**Design Gateway Co.,Ltd**
54 BB Building 13th Fl., Room No.1302 Sukhumvit 21 Rd. (Asoke), Klongtoey-Nua, Wattana, Bangkok 10110
Phone: (+66) 02-261-2277
Fax: (+66) 02-261-2290
E-mail: sales@design-gateway.com
URL: www.design-gateway.com

## Core Facts

| Provided with Core | |
|---|---|
| Documentation | User Guide, Design Guide |
| Design File Formats | NGC Netlist |
| Constraints Files | User constraint file |
| Verification | Test Bench, Simulation Library |
| Instantiation Templates | VHDL |
| Reference Designs & Application Notes | EDK Project, See Reference Design Manual |
| Additional Items | Demo on SP605, AC701, VC707 |
| **Simulation Tool Used** | |
| ISim/Vivado Simulator/ModelSim | |
| **Support** | |
| Support Provided by Design Gateway Co., Ltd. | |

## Features

- TCP/IP stack implementation
- Support IPv4 protocol
- Support one port connection
- Support both Server and Client mode (Passive/Active open and close)
- Support unidirection data transfer for specific time
- Support Jumbo frame
- Transmit/Receive buffer size, adjustable for optimized resource and performance
- Simple data interface by standard FIFO interface
- Simple control interface by standard register interface
- One clock domain interface by fixed 125 MHz clock frequency
- Reference design available on SP605/AC701/VC707 evaluation board

**Table 1: Example Implementation Statistics**

| Family | Example Device | Fmax (MHz) | Slice FFs | Slice LUTs | Slices[1] | IOB[2] | RAMB36E1 | RAMB18E1 | Design Tools |
|---|---|---|---|---|---|---|---|---|---|
| Spartan-6 | XC6SLX45T-3FF1156 | 125 | 2717 | 2878 | 1132 | 135 | - | 73 | ISE® 14.6 |
| Artix-7 | XC7A200T-2FBG676 | 125 | 2702 | 3040 | 1223 | 135 | 1 | 72 | ISE® 14.6 |
| Virtex-7 | XC7VX485T-2FFG1761 | 125 | 2691 | 3218 | 1340 | 135 | 1 | 72 | ISE® 14.6 |

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

2) Assuming all core I/Os and clocks are routed off-chip

3) Block memory resources 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.

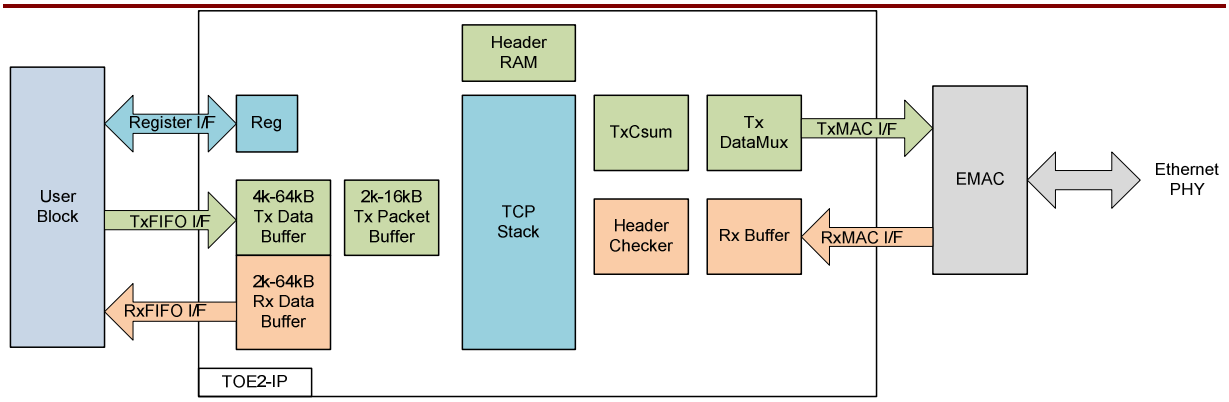


**Figure 1: TOE2-IP Block Diagram**

## Applications

TOE2-IP is designed for network application using TCP/IP protocol with high speed bandwidth transfer such as network data storage, IP camera, Printer server. By using this IP, user can easily transfer data with any device through TCP/IP protocol without CPU usage in system.

## General Description

TOE2-IP core operating with Xilinx EMAC IP core can operate TCP/IP stack, Transport layer, Internet layer, and Link layer for network data transmission. User can send and receive data with any network device through TCP/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 received data signal by FIFO access. During initializing system, user needs to set up system parameter such as packet size, port number, IP number through register interface. Packet data transferring can be operated by two modes, i.e. Active mode and Passive mode. On active mode, user can control through register interface for opening connection, closing connection, and sending data from Tx Data buffer to external network device. On passive mode, port connection will be opened, closed from external device. Also, data from external device will be stored to Rx data buffer within TOE2-IP.

In release items, it includes the netlist files for Tx Data buffer, Tx Packet buffer, and Rx Data buffer which can support many sizes. There are five sizes for Tx Data buffer, four sizes for Tx Packet bufer, and six sizes for Rx Data buffer. The different size is provided to optimize resource utilization for user application. The bigger size takes much resource, but achieve the best transfer performance. Tx Data buffer size and Tx Packet buffer size are effect to transmit performance, while Rx Data buffer is effect to receive performance. Otherwise, Tx Data buffer size and Tx Packet buffer size are related to the packet size which user can be programmed through register interface. 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..

To transmit data, data from Tx Data buffer will be split into packet size and then fed to Tx Packet buffer. Data output from Tx Packet buffer will be combined with header data in Header RAM before sending out to EMAC. TCP and IP checksum will be auto calculated within TOE2-IP. Acknowledge packet at Rx side will be monitored to make decision to send next data packet or resend data packet in Tx buffer. Busy flag within register will be cleared after completed data transfer size is equal to setting value from user. User can monitor this busy flag to check transfer status.

For receiving data, Rx packet will be stored to temp buffer firstly. Header and checksum within Rx packet will be verified. If header or checksum is error, the packet will be rejected and not store to Rx Data buffer. When correct data packet is received, data will be stored to Rx Data buffer and Acknowledge packet will be sent out from TOE2-IP to request more data packet from external network device. TOE2-IP will go back to Idle state (Busy flag='0') when no more packet is received and received packet sequence is correct.

When TOE2-IP is in Idle state (no more data transmit or received), data direction can be switched without closing the port. Also, user can change packet size and total transfer size for new transmit without closing the port if the IP is Idle state.

## Functional Description

TOE2-IP core can be divided into three parts, i.e. control block, transmit block, and received block.

**Control Block**

- **Reg**

User can set parameters for TCP/IP operation by using register interface. Register address of this interface is equal to 4-bit for 11 registers. The description of each register address is defined as shown in Table 2. After system reset is released, all internal parameter will be updated by setting value in each register.

- **TCP Stack**

To operate active command from user, TCP Stack will decode user command and start transmit block to transmit packet out. Also, TCP Stack monitors receive packet from received block for Acknowledge packet.

To operate passive command from external device, TCP Stack receives packet from received block, and then decode it. After that, TCP Stack will start transmit block to transmit Acknowledge packet out.

**Table 2: Register map Definition**

| RegAddr [3:0] | Reg Name | Dir | Bit | Description |
|---|---|---|---|---|
| 0000b | RST | Wr | [0] | Reset IP. '0': Release reset, '1': Reset. Default value is '1'.<br>**After user setting all parameters, set '0' to this register to release reset and start system parameter intialization. Reset needs to be set again if user will change value of SML, SMH, DIP, SIP, DPN, or SPN register.** |
| 0001b | CMD | Wr | [1:0] | User Command in active mode. "00": Send data, "10": Open connection (active), "11": Close connection (active), "01": Undefined.<br>**Before setting this register to start any active command, user needs to check system busy flag by reading bit[0] of this register to confirm that there is no operation running.**<br>Active command will auto-start after this register is set by user. |
| | | Rd | [0] | System busy flag. '0': Idle, '1': System operating. |
| | | | [3:1] | Current operation. "000": Send data, "001": Idle, "010": Active open connection,<br>"011": Active close connection, "100": Receive data, "101": Undefined,<br>"110": Passive open connection, "111": Passive close connection. |
| 0010b | SML | Wr | [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 | [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 | [31:0] | Define 32-bit target IP address.<br>**User needs to set this register before clearing RST register.** |
| 0101b | SIP | Wr | [31:0] | Define 32-bit IP address for this IP.<br>**User needs to set this register before clearing RST register.** |
| 0110b | DPN | Wr | [15:0] | Define 16-bit target port number.<br>**User needs to set this register before clearing RST register if user wants to use active open connection.**<br>Target port number will be auto defined from passive open connection packet which parameters in header are matched with setting value. |
| 0111b | SPN | Wr | [15:0] | Define 16-bit port number for this IP.<br>**User needs to set this register before clearing RST register.** |
| 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 = "00". This value will be latched to internal logic when CMD register is set. So, user can prepare the new value for next transmit after setting CMD register.**<br>If user will transmit data with same length, this register doesn't need to set again. Previous value will be used from internal latch. |
| | | Rd | [31:0] | Remain data transfer length in byte unit which still not transmit. |

| RegAddr [3:0] | Reg Name | Dir | Bit | Description |
|---|---|---|---|---|
| 1001b | TMO | Wr | [31:0] | Define timeout value for waiting Rx packet during running any command. This register is used by 125 MHz counter, so timer unit is about 8 ns. This value must be more than 0x6000. |
| | | Rd | | [0]-Timeout from not receiving ARP reply packet<br>[1]-Timeout from not receiving SYN and ACK flag during active open operation<br>[2]-Timeout from not receiving ACK flag during passive open operation<br>[3]-Timeout from not receiving FIN and ACK flag during active close operation<br>[4]-Timeout from not receiving ACK flag during passive close operation<br>[5]-Timeout from not receiving ACK flag during data transmit operation<br>[6]-Timeout from Rx packet lost, Rx data FIFO full, or wrong sequence number<br>[23]-Rx packet ignored because of Rx data buffer full<br>[27]-Rx packet lost detected<br>[30]-RST flag is detected in Rx packet |
| 1010b | PKL | Wr | [15:0] | Data length of Tx packet in byte unit. Valid from 1-16000. Default value is 1460 byte (Maximum size for non-jumbo frame).<br>**This value must not be changed during data transmission not complete (Busy='1'). If next transmit still use same packet size, user does not need to set this register because the previous value is latched in the logic.** |

Note:

1. Target Mac address is defined from returned value in ARP Reply packet, so user doesn't need to set this parameter.
2. Target Port number is defined from received packet when the port is opened in passive mode.

**Transmit Block**

- **Tx Data Buffer**

User can select five sizes, i.e. 4k, 8k, 16k, 32k, or 64kByte. The buffer size should be at least two times of Tx Packet Size in PKL register. Transmit data from user will be stored within this buffer. To send data output from user command, one packet data will be read out from this buffer to store to Tx Packet Buffer to wait next processing.

This buffer size relates to the total performance. TOE2-IP will transmit the data in Tx Data Buffer continuously without waiting acknowledge packet returned from the targer until no more data available. So, the overhead of each transmit packet will be less.

If user sends data more than the total transmit size, data will be available in the buffer for next transfer. The data will be flushed when the port is closed or reset is detected. If the data in the buffer is not enough for the current transaction, IP will not send out the packet and wait data from user. In this case, system will be busy state and not be ready to receive the data or any new request including closing the port from external device.

- **Tx Packet Buffer**

This buffer size must be more than or equal to Tx Packet size setting in PKL register. This size can be selected to be 2k, 4k, 8k, or 16kByte. 1-Packet Data output from Tx Data Buffer is splitted and store into Tx Packet Buffer to wait the EMAC and target ready before sending out.

- **Header RAM**

This RAM is applied to store header part of Transmit packet. Parameter in Header RAM will be updated by register value when user release RST register. Some parameters will be updated by ARP Reply or Passive open packet.

- **TxCsum**

This module is desigend to calculate checksum of Tx packet before sending out.

- **TxDataMux**

This module is desigend to combine header from Header RAM and data from Tx Packet Buffer, and then send out to EMAC.

**Received Block**

- **Rx Buffer**

This is temporary buffer to store each Rx packet from EMAC for waiting Header Checker processing.

- **Header Checker**

Header in Rx packet will be checked and compared with parameter in register. Packet will be ignored if any parameter is not matched or checksum is error. Only TCP data will be splitted out and store into Rx Data Buffer.

- **Rx Data Buffer**

This buffer is used to store received data and its size can be selcted to be 2k, 4k, 8k, 16k, 32k, or 64kByte. This buffer size will be mapped to be window size of this TCP connection. Setting bigger size to this buffer can increase receive data performance because the data source can continue sending data when data available in buffer without waiting the acknowledge returned from TOE2-IP which may be delayed from network routing or the process within the data source. Also, big buffer allows the chance of TOE2-IP to rearrange data sequence when the received packet sequence is not in correct sequence from network routing.

## 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.

## GEMAC

This block is softIPcore provided by Xilinx.

## Core I/O Signals

Descriptions of all signal I/O are provided in Table 3. All signals in MAC Interface group are designed to connect to Xilinx EMAC port directly.

**Table 3: 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. |
| User Interface | | | |
| RegAddr[3:0] | In | Clk | Register address bus |
| RegWrData[31:0] | In | Clk | Register Write data bus |
| RegWrEn | In | Clk | Register Write enable pulse. Assert with valid value of RegAddr and RegWrData signals. |
| RegRdData[31:0] | Out | Clk | Register Read data bus. Available after asserting RegAddr with 1 Clk period latency |
| ConnOn | Out | Clk | Connection Status (‘1’ : connection is opened, ‘0’: connection is closed) |
| TimerInt | Out | Clk | Timer interrupt. Assert to high for 1 Clk period when time out is detected. User can read TMO[6:0] register to check interrupt status. |
| Tx Data Buffer Interface | | | |
| TCPTxFfFlush | Out | Clk | Transmit buffer within IP is reset. Assert to high only 1 Clk period when connection is closed or reset. |
| TCPTxFfFull | Out | Clk | Transmit buffer full flag. User needs to stop writing data within 4 clock period after this flag is asserted to high. |
| TCPTxFfWrEn | In | Clk | Transmit buffer write enable. Assert to write data to Transmit buffer. |
| TCPTxFfWrData[7:0] | In | Clk | Transmit buffer write data bus. Synchronous with TCPTxFfWrEn. |
| Rx Data Buffer Interface | | | |
| TCPRxFfFlush | Out | Clk | Received buffer within IP is reset. Assert to high only 1 Clk period when connection is opened. |
| TCPRxFfRdCnt[15:0] | Out | Clk | Received buffer data counter to show total number of received data in buffer. |
| TCPRxFfRdEmpty | Out | Clk | Received buffer empty flag. User needs to stop reading data immediately. |
| TCPRxFfRdEn | In | Clk | Received buffer read enable. Assert to read data from Received buffer. |
| TCPRxFfRdData[7:0] | Out | Clk | Received buffer read data bus. Valid after TCPRxFfRdEn assert with 1 Clk period latency. |
| MAC Interface | | | |
| MacRxClk | In | | Received clock from EMAC core. |
| MacRxReset | In | | Active-High Rx software reset from EMAC core. This signal is unused. |
| MacRxData[7:0] | In | RxClk | Received data bus from EMAC core. |
| MacRxValid | In | RxClk | Received data valid signal from EMAC. Synchronous with MacRxData. |
| MacRxLast | In | RxClk | Control signal to indicate the final byte in the frame. |
| MacRxUser | In | RxClk | Control signal asserted at the end of received frame to indicate that the frame has an error. ‘0’: normal packet, ‘1’: error packet. |
| MacTxReset | In | | Active-High Tx software reset from EMAC core. This signal is unused. |
| MacTxData[7:0] | Out | Clk | Transmitted data to EMAC core. |
| MacTxValid | Out | Clk | Transmitted data valid signal to EMAC. Synchronous with MacTxData. |
| MacTxLast | Out | Clk | Control signal to indicate the final byte in a frame. |
| MacTxUser | Out | Clk | Control signal to indicate an error condition. This signal is always ‘0’. |
| MacTxReady | In | Clk | Handshaking signal. Asserted when MacTxData has been accepted. |

## Timing Diagram

User can write/read control signal with TOE2-IP by using Register interface which has timing diagram as shown in Figure 2. Register map address is designed as shown in Table 2. To write control signal, User needs to set RegWrEn='1' with valid value of RegAddr and RegWrData. To read control signal, User set only RegAddr value and then RegRdData will be valid in next clock period.

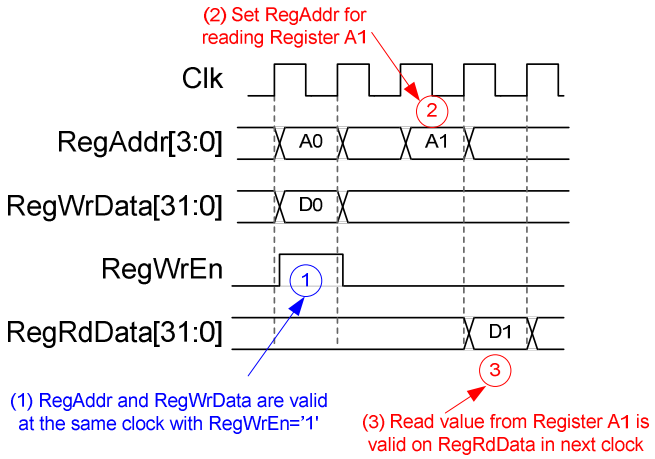


**Figure 2: Register Interface Timing Diagram**

User can send data to IP core by using FIFO interface, as shown in Figure 3. Before sending data, user needs to check full flag (TCPTxFfFull) that is not asserted to '1' and ConnOn is equal to '1'. Then, set TCPTxFfWrEn='1' with valid value of TCPTxFfWrData. TCPTxFfWrEn must be cleared within 4 clock period to stop data sending after TCPTxFfFull is asserted to '1'. TCPTxFfFlush will be asserted to '1' from IP core to inform user that all data in Tx FIFO are cleared which is caused from close connection detect.

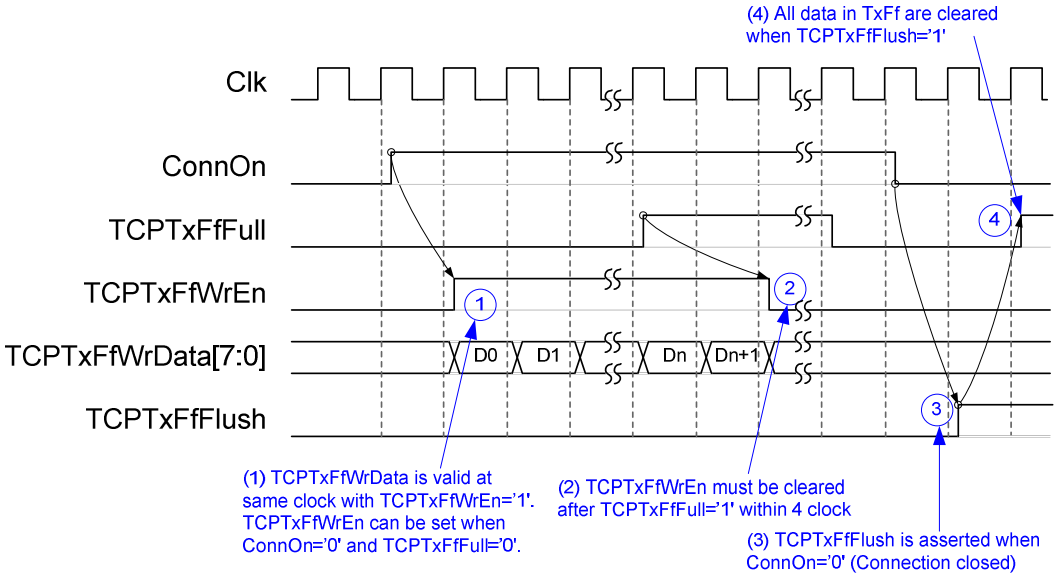


**Figure 3: Tx Data Buffer Interface Timing Diagram**

When IP core receives data from external, data will be stored in Rx Data buffer. User can read data from this buffer by using FIFO interface, as shown in Figure 4. User can monitor data available status from TCPRxFfEmpty. Data can be read when TCPRxFfEmpty is cleared to ‘0’. TCPRxFfRdEn can be set to ‘1’ to read data from Rx data buffer and TCPRxFfRdData will be valid in next clock period. Data reading must be stopped by clearing TCPRRxFfRdEn=’0’ at the same clock with TCPRxFfEmpty settting = ‘1’. Similar to Tx data buffer, all data in Rx data buffer wlil be flushed from open connection detect, so user can monitor flush status from TCPRxFfFlush signal.

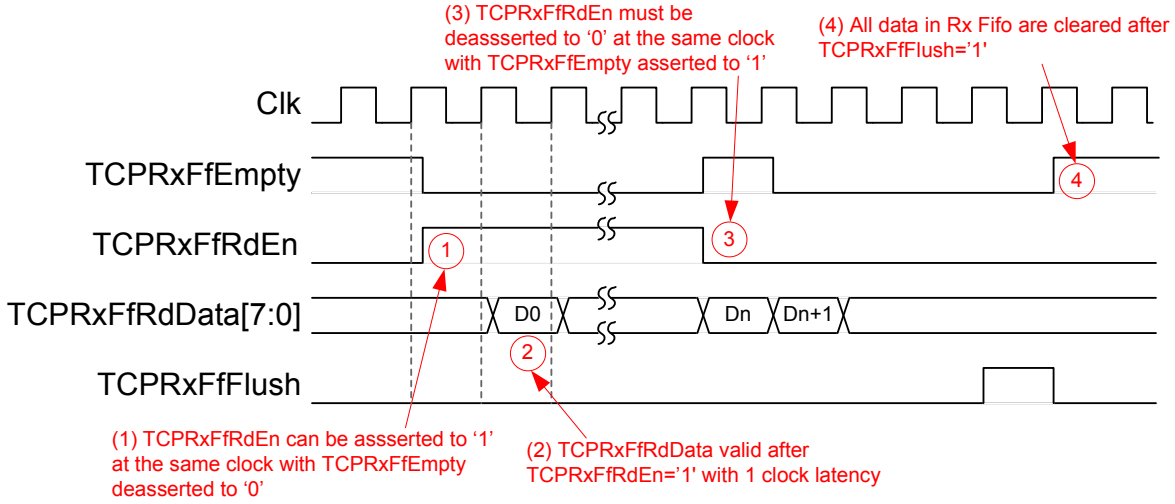


**Figure 4: Rx Data Buffer Interface by Empty flag Timing Diagram**

Rx data buffer status can be also monitored by using TCPRxFfRdCnt. This signal shows total number of available data in Rx data buffer. So, user can assert TCPRxFfRdEn=’1’ for time period equal to total number of data, as shown in Figure 5.

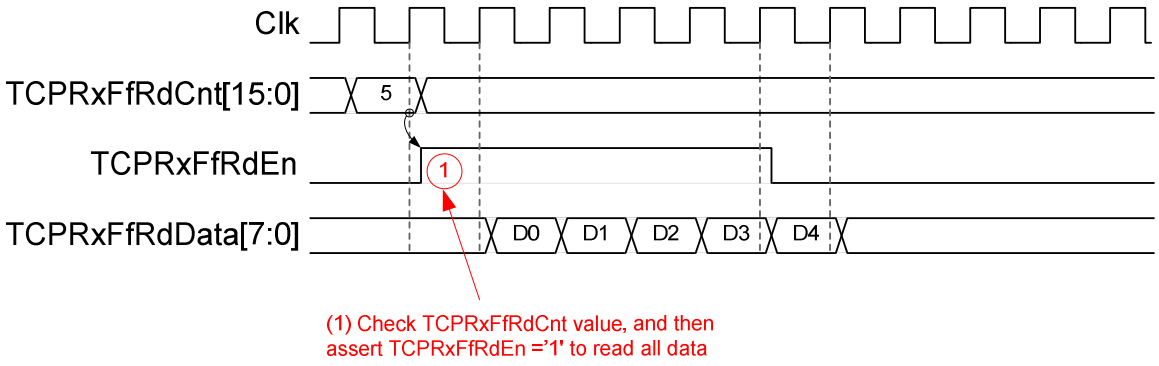


**Figure 5: Rx Data Buffer Interface by Read counter Timing Diagram**

## Verification Methods

The TOE2-IP Core functionality was verified by simulation and also proved on real board design by using SP605/AC701/VC707 evaluation board.

## Recommended Design Experience

User must be familiar with HDL design methodology to integrate this IP into their 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 | Dec-03-2012 | New release |
| 1.1 | Dec-11-2012 | Update IP port and buffer size description |
| 1.2 | Nov-19-2013 | Update Figure1 |
| 1.3 | Nov-28-2013 | Add AC701 Support |
| 1.4 | Mar-13-2014 | Update port name and add more description for Transmit operation |
| 1.5 | Apr-24-2014 | Add VC707 Support |