S i T C P Library

Version 1.2

26th May 2021

Tomohisa Uchida

Electronics system group, IPNS, KEK

Revised/Edited by Bee Beans Technologies Co.,Ltd.

History

Date of modifications	Version	Contents
2011/01/18	1.0	Enacted Version 1
2012/10/24	1.1	Version 1.1
After that, this document is revised by Bee Beans Technologies Co., Ltd., for maintenance.		
2021/05/26	1.2	● "Introduction" is Newly added. ● Modified the description about SiTCP version. ● Added regarding Xilinx tools (Vivado) and file formats (edf). ● Inserted the title in Figure 1.

Introduction

This document is the revised version of "SiTCP Library"(created by Dr. Tomohisa Uchida) by Bee Beans Technologies Co., Ltd. (BBT). SiTCP is the Network Processor developed by Dr. Tomohisa Uchida and managed/distributed by BBT.

Contents

This document describes how to use the library for the network processor (SiTCP) distributed from BBT.

Read the separate document “SiTCP Manual” for the concept of SiTCP and the definition of each signal, etc.

Although we describe the contents using ‘Verilog-HDL’, you can also use ‘VHDL’ for the synthesis.

Background knowledge

- How to use the design tool of ‘ISE Design Suite’ (hereinafter ‘ISE’) or ‘Vivado Design Suite’ (hereinafter ‘Vivado’) from ‘Xilinx’.
- Basic knowledge about TCP/IP as well as Ethernet.

You need to be able to implement a circuit on FPGA from ‘Verilog-HDL’ source codes using ‘Xilinx’ ‘ISE’ or ‘Vivado’ design tool. The background knowledge on the network is assumed so that you are able to check the operation of the circuit via network after its implementation.

Applied version

This document is applied to Version 11.0 of SiTCP library distributed from BBT. You can check the version by referring to the filename. For how to interpret the filename, see the section “Distributed files”.

SiTCP Library Outline

SiTCP library is distributed in the netlist of ‘ngc’ file or ‘edf’ file format.

You can synthesize it using ‘ISE’ or ‘Vivado’ as with the library from ‘Xilinx’.

Notice that different FPGA family requires the different library.

Distributed files

SiTCP library consists of the following 3 files.

1. The ‘ngc’ file or ‘edf’ file which is the main body of the SiTCP library
2. Input/Output definition module of the SiTCP library used for logic synthesis (v file)
3. Wrapper of the SiTCP library used for the logic synthesis (v file)
4. Timer module for SiTCP used for logic synthesis (TIMER.V)

About filename

The filenames of 1 (‘ngc’ file or ‘edf’ file) and 2 (Input/Output definition module) are according to the following format.

SiTCP_XC6S_32K_BBT_V110.v/ngc/edf

Its underscores are the delimiters. It means as follows in this order from the head.

- SiTCP (library name) : SiTCP library
- XC6S (Name of applied FPGA family) : For ‘Spartan6’ family from ‘Xilinx’
- 32K (TCP transmission buffer capacity) : 32Kbytes
- BBT (Compliant MAC writing procedure) : Compliant to ‘BBT’ writing procedure
- V110 (Version) : 11.0

The ‘ngc’ or ‘edf’ files can be used also with the ‘VHDL’ source codes alike.

The filename of 3 (Wrapper) is in the following format.

WRAP_SiTCP_GMII_XC6S_32K.V

Its underscores are the delimiters. It means as follows in this order from the head.

- WRAP : means wrapper
- SiTCP (Name of the library to be wrapped) : Wrapper for the SiTCP library
- GMII (Compliant interface) : Wrapped for GMII
- XC6S_32K (Name details for the libraries to be wrapped) :
  For SiTCP_XC6S_32K_BBT_V110

The meaning of each file and the synthesis process are described below.

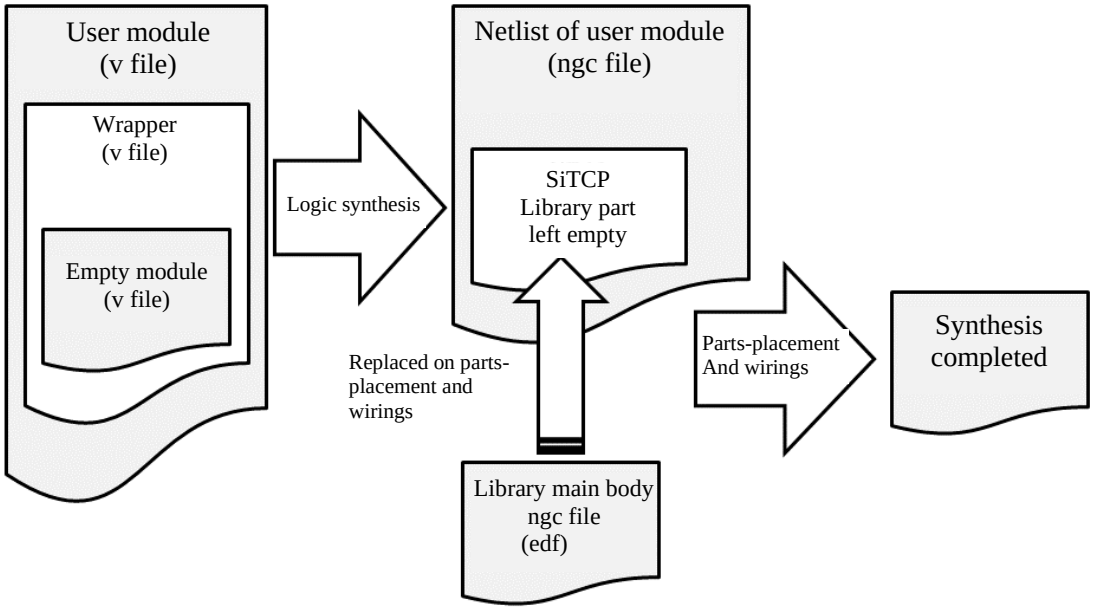

Fig.1 Synthesis process using each file

The library is distributed in the netlist of ‘ngc’ or ‘edf’ file format. The ‘ngc’ or ‘edf’ files have already been logic-synthesized.

The user uses HDL codes for a circuit design in the state before the logic synthesis. From this state, you can logic-synthesize the code with its library empty, provided with the library input/output definition, in order to use any libraries pre-logic-synthesized,

After the logic synthesis, the user’s HDL source file will also become ‘ngc’ file. In this ‘ngc’ file, the SiTCP library is defined as an empty module.

A logic-synthesized file is completed by incorporating the library main body into the empty library. Parts-placement and wirings will be done using this completed netlist to finish the synthesis.

The above shows that a logic synthesis requires empty library HDL module files, and also that parts-placement and wirings require ‘ngc’ file to be read in.

On parts-placement and wiring, ‘ISE’ or ‘Vivado’ automatically search the netlist file (‘ngc’ or ‘edf’ file) for the empty module from its working directory for the synthesis. Therefore, the ‘ngc’ or ‘edf’ file must be placed in the ‘ISE’ or ‘Vivado’ working directory (BBT Note: If you add files by clicking [Add Source] in ISE or [Add Sources] - [Add or create design sources] in Vivado, the directory can be set arbitrarily). The empty HDL module is handled in the same manner as for the other source files (HDL modules).

Simulation model

SiTCP does not provide any simulation model. This is because of the following two reasons. One is that SiTCP does not have any complicated interface that requires a simulation model. The other is that the PC simulation model prepared by the user to simulate TCP/Ethernet should become too complicated.

Notices for Simulation

The simulation covering SiTCP cannot be done since it has no simulation model. So, simulate the circuit you have designed, with due attention to the following.

1. Operation at start & finish of TCP connection
2. Operation on 'FIFO Full' of TCP I/F

About 1:

When the TCP connection is established, TCP_OPEN_ACK becomes H. Start your simulation with TCP_OPEN_ACK being L, and make it transition to H after a certain period of time lapsed in the simulation. Also, simulate the TCP termination as well. Especially, ensure that your simulation makes a transition of TCP_CLOSE_REQ from L to H, and then returns TCP_CLOSE_ACK as in the provision.

About 2:

TCP communication may have its transfer rate lowered largely below the assumed value depending on the receiver PC's performance and network states. Design your TCP communication assuming that the transmission buffer can inevitably be 'Full' at any moment. Most of the problems arising in the development with the SiTCP library can be caused by lack of consideration of the buffer 'Full'. Simulate it as rigorously as possible by generating 'Full' on a randomization.

Sample Circuit

Some sample circuits with the SiTCP library implemented on the evaluation board available from ‘Xilinx’ are distributed for hands-on training and performance evaluation of the library.

The library is similarly implementable on any board mounting of the FPGA from the ‘Xilinx’ and ‘Ethernet PHY’ devices on it. So, implement it with reference to the distributed sample circuits. When switching to the other evaluation board, be sure to change the SiTCP library to the one for the applied FPGA family.

The sample circuit for ‘Spartan6’ evaluation board ‘SP601’ is described below, but the SiTCP version of this circuit is not the latest. The latest version can be downloaded from BBT's GitHub (https://github.com/BeeBeansTechnologies/SiTCP_Netlist_for_Spartan6).

Function and Structure of Sample Circuit

We implement the following functions to the evaluation board ‘SP601’ from ‘Xilinx’.

- SiTCP compliant to Gigabit Ethernet
- TCP loop-back server
- Read-out of DIP SW value by RBCP
- LED control by RBCP

The block diagram for the sample circuit is shown below.

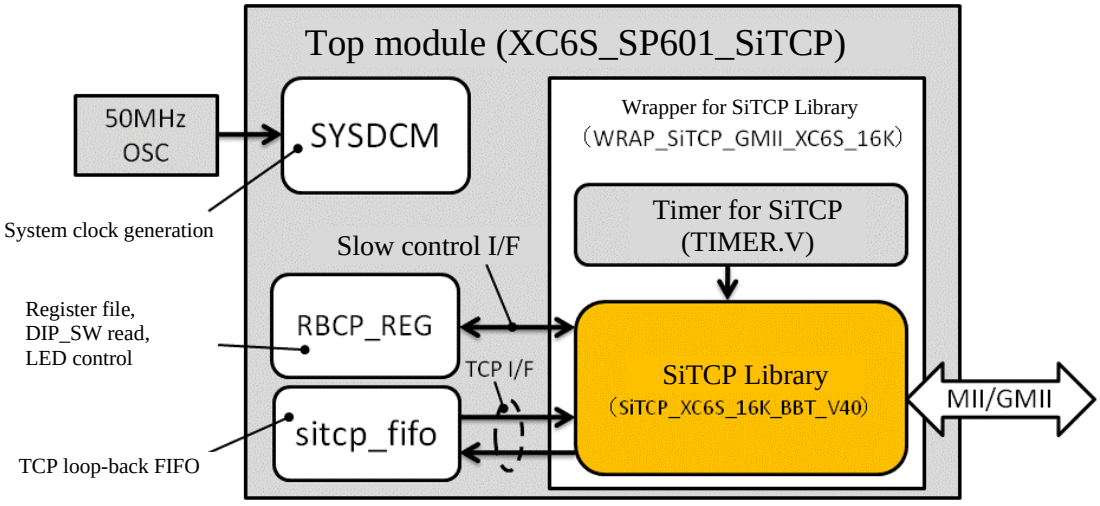

Fig.2 Sample circuit block diagram

Distributed Files

The distributed files and their directory structures are as follows.

- XC6S_SP601_SiTCP (Top directory)
  - CoreGen directory (Xilinx library)
  - ISExx directory (ISExx working directory)
  - SiTCP directory (SiTCP library files)
  - Src directory (‘Verilog-HDL’ source codes for the sample circuits)

COREGEN Directory

This is the working directory for the ‘Core Generator’ where the files related to the ‘Xilinx’ library, used for the sample circuit, are stored.

The following libraries are used for the sample circuits.

1. sys_dcm.* : DCM for system clock generation
2. gmii_dcm.* : DCM for Gigabit Ethernet transmission clock generation
3. sitcp_fifo.* : Synchronous FIFO for TCP loop-back

sys_dcm: generates the clock used within the FPGA from the external clock source (200MHz)

gmii_dcm: DCM that generates GbE transmission clock 125MHz (GTX_CLK) from the external clock source (200MHz)

sitcp_fifo: Synchronous FIFO memory used for TCP loop-back

ISEXX Directory

This is the working directory for ISE. The ‘xx’ of its folder name corresponds to the ISE’s version number. For example, the folder name ‘ISE11’ indicates that the synthesis was done with the ISE version 11.

The files required to generate ‘bit’ file and the synthesized result are stored, herein. ISE can read a project file ‘isexx.xise’ to re-synthesize the circuit.

The sample circuit can be re-synthesized with the ISE of its version that is the same as, or newer than, the one used for the previous synthesis. Notice that any lower versions cannot make a re-synthesis.

The following ‘UCF’ files are stored in this directory.

- XC6S_SP601_SiTCP.ucf

SITCP Directory

The previously described SiTCP library files are stored herein.

SRC Directory

The source codes are stored herein.

- Top module : XC6S_SP601_SiTCP.V
- Slow control register file: RBCP_REG.v

Notice for Synthesis

NGC File

Ensure to copy the ‘ngc’ files of the library on any ISE synthesis. The following files must be copied to the working directory.

- SiTCP_XC6S_16K_BBT_V40.ngc (from SiTCP directory)
- sitcp_fifo.ngc (from ‘CoreGen’ directory)

How to Designate Source File

The ‘include’ sentence in the top module is to read the source file. Designate two files, the top module (XC6S_SP601_SiTCP.V) and the ‘ucf’ file (XC6S_SP601_SiTCP.ucf) for the ISE source file to read.

(Notice) Unsuccessful synthesis has been reported with the ‘include’ sentence used in ISE12. If your synthesis does not work with ISE12, delete the ‘include’ sentence, and specify all the files designated in the ‘include’ sentence to be read directly from the ISE project, for your synthesis.

Operation Check

Procedures

- Ping command operation check
- Network settings of PC
- Response check with Ping command
- TCP communication
- Slow control

PING Command Operation Check

The ‘ping’ command is used to check the connection.

Since the ‘PING’ command may not be used depending on the OS settings, test SiTCP after checking if the ‘Ping’ command works.

How to Check the PING Command Operation

For Windows,

1. Connect your PC to the internet.
2. Open the command prompt.
3. Input ‘ping yahoo.co.jp’.
4. If ‘Reply from xxx.xxx.xxx.xxx’ is displayed, it is ok (The ‘xxx’ is the IP address of ‘Yahoo’).

For Linux,

1. Connect your PC to the internet.
2. Open the terminal.
3. Input ‘ping yahoo.co.jp’
4. If ‘Reply from xxx.xxx.xxx.xxx’ is displayed, it is ok (The ‘xxx’ is the IP address of ‘Yahoo’).
5. Terminate the terminal with ‘Ctrl+C’.

When ‘ping’ command cannot be used,

It should be due to the high security level of your PC or in-house network. Try to make the operation check by …

- Sending the ‘ping’ command to the PC within your laboratory instead of ‘Yahoo’.
- Lowering the security level.

Etc.

Network settings of PC

Do not connect the sample circuit to the public network. Test it on a network independent from the public one. For example, you can test it using a dedicated network where the PC and the sample circuit are directly connected via HUB (not connected to any public network). Refer to the figure below.

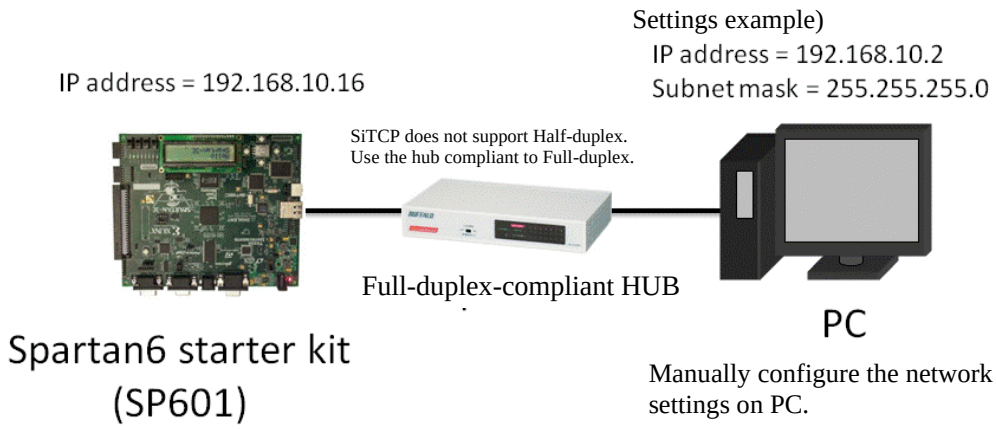

Fig.3 An example of testing environment

Since the sample circuit works with the IP address 192.168.10.16, configure the network environment of your PC communicable with the circuit, accordingly.

Configure the network settings of your PC as follows.

- IP address : 192.168.10.2
- Subnet mask : 255.255.255.0

With the proper network knowledge, you may set the values other than the above.

Response check using PING command

Check the response by the following command.

ping 192.168.10.16

If without any response, your implementation is unsuccessful.

TCP Communication

Loop-back circuit is implemented (Port No. 24). The data transmitted from PC will be received in SiTCP, and, in turn, sent back to the PC.

Below, we describe how to check the TCP operation using 'Telnet'.

```
telnet 192.168.10.16 24
```

Typing the above characters in the input window will display the characters as input. With the local echo setting enabled, the input character is displayed in doubles (two characters; one for the input character and the other for the returned character). With the local echo disabled, only the input character is displayed. Although a few characters at the beginning may be corrupted depending on your terminal software, this is not a problem.

When programming the operation check by yourself, create a TCP client by socket programming.

Slow Control

You can, for example, light on/off the LED using RBCP.

Address map

Address	Register name	**R/W**	Description
0x00-03	FPGA version	R	The value in the 'xxxx xxxx' part of ``define FPGA_VER 32'hxxxx_xxxx' defined in the top module, is readable.
0x04-07	FPGA ID	R	The value in the 'xxxx_xxxx' part of ``define FPGA_ID 32'hxxxx_xxxx' defined in the top module, is readable.
0x08	LED	R/W	The written value is displayed on LED. 'Bit2-0' is connected to LED. 1 for light on and 0 for off.
0x09	DIP SW	R	Read-in of DIP SW setting value. 'Bit3-0' is connected to DIP SW.
0x0A-0F	GPR	R/W	Register file. Read/write-able. No influence on the circuit operation.

References

A paper on SiTCP : T. Uchida, “Hardware-Based TCP Processor for Gigabit Ethernet,”
http://hdl.handle.net/2261/15490

SiTCP Manual : Tomohisa Uchida / BBT