Implementation of multi-serials to Ethernet Gateway based on Field Programmable Gate Array

D.YESUBABU, B.SRIHARI
[1]*PG-Student, Dept.of ECE, CRV INSTITUTE OF TECHNOLOGY&SCIENCE , SHAMEERPET RANGAREDDY DISTRIC*
[2]*Associate Professor, Dept. Of ECE, CRVITS, SHAMEERPET, RANGAREDDY DISTRIC, HYDERABAD, India*
1devarakondayesubabu@gmail.com
2srihari.royal@gmail.com

***Abstract*—The Unmanned Aerial Vehicle (UAV) data link is composed of ground control station and airborne data terminal, all of which communicate in UART mode. In order to achieve multi-serials communication between UAV and ground control station, a method of multi-serials to Ethernet Gateway based on field programmable gate array (FPGA) +network interface chip is given. In this paper the structure and working principle of system is introduced first. Then the design of system's hardware and software programming were described. The modules of transfer and receiver were completed by state machine. Finally, the timing simulation of two serial ports was shown. It has been proved in the real test that the design can communicate with computer and work well, and achieve the prospective purpose. It is of much great practical significance and operational benefits.**

Keywords-Ethernet; FPGA; multi-serials; Gateway; UAV; UART

I. INTRODUCTION

The Unmanned Aerial Vehicle (UAV) data link [1] is composed of ground control station and airborne data terminal. The computers on the ground need control and check more than ten assemblies, containing main and sub remote control transmitters, telemetry receivers, image decompression board, ground positioning receiver and radio location tracking servo system etc. The communication between them is standard asynchronous serial form and at the current, the serial ports are expanded by MOXA card, whose weaknesses are: complicated cable wiring, frequently interruption of CPU, which greatly reduces the CPU's efficiency and impacts the system's real-time Processing. To resolve these problems, a method of multi serials to Ethernet Gateway based on the field programmable gate array (FPGA) +network interface chip is presented. The Gateway will send data as Ethernet frame format after receiving serial data, indirectly achieves multi-serials communication, simplifies cabinet wiring and improves CPU's efficiency.

II. SYSTEM STRUCTURE AND WORKING PRINCIPLE

The Gateway mainly consists of FPGA, Ethernet module and level converter. Using a flexible FPGA programming Feature, a UART [2] can be designed in it. If several UARTs are in it, the system has the capacity of communication with multiple serial ports. The Ethernet module implements Ethernet communication and is configured at the time of initialization as we can see in Fig. 1, the Gateway's main function is to achieve communication between the serial devices and Ethernet. When it receives data from devices, Gateway will choose useful data from serial data frame following the communication protocol, and send data after packaged. When it receives data from Ethernet, it firstly unpacks the frame and determines the port number to transfer data to its buffer and adds the synchronous heads.

III. HARDWARE DESIN

The FPGA [3] [4], ep2c70, belongs to Alter a Cyclone II family, ups to 1.1Mbits of embedded memory, satisfies high-speed and large-capacity communication completely. Asynchronous serial communication level is different from FPGA level, so it must shift level. SP208E is selected, which integrates a 4-channel RS232 transceiver and is fully compatible with CMOS/TTL levels. The Network controller is RTL8019AS, which supports PNP automatic detection, embeds 16KB SRAM, includes a full-duplex communication interfaces. It is designed for the ISA bus and used to implement the network physical layer protocol. The overall structure of the hardware is shown in Fig.2.

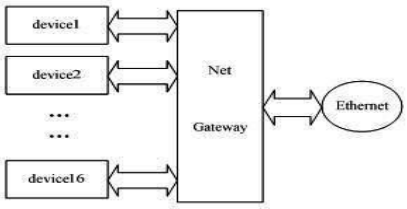


Figure 1. Basic Structure of the System

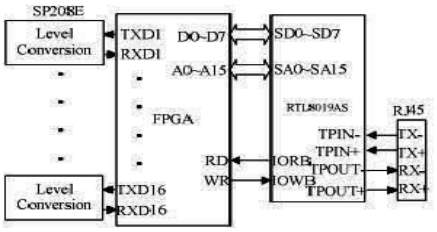


Figure 2. Basic Structure of the Hardware

A. Serial ports module

The code file of serial ports is programmed in VHDL language with the Quartus II7.2 platform.

A single serial port module consists of three parts: control module, data receiver module and data transfer module. Working diagram is shown in Fig. 3.

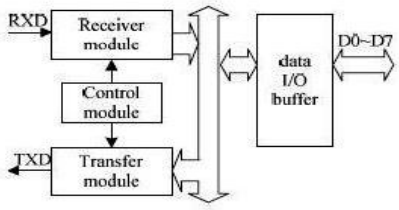


Figure 3. Sketch map of serial port

1) Control module: This module is mainly to complete the registers configuration, the data format setting, the baud rate and the size of the receive the block setting, as well as the controlling of sending or receiving data and the judgement of data status. Registers need to be configured are shown in Table I. In this design, a 40.6MHz crystal oscillator is used. For the purpose of improving the accuracy of sampling, we use FPGA's phase-locked loop (PLL) function [3]. Firstly, converting this frequency to 19.2MHz, then introducing it to each port module and dividing it into different baud rates as you need. For example, we need a rate of 19200b/s. What we should do is writing 1000 to baud rate register. Counter will jump when it notes to 500 and jump again when it notes to 1000. And then clears zero to start over. In this way, we can get a 50% duty cycle clock.

TABLE I SEND AND RECEIVE REGISTERS

Send Data	Receive Data
Send hold register	Receive buffer register
Send baut rate control register	Receive counter
Send counter	Receive enable control register
Send control and state register	

2) Receiver module: This module is mainly responsible for data reception and the conversion of data. It mainly contains data registers and receivers. When the valid stop bit is detected, data are sent into the register. When the size equals to the setting value, data in the register are transferred into the I/O buffer. Receive controller mainly includes a counter and a 8 bits shift-register. It is controlled by the state machines [4], as shown in Fig. 4.

3) Transfer module: This module is used to send bus data received from Ethernet. It consists of transmitter and data registers. When the command is executed, transfer will send data in serial form until the send counter is 0, which means over. At last, the flag bit is set. The transmitter is a 8-bit shift register. As long as data register is not empty, shift register would constantly read data, add start and stop bits and send them in asynchronous frame format. It is also controlled by state machines, as shown in Fig. 5. When the send clock is high, if the data register is full, the data will sent in order. The coordinated communication of multiple serial ports: in order to allow ports communicate effectively, there is a need to administer them. As we know, the FPGA processing speed is far greater than peripheral transmit data rate, so once I/O buffer receives data from any serial port, the data can be read
by Ethernet interface chip directly. So long as the serial data start address and end address doesn't conflict.

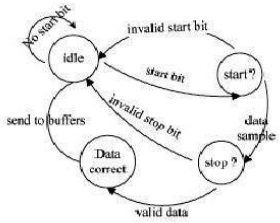


Figure 4. The state of receiving register is not full

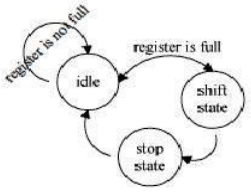


Figure 5. The state of sending

B. Ethernet interface chip drive

The chip is used to write data into I/O buffers in right format and start sending commands. The 8019 will
Automatically convert data packets into appropriate form and transmit them in physical channel. On the contrary, the 8019 will restore physical signals into data, store them in I/O buffers. In short, it implements conversion between data packets and electrical signals. Ethernet protocol will be done automatically by the chip. Drive program includes the initialization of the 8019, receiving program and sending program. Reg00, named command register (CR) is a byte whose address offset is 0x00, as listed in Table II.

TABLE II. COMMAND REGSITERS

bit	7	6	5	4	3	2	1	0
name	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP

bit 7 6 5 4 3 2 1 0 name PS1 PS0 RD2 RD1 RD0 TXP STA STP In Table 2, PS0 and PS1 bits are used to select register pages. When PS1:PS0=0x00, page 0 is selected.

When PS1:PS0=0x01, page 1 is selected. 0x10, page 2, 0x11, page 3. The parameter is page-number used to specify the number of page. The registers configuration is as follows:

• reg00=0x21; //page 0, stop running, wait for initialization.
• reg01=0x4c; //address of receiver buffer start page (PSTART).
• reg02=0x80; //address of receiver buffer stop page (PSTOP).
• reg03=0x4c; //read pointer, point to the last page has been read (BNRY).
• reg04=0x45; //transmitter page start register (TPSR).
• reg0c=0xcc; //receiver construction register (RCR).
• reg0d=0xe0; //transmitter configuration register (TCR)
• reg0e=0xc8; //data configuration register (DCR).
• reg0f=0x00; //disable all interrupt.
• reg07=0x4d; //write pointer, point to the current stop page of receiving (CURR).
• reg00=0x22; //select page 0 register, start to execute the command. As depicted in front, serial port data need to package and Ethernet data need to unpack. All data need a unified frame format. Ethernet format [5] is in Table 3. When 8019 receives a valid packet, the pointer will return the received data, otherwise it returns NULL. There are two registers (CURR, BNRY) to control the receiver buffers at the start of initialization, BNRY=CURR-1, denotes that there is no packet. Before calling the receiver function, run the check function firstly to judge whether a new packet comes. Flow chart is shown in Fig. 6.

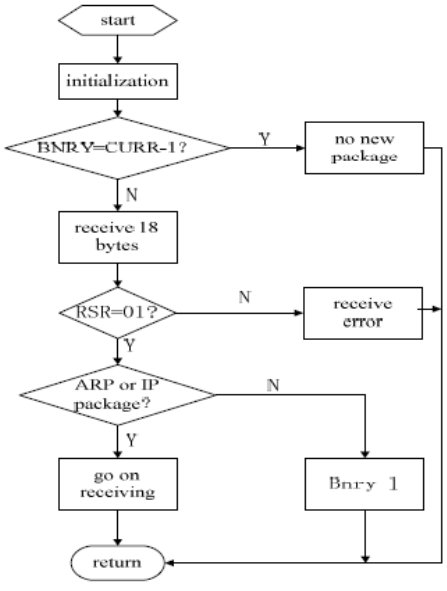


Figure6.Flow chart of receiving data

When 8019 sends data, it will choose send buffer accordant with the value of txd_buffer_select bit variable. Through Remote direct memory access (DMA), data are written to TXD buffer, and then local DMA starts to work. Flow chart is shown in Fig. 7.

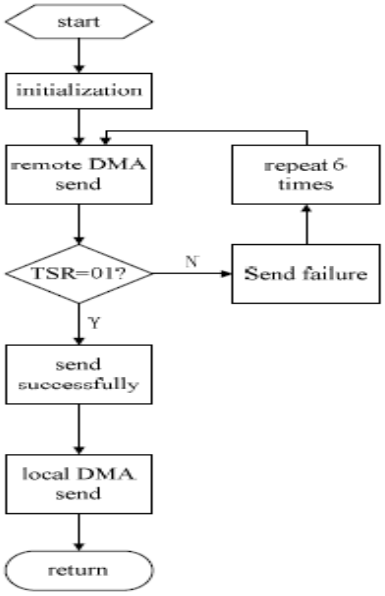


Figure7.Flow chart of sending data

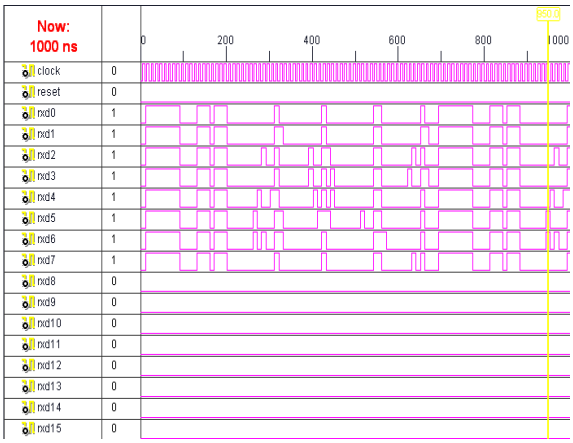
C. Analysis of the whole data flow

In order to save CPU's resources, serial ports don't send data if only receive a frame but wait for the time 10 correct frames reception. At the same time, considering the real time of system, data from serial ports would be packaged every 100ms, and sent to Ethernet data buffers. In this way, CPU can process more data at the same time, reduce the possibility of wait in vain, advance the performance of real time. After the system receiving data from Ethernet, it will send them to corresponding port according to the port number; of course, adding the synchronous heads before start.

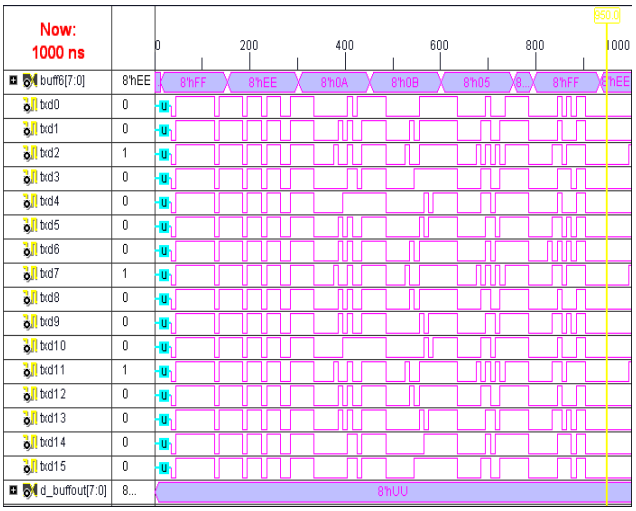
V. SIMULATION AND RESULTS

In this simulation, two ports are used to test the accuracy of receiving and transmitting, by using upper machine data Analysis tools programmed in C++. The UARTs have worked correctly for a long time. The timing simulation is shown in Fig.

Receiving Data Results

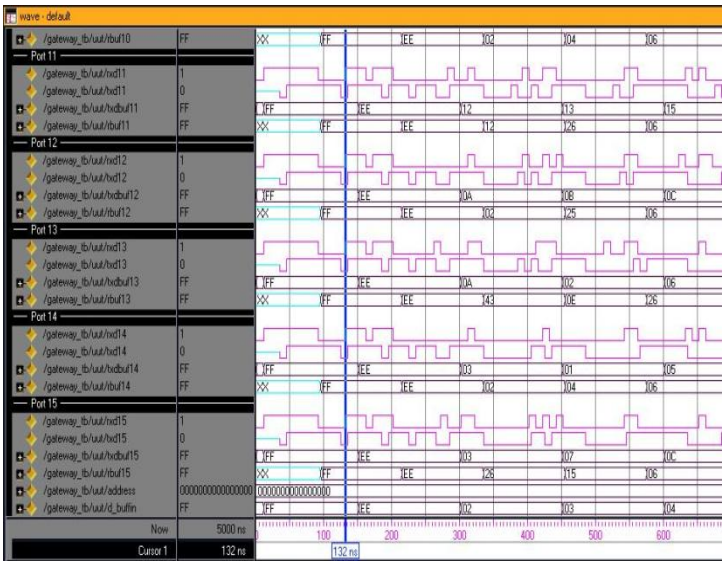


Transmitting Data Results

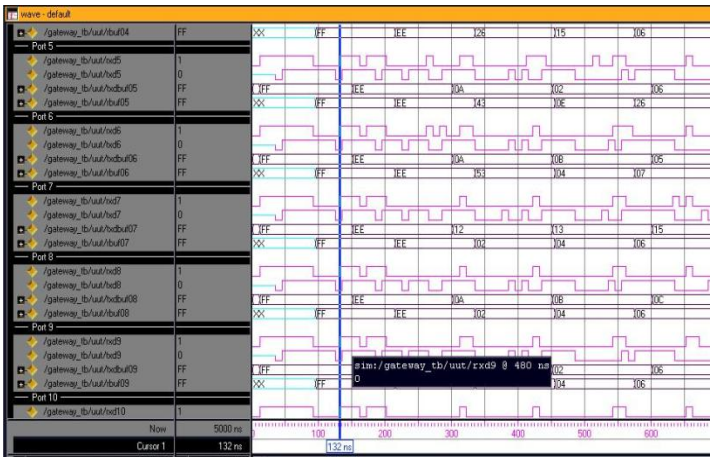


Port0 – Port4 Results

Port5 – Port10 Results



Port11 – Port15 Results



Design Summary

GATEWAY5 Project Status			
Project File:	gateway5.ise	Current State:	Mapped
Module Name:	Gateway_Top	• Errors:	1 Error (1 new)
Target Device:	xc3s250e-5tq144	• Warnings:	104 Warnings (104 new, 0 filtered)
Product Version:	ISE 9.1i	• Updated:	Sun Sep 30 20:06:05 2012

GATEWAY5 Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Total Number Slice Registers	734	4,896	14%	
Number used as Flip Flops	606			
Number used as Latches	128			
Number of 4 input LUTs	1,248	4,896	25%	
Logic Distribution				
Number of occupied Slices	777	2,448	31%	
Number of Slices containing only related logic	777	777	100%	
Number of Slices containing unrelated logic	0	777	0%	
Total Number of 4 input LUTs	1,248	4,896	25%	
Number of bonded IOBs	124	108	114%	OVERMAPPED
IOB Flip Flops	16			
Number of GCLKs	2	24	8%	
Total equivalent gate count for design	13,566			
Additional JTAG gate count for IOBs	5,952			

8. The “rxdr01” is receive signals from port 1, the “rbuf01” means read buffer. When it receives data from serial port, it will add synchronous heads “FF”, “EE”, and then hold those data or send them to Ethernet. The “txdbuf01” is transmitter buffer. Its contents are data from Ethernet. The “txd01” is send signals sent to port 1.

VI. CONCLUSION

The results of test denote that the method adopted in this paper can achieve the original purpose. Using this method, we simplify the communication between monitor computer and port devices, improve the efficiency of CPU, and

ensure the processing of system in real time. FPGA's flexible programming features also allow further upgrade for system. This design could be used in the domain of net control and information management system. It has a certain value for application.

ACKNOWLEDGMENT

I am extremely thankful to CRVITS Electronics Communication Engineering Department for providing excellent lab facilities which were helpful in successful completion of my project.

REFERENCES

- A.Ding Wang, Jiadong Xu, Rugui Yao, Ruifeng Miao, "Simulation System of telemetering and telecontrol for unmanned aerial vehicle," Aerospace and Electronic Systems Magazine, IEEE, Sept, 2006, pp. 3-5 [1]
- Shouqian Yu . Lili Yi, Weihai Chen, Zhaojin Wen, "Implementation of a Multi-channel UART Controller Based on FIFO Technique and. FPGA," industrial Electronics and Applications, Harbin, China, May 2007, pp. 2633-2638 [2]
- DUAN Peng, HE Mingyi, XUE Minbiao, "Design and Simulation of OFDM Synchronization System Based on FPGA," Meansurment and Control Technology, vol. 28, n. 11, 2009, pp. 63-67 [3]
- Yu Zhang, Yugui Hu, Kuixi Yin, Jilian Li, "Implementation of Multiserials Expand Based on FPGA," Electronic Instrument, vol. 32, n. 1 2009, pp. 233-236 [4]
- Yonghong Hu, Lu Ding, "Design and Realization of Multi-functional Gateway Based on SingleChip," 2009 2nd International Congress on Image and Signal Processing, Tianjin, China, October 2009, pp. 3709- 3712 [5]

