**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 File |
| Verification | Test Bench, Simulation Library |
| Instantiation Templates | VHDL |
| Reference Designs & Application Notes | QuartusII Project, See Reference Design Manual |
| Additional Items | Demo on CycloneV E/ ArriaV GX/Arria10 SoC Development Board |
| **Simulation Tool Used** | |
| ModelSim-Altera 10.1e | |
| **Support** | |
| Support Provided by Design Gateway Co., Ltd. | |

## Features

- UDP/IP stack implementation
- Support IPv4 protocol
- Support one session per one UDP1G-IP (Multisession can be implemented by using multiple UDP1G-IP)
- Transmit/Receive buffer size, adjustable for resource and performance optimization
- Simple data interface by standard FIFO interface
- Simple control interface by standard register interface
- 8-bit Avalon stream to interface with Triple-Speed Ethernet MAC from Intel
- One clock domain interface by fixed 125 MHz clock frequency
- Reference design available on CycloneV E/ArriaV GX/Arria 10 SoC Development Board
- Support IP fragmentation

**Table 1: Example Implementation Statistics**

| Family | Example Device | Fmax (MHz) | ALMs | Registers[1] | Pin | Block Memory bit[2] | Design Tools |
|---|---|---|---|---|---|---|---|
| StratixIV GX | EP4SGX230KF40C2 | 125 | 1,133 | 1,452 | - | 1,181,696 | QuartusII 16.0 |
| CycloneV E | 5CEFA7F31I7 | 125 | 1,023 | 1,656 | - | 1,181,696 | QuartusII 16.0 |
| ArriaV GX | 5AGXFB3H4F35C5 | 125 | 1,038 | 1,659 | - | 1,181,696 | QuartusII 16.0 |
| Arria10 SX | 10AS066N3F40E2SGE2 | 125 | 1,051 | 1,749 | - | 1,181,696 | QuartusII 16.0 |

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Block memory resource s are based on 64k Tx data buffer size, 16k Tx packet buffer size, and 64k Rx data buffer size. Minimum size of each buffer are 4k Tx data buffer size, 2k Tx packet buffer size, and 2k Rx data buffer size.

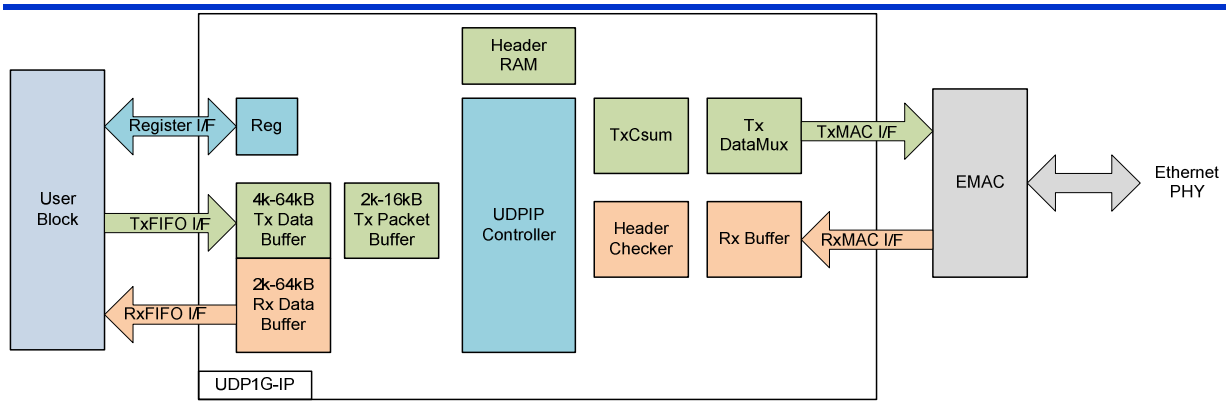


**Figure 1: UDP1G-IP Block Diagram**

## Applications

UDP1G-IP is designed for network application such as video data streaming by using UDP/IP protocol to transfer high speed data through 1 Gb Ethernet. By using this IP, user can easily transfer data with some devices through UDP/IP protocol without CPU and external memory in the system.

## General Description

UDP1G-IP core operating with Intel Triple-Speed Ethernet MAC implements as UDP/IP stack, Transport layer, Internet layer, and Link layer for network data transmission. User sends and receives 1 Gb Ethernet data with some network devices through UDP/IP protocol by using this system and external PHY chip.

There are three types of user interface, i.e. control signal by register access, transmit and receive data signal by FIFO access. To initialize system, user needs to set up network parameters such as MAC address, port number, IP address through register interface. After complete initialization, UDP1G-IP is ready to receive command from user.

After user sends command, data is transferred from user logic through TxFIFO interface. Transmit data from user is stored within Tx Data buffer before forwarding to Ethernet MAC. For the receive direction, if the header of UDP packet is valid, UDP1G-IP will extract only UDP data to store in Rx Data buffer. User reads received data through RxFIFO interface.

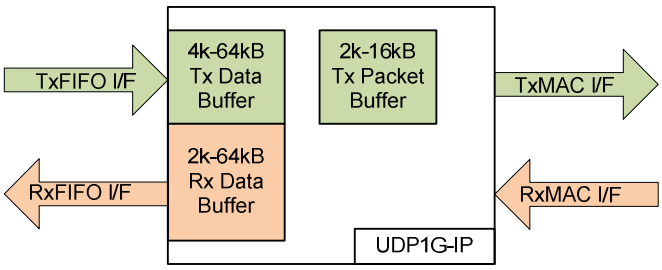


**Figure 2: Adjustable Tx/Rx Buffer Size**

User sets the size of three buffers in UDP1G-IP (Tx Data buffer, Tx Packet buffer, and Rx Data buffer) by defining as constant value to UDP1G-IP HDL code. The different size is provided to optimize resource utilization for user application. The recommended value of Tx Data buffer size and Tx Packet buffer size depends on Tx packet size (PKL register from user). Tx Packet Buffer must be more than the Tx packet size while Tx Data buffer size should be at least two times of the Tx packet size.

For Rx data buffer, the size should be at least two times of received packet size for storing new receive packet and sending out data to user at the same time.

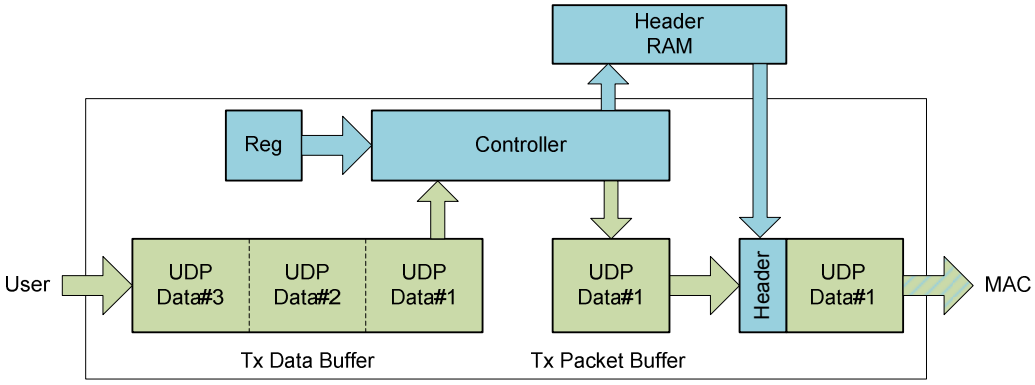


**Figure 3: Transmit Data Flow**

To transmit data, data from Tx Data buffer is split into packet size to store in Tx Packet buffer. Data output from Tx Packet buffer is combined with header data in Header RAM before sending out to EMAC. UDP and IP checksum are auto calculated within UDP1G-IP. Busy flag of UDP1G-IP is de-asserted to '0' after sending all data. User sets total size of transmit data and packet size through register interface.

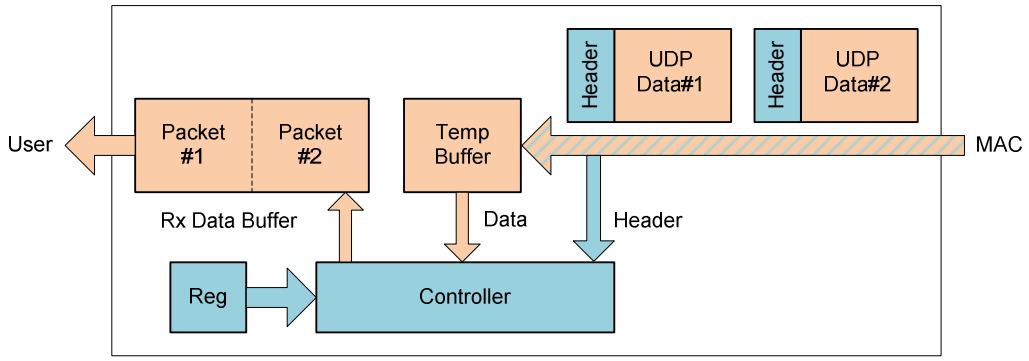


**Figure 4: Receive Data Flow**

When new packet is received from MAC, received packet is stored to temp buffer firstly. Header and checksum of Rx packet are verified. If network parameters in the header do not match to set value from user or checksum of received packet is error, Rx packet will be ignored. Data of ignored packet from Temp Buffer does not store to Rx Data buffer. Only data of valid packet is extracted and stored to Rx Data buffer.

## Functional Description

UDP1G-IP core can be divided into three parts, i.e. control block, transmit block, and receive block.

### Control Block

- **Reg**

User sets parameters for UDP/IP operation by using register interface. Register address of this interface is 4-bit (maximum register = 16). The description of each register is defined as shown in Table 2**.** After system reset is released to '0', all internal parameters are loaded to internal signals.

- **UDPIP Controller**

After IP reset is changed from '1' to '0', UDP1G-IP starts initialization process which can be selected in two modes (set by user through register interface), i.e. server mode and client mode. In client mode, UDP1G-IP sends ARP request to extract Target MAC address from ARP reply. In server mode, UDP1G-IP waits ARP request from the target to get MAC address, and then returns ARP reply to complete intialization process.

After complete initialization process, UDP1G-IP is in Idle status to wait new command (Send data command) from user and ready to extract data from received packet (Receive direction).

**Table 2: Register map Definition**

| RegAddr [3:0] | Reg Name | Dir | Bit | Description |
|---|---|---|---|---|
| 0000b | RST | Wr /Rd | [0] | Reset IP. '0': Release reset, '1': Reset. Default value is '1'.<br>**After all parameters are assigned, user sets '0' to this register to load parameters and start system initialization. Reset needs to be set/clear again to reload parameter if user changes the value of SML, SMH, DIP, SIP, DPN, or SPN register.** |
| 0001b | CMD | Wr | [0] | User command. Set '1' to send data.<br>**Before setting this register, user needs to confirm system busy flag = '0' by reading bit[0] of this register or checking from Busy output signal.** |
| | | Rd | [0] | IP busy flag. '0': Idle, '1': IP is in initialization or IP is in sending data operation.<br>Mapped to Busy output signal. |
| 0010b | SML | Wr /Rd | [31:0] | Define 32-bit lower MAC address (bit [31:0]) for this IP.<br>**User needs to set this register before clearing RST register.** |
| 0011b | SMH | Wr /Rd | [15:0] | Define 16-bit upper MAC address (bit [47:32]) for this IP.<br>**User needs to set this register before clearing RST register.** |
| 0100b | DIP | Wr /Rd | [31:0] | Define 32-bit target IP address.<br>**User needs to set this register before clearing RST register.** |
| 0101b | SIP | Wr /Rd | [31:0] | Define 32-bit IP address for this IP.<br>**User needs to set this register before clearing RST register.** |
| 0110b | DPN | Wr /Rd | [31:0] | [15:0]-Define 16-bit target port number for IP sending data.<br>[31:16]-Define 16-bit target port number for IP receiving data.<br>**User needs to set this register before clearing RST register.** |
| 0111b | SPN | Wr /Rd | [15:0] | Define 16-bit port number for this IP.<br>**User needs to set this register before clearing RST register.** |

| RegAddr [3:0] | Reg Name | Dir | Bit | Description |
|---|---|---|---|---|
| 1000b | TDL | Wr | [31:0] | Total Tx data length transfer in byte unit. Valid from 1-0xFFFFFFFF.<br>**User needs to set this register before setting CMD register='1'. This value are loaded when CMD register is set. User can prepare the new value for next transmit after UDP1G-IP starts sending operation.**<br>If the next data transmission uses the same length, this register will not need to set again. UDP1G-IP uses the lastest value for the next data transmission. |
| | | Rd | [0] | Remaining data transfer length in byte unit which still not transmit. |
| 1001b | TMO | Wr | [31:0] | Define timeout value for waiting ARP reply packet after sending ARP request. The counter runs by using 125 MHz, so timer unit is about 8 ns. Set value should be more than 0x6000. |
| | | Rd | | [0]-Timeout from not receiving ARP reply packet<br>After timeout, IP resends ARP request until ARP reply is received.<br>[8]-Rx packet ignored because of Rx data buffer full<br>[9]-Rx packet ignored because of checksum failed<br>[10]-Rx packet ignored because of MacRxUser error |
| 1010b | PKL | Wr /Rd | [15:0] | Data size of Tx packet in byte unit. Valid from 1-16000. Default value is 1472 byte (Maximum size for non-jumbo frame).<br>**This value must not be changed when data transmission still not complete (Busy='1'). If the next data transmission uses same packet size, user will not need to set this register. The IP loads the previous value from latched register.** |
| 1110b | SRV | Wr/ Rd | [0] | '0': Client mode. The IP sends ARP request to get Target MAC address from IP address after IP reset is released to '0'. After IP receives ARP reply, IP busy is de-asserted to '0'.<br>'1': Server mode. The IP waits ARP request from the Target to get Target MAC address after IP reset is released to '0'. After IP receives ARP request and returns ARP reply, IP busy is de-asserted to '0'.<br>Default value is '0' (Client mode)<br>**Note: In Server mode, after UDP1G-IP is reset, the Target needs to resend ARP request to UDP1G-IP to complete IP initialization.** |

**Table 3: TxBuf/TxPac/RxBufBitWidth Parameter description**

| Value of BitWidth | Buffer Size | TxBufBitWidth | TxPacBitWidth | RxBufBitWidth |
|---|---|---|---|---|
| 11 | 2kByte | No | Valid | Valid |
| 12 | 4kByte | Valid | Valid | Valid |
| 13 | 8kByte | Valid | Valid | Valid |
| 14 | 16kByte | Valid | Valid | Valid |
| 15 | 32kByte | Valid | No | Valid |
| 16 | 64kByte | Valid | No | Valid |

**Transmit Block**

- **Tx Data Buffer**

This buffer size is set by "TxBufBitWidth" parameter of the IP. The valid value is 12-16 which is equal to the address size of buffer, as shown in Table 3.

The buffer size should be at least two times of Tx Packet Size in PKL register. During sending operation, one packet data is forwarded from this buffer to Tx Packet buffer. At the same time, next packet is received from user logic. Data in Tx Data buffer is flushed after the data is forwarded to EMAC. If Tx Data buffer size is much enough (more than two times of Tx Packet size), current packet will be transmitted to EMAC continuously. This buffer is also used to be data buffer between user logic and UDP1G-IP for sending data operation.

- **Tx Packet Buffer**

The size is set by "TxPacBitWidth" parameter of the IP. The valid value is 11-14 and the description of the parameter is shown in Table 3. This buffer size must be more than Tx Packet size + 4 to store at least one packet data splitting from Tx Data Buffer (Tx Packet size is set by PKL register). Tx Packet buffer stores data about 1-2 packet sizes. There is no advantage to set Tx Packet buffer size more than two times of Tx Packet size.

- **Header RAM**

This RAM is applied to store header part of Transmit packet. The header part consists of network parameters which are set by user through register interface and packet checksum which is calculated by TxCsum block.

- **TxCsum**

This module is designed to calculate checksum of Tx packet before sending out. After complete to calculate checksim, the checksum output is loaded to Header RAM to be the header of Tx packet.

- **TxDataMux**

This module is desigend to merge header from Header RAM and data from Tx Packet buffer. The ethernet packet including header and data is forwarded to EMAC.

**Receive Block**

- **Rx Buffer**

  This is temporary buffer to store Rx packets from EMAC. The objective of this buffer is to wait Header Checker to validate the received packet before forwarding to Rx Data buffer. Only UDP data of valid packet is extracted and forwarded to Rx Data buffer.

- **Header Checker**

  Header in Rx packet consists of network parameters and checksum value. If some network parameters is not matched to the set value in register or checksum is not correct, Rx packet will be rejected.

- **Rx Data Buffer**

  This buffer size is set by “RxBufBitWidth” parameter of the IP. The valid value is 11-16 which more details are shown in Table 3. This is the data buffer between user logic and UDP1G-IP for receiving data operation. If buffer is full, new received packet will be ignored. It is recommended to set Rx data buffer size to be more than or equal to two times of received packet size.

## User Block

This block is user module for setting and monitoring register interface, writing data to Tx FIFO, and reading data from Rx FIFO. This module can be designed by simple hardware logic.

## Triple-Speed Ethernet MAC

This block is softIPcore provided by Intel. Please read more details from following website.
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-ethernet-mac.html

## Core I/O Signals

Descriptions of all parameters and signal I/O are provided in Table 4 and Table 5. MAC Interface signals are Avalon stream interface to connect with Intel Triple Speed EMAC.

**Table 4: Core Parameters**

| Generic Name | Value | Description |
|---|---|---|
| TxBufBitWidth | 12-16 | Setting Tx Data buffer size. The value is referred to address bus size of this buffer. |
| TxPacBitWidth | 11-14 | Setting Tx Packet buffer size. The value is referred to address bus size of this buffer. |
| RxBufBitWidth | 11-16 | Setting Rx Data buffer size. The value is referred to address bus size of this buffer. |

**Table 5: Core I/O Signals**

| Signal | Dir | Clk | Description |
|---|---|---|---|
| Common Interface Signal | | | |
| RstB | In | | Reset IP core. Active Low. |
| Clk | In | | 125 MHz fixed clock frequency input for user interface and MAC transmit interface for 1 Gbps mode |
| User Interface | | | |
| RegAddr[3:0] | In | Clk | Register address bus |
| RegWrData[31:0] | In | Clk | Register write data bus. Synchronous to RegAddr signal for write process. |
| RegWrEn | In | Clk | Register write enable pulse. Synchronous to RegAddr and RegWrData signals. |
| RegRdData[31:0] | Out | Clk | Register Read data bus. Available the next clock after RegAddr is valid. |
| Busy | Out | Clk | IP busy status (‘0’-Idle, ‘1’-IP Intialization or IP sending data).<br>This signal is mapped to bit0 of CMD register. |
| IntOut | Out | Clk | Assert to high for 1 clock cycle when timeout is detected or Rx packet is ignored. More details of Interrupt status could be checked from TMO register. |
| Tx Data Buffer Interface | | | |
| UDPTxFfFull | Out | Clk | Transmit buffer full flag. User needs to stop writing data within 4 clock cycles after this flag is asserted to high. |
| UDPTxFfWrEn | In | Clk | Transmit buffer write enable. Assert to ‘1’ to write data to Transmit buffer. |
| UDPTxFfWrData[7:0] | In | Clk | Transmit buffer write data bus. Synchronous with UDPTxFfWrEn. |
| Rx Data Buffer Interface | | | |
| UDPRxFfRdCnt[15:0] | Out | Clk | Received buffer data counter to show total received data in buffer. |
| UDPRxFfRdEmpty | Out | Clk | Received buffer empty flag. User needs to stop reading data immediately when this signal is asserted to ‘1’. |
| UDPRxFfRdEn | In | Clk | Received buffer read enable. Assert to ‘1’ to read data from Received buffer. |
| UDPRxFfRdData[7:0] | Out | Clk | Received buffer read data bus. Valid in the next clock after UDPRxFfRdEn asserts to ‘1’. |

| MAC Interface | | | |
|---|---|---|---|
| MacTxSOP | Out | Clk | Transmit start of packet. Assert this signal to '1' when the first byte in the frame is driven on MacTxData. |
| MacTxData[7:0] | Out | Clk | Transmit data. |
| MacTxEOP | Out | Clk | Transmit end of packet. Assert this signal to '1' when the last byte in the frame is driven on MacTxData. |
| MacTxValid | Out | Clk | Transmit data valid. Assert this signal to '1' to indicate that the data on the following signals are valid: MacTxSOP, MacTxData, and MacTxEOP. |
| MacTxReady | In | Clk | Mac ready. Assert this signal to '1' to indicate that MAC is ready to accept data. MacTxReady must not be de-asserted to '0' during packet transmission. |
| MacRxClk | In | | Receive clock from MAC. |
| MacRxSOP | In | MacRxClk | Receive start of packet. This signal is unused. |
| MacRxData[7:0] | In | MacRxClk | Receive data. |
| MacRxEOP | In | MacRxClk | Receive end of packet. Assert to '1' when the last byte of frame is driven on MacRxData. |
| MacRxValid | In | MacRxClk | Receive data valid. Assert to '1' when data on the following signals are valid: MacRxSOP, MacRxData, and MacRxEOP. During one packet transmission, MacRxValid must be asserted to '1' to send data continuously. |
| MacRxError | In | MacRxClk | Receive error. Assert to '1' with the final byte in the frame to indicate that receive frame has the error. |
| MaxRxReady | Out | RxClk | Assert to '1' when UDP1G-IP is ready to receive data from MAC. MacRxReady is de-asserted to '0' when time gap between each received packet is less than 3 clock cycles. More details is shown in Figure 13 and Figure 14. |

## Timing Diagram

**IP Initialization**

For initialization process after user changes value of RST register from ‘1’ to ‘0’, UDP1G-IP can operate in two modes depending on SRV register setting, i.e. client mode and server mode.

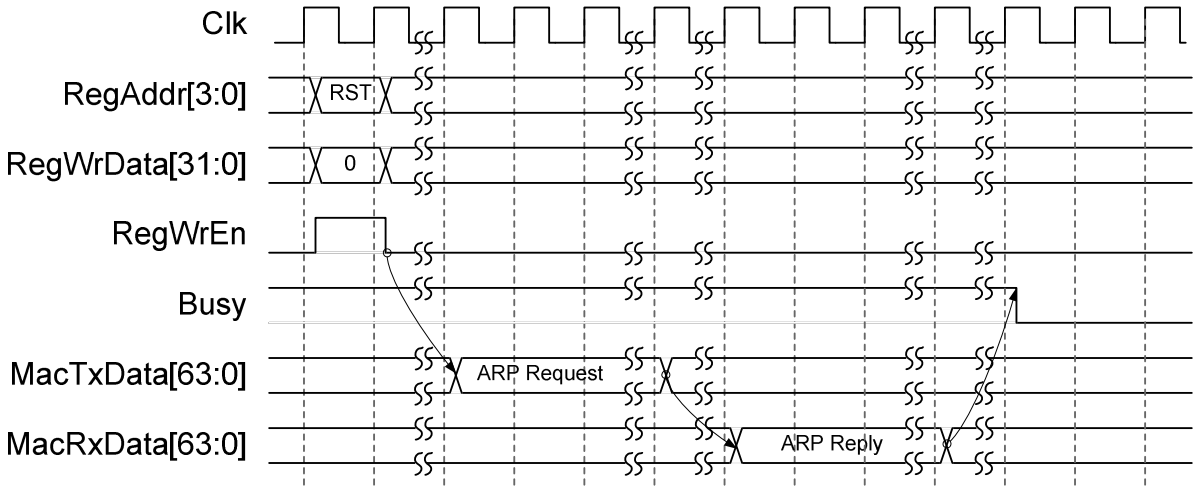


**Figure 5: IP Initialization in client mode**

In client mode, UDP1G-IP sends ARP request and waits ARP reply from the target. Target MAC address is extracted from ARP reply packet. After that, busy signal is de-asserted to ‘0’.

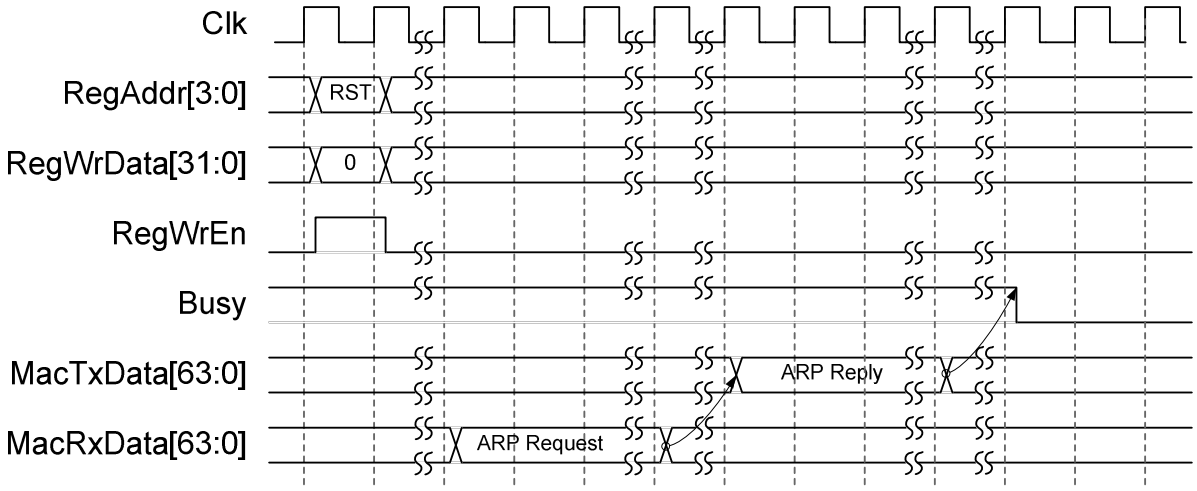


**Figure 6: IP Initialization in server mode**

In server mode, after UDP1G-IP reset is released to ‘0’, UDP1G-IP waits ARP request from the target. After receiving ARP request which the header is matched to setting value, UDP1G-IP returns ARP reply to the target. Target MAC address is extracted from ARP request packet. Final stage, busy signal is de-asserted to ‘0’.

**Register Interface**

User can write/read control signals of UDP1G-IP through Register interface. Timing diagram of register interface is shown in Figure 7. Register map address is defined as shown in Table 2. To write register, user sets RegWrEn='1' with valid value of RegAddr and RegWrData. To read register, user sets RegAddr value and then RegRdData is valid in the next clock.

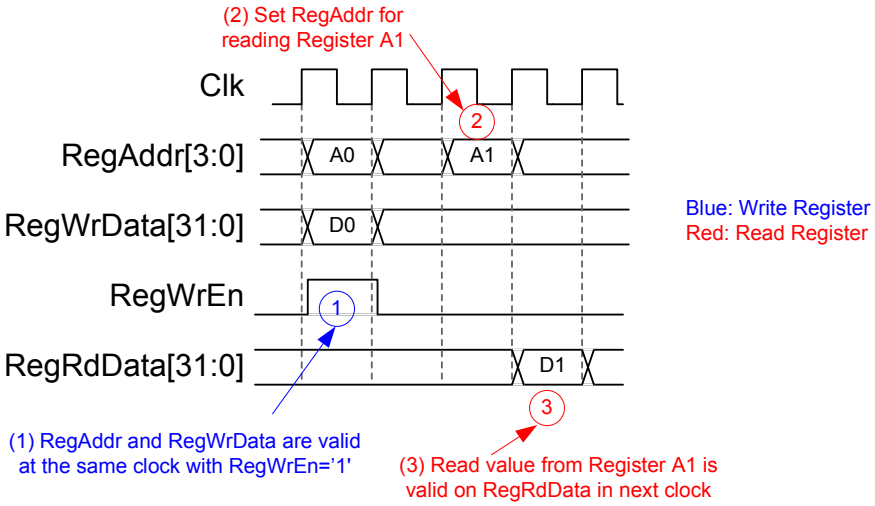


**Figure 7: Register Interface Timing Diagram**

Before starting sending data operation, busy flag must be monitored that it is equal to '0' (IP is in Idle status). After CMD register is set, busy flag is asserted to '1', as shown in Figure 8. Busy is de-asserted to '0' when send data command is completed.

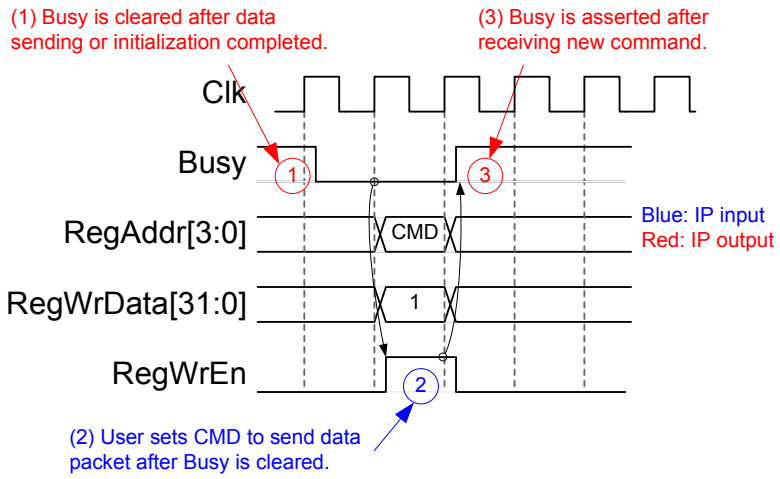


**Figure 8: Set CMD register when Busy is de-asserted**

**Tx FIFO Interface**

User sends data to UDP1G-IP by using FIFO interface, as shown in Figure 9. Before sending data, user needs to check full flag (UDPTxFfFull) that is not asserted to ‘1’. Then, set UDPTxFfWrEn=’1’ with valid value of UDPTxFfWrData. UDPTxFfWrEn must be de-asserted to ‘0 within 4 clock cycles to stop data sending after UDPTxFfFull is asserted to ‘1’. During IP is in reset condition, UDPTxFfFull is asserted to ‘1’ and all data in FIFO are flushed.

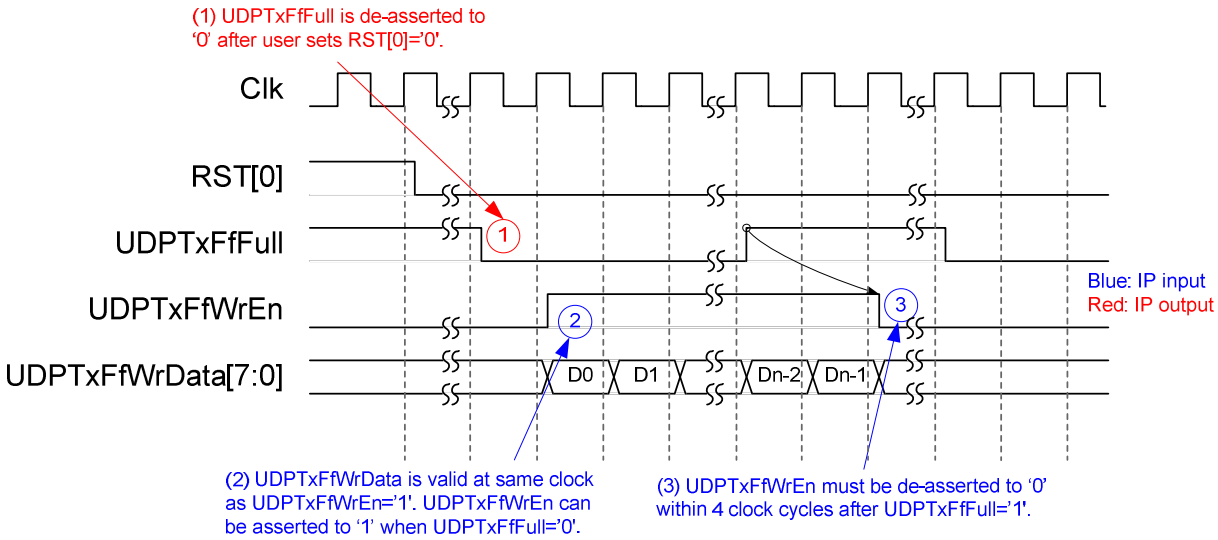


**Figure 9: Tx Data Buffer Interface Timing Diagram**

**Rx FIFO Interface**

After the received data extracted from valid received packet is stored in Rx Data buffer, UDPRxFfEmpty is de-asserted to '0'. When user logic detects that new data is received by monitoring UDPRxFfEmpty signal, UDPRxFfRdEn could be asserted to '1' to read data through RxFIFO interface, as shown in Figure 10. UDPRxFfRdData is valid in the next clock. When UDPRxFfEmpty = '1', UDPRxFfRdEn must be de-asserted to '0' at the same clock to stop data reading process. All data in Rx data buffer are flushed when IP is reset. UDPRxFfEmpty is asserted to '1' during reset condition.

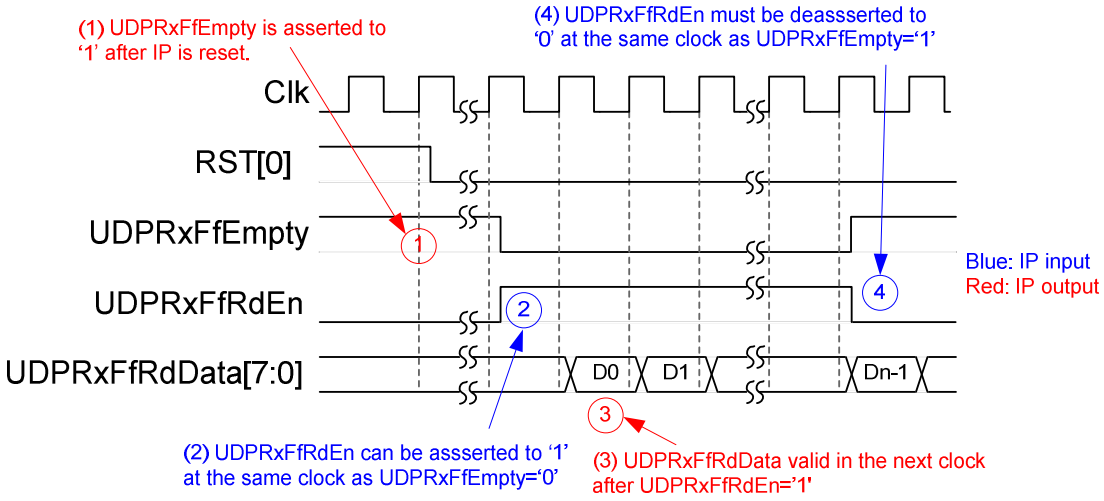


**Figure 10: Rx Data Buffer Interface by Empty flag Timing Diagram**

In addition, Rx data buffer status can be monitored by using UDPRxFfRdCnt. RdCnt is used when the logic to read data is designed as burst transfer. RdCnt shows total data in Rx data buffer. So, user asserts UDPRxFfRdEn='1' for many clocks to read more than one data from the buffer, as shown in Figure 11.

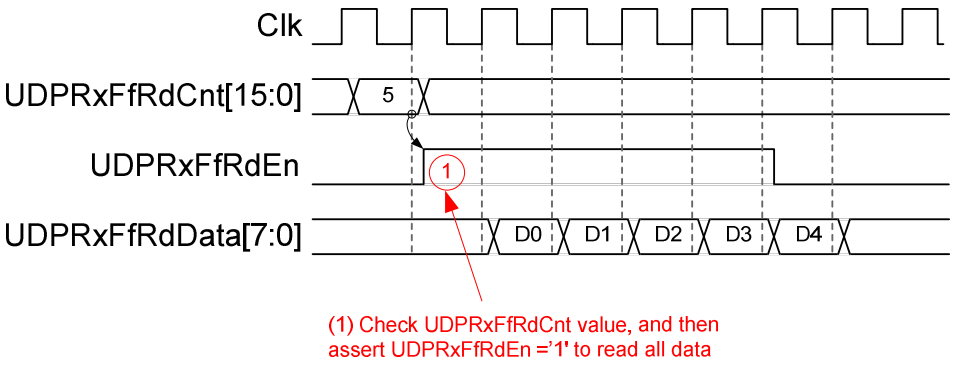


**Figure 11: Rx Data Buffer Interface by Read counter Timing Diagram**

**EMAC Interface**

To transmit packet, the IP asserts MacTxSOP and MacTxValid with the first data of the packet on MacTxData. The signal is latched until MacTxReady is asserted to '1' to acknowledge data transmit request. MacTxReady must be asserted to '1' until the packet is end of transmission. MacTxValid is asserted to '1' to send data on MacTxData continuously until the end of the packet. MacTxEOP and MacTxValid are asserted to '1' with the last transmit data to show end-of-packet status.

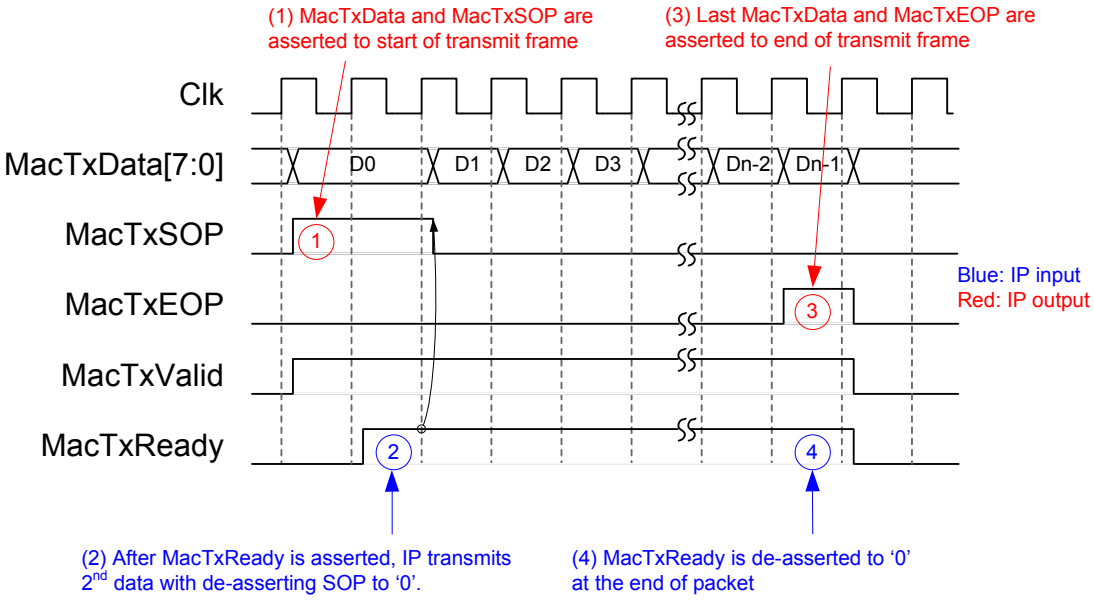


**Figure 12: Transmit EMAC Interface**

Figure 13 shows timing diagram of received side of Intel EMAC. UDP1G-IP monitors start of received frame from MacRxValid which changes from '0' to '1'. MacRxData is received continuously until end of packet though the last data is transferred with MacRxValid='0' and MacRxEOP='1'. The real end-of-frame is found when MacRxEOP='1' and MacRxValid='1'. At the real end-of-frame, MacRxError is read to check packet status. If MacRxValid gap size between the last data available and the real end-of-frame is more than or equal to 3 clock cycles, MacRxReady will be always asserted to '1'.

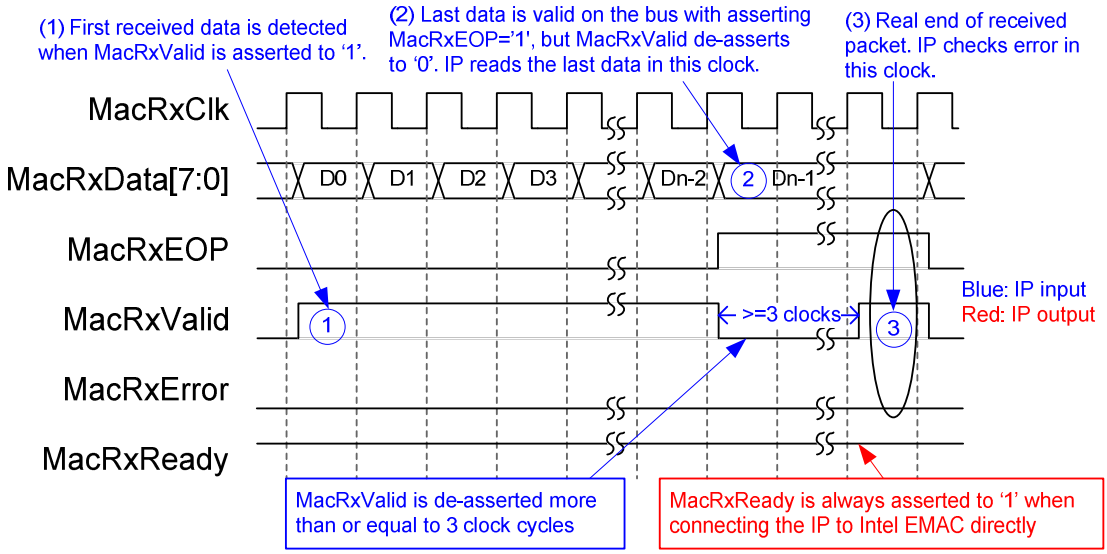


**Figure 13: Receive EMAC Interface when connecting UDP1G-IP to Intel EMAC**

In some user application, UDP1G-IP does not connect to Intel EMAC directly, but connects to other module through Avalon-ST bus instead. In this situation, MacRxValid may be asserted to '1' continuously until end of packet. UDP1G-IP requires at least 3 clock cycles as minimum gap size between each received packet, so MacRxReady is de-asserted to '0' for 3 clock cycles at the end-of-frame, as shown in Figure 14.

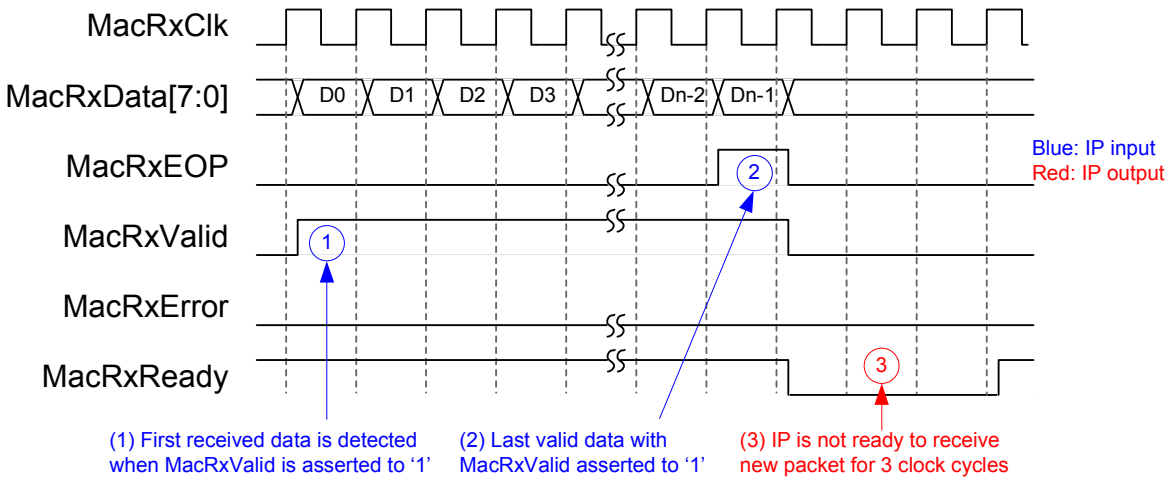


**Figure 14: Receive EMAC Interface when connecting UDP1G-IP to other Avalon-ST modules**

## Example usage

**Client mode (SRV[0]='0')**

The example of the sequence to set register for data transmission and reception in client mode is shown as follows.

1) Set RST register='1' to reset the IP.

2) Set SML/SMH for MAC address, DIP/SIP for IP address, and DPN/SPN for port number.

3) Set RST register='0' to clear the reset. IP starts initialization by sending ARP request to get Target MAC address from ARP reply. After end of intialization, busy signal is cleared to '0'.

4) a. For data transmission, set TDL for total transfer length and PKL for packet size. Next, set CMD register to start data transmission. User sends data to TxFIFO and monitors busy flag until it is equal to 0'. After complete the command, user can change TDL/PKL value for new data transmission without IP reset.

b. For data reception, user monitors RxFIFO status and reads data when RxFIFO is not empty.

**Server mode (SRV[0]='1')**

The different point between server mode and client mode is the initialization process to get MAC address of the target. In client mode, MAC address is extracted from ARP reply after UDP1G-IP sends ARP request. In server mode, MAC address is extracted from ARP request which has matched Target IP address. The process to send and receive data is same as client mode. The example sequence of server mode is shown as follows.

1) Set RST register='1' to reset the IP.

2) Set SML/SMH for MAC address, DIP/SIP for IP address, and DPN/SPN for port number.

3) Set RST register='0' to clear the reset. IP starts initialization by waiting ARP request to get Target MAC address. Next, IP creates ARP reply to the Target. After end of intialization, busy signal is cleared to '0'.

4) Data process is same as client mode.

## Verification Methods

UDP1G-IP Core functionality was verified by simulation and also proved on real board design by using CycloneV E/ArriaV GX/Arria10 SoC development board.

## Recommended Design Experience

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

## 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 | Feb-27-2017 | New release |
| 1.1 | Mar-1-2017 | Update Figure11 |
| 1.2 | Aug-10-2018 | Add SRV register and MacRxReady |