ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014

Implementation of Soft-Core Processor Based Ethernet Data Transfer

Indu Raj[1], Rejani Krishna P[2]

PG Student [VLSI & Embedded Systems], Department of Electronics and Communication Engineering,

Sree Narayana Gurukulam College of Engineering, Kadayiruppu, Kerala, India[1]

Scientist-D, Naval Physical and Oceanographic Laboratory, Cochin, Kerala, India[2]

ABSTRACT: This paper presents the implementation of embedded processor inside FPGA such that it can receive Ethernet packets, retrieve actual data, modify it and finally transmit it to any other destinations required. The implementation platform is an evaluation board that have a Virtex-5 FPGA. Using the XPS tool, the embedded processor MicroBlaze is configured inside the Virtex-5 FPGA. The software part of the processor is configured using the lwIP Echo Server template available in SDK. The implementation requires ML505 evaluation board, Ethernet cable that supports GbE data transfer, RS232 serial cable and a PC with gigabit NIC.

KEYWORDS: FPGA, MicroBlaze, Virtex-5, EDK, XPS, SDK, lwIP.

I. INTRODUCTION

The reconfigurable devices, such as FPGA are flexible and reusable high-density circuits. That is, any portion of the system can be reconfigured at any time while the rest of the design is still working. With the advancement of Field Programmable Gate Arrays (FPGAs) a new trend of implementing the microprocessors on the FPGAs has emerged in the design community [1]. The design of such microprocessors is available in the form of software bitstream, so they are called soft-processors or soft cores. System designers can embed these cores into their designs and have the option to customize them as required. Soft cores are technology independent and require only simulation and timing verification. This reduces the design time as compared to a hard core processor.

In Ethernet communication the data will be secured with a header, so there requires a method that can be used to retrieve actual data from the Ethernet data. If a processor is used, this can be done quite easily with good speed and also the actual data can be further processed as required. While a processor based system could simplify data retrieval and processing, the processor embedded inside the system simplifies the technique further since it requires only less number of components thereby reducing the device utilization. So all these factors prompted us to implement a system using an embedded processor that receives Ethernet packets, retrieve actual data, modify it and finally transmit it to any other destinations required with the help of Embedded Development Kit. The implementation platform is the ML505 board which supports the embedded processor MicroBlaze. MicroBlaze is a 32-bit RISC Harvard architecture soft processor IP core with a rich instruction set optimized for embedded applications [2]. The implementation is achieved using the Embedded Development Kit provided by Xilinx, which helps to design the complete embedded processor more quickly and easily [3].

II. SOFTWARE OVERVIEW

Embedded Development Kit (EDK) is an integrated software tool for developing embedded systems [4]. The EDK package has Xilinx Platform Studio (XPS) software for configuring the hardware portion and Software Development Kit (SDK) for configuring the software portion. XPS includes a graphical user interface that provides a set of tools to help in the project design. The Base System Builder (BSB) is a wizard in the XPS that quickly and efficiently establishes a working design, which can then be customized by the designer. XPS supports the creation of the Microprocessor Hardware Specification (MHS) and Microprocessor Software Specification (MSS) files needed for embedded tools flow. The MHS file defines the system architecture, peripherals and embedded processors [5]. The software platform is defined by MSS file which defines driver and library customization.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014

After designing the hardware portion, the design has to undergo different processes like Synthesis, Simulation, Translate, Map and Place and Route. Finally, a bit file is generated. This bit file is exported to the SDK along with the MHS and MSS files. SDK is an integrated development environment, complimentary to XPS. It can create different software applications for peripherals and processors that have been connected in XPS. The code can be written in the high level programming language C/C++. The block diagram that represents the basic embedded design flow can be seen in Fig. 1[3].

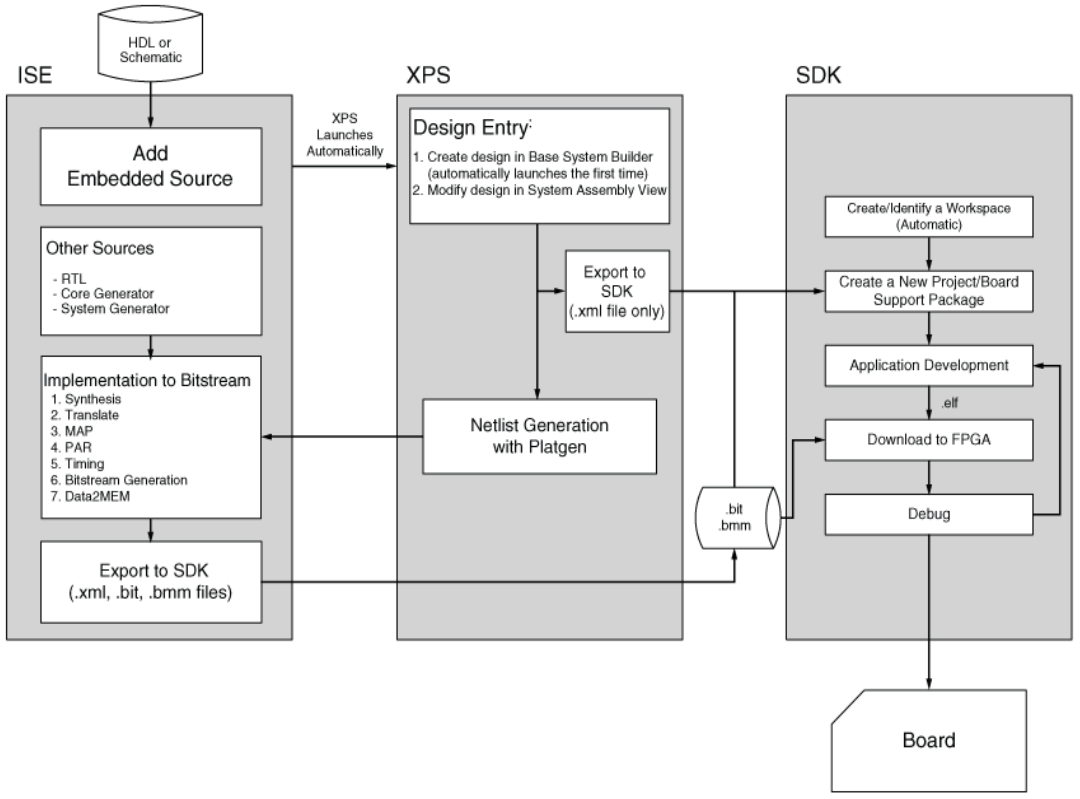

HDL or Schematic
ISE
XPS
SDK
XPS Launches Automatically
Add Embedded Source
Other Sources
- RTL
- Core Generator
- System Generator
Implementation to Bitstream
1. Synthesis
2. Translate
3. MAP
4. PAR
5. Timing
6. Bitstream Generation
7. Data2MEM
Export to SDK (.xml, .bit, .bmm files)
Design Entry:
1. Create design in Base System Builder (automatically launches the first time)
2. Modify design in System Assembly View
Export to SDK (.xml file only)
Netlist Generation with Platgen
.bit
.bmm
Create/Identify a Workspace (Automatic)
Create a New Project/Board Support Package
Application Development
.elf
Download to FPGA
Debug
Board

Fig. 1 Basic embedded design flow

III.SYSTEM DESIGN

The implementation platform is the ML505 evaluation board that have Virtex-5 FPGA. The ML505 board support MicroBlaze soft core processor. The system design is divided into two. One is hardware design, which includes the designing methods using XPS and the other is Software design, which include designing methods using SDK.

A. Hardware Design

Besides the MicroBlaze, other components required by the design are GPIO LEDs, a timer, an interrupt handling controller and a serial communication device UART. The MicroBlaze is implemented entirely in the general-purpose memory and logic fabric of FPGAs using the EDK design environment. The BSB (Base System Builder) Wizard inside XPS is used for generating the embedded system around the MicroBlaze that is supported on the ML505 board. This wizard allows the selection of the board and processor.

As a first step, the MicroBlaze processor is configured. Here we set the system clock frequency to 125 MHz and the local memory to 64 KB. The next step is the selection and configuration of peripherals, which include the

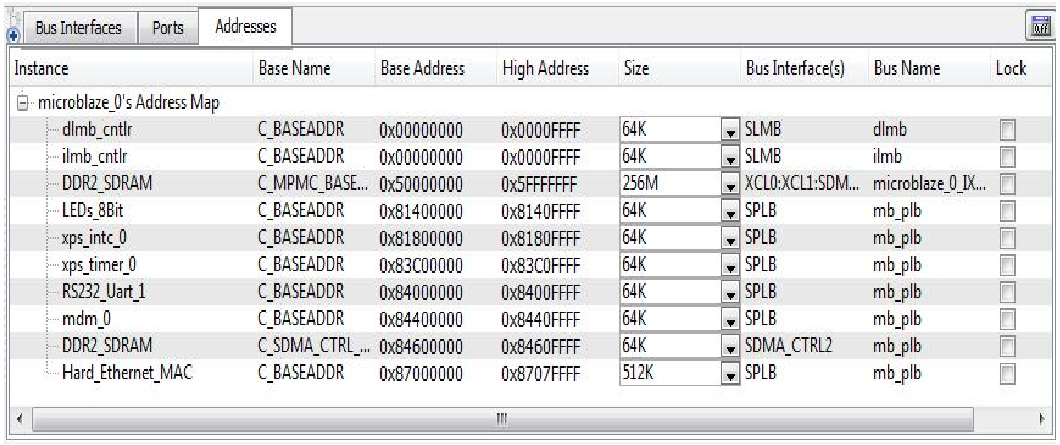
ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014

DDR_SDRAM memory, the serial communication device RS232_UART, GPIO LEDs for display, Hard Ethernet MAC – 'xps_ll_temac' and timer - 'xps_timer'. The timer was used for the reference time generation part.



Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock
microblaze_0's Address Map							
dlmb_cntlr	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB	dlmb	
ilmb_cntlr	C_BASEADDR	0x00000000	0x0000FFFF	64K	SLMB	ilmb	
DDR2_SDRAM	C_MPMC_BASE...	0x50000000	0x5FFFFFFF	256M	XCL0:XCL1:SDM...	microblaze_0_IX...	
LEDs_8Bit	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	mb_plb	
xps_intc_0	C_BASEADDR	0x81800000	0x8180FFFF	64K	SPLB	mb_plb	
xps_timer_0	C_BASEADDR	0x83C00000	0x83C0FFFF	64K	SPLB	mb_plb	
RS232_Uart_1	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	mb_plb	
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb_plb	
DDR2_SDRAM	C_SDMA_CTRL_...	0x84600000	0x8460FFFF	64K	SDMA_CTRL2	mb_plb	
Hard_Ethernet_MAC	C_BASEADDR	0x87000000	0x8707FFFF	512K	SPLB	mb_plb	

Fig. 2 The automatically generated addresses

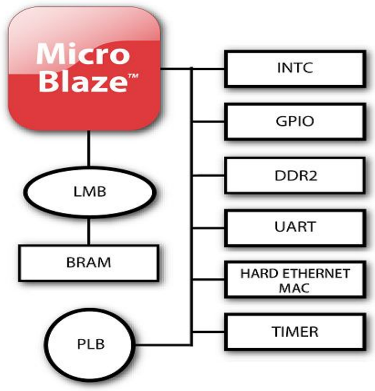


Fig. 3 The block diagram

Interrupt is enabled for both Hard Ethernet MAC and the timer. Interrupt handling was done with the help of the interrupt handling controller (INTC) named xps_intc_0. The instruction and data cache configuration is the final part. The automatically generated addresses by the XPS tool for each IP cores are shown in Fig. 2. The resulted block diagram of the design is shown in Fig. 3 [6].

B. Software Design

The software part of the design is configured using the SDK tool. For configuring the software portion we have to export the hardware design to the SDK along with the BIT file. SDK provides us with a great number of project templates that we can pick from. In our design we have to configure the Ethernet MAC inside the ML505 board such that it can receive UDP packets. So to meet our requirement, we can select the lwIP Echo Server template from the available templates [7]. The lwIP Echo Server template provides a simple demonstration of how to use the light weight IP stack. The server works on TCP data and listen for the input at the specified port and simply echoes back whatever data is sent to that port. Our requirement is to receive UDP packets, modify them and either transmit them or else display them on the LEDs. So we developed our C-program accordingly. In our design we will bind the IP address

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

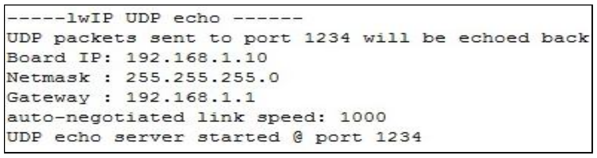
(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014

192.168.1.10 with our board's MAC address. Also our design is configured such that it will be listening for the input at the port 1234.

IV. IMPLEMENTATION

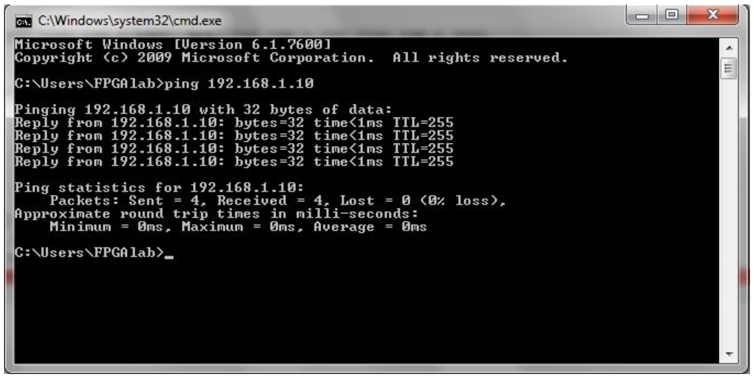
The FPGA board is connected to an Ethernet port on the host computer via an Ethernet cable. Next an IP address is assigned to the Ethernet interface on the host computer. The IP address of the PC and the board must be in the same subnet. The software application assigns a default IP address of 192.168.1.10 to the board. So in our design we are assigning the PC with the IP address 192.168.1.100.



```
-----lwIP UDP echo ------
UDP packets sent to port 1234 will be echoed back
Board IP: 192.168.1.10
Netmask : 255.255.255.0
Gateway : 192.168.1.1
auto-negotiated link speed: 1000
UDP echo server started @ port 1234
```

Fig. 4 Output at the serial port

The C-code written in SDK is compiled with the GNU Compiler tool. The compiled C-files along with the libraries generate the executable Executable and Linkable File (ELF) file. The final stage of designing is the association between the hardware and software parts and the download of the entire image into the FPGA. For this we will use the Data2MEM tool which links the BIT (bitstream) file generated at the end of hardware implementation and the compiled ELF file. The result is a download.bit file and this is downloaded into the FPGA. After successful download, we will get the output shown in Fig. 4 at the serial port.



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\FPGAlab>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255
Reply from 192.168.1.10: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\FPGAlab>_
```

Fig. 5 PING result

Now we will be able to ping to the IP address 192.168.1.10 from the PC. The ping result can be seen in the Fig. 5. We use a LabVIEW program which can transmit UDP packets to the IP address 192.168.1.10. So as we run the LabVIEW program we can see that the Ethernet LED's in the ML505 board lights up indicating that Ethernet reception and transmission are in progress. The Ethernet connection LED, transmission LED and reception LED lights up along with the reception and transmission speed indicator LED indicating 1 Gbps operation. If we have interfaced the design with the GPIO LEDs, we can see the result on LEDs. In the Fig. 6, you can see the data "1D" displayed on the GPIO LEDs.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014

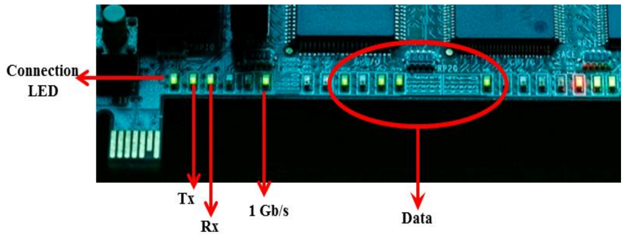
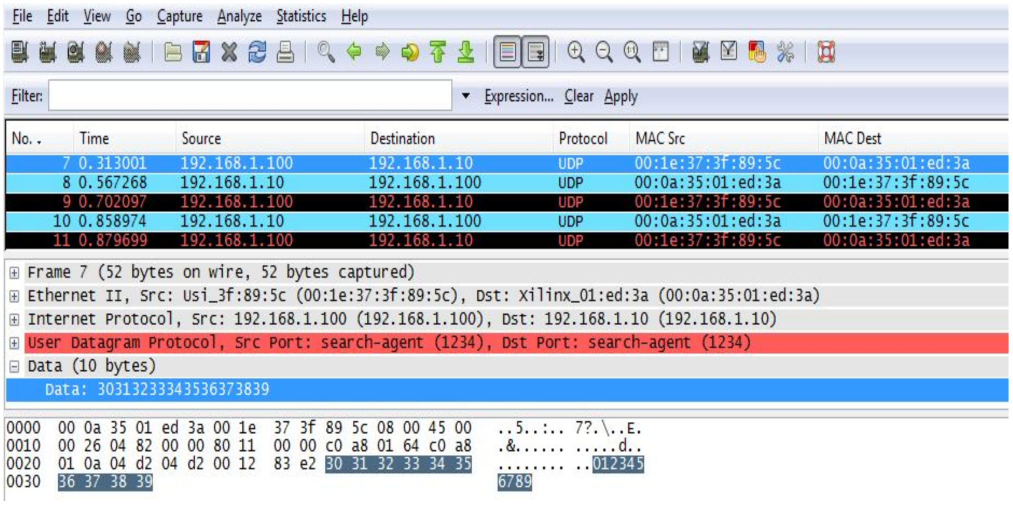


Fig. 6 Interfacing with LED

To verify that the frames are correctly received and transmitted, the frames are captured using Wireshark. From the packets captured using Wireshark (see Fig. 7), it is noted that the packets that are transmitted from the PC using the LabVIEW program are received on the board. The data that are transmitted are 1, 2, 3, 4, 5, 6, 7, 8 and 9. By checking the data field of the received packet, we have seen that the ASCII values of the data are received on the board. That is, 31, 32, 33, 34, 35, 36, 37, 38 and 39.


File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. .  Time       Source          Destination     Protocol  MAC Src            MAC Dest
   7 0.313001     192.168.1.100   192.168.1.10    UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
   8 0.567268     192.168.1.10    192.168.1.100   UDP       00:0a:35:01:ed:3a  00:1e:37:3f:89:5c
   9 0.702097     192.168.1.100   192.168.1.10    UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
  10 0.858974     192.168.1.10    192.168.1.100   UDP       00:0a:35:01:ed:3a  00:1e:37:3f:89:5c
  11 0.879699     192.168.1.100   192.168.1.10    UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a

⊞ Frame 7 (52 bytes on wire, 52 bytes captured)
⊞ Ethernet II, Src: Usi_3f:89:5c (00:1e:37:3f:89:5c), Dst: Xilinx_01:ed:3a (00:0a:35:01:ed:3a)
⊞ Internet Protocol, Src: 192.168.1.100 (192.168.1.100), Dst: 192.168.1.10 (192.168.1.10)
⊞ User Datagram Protocol, Src Port: search-agent (1234), Dst Port: search-agent (1234)
⊟ Data (10 bytes)
    Data: 30313233343536373839

0000  00 0a 35 01 ed 3a 00 1e  37 3f 89 5c 08 00 45 00   ..5..:.. 7?.\..E.
0010  00 26 04 82 00 00 80 11  00 00 c0 a8 01 64 c0 a8   .&...... .....d..
0020  01 0a 04 d2 04 d2 00 12  83 e2 30 31 32 33 34 35   ........ ..012345
0030  36 37 38 39                                        6789
```

Fig. 7 Received frames captured in Wireshark

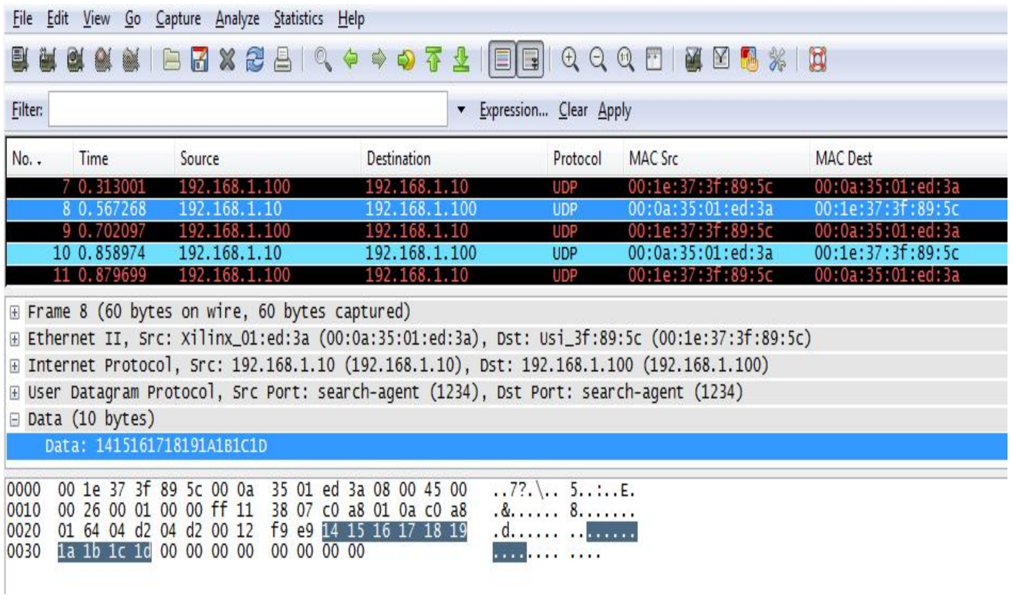
In the present case, the modifications include converting the data in ASCII format into the corresponding decimal format, adding a constant decimal value “20” to them and then transmit them back to the PC. In Fig. 8, we can see the corresponding hex values of the modified data are transmitted back to the PC. That is, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C and 1D.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014


File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

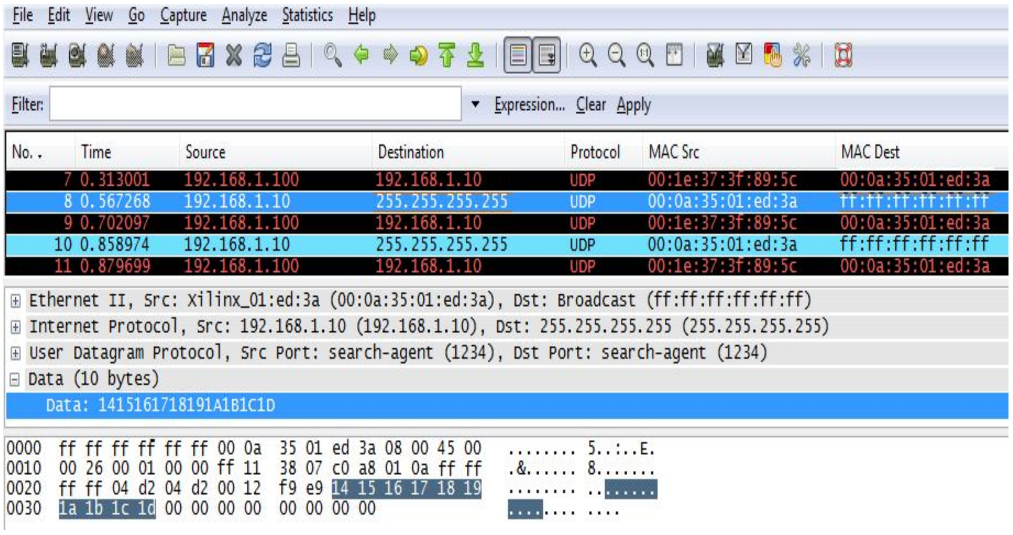
```
No. .  Time       Source          Destination     Protocol  MAC Src            MAC Dest
    7 0.313001   192.168.1.100   192.168.1.10    UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
    8 0.567268   192.168.1.10    192.168.1.100   UDP       00:0a:35:01:ed:3a  00:1e:37:3f:89:5c
    9 0.702097   192.168.1.100   192.168.1.10    UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
   10 0.858974   192.168.1.10    192.168.1.100   UDP       00:0a:35:01:ed:3a  00:1e:37:3f:89:5c
   11 0.879699   192.168.1.100   192.168.1.10    UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
```

```
⊞ Frame 8 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Xilinx_01:ed:3a (00:0a:35:01:ed:3a), Dst: Usi_3f:89:5c (00:1e:37:3f:89:5c)
⊞ Internet Protocol, Src: 192.168.1.10 (192.168.1.10), Dst: 192.168.1.100 (192.168.1.100)
⊞ User Datagram Protocol, Src Port: search-agent (1234), Dst Port: search-agent (1234)
⊟ Data (10 bytes)
    Data: 1415161718191A1B1C1D
```

```
0000  00 1e 37 3f 89 5c 00 0a  35 01 ed 3a 08 00 45 00   ..7?.\.. 5..:..E.
0010  00 26 00 01 00 00 ff 11  38 07 c0 a8 01 0a c0 a8   .&...... 8.......
0020  01 64 04 d2 04 d2 00 12  f9 e9 14 15 16 17 18 19   .d...... ........
0030  1a 1b 1c 1d 00 00 00 00  00 00 00 00              ........ ....
```

Fig. 8 Modified frames transmitted back to PC captured in Wireshark

We are able to transmit the data to any IP address by making required changes in the corresponding C file. In the Wireshark result shown in Fig. 9, it can be seen that the board is receiving data from the IP address 192.168.1.100 and after modifying the received data, it is broadcasted.


File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

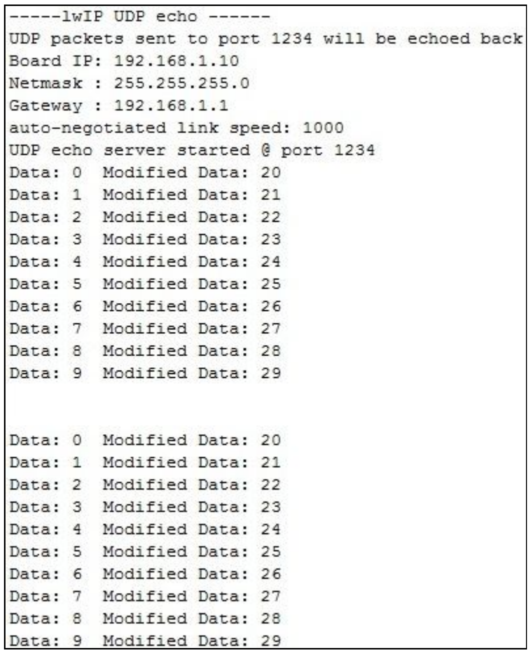
```
No. .  Time       Source          Destination       Protocol  MAC Src            MAC Dest
    7 0.313001   192.168.1.100   192.168.1.10      UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
    8 0.567268   192.168.1.10    255.255.255.255   UDP       00:0a:35:01:ed:3a  ff:ff:ff:ff:ff:ff
    9 0.702097   192.168.1.100   192.168.1.10      UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
   10 0.858974   192.168.1.10    255.255.255.255   UDP       00:0a:35:01:ed:3a  ff:ff:ff:ff:ff:ff
   11 0.879699   192.168.1.100   192.168.1.10      UDP       00:1e:37:3f:89:5c  00:0a:35:01:ed:3a
```

```
⊞ Ethernet II, Src: Xilinx_01:ed:3a (00:0a:35:01:ed:3a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol, Src: 192.168.1.10 (192.168.1.10), Dst: 255.255.255.255 (255.255.255.255)
⊞ User Datagram Protocol, Src Port: search-agent (1234), Dst Port: search-agent (1234)
⊟ Data (10 bytes)
    Data: 1415161718191A1B1C1D
```

```
0000  ff ff ff ff ff ff 00 0a  35 01 ed 3a 08 00 45 00   ........ 5..:..E.
0010  00 26 00 01 00 00 ff 11  38 07 c0 a8 01 0a ff ff   .&...... 8.......
0020  ff ff 04 d2 04 d2 00 12  f9 e9 14 15 16 17 18 19   ........ ........
0030  1a 1b 1c 1d 00 00 00 00  00 00 00 00              ........ ....
```

Fig. 9 Broadcasted frames captured in Wireshark

If we are trying to display the modified data by adding suitable print commands in the main C-file, we will get the output shown in Fig. 10, at the serial port.



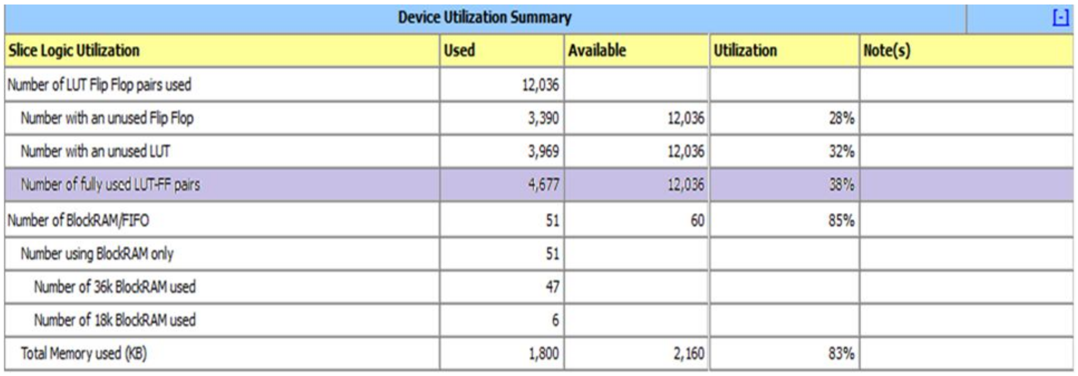
```
-----lwIP UDP echo ------
UDP packets sent to port 1234 will be echoed back
Board IP: 192.168.1.10
Netmask : 255.255.255.0
Gateway : 192.168.1.1
auto-negotiated link speed: 1000
UDP echo server started @ port 1234
Data: 0  Modified Data: 20
Data: 1  Modified Data: 21
Data: 2  Modified Data: 22
Data: 3  Modified Data: 23
Data: 4  Modified Data: 24
Data: 5  Modified Data: 25
Data: 6  Modified Data: 26
Data: 7  Modified Data: 27
Data: 8  Modified Data: 28
Data: 9  Modified Data: 29


Data: 0  Modified Data: 20
Data: 1  Modified Data: 21
Data: 2  Modified Data: 22
Data: 3  Modified Data: 23
Data: 4  Modified Data: 24
Data: 5  Modified Data: 25
Data: 6  Modified Data: 26
Data: 7  Modified Data: 27
Data: 8  Modified Data: 28
Data: 9  Modified Data: 29
```

Fig. 10 Serial Output after modification

V. CONCLUSION

MicroBlaze processor with Hard Ethernet MAC was configured using the EDK tool. Using SDK, lwIP was configured inside the processor. The configuration is capable of receiving packets sent to the board’s IP address. The processor will modify the data accordingly and will display it on the LEDs. We were also able to transmit the modified data to any IP address required. The details of FPGA resource utilization is given in Fig. 11.



Device Utilization Summary

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of LUT Flip Flop pairs used	12,036			
Number with an unused Flip Flop	3,390	12,036	28%	
Number with an unused LUT	3,969	12,036	32%	
Number of fully used LUT-FF pairs	4,677	12,036	38%	
Number of BlockRAM/FIFO	51	60	85%	
Number using BlockRAM only	51			
Number of 36k BlockRAM used	47			
Number of 18k BlockRAM used	6			
Total Memory used (KB)	1,800	2,160	83%	

Fig. 11 Device utilization summary

ACKNOWLEDGMENTS

The authors wish to thank Director, NPOL for permitting to carry out this project. Authors also wish to thank Mr. Suresh M., Scientist G, NPOL and Mrs. Jayamma T. M., Scientist F, NPOL for their valuable guidance, help and insightful comments.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2014

REFERENCES

[1] Xilinx Inc. MicroBlaze Reference Manual, UG081 (v13.2)
[2] Xilinx,http://www.xilinx.comlproducts/design
[3] EDK Concepts, Tools and Techniques, A Hands-On Guide to Effective Embedded System Design, UG683 April 24, 2012
[4] Embedded System Tools Guide, October 6, 2003
[5] Platform Specification Format Reference Manual, UG642 July 6, 2011
[6] ml505_overview_setup
[7] Xilinx, Embedded System Example, XAPP1026, version 2.2, 2006
[8] National Instruments LabVIEW Fundamentals and Getting Started Guide.