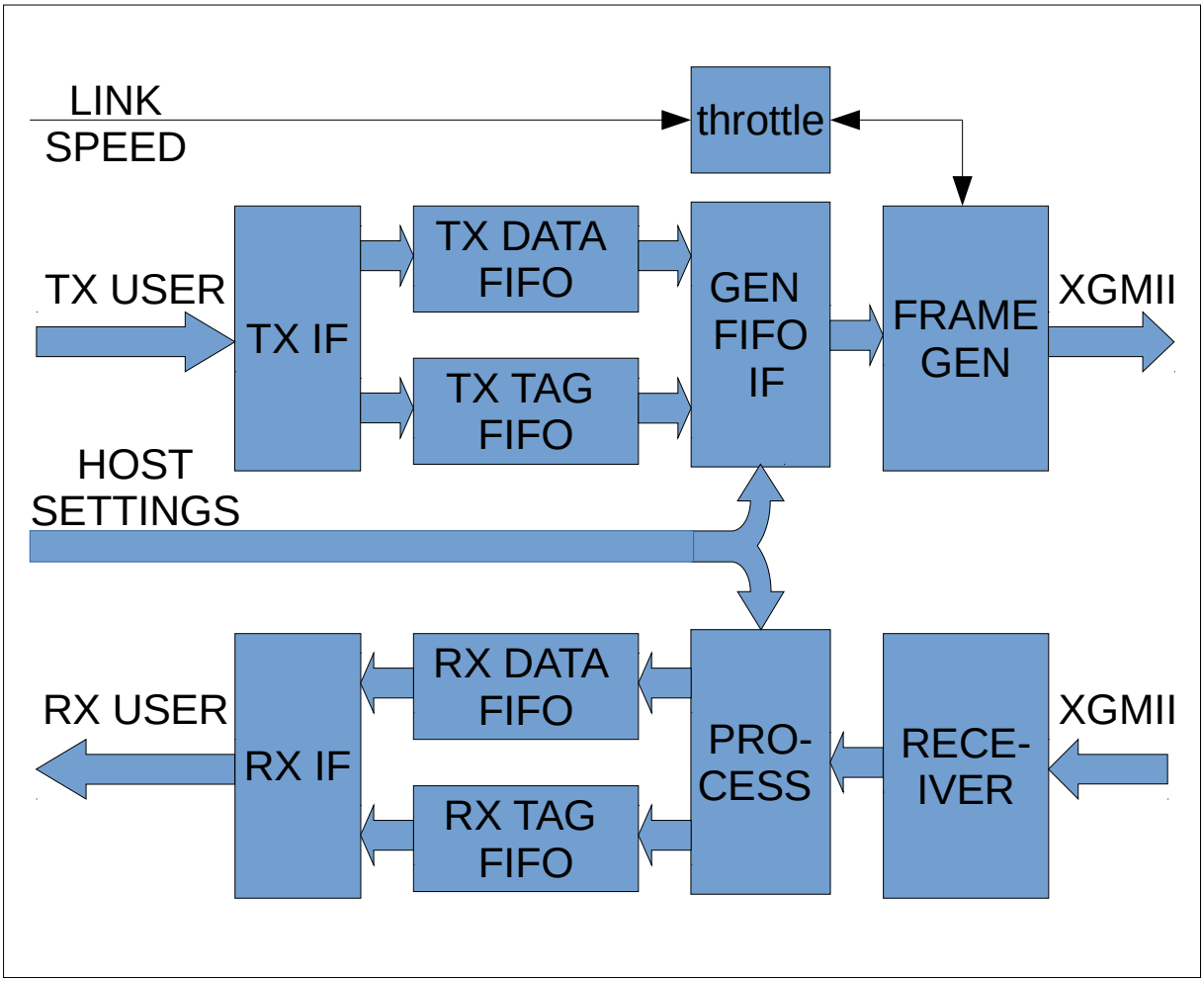

Figure 1: UDP/IPv4 core block diagram

3.1 RX path

Data are received via XGMII interface by RECEIVER block in the block diagram. It ensures data are properly aligned for further processing and checks CRC of incoming packets. Normalized data together with information about validity of CRC are passed to PROCESS block. It implements part of Ethernet MAC layer.

The PROCESS block consists of Ethernet MAC layer, IP layer and UDP layer. Each one is responsible for filtering and consistency checks on its particular layer. Each layer also extracts information that is necessary to send response to sender of the received packet (return info). The return info consists of source MAC and IP addresses and source UDP port. As destination UDP port needs to be passed to the user, it is extracted too.

The Ethernet MAC layer of the RX path accepts only Ethernet frames with destination MAC address matching either broadcast MAC address (FF:FF:FF:FF:FF:FF) or host MAC address given by the HOST SETTING interface. Each accepted frame must have valid CRC too.

The IP layer accepts packets that have valid IP header checksum and match one of the address filtering rules. Address filtering rules let pass packet that fulfils one of these conditions:

- Destination IP address of the packet matches exactly host IP address given by HOST SETTINGS interface.
- Destination IP address of the packet is broadcast for subnet that host IP address belongs to. The subnet broadcast address is derived form host IP address and subnet mask given by HOST SETTINGS interface. If host IP is e.g. 192.168.2.5 and subnet mask is 255.255.255.0, subnet broadcast address is 192.168.2.255.
- Destination IP address of the packet is “this network” broadcast address 255.255.255.255.

The UDP layer does no filtering based on UDP ports. Both source and destination UDP port are passed to the user. UDP header CRC is not checked (it is optional according to the RFC and it doesn’t limit any functionality).

As data are coming they are stored in the RX DATA FIFO. Necessary return info is stored separately in RX TAG FIFO. As soon as packet ends, tag containing length of stored data and information about its validity is written to the RX TAG FIFO.

When there is a valid tag in the TAG FIFO, RX IF block extracts it (together with return info) and decides whether packet shall be discarded (e.g. because of CRC) or passed to user. If packet is valid, return info and UDP payload is provided to the user via RX user interface.

3.2 TX path

User can send UDP data via TX USER interface. User must provide destination addresses (MAC, IP) and both source and destination UDP port together with the first word of data. Length of the data doesn’t have to be known in advance, but user is responsible to ensure, that data fit into the TX DATA fifo, resulting frame is shorter than MTU.

User provides UDP payload only – header are added by the core. If MTU is 1500 Bytes (most often case in gigabit networks) maximum UDP payload size is 1472 bytes. If jumbo frames are enabled and MTU is 9000 bytes, payload can have 8972 bytes.

When user is done with writing the data and marks end of packet, TX IF block writes tag with length of data to be sent to the TAG FIFO.

When GEN FIFO IF block gets valid tag from TX TAG FIFO it initiates transmission of Ethernet frame in FRAME GEN block and provides all data it needs.

Data are sent out towards lower layers using the XGMII interface.

3.3 Link speed changes – throttling

The XGMII interface is defined to run at 156.25 MHz if 64 bits wide. This corresponds to 10 Gbps link speed. The XGMII operating frequency is constant and cannot change during operation. Moreover the XGMII doesn't contain any flow-control.

If the PHY with ability to change link speeds to slower standards is used, outgoing traffic must be throttled. There is a buffer in the PHY in such a case. Source of the data (in this case the UDP/IPv4 core) must ensure that average speed of data transmitted via the XGMII matches the real link speed. When this is fulfilled, the buffer inside the PHY ensures correct operation.

Interface called LINK SPEED in the diagram must be used by the user to provide valid information about current link speed. If information is incorrect, strange packet loss and fragment transmission may be observed. This effects may not be observed under low traffic because the PHY buffer may suffice to compensate.

Link speed changes don't affect the RX path – they are always compensated by the PHY. If LINK SPEED setting doesn't match the real link speed, incoming data are not disrupted.

3.4 Bandwidth consideration

When maximum achievable bandwidth shall be estimated it is necessary to consider these factors:

- Ethernet framing overhead – 38 bytes. Consists of:
  - Ethernet preamble (7) + Start Of Frame delimiter (1) = 8 bytes.
  - Ethernet header - 14 bytes.
  - CRC - 4 bytes.
  - IPG - 12 bytes.
  - For very short datagrams requirement of 64 bytes as minimal frame length can introduce additional overhead.
- IP header overhead – 20 bytes.
- UDP header overhead – 8 bytes.

Depending on frame length another 4 bytes of IPG may need to be added due to alignment. This makes up to 70 bytes overhead to each transferred datagram. If maximum jumbo frames (MTU=9000) are used 8972 payload bytes can be carried in each datagram. This makes approximately 99.2 % utilization of the link. If we don't consider headers as overhead, we get to 99.7 %.

If standard frames (MTU=1500) are used, we get 95.5 % and 98.2 % utilization.

As datagram size go lower utilization falls down quickly.

Except for aforementioned 4 alignment bytes, the core adds no additional overhead.

For the TX path, the core wraps datagram payload to 70 bytes of envelope. It means that even with full utilization of the link in average 8 or 9 idle (RDY low) cycles will be observed between (or inside) packets.

The RX path is designed to cope with full bandwidth too. As headers are discarded inside the core and only payload is transferred to the user, the same 8 or 9 idle cycles (VALID low) can be observed at RX user interface between (core itself never inserts idles inside datagrams) datagrams.

3.5 Latency consideration

Both interfaces TX and RX are buffered in such a way that datagrams must be completely stored to buffer before transmission or user processing may be started.

This design has different reasons for TX and RX path. In case of RX path it ensures that only valid datagrams are passed to the user as CRC check can be done only after whole frame is received. The TX path was designed this way in order to get rid of necessity to know length of transmitted data in advance and in order to provide two-way flow control mechanism on the TX user interface.

Mechanisms described above cause that latency is given mostly by time that takes to store data into buffers so it varies depending on datagram length.

4 Parametrization

There are four FIFOs inside the core. All of them can be parametrized by the user using VHDL generics (if used as VHDL source) or by GUI parameters when used as IPI or IP catalogue component.

There are two properties that can be set for each FIFO – size and implementation type.

There are three possible options for the type. When FIFOs shall be implemented in block RAM, which is most probable choice for both RX and TX DATA FIFOs, property must be set to string “block”. If distributed memory shall be used, property must be set to string “distributed”. Implementation in distributed memories can make sense for small TAG FIFOs under certain circumstances. The last option is “auto”, which says that implementation tools shall decide.

Sizes of FIFOs are given as their depth. Each data FIFO can store 8 bytes of UDP payload data in one word. So if depth is 1024, FIFO can store 1024*8 = 8196 bytes.

Sizes of DATA FIFOs must be always higher than the largest frame that can be transmitted/received. For TX DATA FIFO 3 words overhead per each packet waiting in the FIFO must be added (so the minimum depth is ceil(MTU/8) + 3).

For the TX TAG FIFO, one word (or depth level) is used for each outstanding datagram. Minimum depth of 4 is recommended.

For the RX TAG FIFO, 3 words are used for each received waiting datagram.

Generic Name	Description	
g_tx_dfifo_depth	Depth of TX DATA FIFO – 8 data bytes per level. Each stored datagram needs 3 levels (24 bytes overhead). Should be at least ceil(MTU/8) + 3	
g_tx_tfifo_depth	Depth of TX TAG FIFO – for each datagram stored in the data fifo one word is needed.	
g_rx_dfifo_depth	Depth of RX DATA FIFO – 8 data bytes per level. Should be at least ceil(MTU/8) + 1	
g_rx_tfifo_depth	Depth of RX TAG FIFO – for each datagram stored in the data fifo one word is needed.	
g_tx_dfifo_type	TX DATA FIFO type	“block” - block RAM “distributed” - distributed memory (LUTs) “auto” - implementation selected by tools
g_tx_tfifo_type	RX TAG FIFO type	
g_rx_dfifo_type	TX DATA FIFO type	
g_rx_tfifo_type	RX TAG FIFO type	

5 Interfaces

All signals of all interfaces are synchronous to single clock signal CLK.

5.1 Control signals

Signal Name	Direction	Description
CLK	In	XGMII clock 156.25 MHz. The main clock that all signals are synchronous to.
RST	In	Active-HIGH synchronous reset. Reset can be one cycle long, but it takes 16 cycles to reset core internally. All incoming data are ignored during this period.
LINK_SPEED[2:0]	In	Link speed must be provided by the user in order to ensure correct and reliable data transmission. Receiver path doesn't depend on this settings. There are six available speeds: "000" - 10 Gbps "001" - 5 Gbps "010" - 2.5 Gbps "011" - 1 Gbps "100" - 100 Mbps "101" - 10 Mbps

5.2 XGMII

XGMII interface connects UDP/IPv4 core to lower layers – e.g. Xilinx RXAUI core.

Signal Name	Direction	Description
XGMII_RXC[7:0]	Out	Receive control bits, 1-bit per received data byte.
XGMII_RXD[63:0]	Out	Received data, 8 bytes wide.
XGMII_TXC[7:0]	In	Transmit control bits, 1-bit per transmit data byte.
XGMII_TXD[63:0]	In	Transmit data, 8 bytes wide.

5.3 Host settings

This interface is common for TX and RX path of the core. When this settings are changed they are applied immediately and even datagrams waiting in the queue for transmission will be sent with the new settings.

Signal Name	Direction	Description
HOST_MAC[47:0]	In	MAC address (address A1:A2:A3:A4:A5:A6 is represented by 0xA1A2A3A4A5A6).
HOST_IP[31:0]	In	IPv4 Address (address 10.0.0.5 is represented by 0x0A000005.
HOST_IP_NETMASK[31:0]	In	IPv4 subnet mask (255.255.255.0 is represented by 0xffffff00).

5.4 TX user interface

The TX user interface enables user to transmit UDP datagrams. Signals can be divided into two groups. Signals with prefix TX_FRAME_ form AXI4-stream compatible interface that is used to start new datagram, provide payload data, and end the datagram. The other signals with prefixes TX_SRC_ or TX_DST_ are used to provide destination addresses – MAC and IP address – and UDP source and destination port. These additional signals don't fit into the AXI4-stream semantics. List of signals of this interface is provided in the table below.

Signal Name	Direction	Description	
TX_DST_MAC[47:0]	In	Destination MAC address.	Must be valid from the first assertion of the VALID for the new datagram until the RDY is asserted by the core.
TX_DST_IP[31:0]	In	Destination IP address.	
TX_SRC_UDP[15:0]	In	Source UDP port.	
TX_DST_UDP[15:0]	In	Destination UDP port.	
TX_FRAME_VALID	In	By the assertion the user informs the core that BE and DATA signals are valid. If once asserted, it must not be deasserted until the core accepts the data word by the assertion of the RDY signal. The first valid word after the reset or the last word of the previous datagram starts a new datagram. MAC address, IP address and UDP ports must be valid from the first assertion of the VALID for the new datagram until the RDY is asserted by the core.	
TX_FRAME_RDY	Out	By assertion, the core informs the user that it is ready to accept data and if VALID is asserted at the same time, it acknowledges that the data word is accepted.	
TX_FRAME_LAST	In	Must be asserted for the last word of each datagram. Until the datagram is finished its transmission to physical media cannot begin.	
TX_FRAME_BE[7:0]	In	Byte enable. One bit per data byte. Must be set to all ones for all words except for the last one. May be all zeroes if empty datagram shall be sent or if no more data are available and datagram must be ended.	
TX_FRAME_DATA[63:0]	In	UDP data to send. LSB is sent first.	

Figure 2 shows typical start of frame and relationship between VALID signal and destination infos (MAC, IP, ports). Lets assume that state in the cycle 0 is after reset or after the last datagram was correctly ended.

In the cycle 1 the user decides to start a new datagram. The VALID signal is asserted together with setting info signals and data-related signals (DATA, BE, LAST) to their valid value.

Depending on the state of the FIFOs of the TX path, it may take number of cycles before the core asserts the RDY signal, acknowledging start of the new datagram and acceptance of the first data word (occurs in the cycle marked 4 in the figure). During this period none of the signals is allowed to change.

The destination info signals are not used after the cycle 4 and may change. The waveform can continue in number of ways in the cycle 5 and further.

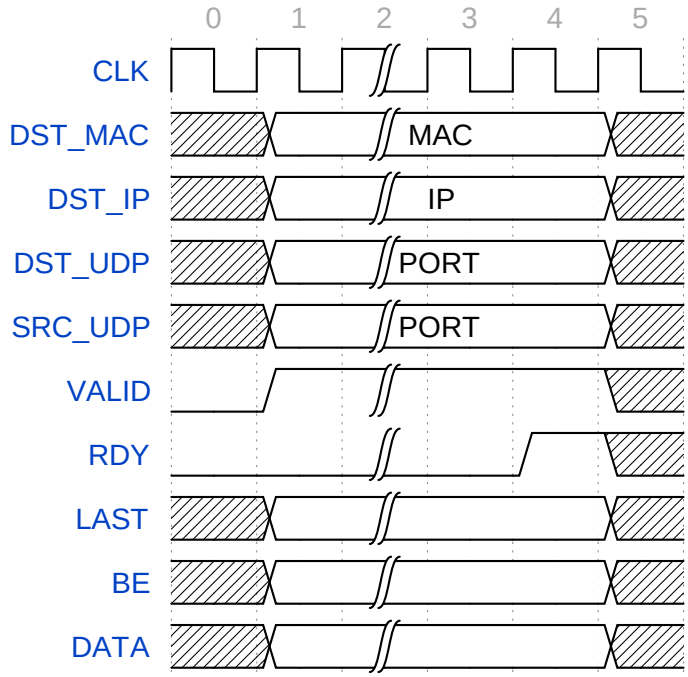

Figure 2: Start of frame - relation between infos and valid signal

Figure 3 shows transmission of one whole datagram. MAC, IP and UDP port info signals are all grouped to Infos in the figure. It is almost ideal case. Datagram is started in the same way as in previous case in cycle 1. The core accepts start of datagram and the first data word in the cycle 3. TX FIFO has enough space and the user has enough data ready so no flow-control is performed. Datagram is ended in the cycle 8, where the user asserted LAST signal. Please notice that BE signal must be all-ones for all data words except for the last one. In this particular case BE has value 0x1f that indicates that only 5 bytes of the word are real data – byte 4 of the data word will be the last one in the datagram. UDP payload length will be 45 bytes in this case.

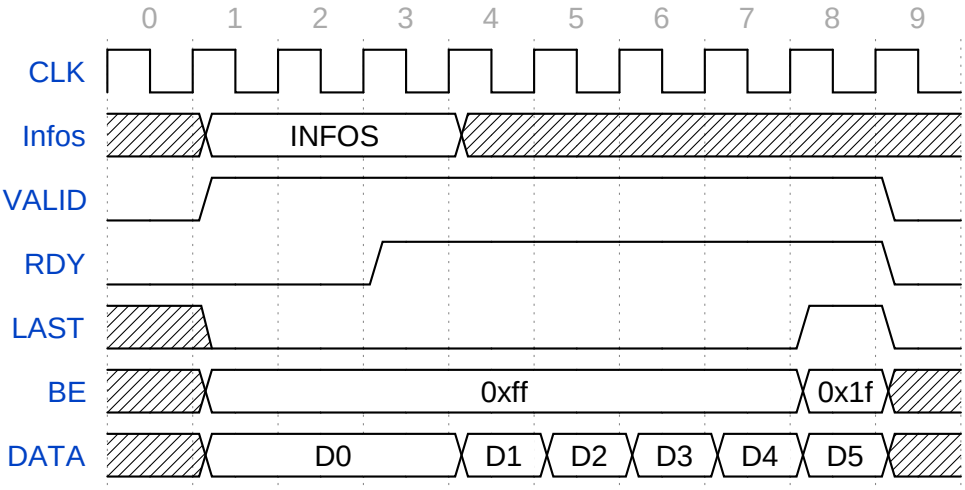

Figure 3: Whole frame transmission - no flow-control

More complex situation is depicted in the Figure 4. Start of the datagram is the same as in previous cases. The first difference appears in the cycle 4, where the user didn't have new data word ready after data word D0 had been accepted in the cycle 3. The user deasserted VALID signal to indicate this to the core. Data word D1 became available in the cycle 5, so VALID was asserted again. Another difference appears in the cycle 7, where RDY was deasserted because TX FIFO was temporarily full. RDY was asserted again in the cycle 8 and the data word D3 was accepted in this cycle. VALID is deasserted in the cycle 9 as no new data are available. The user then waited for new data until cycle 11 where the user decided to end the datagram and let it be transmitted. BE was set to 0x00 which meant no new data should be added.

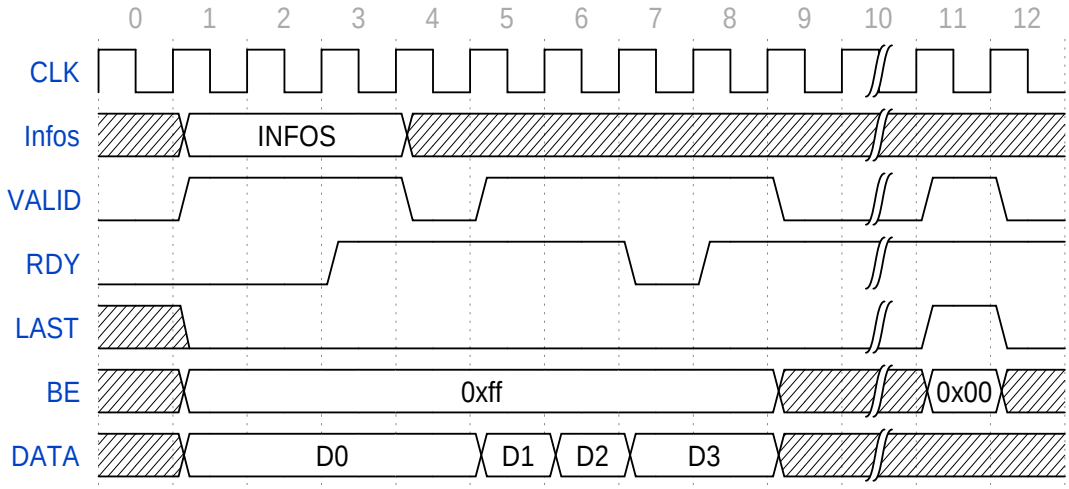

Figure 4: Whole frame transmission - flow-control

5.5 RX user interface

The RX user interface is analogous to the TX user interface described earlier. It consist of AXI4-stream compatible set of signals prefixed with RX_FRAME_, set of signals identifying datagram source and service prefixed with RX_SRC_ or RX_DST_ and additional signal RX_FRAME_LENGTH that is added for user convenience. It informs of payload length of the whole datagram. Complete list of signals can be found in the table below.

Signal Name	Direction	Description	
RX_SRC_MAC[47:0]	Out	Source MAC address.	This signals become valid in the same cycle VALID signal is asserted for the first time for datagram (datagram start) and remains valid until the last data word is accepted by the user.
RX_SRC_IP[31:0]	Out	Source IP address.	
RX_SRC_UDP[15:0]	Out	Source UDP port.	
RX_DST_UDP[15:0]	Out	Destination UDP port.	
RX_FRAME_LENGTH	Out	Length of datagram.	
RX_FRAME_VALID	Out	By the assertion the core informs the user that data of received datagram are ready to be read out. If it is asserted for the first time since reset or previous packet ended. It also indicates the source addresses and length are also valid up until the end of the datagram. Once asserted it remains asserted until the whole frame is read. It is guaranteed to be deasserted for at least one cycle between datagrams.	
RX_FRAME_RDY	In	By assertion, the user informs the core it is ready to accept data and if VALID is asserted at the same time, it acknowledges that the data word is accepted.	
RX_FRAME_LAST	Out	Asserted for the last word of the datagram.	
RX_FRAME_BE[7:0]	Out	Byte enable. One bit per data byte. It is set to all ones for all words except for the last one. May be all zeroes if empty datagram was received.	
RX_FRAME_DATA[63:0]	Out	Data of the received UDP datagram. LSB was received first.	

As the TX user interface is AXI4-stream compatible too, its behaviour is similar to behaviour of the RX user interface. Figure 5 depicts how one datagram is completely received and the consecutive one started.

As interface is buffered the core guaranties that once it offers start of the datagram by assertion of VALID, it can provide new data word in each cycle – so VALID remains asserted until the end of the datagram. However the user can use flow-control signal RDY to interrupt process if it needs more time to handle data. This situation is shown in the cycles 1,2 and 7. It is also useful to know that there is always at least one cycle idle between datagrams (deasserted VALID) – the LAST signal may not need to be evaluated.

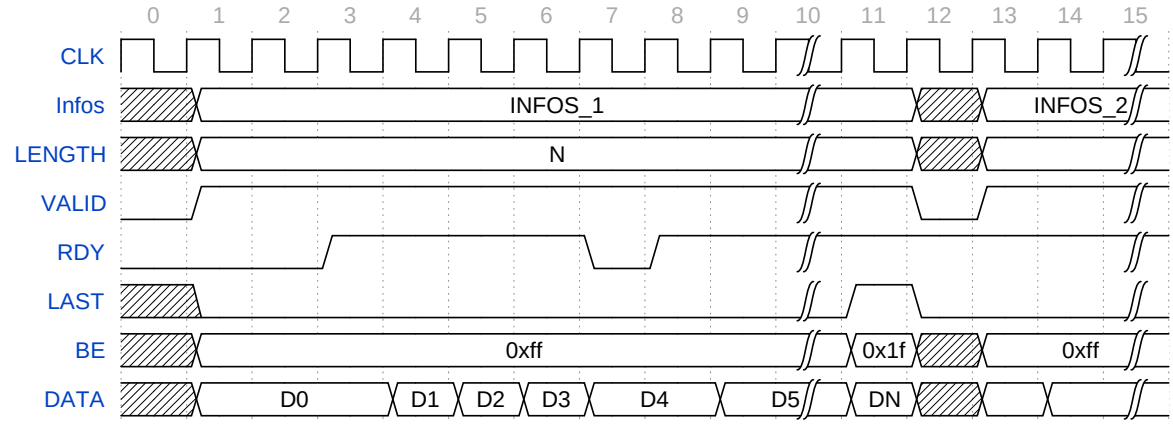

Figure 5: Two consecutive frames received