RESEARCH AND BUILD A SOFT MICROPROCESSOR AND ETHERNET INTERFACE USING EDK OF XILINX

***Đặng Hồng Ngọc Quý*[1]**
***Tưởng Thị Thu Hường*[1]**
***Châu Thị Như Quỳnh*[1]**
***Trần Anh Khôi*[1]**
***Hoàng Minh Vũ*[1]**

ABSTRACT

Processor is an integral part of automated metering and control systems. This article covers the fabrication of a soft microprocessor (microblaze) (32bit, on-chip block of RAM 64K, clock 50M, internal buses and peripheral buses) that receive Ethernet data packets, process and send the data packets back to the required station. This soft microprocessor is made from the EDK software tool's XPS (Xilinx), configured into the Xilinx XC3S500E FPGA on the Xilinx Spartan XC3S500 E board. The software operating the interface between the microprocessor and the Ethernet interface port written in C language and functioned as a HOST (station) has its own IP to connect to the Ethernet network. To perform data transmission, it requires an Ethernet cable supporting up to 100Mbs and the computer program that runs the data collection and communication with a soft microprocessor written in Labview.

***Keywords**: EDK, soft microprocessor, Ethernet interface*

I. Introduction

Currently, there are numerous types of soft processors that interface via ethernet such as PIC, Arduino. However, with the purpose of transmitting and receiving data in the coincidence and multichannel system in DaLat nuclear research institute which are self-designed and manufactured on FPGA. Instead of using external soft processors, we use microblaze that are available on FPGA chip to reduce computational time and simplifies designs. In the framework of the topic, in order to communicate with Ethernet via soft-processor, we had to use the XPS (support of C command set) to generate hardware components and software (firmware) of the soft-processor and an FPGA board named Xilinx Spartan XC3S500E.

In order to communicate devices with computers, parallel and serial interface standards such as RS232, USB have been studied. However, Ethernet interface which has not been studied and applied in nuclear electronic devices at Dalat institite.

Nowadays, FPGA is a device which has generally been used because its properties as follows: it can be reconfigurated, it has a high flexibility in design and fabrication. Owning to the FPGA with the afore-mentioned properties, a generation of a specific processor playing a role of hardware as well as a firmware for control of hardware components is created inside the FPGA.

In the world, with the advancement of FPGAs, a new trend of implementing the microprocessors on the FPGAs has emerged in the design community. They implemens Gigabit Ethernet data transmission and reception in FPGA, using the embedded processor

[1]Viện nghiên cứu Hạt nhân
Email: danghongngocquy@gmail.com

MicroBlaze [1] which is implemented on the evaluation board ML505 or Spartan3 XC3S200-4 [2].

The purpose of the article is design and fabricate a soft processor (called microblaze) able to communicate with Ethernet. This microblaze can be embedded into the FPGA by using the supported XPS. Application program running under window XP environment is written in LabView language to link and receive/transmit information between PC and the soft-processor.

II. Methods

The overall design is shown in Figure 1. The design of system is divided into two parts. Hardware part is the 32-bit microprocessor while software part is a firmware. Both of them are created by XPS from Xilinx Company and integrated in a FPGA named XC3S500E board.

A product created through the project is a transfer system including of: the self-excuted application program running under PC, a modem and the soft-processor as Figure1.

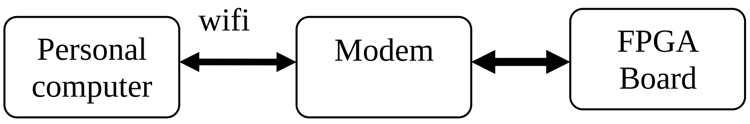


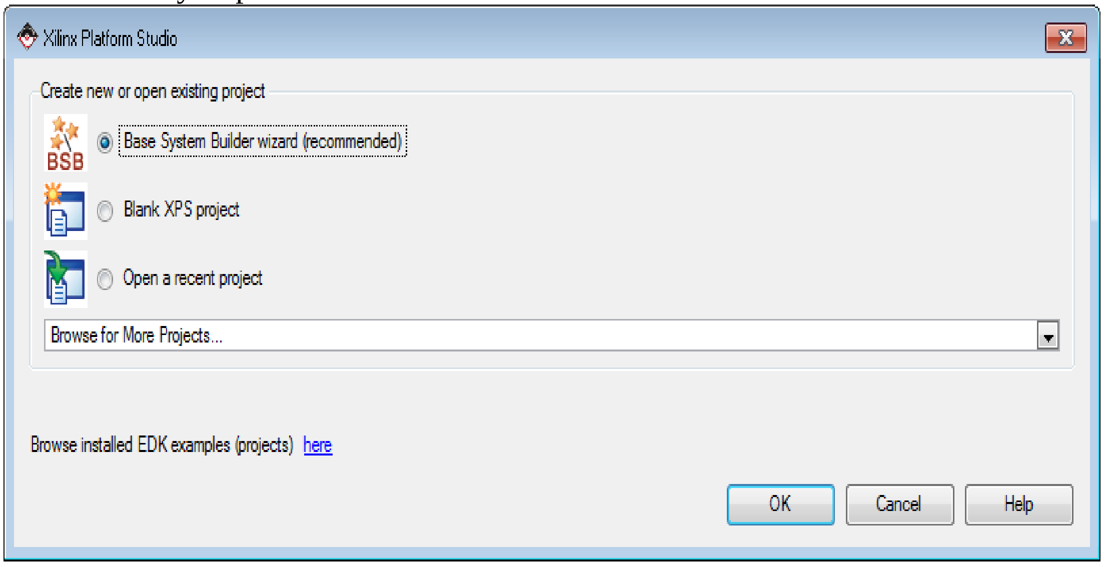
Figure 1: The diagram of overall design

A diagram of creating and embedding a microblaze including software and hardware is shown as follows

II.1. The methology of creating a Microblazer

To create a microblazer in FPGA, there are many steps to as follows

Create a project from EDK: Choose menu File -> new project, the screen to create a new project appears as shown in Figure 2 below, clicking OK to appear a small screen as shown in Figure 2.


Xilinx Platform Studio

Create new or open existing project

[✓ Base System Builder wizard (recommended)]

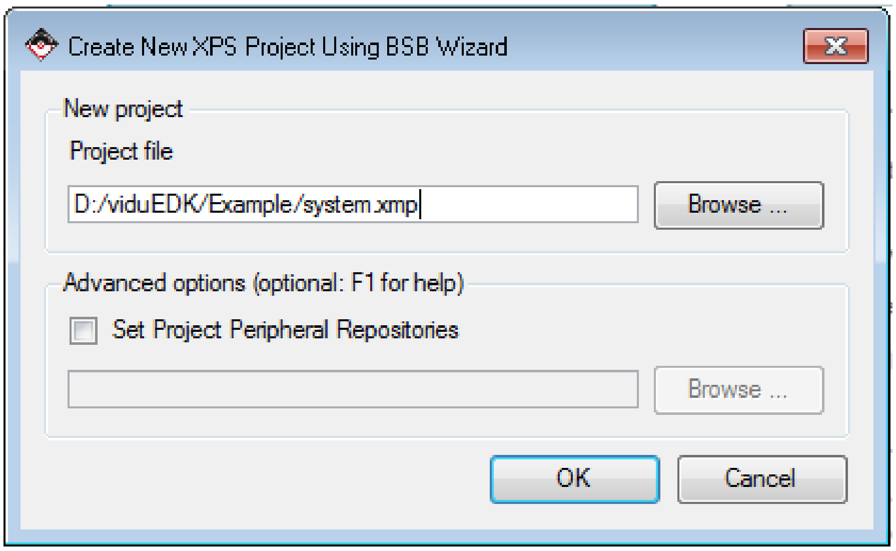
Blank XPS project

Open a recent project

Browse for More Projects...

Browse installed EDK examples (projects) here

Figure 2: *Creating a new project*


Create New XPS Project Using BSB Wizard

New project

Project file

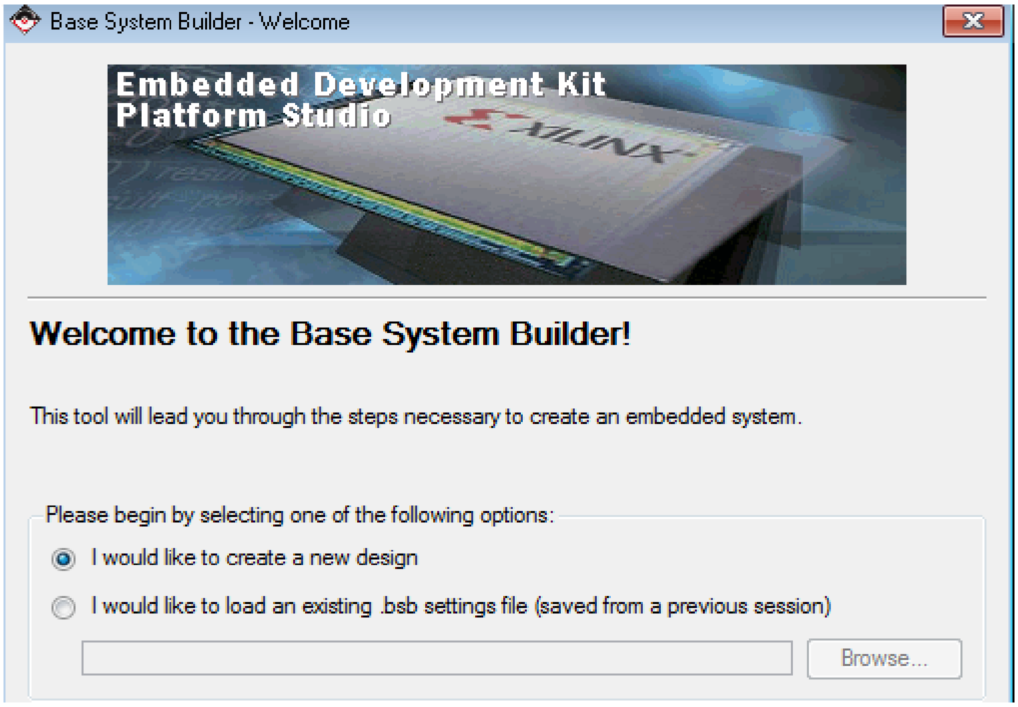
D:/viduEDK/Example/system.xmp

Advanced options (optional: F1 for help)

[ ] Set Project Peripheral Repositories

Figure 3: *The directory of saving project*

In Figure 3, the user selects a directory to save the project and click OK. A screen appears as shown in Figure 4. Users choose Next to start designing a Microblaze.


Base System Builder - Welcome

Embedded Development Kit
Platform Studio

Welcome to the Base System Builder!

This tool will lead you through the steps necessary to create an embedded system.

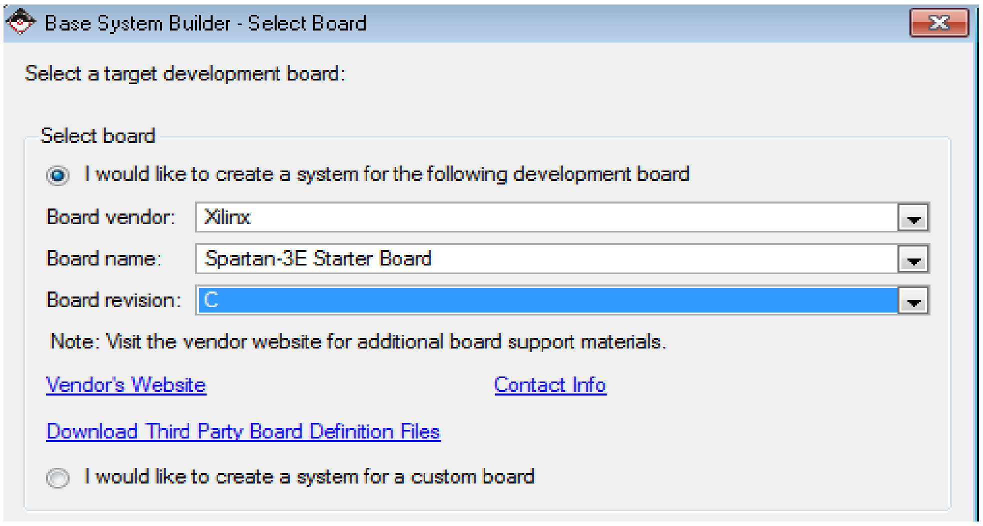
Please begin by selecting one of the following options:

[✓ I would like to create a new design]

I would like to load an existing .bsb settings file (saved from a previous session)

Figure 4: *Creat a new Microblaze*

Choosing the type of board, brand and revision in Figure 5.


Base System Builder - Select Board

Select a target development board:

Select board

(●) I would like to create a system for the following development board

Board vendor: Xilinx

Board name: Spartan-3E Starter Board

Board revision: C

Note: Visit the vendor website for additional board support materials.

Vendor's Website   Contact Info

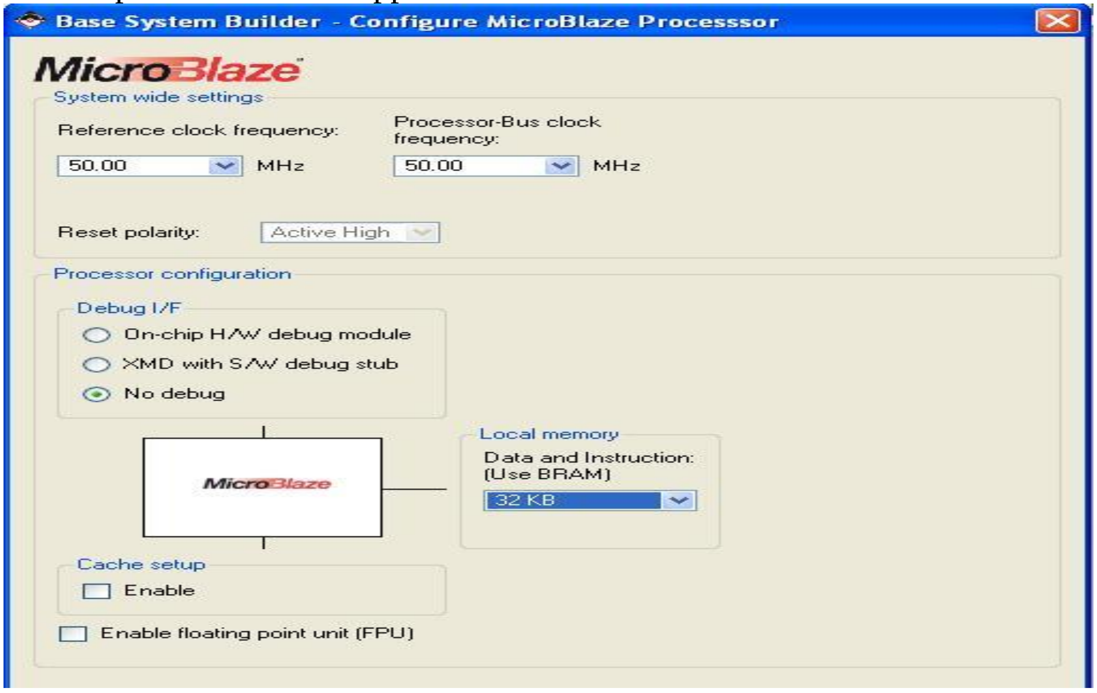
Download Third Party Board Definition Files

( ) I would like to create a system for a custom board

Figure 5: *Screen to select the kit uses to create Microblaze*

After selecting the parameters as shown above, select Next to continue. A small screen asking to select a processor should appear.

Select the frequency, capacity BRAM and Debug option suitable for the board we are using, then select Next.


Base System Builder - Configure MicroBlaze Processor

MicroBlaze

System wide settings

Reference clock frequency: 50.00 MHz

Processor-Bus clock frequency: 50.00 MHz

Reset polarity: Active High

Processor configuration

Debug I/F

( ) On-chip H/W debug module

( ) XMD with S/W debug stub

(●) No debug

Local memory

Data and Instruction: (Use BRAM) 32 KB

Cache setup

☐ Enable

☐ Enable floating point unit (FPU)

Figure 6: *Window of Microblaze 8/16/32/64 bits*

II.2. A design methodology for soft-processor

The process of creating a soft microprocessor is performed sequentially according to the following 5 steps:

1. Create soft processor with hardware and peripheral parameters:

32bit softwares, BRAM 64K, LMBs, PLBs, OPBs, Ethernet, 8 LEDs, RS232. All are created in files * .MSS, * .MPD, * .MHS

2. Creating the library related to the soft-processor and peripheral

3. Compiling firmware file and C-function library (mb-GCC), *.OUT file created

4. Creating the platform to have a core and netlists

5. A collection of *.UCF, *.ED files, core and netlists, use of iMPACT for creating *.Bit file, *.MCS file. Finally, the MCS file will be loaded into ROM for operating.

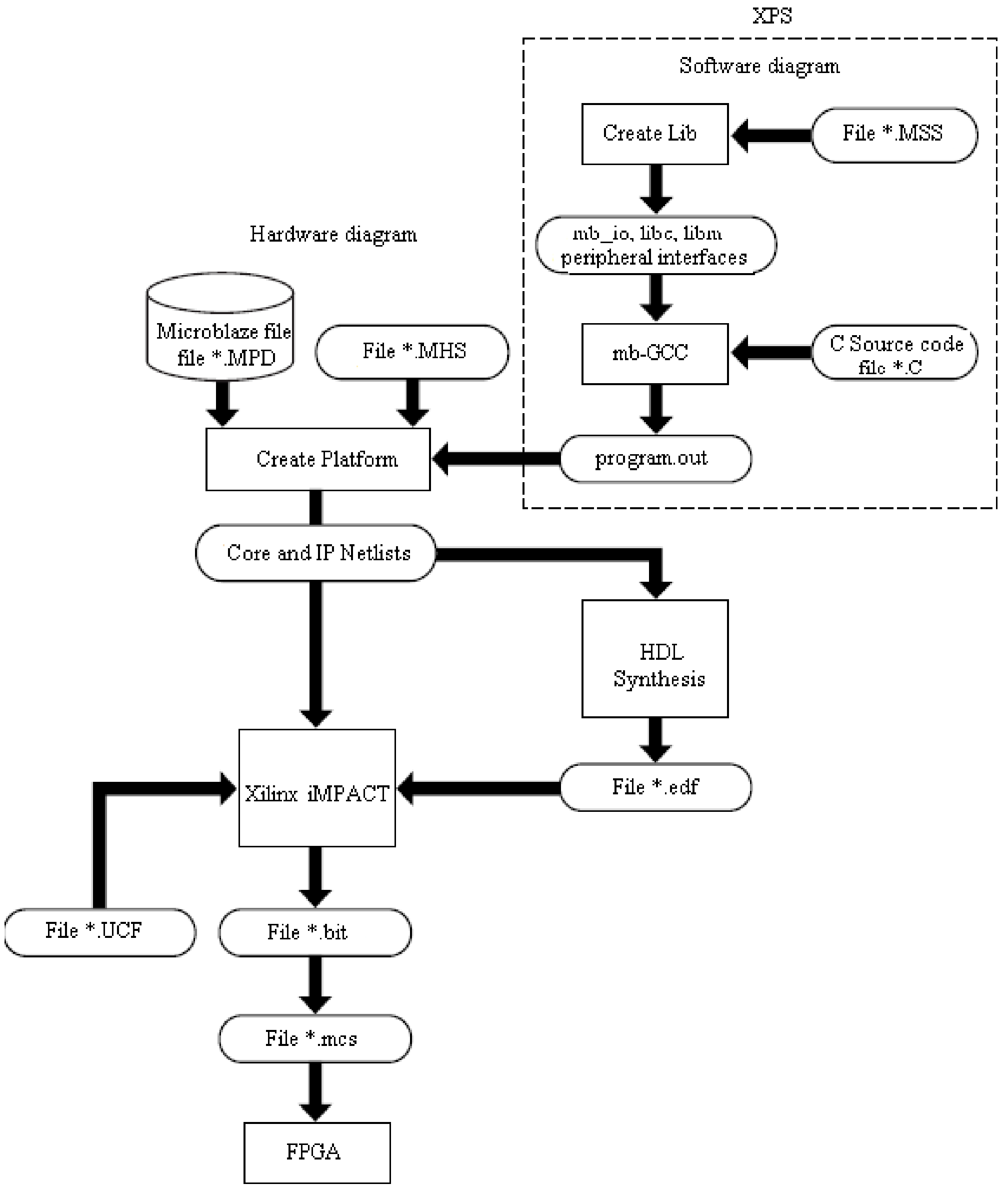
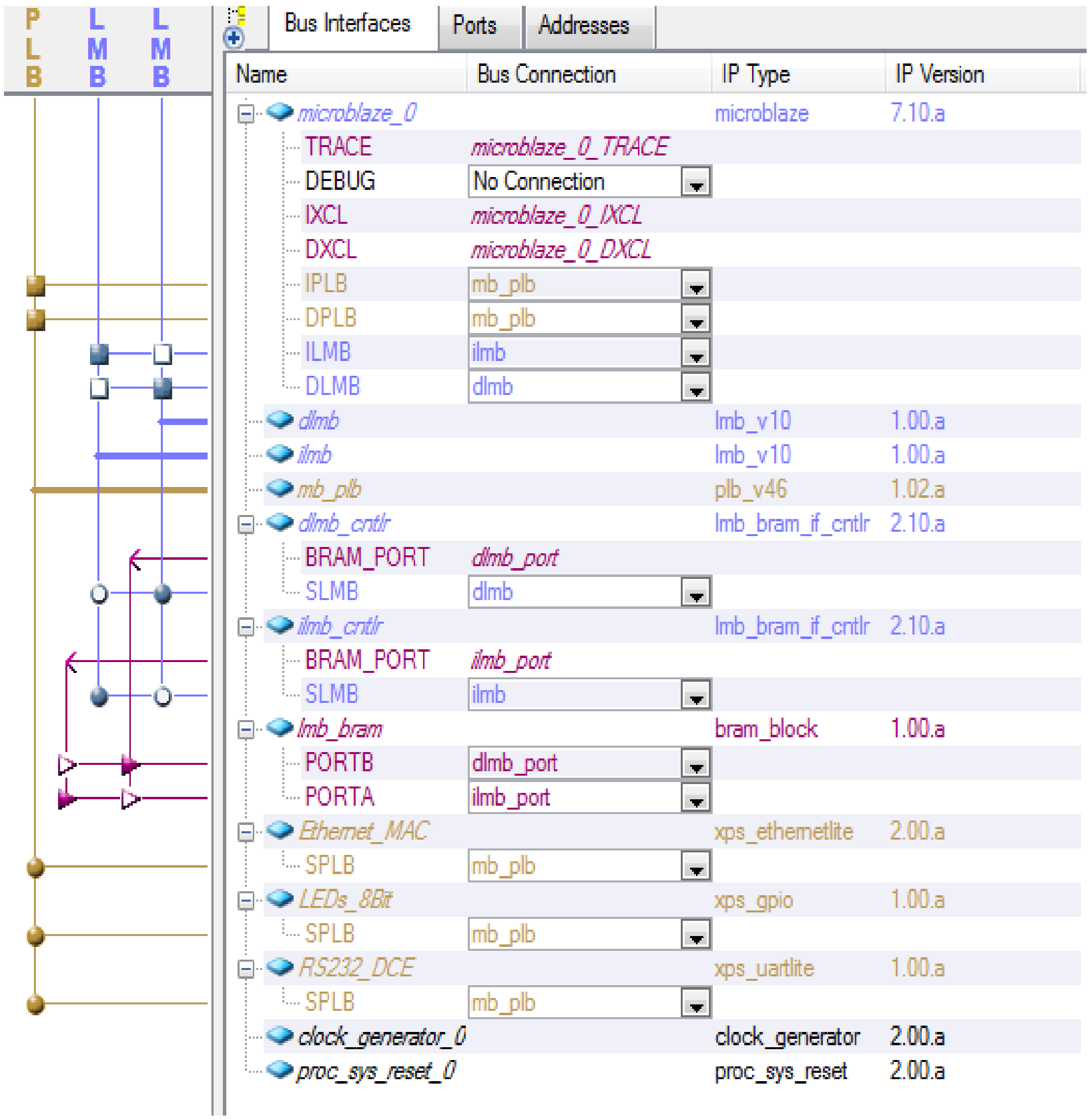


Figure 7: *The diagram paints all steps create and embed an FPGA soft-processor*

The XPS has Wizard supporting users how to design conveniently. In addition to creating a 32bit soft-processor, 64K internal RAM, we can also create more peripherals such as Ethernet, RS232, LEDs...

After that, we carry out connection of input/output port with peripheral (*.UCF file), producing address automatically, generating library functions relating to microprocessor software and peripheral (Figure 8).



Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.a
TRACE	microblaze_0_TRACE		
DEBUG	No Connection		
IXCL	microblaze_0_IXCL		
DXCL	microblaze_0_DXCL		
IPLB	mb_plb		
DPLB	mb_plb		
ILMB	ilmb		
DLMB	dlmb		
dlmb		lmb_v10	1.00.a
ilmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.02.a
dlmb_cntlr		lmb_bram_if_cntlr	2.10.a
BRAM_PORT	dlmb_port		
SLMB	dlmb		
ilmb_cntlr		lmb_bram_if_cntlr	2.10.a
BRAM_PORT	ilmb_port		
SLMB	ilmb		
lmb_bram		bram_block	1.00.a
PORTB	dlmb_port		
PORTA	ilmb_port		
Ethernet_MAC		xps_ethernetlite	2.00.a
SPLB	mb_plb		
LEDs_8Bit		xps_gpio	1.00.a
SPLB	mb_plb		
RS232_DCE		xps_uartlite	1.00.a
SPLB	mb_plb		
clock_generator_0		clock_generator	2.00.a
proc_sys_reset_0		proc_sys_reset	2.00.a

Figure 8: *Bus Interface of soft-processor*

II.3. The create of software design (firmware)

Using the libraries were created when we designed the hardware to designed software (firmware). These libraries support functions read/write registers, cache.... Connection, transferability in this project, we used TCP/IP protocol [3]. When the firmware had received a TCP packet from PC, the firmware performed the packet and created a require package.

This package was sent to PC via Ethernet.

In this article, we using C language anđ some standard functions to process Microblazer [4-5] and the algorithm flowchart of firmware is describered in figure 9.

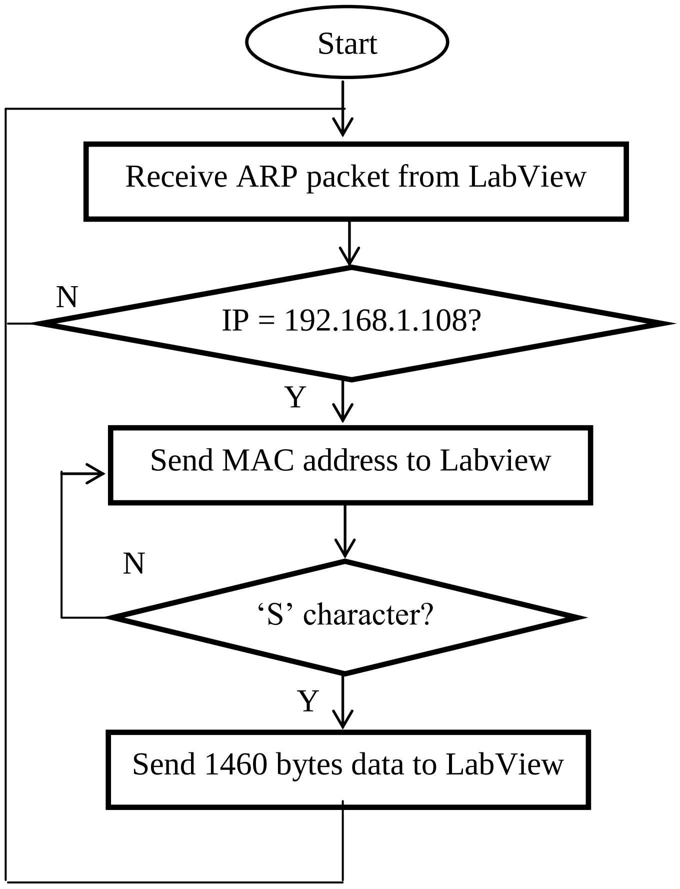

Start
Receive ARP packet from LabView
N
IP = 192.168.1.108?
Y
Send MAC address to Labview
N
'S' character?
Y
Send 1460 bytes data to LabView

Figure 9: *The algorithm flowchart of firmware*

Firmware for Microblaze with function as a server. In Firmware, set the parameters necessary to connect to the Ethernet network such as IP 192.168.1.108, MAC address 00: 0A: 35: 02: 22: 5E, PORT is 53326. The server sends data to the computer only when receiving get an "S" command from LabVIEW.

The application program with LabVIEW as a client with a computer IP of 192.168.1.105, MAC 94: 39: E5: 59: 0B: A3, PORT is 3363. The program sends the "S" command and wait to receive 16,364 bytes of data from the server, then end the connection. The following is a detailed section on how to create data transmission systems over Ethernet.

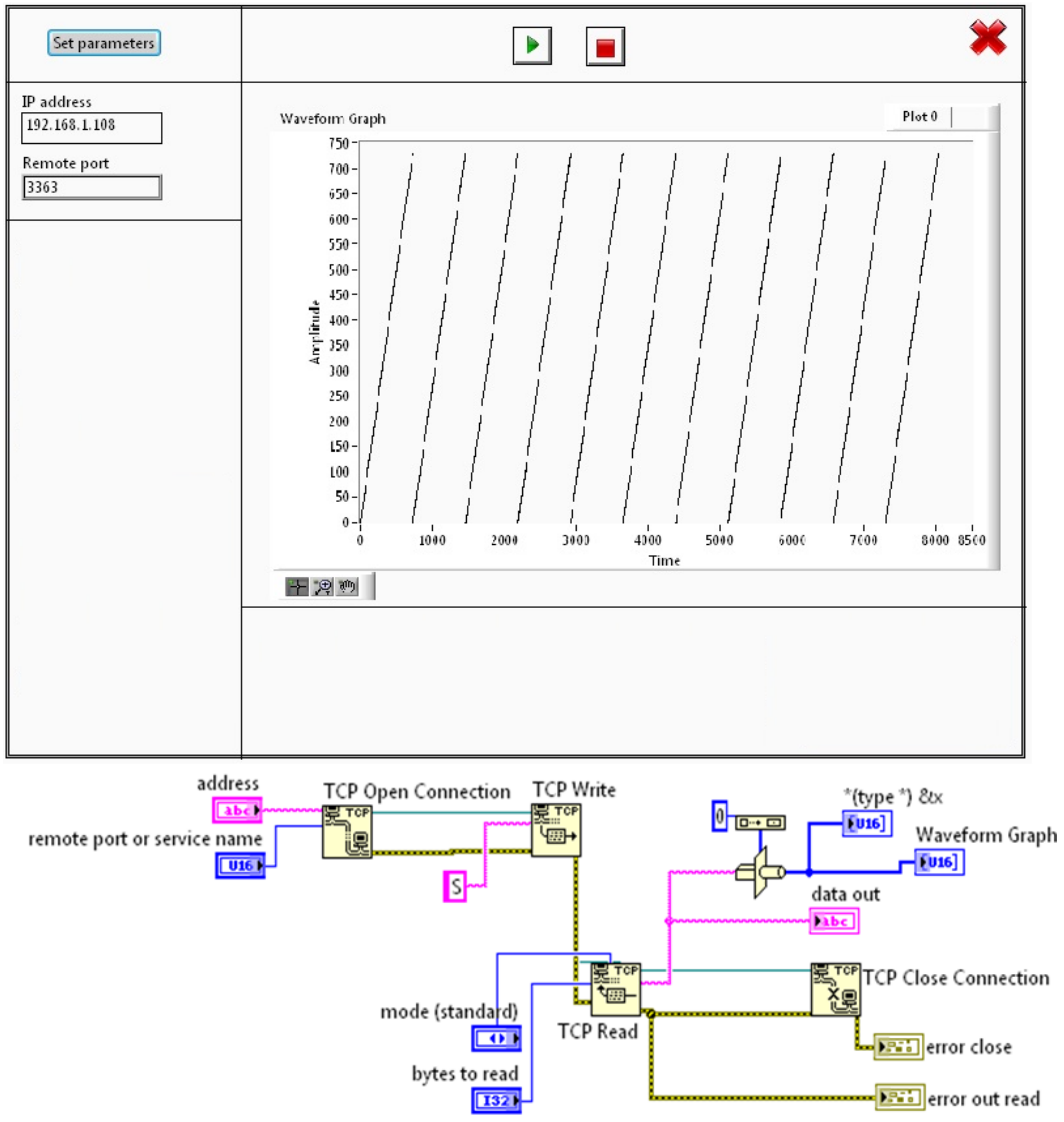
III. Results

In the project, using FPGA component named XC3S500E, 50MHz internal clock, LAN83C185 Ethernet chip, 8 LEDS display, RS232 port, Platform Flash PROM (configure the FPGA and program the soft-processor), 10/100 Mbs Ethernet port, USB for JTAG, MAC address is Xilinx_02:22:5E (00:0A:35:02:22:5E)

MAC address of the computer is HonHaiPr_59:0B:A3 (94:39:E5:59:0B:A3), IP is 192.168.1.105, PORT is 3363.

Wireshark software is used to capture all packets going through the computer network card that we can see in Figure 9.

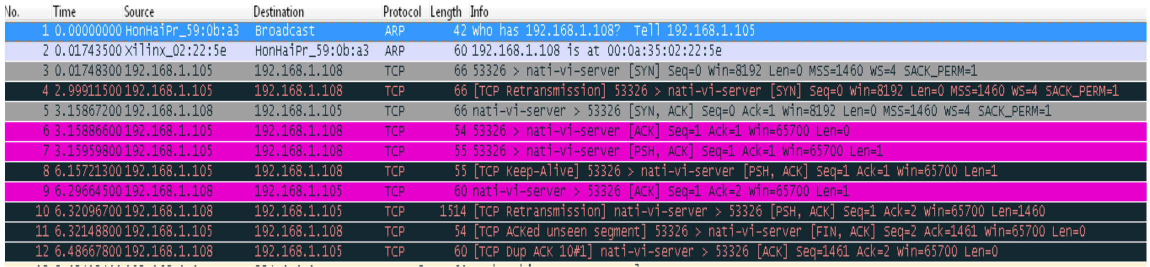
In order to check the accuracy of the programmed data packages, the project team used wireshark software to capture the data packets transferred through the computer network card. The munber of the package is 1, 2, 3, 5, 6, 7, 9, 10, 11, 12, respectively are communication packets between labview and spartan 3E.



IP address: 192.168.1.108

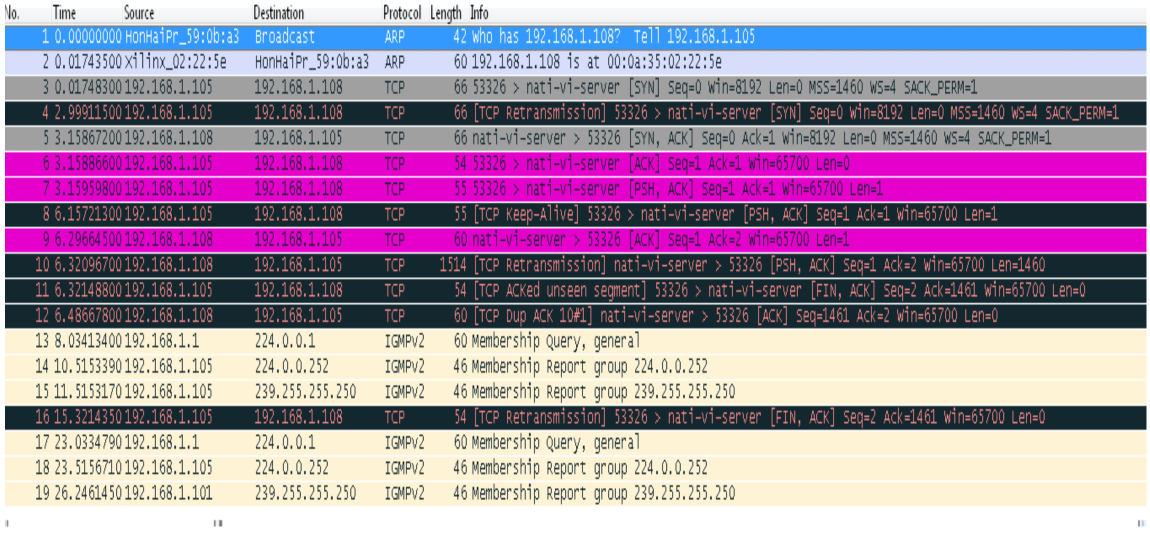
Remote port: 3363

Figure 9: *Data acquisition program*



```
No. Time       Source             Destination        Protocol Length Info
  1 0.00000000 HonHaiPr_59:0b:a3  Broadcast          ARP        42 Who has 192.168.1.108?  Tell 192.168.1.105
  2 0.01743500 Xilinx_02:22:5e    HonHaiPr_59:0b:a3  ARP        60 192.168.1.108 is at 00:0a:35:02:22:5e
  3 0.01748300 192.168.1.105      192.168.1.108      TCP        66 53326 > nati-vi-server [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  4 2.99911500 192.168.1.105      192.168.1.108      TCP        66 [TCP Retransmission] 53326 > nati-vi-server [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  5 3.15867200 192.168.1.108      192.168.1.105      TCP        66 nati-vi-server > 53326 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  6 3.15886600 192.168.1.105      192.168.1.108      TCP        54 53326 > nati-vi-server [ACK] Seq=1 Ack=1 Win=65700 Len=0
  7 3.15959800 192.168.1.105      192.168.1.108      TCP        55 53326 > nati-vi-server [PSH, ACK] Seq=1 Ack=1 Win=65700 Len=1
  8 6.15721300 192.168.1.105      192.168.1.108      TCP        55 [TCP Keep-Alive] 53326 > nati-vi-server [PSH, ACK] Seq=1 Ack=1 Win=65700 Len=1
  9 6.29664500 192.168.1.108      192.168.1.105      TCP        60 nati-vi-server > 53326 [ACK] Seq=1 Ack=2 Win=65700 Len=1
 10 6.32096700 192.168.1.108      192.168.1.105      TCP      1514 [TCP Retransmission] nati-vi-server > 53326 [PSH, ACK] Seq=1 Ack=2 Win=65700 Len=1460
 11 6.32148800 192.168.1.105      192.168.1.108      TCP        54 [TCP ACKed unseen segment] 53326 > nati-vi-server [FIN, ACK] Seq=2 Ack=1461 Win=65700 Len=0
 12 6.48667800 192.168.1.108      192.168.1.105      TCP        60 [TCP Dup ACK 10#1] nati-vi-server > 53326 [ACK] Seq=1461 Ack=2 Win=65700 Len=0
```

Figure 10: *Connected, transmited, received frames captured in Wireshark*



```
No. Time       Source             Destination        Protocol Length Info
  1 0.00000000 HonHaiPr_59:0b:a3  Broadcast          ARP        42 Who has 192.168.1.108?  Tell 192.168.1.105
  2 0.01743500 Xilinx_02:22:5e    HonHaiPr_59:0b:a3  ARP        60 192.168.1.108 is at 00:0a:35:02:22:5e
  3 0.01748300 192.168.1.105      192.168.1.108      TCP        66 53326 > nati-vi-server [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  4 2.99911500 192.168.1.105      192.168.1.108      TCP        66 [TCP Retransmission] 53326 > nati-vi-server [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  5 3.15867200 192.168.1.108      192.168.1.105      TCP        66 nati-vi-server > 53326 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  6 3.15886600 192.168.1.105      192.168.1.108      TCP        54 53326 > nati-vi-server [ACK] Seq=1 Ack=1 Win=65700 Len=0
  7 3.15959800 192.168.1.105      192.168.1.108      TCP        55 53326 > nati-vi-server [PSH, ACK] Seq=1 Ack=1 Win=65700 Len=1
  8 6.15721300 192.168.1.105      192.168.1.108      TCP        55 [TCP Keep-Alive] 53326 > nati-vi-server [PSH, ACK] Seq=1 Ack=1 Win=65700 Len=1
  9 6.29664500 192.168.1.108      192.168.1.105      TCP        60 nati-vi-server > 53326 [ACK] Seq=1 Ack=2 Win=65700 Len=1
 10 6.32096700 192.168.1.108      192.168.1.105      TCP      1514 [TCP Retransmission] nati-vi-server > 53326 [PSH, ACK] Seq=1 Ack=2 Win=65700 Len=1460
 11 6.32148800 192.168.1.105      192.168.1.108      TCP        54 [TCP ACKed unseen segment] 53326 > nati-vi-server [FIN, ACK] Seq=2 Ack=1461 Win=65700 Len=0
 12 6.48667800 192.168.1.108      192.168.1.105      TCP        60 [TCP Dup ACK 10#1] nati-vi-server > 53326 [ACK] Seq=1461 Ack=2 Win=65700 Len=0
 13 8.03413400 192.168.1.1        224.0.0.1          IGMPv2     60 Membership Query, general
 14 10.5153390 192.168.1.105      224.0.0.252        IGMPv2     46 Membership Report group 224.0.0.252
 15 11.5153170 192.168.1.105      239.255.255.250    IGMPv2     46 Membership Report group 239.255.255.250
 16 15.3214350 192.168.1.105      192.168.1.108      TCP        54 [TCP Retransmission] 53326 > nati-vi-server [FIN, ACK] Seq=2 Ack=1461 Win=65700 Len=0
 17 23.0334790 192.168.1.1        224.0.0.1          IGMPv2     60 Membership Query, general
 18 23.5156710 192.168.1.105      224.0.0.252        IGMPv2     46 Membership Report group 224.0.0.252
 19 26.2461450 192.168.1.101      239.255.255.250    IGMPv2     46 Membership Report group 239.255.255.250
```

Figure 11: *The process of connecting, closing the connection and receiving a data packet*

In this program, microblaze acts as a server, a PC is a client. PC uses LabVIEW to connect to the server.

+ VI "TCP open connection" creates a handshake connection with microblaze as follows:

- No.1: LabVIEW sends an ARP breadcast packet asking which device has an IP of 192.168.1.108

- No.2: Microblaze send to 192.168.1.105 (LabVIEW) that the network card on the xilinx board is 192.168.1.108 and the MAC address is 00: 0A: 35: 02: 22: 5E

- No.3: LabVIEW (192.168.1.105) sent to microblaze (192.168.1.108) TCP packet [SYN] confirms the message specification

- No.5: Microblaze receives TCP packet [SYN] then returns TCP packet [SYN, ACK] to confirm sending method

- No.6: LabVIEW (192.168.1.105) sent to microblaze (192.168.1.108) TCP packet [ACK] confirms connection completion

+ VI "TCP write" sends data to microblaze as follows:

- No.7: LabVIEW (192.168.1.105) sent to microblaze (192.168.1.108) TCP packet [PSH, ACK], len = 1, the letter "S" tells microblaze LabVIEW wants to receive data

- No.9: Microblaze receives TCP packet [PSH, ACK] then returns TCP packet [ACK] confirms receipt of data sending command and prepares data packet to send to LabVIEW

+ VI "TCP read" waiting to receive data from microblaze:

- No.10: Microblaze (192.168.1.108) sent to LabVIEW (192.168.1.105) TCP packet [PSH, ACK], len = 1460 bytes

+ VI "TCP close connection" sent to microblaze closed the connection:

- No.11: LabVIEW (192.168.1.105) receives a packet that sends back microblaze (192.168.1.108) TCP packet [FIN, ACK] to close the connection

- No.12: Microblaze (192.168.1.108) receives the packet of ending connection and sends to LabVIEW (192.168.1.105) the TCP [ACK] packet to report disconnection.

By capturing packets of information transmitted between the device and the computer using Wireshark as mention-above, the detailed information displayed in the packets represents the accuracy, integrity and correct sequence of data. Microblaze and the FPGA interface circuits have functioned properly.

IV. Conclusion

A soft-processor is sucessfully created including software and hardware. The success of the project a new way for fabricating instruments. The advantages of this approach as follows:

- Economics: not having to buy hardware devices, small chip, small size of ROM
- Flexibility: design of 8, 16, 32, 64 bits microcontroller, depending on demands, creating arbitrary RAM only changed the parameters on software, connecting with expected peripherals easily
- Convenience: embedded directly into the FPGA and reusable
- Compact: all of needed components was only inside FPGA

We also succeeded in firmware design for TCP/IP protocol transfer data via Ethernet. This success will help remote measuring systems transfer data going further, faster and also help users remote control instruments conveniently.

REFERENCES

1. Indu Raj, and Rejani Krishna, “FPGA Based Platforms for Ethernet Data Transfer”, International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 3, Issue 1, January 2014

2. Anjali S S, Rejani Krishna P, Aparna Devi P S, “High Speed Data Transfer Using FPGA”

International Journal of Engineering Research and General Science Volume 4, Issue 3, May-June, 2016.

3. E.Bryan Carne, *A profession guide to Data Communication in a TCP/IP World*, Artech House Publishers, London

4. Xilinx (2008), *Embedded Development Kit EDK 10.1i, X*ilinx Inc, USA

5. Xilinx (2010), *XPS Ethernet Lite Media Access Controller*, Xilin Inc, USA

NGHIÊN CỨU, XÂY DỰNG MỘT VI XỬ LÝ MỀM VÀ GIAO DIỆN ETHERNET SỬ DỤNG CÔNG CỤ PHẦN MỀM EDK (XILINX)

TÓM TẮT

Bộ xử lý là một thành phần không thể thiếu trong các hệ thống đo đạc và điều khiển tự động. Bài báo này đề cập đến việc chế tạo một vi xử lý mềm (microblaze) (32bit, on-chip block RAM 64K, clock 50M, các bus nội và bus ngoại vi) nhận gói dữ liệu Ethernet, xử lý và gửi gói dữ liệu trở lại trạm yêu cầu. Vi xử lý mềm này được chế tạo từ bộ XPS của công cụ phần mềm EDK (Xilinx), được cấu hình vào FPGA XC3S500E trên board mạch Xilinx Spartan XC3S500E. Phần mềm điều hành giao diện giữa vi xử lý và cổng giao diện Ethernet viết bằng ngôn ngữ C và có chức năng như một HOST (trạm), có IP riêng để kết nối với mạng Ethernet. Để thực hiện truyền nhận dữ liệu cần có cáp ethernet hỗ trợ tốc độ đến 100Mbs, chương trình máy tính điều hành việc thu nhận dữ liệu và giao tiếp với vi xử lý mềm viết bằng Labview.

***Từ khóa**: EDK, vi xử lí mềm, Giao diện Ethernet*

(Received: 21/5/2020, Revised: 30/7/2020, Accepted for publication: 1/12/2020)