University at Buffalo
The State University of New York

Implementation of UDP over IP with Walkie-Talkie Application

Project Report

EE550: Networked Systems Design

Keyur Ajay Ketkar (50206159)

Rishabh Ganjoo (50206867)

Roshan Baby Reji (50205771)

Introduction

Ethernet is a group of frame-based computer networking technologies for local area networks. It is a standard which characterizes various wiring and signaling standards for Physical Layer of OSI model. It supports very high data communications up to Gigabit level. Thus, in most of the situations which demands very high speed data communications, Ethernet is preferred. Hence, implementation of Ethernet standard on FPGA is fundamental for fast prototyping of a wide range of digital systems. The primary objective of this project is to implement an Ethernet core with a simplified upper layer interface on FPGA which implements a communication interface between two computers using Triple Speed Ethernet on De2i-150 board.

Project Goal

For our project, we designed a software interface that communicated with the board via the Altera Monitor. Through this interface, the board could receive data packets sent over an Ethernet network. Additionally, the board was able to transmit data packets of its own. The User provides input message to transmit, it is displayed at the receiving end on successful reception.

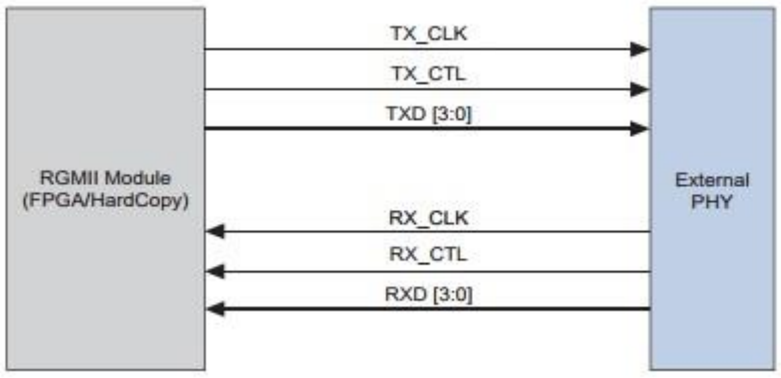
System is half-duplex, defined by switch condition for transmission and reception. (Switch=>OFF>Transmit mode, Switch=> ON-> Receive mode). Thus, at a time only one computer will transmit the signal and the other one will receive.

Layers and Protocols

Physical Layer

The Physical layer establishes how signals can be transmitted on a network, gives interfaces for a network and characterizes the different types of physical aspects. So as to have an Ethernet connection, physical layer uses PHY device. For associating PHY device to the FPGA, RGMII interface is used.

RGMII is a contrasting to the IEEE 802.3z GMII with reduced pin count.



Signal Diagram of RGMII

Data Link Layer (MAC)

One of two sub layers of the Data Link layer is the Media Access Control Layer (MAC). The MAC layer is responsible for delivering data packets over a shared channel. MAC protocol has been used in order to have raw packet communication and to verify solution then it has been improved to UDP communication.

Network Layer

IP (Internet Protocol) is an important Protocol at network layer because it introduces a way to deliver messages from source to destination. The IPv4 (Internet Protocol version 4) is the most extensively used Internet Layer protocol. We found that, the combination of IPv4 with UDP, represents the ideal solution in terms of hardware resource requirements for data transmission between a host PC and a FPGA board. Because of the comparatively small protocol overhead, UDP allows for high transmission rates whereas meanwhile requiring low programming overhead on the PC side, i.e., programming interfaces for UDP are well documented and promptly accessible for all common operating systems.

Transport Layer

Transport Layer has two protocols; the Transmission Control Protocol(TCP), which provides a communication with reliable data delivery, and the User Datagram Protocol (UDP) which gives an unreliable communication. UDP is such a simple transport protocol that allows applications to send datagram and handle translation between ports and sockets.

UDP packets are simple packets of data that are sent from one device to another, with no expectation of acknowledgement. This is usually used in cases when a single missing packet will not noticeably degrade message quality, or the packet is expected to be used and immediately discarded.

Implementation

Hardware

The Nios II processor core is a soft-core central processing unit that you could program onto an Altera field programmable gate array (FPGA). The system can be built using Qsys tool in Quartus Prime.

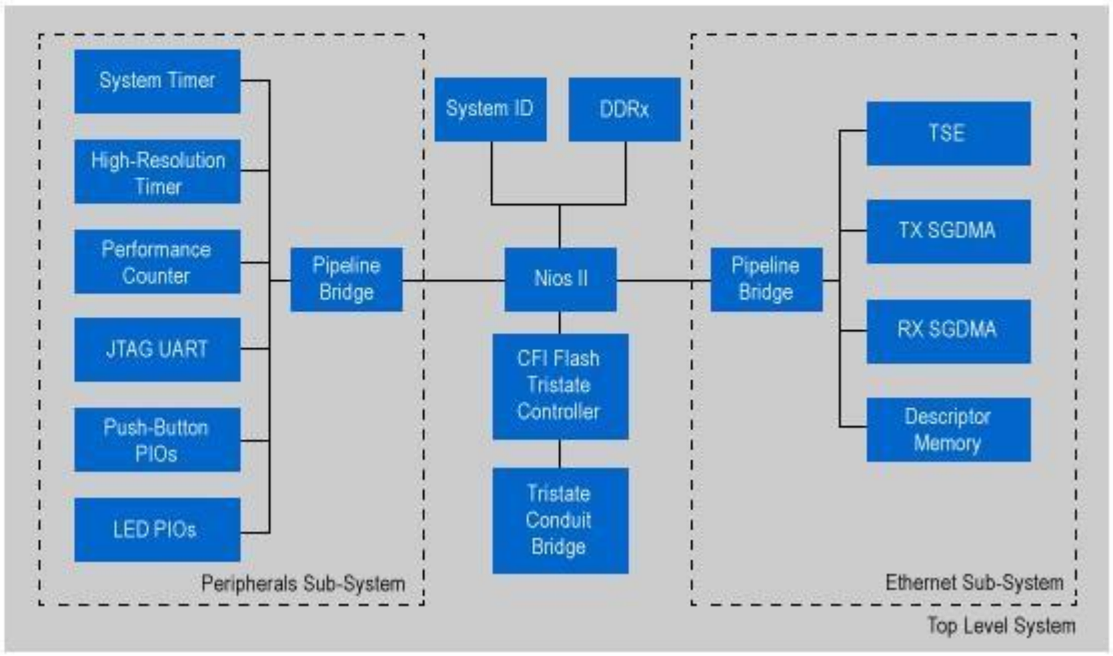
The example NIOS II standard hardware system provides the following necessary components:

• Nios II processor core, that’s where the software will be executed

• On-chip memory to store and run the software

• JTAG link for communication between the host computer and target

• hardware (typically using a USB-Blaster cable) • LED peripheral I/O (PIO), be used as indicators



Block Diagram of Nios II subsystem

This subsystem takes a clock signal and an active-low reset signal as inputs, and communicates with the external PHY chip through a chosen interface. There is a Nios II processor to run application programs, a JTAG UART component to support communication between the processor and the host computer, and a Triple-Speed Ethernet IP Core to implement the MAC sublayer and a partial Physical layer when needed based on the interface type. The two SGDMA controllers are used for transmit and receive functions of the core. The on-chip memory is used for the program code, data, as well as descriptors for the SGDMA controllers.

- After adding all the components using Qsys, we assigned Base addresses, and then generate the Verilog HDL code.
- Integrated the Nios system and three additional modules within the top-level module.
- Created a new TimeQuest SDC File.
- We assigned FPGA Pin locations, compiled the design and downloaded the SRAM object file that contains the Nios II system on the board.

Software

We have defined structures to encapsulate data at each layer, where upper layer will pass parameters to the lower layer.

IP protocol is simplified, not including option filed and the checksum. Each layer has "layer_send()" function, which passes the parameters to the lower layer. The structure builds the header, and the packet. Rx_ethernet_isr() function is called when Transmit and Receive interrupts are invoked.

Application Layer:

User is asked to provide input message in the console at the transmitting end and the received message is displayed on the receiving console on the receiving end. The function gets(mssg) is invoked to accept the input from the user. If the user enters ***over and out*** the program is terminated.

Transport Layer:

Here we use the UDP protocol. For the source and destination port, we have created structure "socket_address()". This structure provides IP address and the port address. This is done because in the actual socket programing, socket address is taken as the combination of IP address and port address. Structure "udp_header()" builds the header for the UDP. It takes parameters source port, destination port, udp_length and udp_crc. For simplification, we have set udp crc as 0. Using these parameters, UDP frame is built by structure udp_datagram().

- This structure takes parameters from the udp_header() and payload is the user input message.
- Udp_send() function is used to pass the UDP segment to network layer. UDP length is calculated starting from the header.

Network layer:

Here we use the IPV4 protocol.

We use the structure ip_header() to build the header. Source and destination IP addresses are taken from socket, and are entered manually. For simplification we are not using options and padding field of the conventional IPV4 header. Time to live is set to 255. We are not using checksum to avoid complexity.

- IP packet is built by structure ip_packet(). It takes parameters from ip_header() and payload is the entire udp frame.
- Ip_send() function is used to pass the IP packet to link layer.IP length is calculated starting from IP header.

Link Layer:

Here we use the Ethernet protocol.

We use the structure ethernet_frame() to build the ethernet frame . Destination and source MAC addresses are entered manually. Payload is the IP packet. We enter two "0" bytes at the beginning of the etherent frame, as the tse strips of first two bytes due to 32-bit alignment.

- ethernet_send() function is used to send this frame to tse for transmission.

Transmission and Reception:

We take the decision of transmission and reception based on the switch condition. If the switch is “ON”, the system is in transmit mode. If the switch is “OFF” the system is in the receiving mode.

Avalon provided functions are used to invoke transmit and receive interrupts. Creating sgdma transmitter and receiver and setting up non-blocking transfer of sgdma descriptor at transmitter and receiver.

- We create transmit and receive vectors using structure ethernet_frame().
- We use rx_etherent_isr() subroutine for reception of the ethernet frame.
- TSE is configured for transmitter and receiver using command word.

Max message size:

Conventional Ethernet frame has max data load of 1500 bytes, we have restricted the max length to 1024 bytes.

Conclusion

We successfully implemented UDP over IP and established a communication between two De2i150 boards, where the user is able to send and receive the message hence the walkie-talkie application is successfully accomplished.

UDP packets are recognized and their data is correctly displayed on the terminal.

Improvements

Full-duplex system can be implemented, but we will have to take care of the collisions when both the devices are in transmit mode.

Also, Checksums can be implemented, we have to consider the capacity of processor to handle complex algorithms.