TOE-IP Reference Design Manual

Rev 2.1 9/22/2010

This document describes about TCP off-load engine IP Core reference design.

1. Recommended knowledge for this reference design

User must be have knowledge for understanding this reference design as follow.

1. Knowledge of how to use Xilinx FPGA implement tool "ISE".
2. Knowledge of how to use Xilinx FPGA implement tool "EDK"
3. Knowledge about C language.
4. Knowledge about TCP/IP.

2. Environment

This reference design is based on the following environment as shown in Figure1.

1. Xilinx FPGA evaluation board(ML506, ML605, SP605 or Spartan3A DSP 1800 board)
2. Xilinx Platform cable USB
3. Serial cable
4. ISE 11.5 / EDK 11.5
5. PC has Gigabit Ethernet Port for receiving TCP data. (Recommended spec is dual core processor over 2GHz.)
6. Ethernet cable(Cat5e or Cat6)

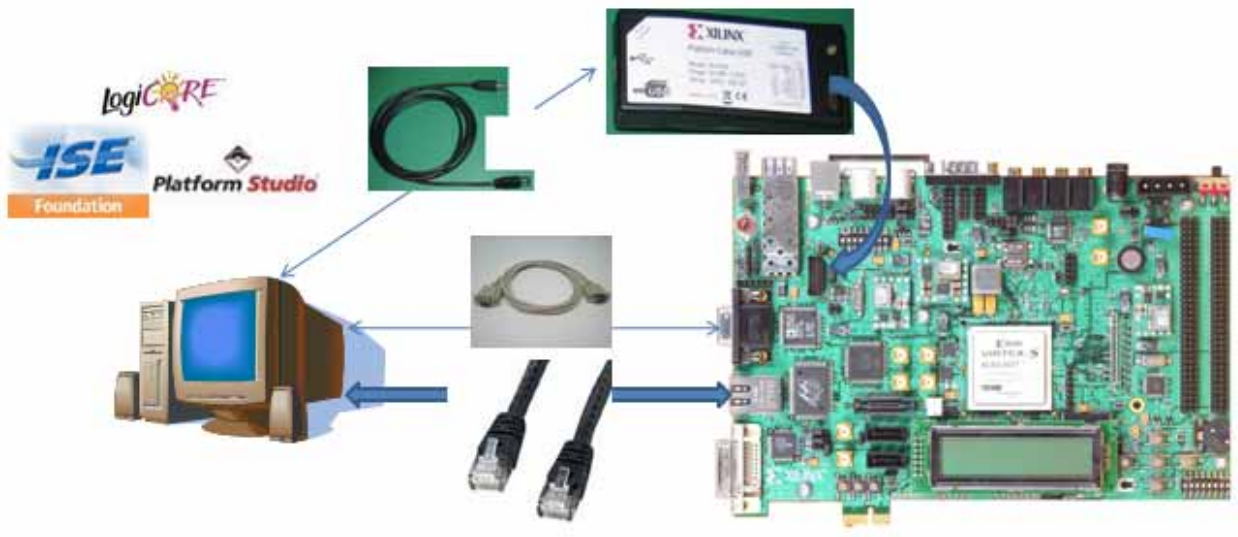

Figure 1: Reference design operation environment

3. Reference Design component

This reference design is built by both ISE and EDK. Customer must install both ISE and EDK on the PC to re-compile design.

Note that such FPGA device that does not include hardware TEMAC core internally such as Spartan-3A DSP, user must prepare Gigabit Ethernet MAC IP license from Xilinx for re-compilation.

Important files are as follows.

- ise/ise.xlise: ISE project file. To confirm/edit Reference design, open this file from ISE.
- system.xmp: EDK project file. User can refer to EDK design and TCP/IP stack sample software from EDK. Note that user must not open this file directly from EDK but open from ISE.
- ise/system_stub.vhd: This file describes hardware logic other than EDK part.
- test/recv_tcp_client.exe(test/recv_tcp_client.c) : TCP data receiver software for high speed. This software checks data pattern and verify with expected pattern to confirm correct data receive, and also displays data receive transfer speed.
- test/echo_tcp_client.exe(test/echo_tcp_client.c) : TCP data send/receive software for high speed. This software sends data to designated port by designated timing and receives data.
- ise/gemac_core.xco: CoreGenerator file of GbE MAC Core from Xilinx. This file may also not exist in case of using hardcore TEMAC.
- ise/gemac_core_block.vhd, gmii_if.vhd: Wrapper file and GMII interface logic of GbE MAC Core generated from gemac_core.xco.
- tcp_hw Folder: Folder that store TCP/IP sample stack software.

Refer to “User Guide” for IP-Core files.

4. Hardware description

TOE-IP Core is a hardware accelerator that processes some part of TCP operation, so that external host processor must process connection open/close, ACK process, another protocol except TCP.

If system uses OS (such as Windows or Linux) that already support TOE, user can apply this TOE-IP Core by simply building original driver. But if system OS does not support TOE or if system does not use OS itself, user must build TCP/IP stack or modify from existing TCP/IP for TOE-IP application.

This reference design uses MicroBlaze and can operate TCP data transmission without OS. So that this design includes simple TCP/IP stack, TOE-IP Core, GEMAC-IP bridge logic, frame receive logic, and bridge logic for MicroBlaze connection.

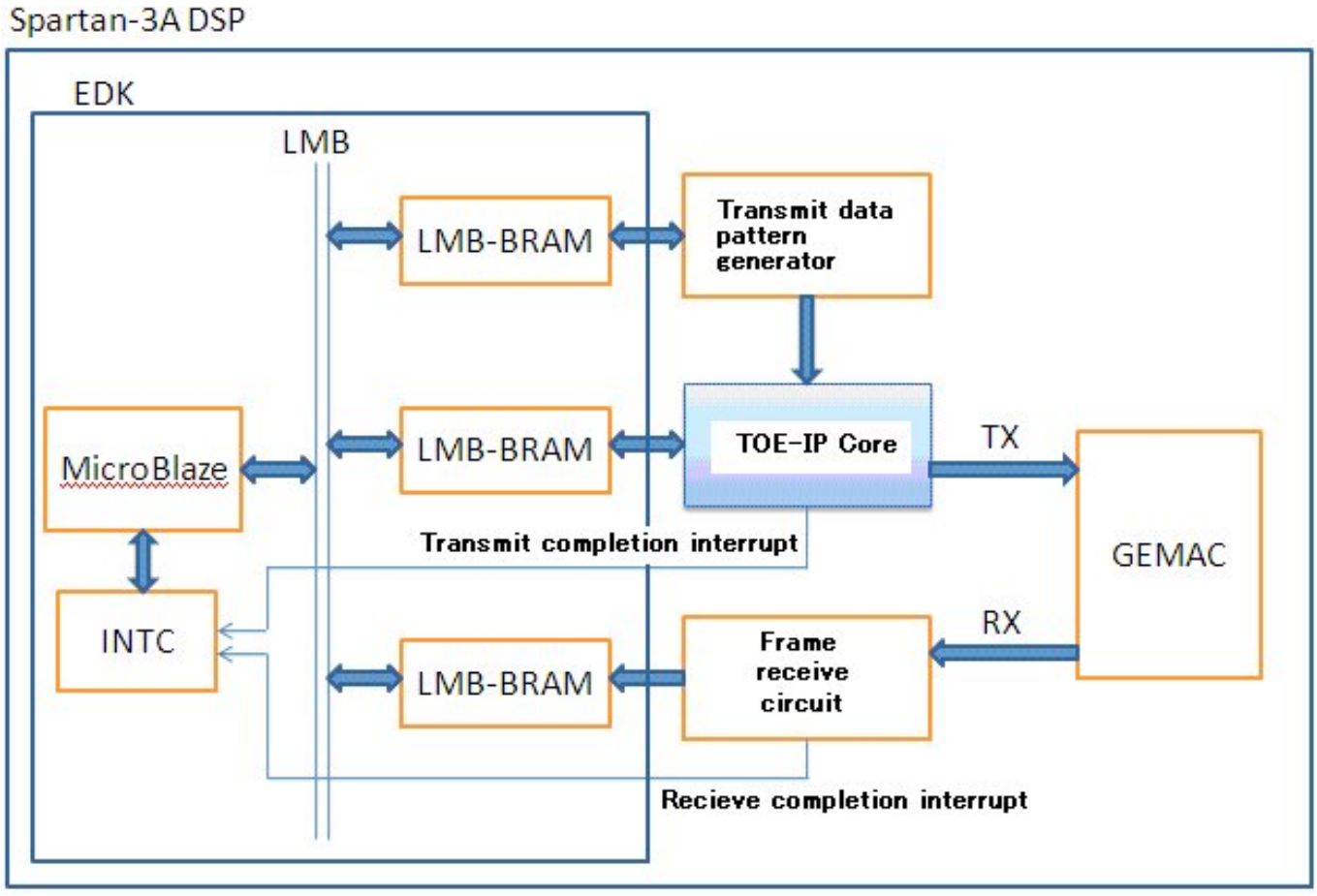

Figure 2: TCP data transmission block diagram.

- GEMAC

This reference design uses Gigabit Ethernet MAC-IP core from Xilinx. TOE-IP is designed for easy connection with this MAC-IP. TOE-IP is an accelerator of transmission operation function only and there is no receive function in this core. So that reference design covers receive function

- Frame receive logic

This block is implemented as 4Kbytes ring buffer and stores received frame data successively. Each frame address is stored in the other 64words ring buffer together with receive status(flag information of good or bad frame). User also can read last store address always so that whole frames can be read by reading until last address. In each frame receive completion, it generates interrupt.

- Transmit data pattern generator

This block generates data to transmit by TCP and connected with data port of TOE-IP. It generates incremental 32bit data by each specified data count. User also can specify generation data count. By verifying data at receive PC side, user can check that data can be transmitted correctly or not.

- EDK Design

This design consists of MicroBlaze, Interrupt Controller, and LMB-BRAM interface. TOE-IP and each user logics are connected via LMB-BRAM interface.
Note: LMB-BRAM interface IP core attached in EDK11.5 does not permit external logic connection. So this reference design includes modified LMB-BRAM interface that can connect with external logics, and stores it under "pcore" folder as a user IP core.

5. Memory Map

The memory map of this reference design is as follows.

Reginster Name	Address	R/W	Valid bit	Description
Frame receive register				
Receive frame store address	0x10000000 –0x100000ff	R	[29:0]	This ring buffer stores receive frame address. User can read from top to bottom, and return to top when arrived at bottom. [13:0] Receive frame store address [14] Set ‘1’ in case of “Good frame” is detected [15] Set ‘1’ in case of “ Bad frame” is detected [29:16] Last received frame address
Receive interrupt Clear	0x1000100c	R/W	[31:0]	R : Bit 31 becomes '1' in case of interrput is occurred. It becomes '0' when interrput is cleared. W : Recive interrupt is cleared by setting ‘1’ to this register.
Receive buffer	0x20000000 –0x2000ffff	R	[31:0]	Ring buffer that stores received frame data. User can get the top address of each frame data by adding read value from Receive frame store address and receive buffer base address. As an address range, it is assigned 64KBytes space, but in real logic, it is implemented as 4Kbytes buffer. Because upper 4bits of 64Kbytes space is ignored, even if frame data is arrived at the bottom of the ring buffer and return to the top of the ring buffer during receiving one frame, MicroBlaze can detect that whole data is stored in the successive address space.

Reginster Name	Address	R/W	Valid bit	Description
Transmit data pattern generator register				
Pattern length	0x10001000	W	[31:0]	Data pattern increments in each “the value set by this register + 1”.
Pattern generation start	0x10001008	W	[0:0]	Set ‘1’ to start pattern generation and send to TOE-IP core,. Set’0’ to stop pattern generation.
Generation data count	0x10001014	W	[31:0]	Set generation data count. Data is genereated “Pattern length+1” x “Generation data count” bytes.

Reginster Name	Address	R/W	Valid bit	Description
Other register				
TOE-IP Core Control	0x30000000	W	[31:0]	Base address for TOE-IP core control interface. Refer to TOE-IP Core datasheet for each TOE-IP internal register details
IFG length	0x10001004	W	[7:0]	User can set IFG length of the Ethernet by this register.
Reset	0x10001010	W	[0:0]	Set ‘1’ to reset TOE-IP core and dada pattern generation circuit. Set’0’ to release reset condition.

6. Operation procedure

Refer to (dg_toe_ip_demoguide_jp.doc) for demo design operation procedure.

7. Software description

Software general flow of this reference design is as below. Refer to the comment of the source code for more detail.

1. Send ARP Request and get MAC address of the transmit destination.
2. Wait request of TCP connection
3. Execute data transmit in case of fast transmit connection is opened. When all data transmit is completed, it closes TCP connection and go back to the state of waiting request of TCP connection.
4. Receive data and verify and echo back in case of slow transmit connection is opened. When connection is closed, go back to the state of waiting request of TCP connection.
5. Repeat 2 – 4.

This reference design supports multi-connection. Max number of connection is defined MAX_CONN in mytcp.h. Sequence number, ACK number, receive buffer is provided each MAX_CONN number as global variable. This reference design manages these connections by the numbers called as "Connection Management Number".

At first, host processor sends ARP to target IP address. After that, host processor sends packet by using MAC address get by this.

After that, host processor waits request of opening connection. This is executed by repeating checking received packet include SYN flag with changing "Connection Management Number" and executing "open connection" function that proceeds opening connection in case of SYN flag is detected.

In this time, "Connection Management Number" 0 is for fast transmit connection. In case of this connection is opened, data is transmitted by using TOE-IP.

For fast transmission, host processor will set sequence number, header length, data length, and data pattern generation ("fastsend_setup" function). After that, issue sending to TOE-IP ("fastsend" function). Host processor repeat execution "fastsend" function for completing transmit whole data. This function executes calculation of "window size divide by MSS size" and set this value to the transmit packet count register.

After this 1st calculation, host processor will execute …

(1) Calculate { "window size" – ("transmit sequence number" – "received ACK number") }

(2) Then execute "(1) value divide by MSS size".

(3) Set (2) result value to the transmit packet count register.

This method enables data transmission with full window size data.

When user sets window size as 65,000 Bytes and set enough large value to MSS size (1460 Bytes), this reference design can show more than 900Mbps transfer speed in average on assumption that receive PC performance is enough high.

To discard transmission complete data, it can be done by writing same received ACK number subtract by initial transmit sequence number to the transmission complete address register.

For data re-transmission, host processor will execute it either of the following case…

(1) Host processor detects that 3 received ACK number is identical.

(2) Host processor detects that target receive window size becomes to zero.

(3) Host processor did not detect ACK within specified timing. (ACK timeout condition).

Re-transmission routine will start when host processor sets "retransmit" flag. In this re-transmission routine, host processor sets sequence number, issue transmit reset, and send 2 packets (Sending 2 packets prevent speed reducing caused by Delayed ACK.). After ACK return is confirmed, host processor will proceed to the next process.

Because there are many packet sending in one time, when only some packet is lost, it is possible case that current sending sequence number is older than received ACK number after lost packet re-transmission is executed. So this ACK will be set as the next transmission sequence number.

At the last transmission data, it is possible that final data do not match with MSS size. But TOE-IP does not support data transmission size less than 200 bytes, so in this case, when host processor sends final data, rewinds data position for last packet data size can match with MSS size. This process is done by using re-transmission routine.

In the other method, adding '0' after frame. Receiver device can receive valid data by using data length field in IP header even if additional dummy data presents. And then the checksum don't change because '0' is added.

"fastsend" function returns transmitted bytes after connection established. So when this value is equal to the value user want to send, it means all data is completed to transmit, so host processor executes "close_connection" function for closing connection.

In case of slow transmit connection, connection is opened same as fast transmit connection. Received data in received buffer is copied to temporary buffer in interrupt handler. After that the specified number of data is copied to user buffer. When host processor receives FIN flag, host processor closes connection. In main function, host processor verifies data and send back as is.

During long time transmission, target device sometimes sends ARP request. To support this, ARP acknowledgement routine is inserted inside of the loop part. At the ARP acknowledgement routine, it overwrites header data with ARP acknowledge frame and transmits ARP acknowledge, and then write back to the original header information.

This implementation is one sample design. System operation behavior may change when user inserts congestion control, change re-transmission timing, or improve algorithm. To optimize data transmission, it is recommendable to try many possible implementations at the user side.

8. Limitation and Remark

This reference design is based on 32bit processor application, so that this design does not support data transmission for more than 4GBytes due to 32bit address space limitation. However as the TOE-IP hardware itself, it can support more than 4GBytes transfer length. So even in 32bit processor environment, when user improves software to handle more than 4GBytes data, it is possible to support such huge data transmission application.

9. Revision History

Revision	Date	Description
0.1	2010/06/21	Temporary release for DG in-house usage (English translation not complete)
2.1	2010/09/03	English version of Reference Design Guide v2.1.