IMPLEMENTATION OF LAN PROTOCOL IN FPGA BASED C-BAND SYNTHESIZER IN RADAR

[1]Chandana B.Y, [2]J Pushpanjali, [3]Ruchit M.S.

[1]M.Tech. Student, [2]Assistant Professor, [3]Senior Engineer
[1,2]Bangalore institute of technology, [3]Bharat Electronics Limited
Bangalore, India

***Abstract*—Ethernet is a mode to connect devices remotely. Local Area Network (LAN) is a protocol which governs the set of laws for data transfer across devices. This paper involves the implementation of LAN protocol into a frequency synthesizer unit which operates in C-Band. LAN protocol is necessary as the data communication needs to happen at very fast rates as the unit requires very low switching times between the frequencies it needs to hop. This protocol has been implemented on a soft processor (Microblaze) invoked in a Xilinx (Artix7) FPGA (Field Programmable Gate Array).**

***Keywords*— LAN; RJ45; Phy; FPGA; MicroBlaze; Synthesizer;**

I. INTRODUCTION

LAN protocol consists of set of rules which govern data transfer across the devices. It requires a RJ45 connector on the board which transfers the data from the LAN cable to the PCB of the module. Connector also has filters which decouples noise from the signal lines. The choice of the connector and LAN cable being used needs to optimized for the application. FPGA can interpret only converted physical layer instruction from arrived data on the connector.. This operation is being carried out by a phy device. The phy device needs to be configured to auto negotiation mode to let the data transfer happen to the best possible speed, either 10/100/1000 Mbps, which can be achieved by the channel. There are LED pins which indicate the mode of operation of the phy device.

Data coming into the phy device is converted into nibble or byte data along with necessary signals such as enable, clock etc. Similarly, data which needs to transmit to the RADAR controller[1] should be sent to the phy device along with enable, clock and other necessary signals. The mode of the phy device can be changed by changing the hardware connections to the configuration pins or by soft controlling the device through the FPGA by writing its internal registers. In this paper, only four receive bits are used as it operates in 10/100 Mbps modes.

Figure 1 depicts a block level representation of how data transfer takes place between the module and RADAR controller, shown as a PC.

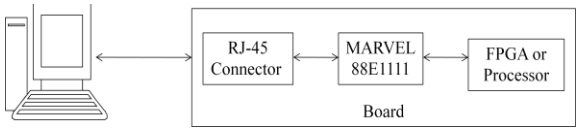

Fig. 1. Connection between Radar controller and module

II. THE FUNDAMENTAL COMPONENTS

A. Block diagram of synthesizer module

This protocol has been implemented in a frequency synthesizer project which operates in C band frequency range. Data which is sent to the module from a remote controller through LAN needs to be extracted and implemented by the FPGA in quick time as the module needs to work in a very agile environment. The block level diagram of the module is depicted in figure 2.

The module includes a PLL (Phase Locked Loop) [2] which locks to a particular frequency governed by the hardware connected to it and the data written into the PLL by the FPGA. The module also contains a DDS (Direct Digital Synthesizer) [3] which is used to switch among the programmed set of frequencies in less time. It also includes multipliers, switch filter banks (band pass), discrete filters and micro-strip based filters.

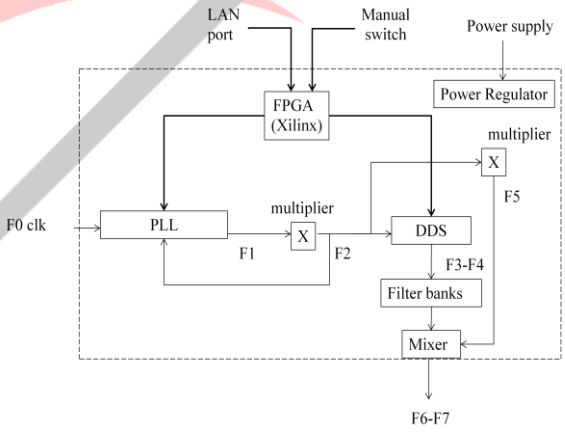

Fig. 2. Block diagram of the synthesizer module

B. Microblaze processor

A soft processor needs to be invoked into the FPGA. Here, a processor named microblaze[4] has been developed into a Xilinx FPGA. Various blocks need to be invoked for the microprocessor to function correctly. The various blocks for the microprocessor are listed below.

1) Clocking wizard: The processor needs a clocking wizard block to provide the clock frequency for its operation. Currently, 100 MHz clock is being generated by

the wizard through an inbuilt PLL mechanism. The input clock to the block is provided by the FPGA.

2) Microblaze debug module: It requires a Debug module block to be invoked which assigns the external interrupts to the processor.

3) Processor system reset: A reset module block has to be called which provides reset commands to various other blocks used.

4) Memory: The processor needs a localized memory of 32kb width. BRAM is the memory used to map the microprocessor.

5) Microblaze: A processor block needs to be called for which has a trace buffer size of 8kb.

6) AXI interface: The microprocessor pinouts are interfaced to the FPGA via An AXI interface. The AXI interface translates the data which can be interpreted by the FPGA.

7) AXI GPIO: GPIO is AXI based which carry the inputs and outputs to and from the processor. In this application, there are 15 AXI blocks used out of which 6 are inputs and remaining 9 are outputs. Except for one output block which is of 4 bits and provides the frequency information, all the other bocks are 32 bits in size.

III. SOFTWARE DESIGN

The FPGA which needs to be programmed requires software known as VIVADO[5]. This software helps to create the HDL and microprocessor blocks, synthesize, compile, implement and generate a bit file that can be loaded onto the FPGA.

The microprocessor which is invoked the VIVADO then needs to assign a file which tells the microprocessor what it needs to do. Here, applications know as Software Development Kit (SDK) is used[6].

The blocks mentioned in the previous sections need to invoked from the IP catalog the FPGA which is selected during the creation of the project in VIVADO.

The block of microblaze processor and its peripherals are shown in figure 3.

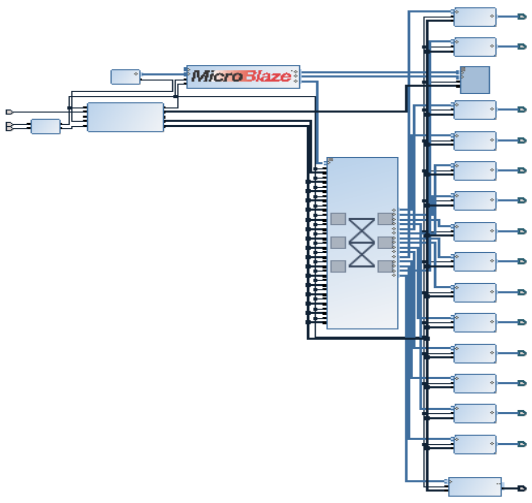

Fig. 3. Block diagram of Microblaze processor

During the development of the microprocessor block, every pin needs to be carefully connected. If not connected properly, the processor block will throw errors and will not be invoked properly. The pins of the FPGA which are actually interfaced with the processor need to be mapped properly during the implementation of the project. The microprocessor block when generated also generates an instantiation file which can be used as a component in the main VHDL code.

The bit file which is generated by the VIVADO in the initial stage includes only the hardware configuration of the processor. It needs to be assigned a file which tells it to function in a specified manner. The file with .elf extension is generated is known as an ELF file. This file is generated by the SDK software.

To SDK software export the generated bit file, along with the microblaze block. A C-project needs to be created where the functions to be executed the processor need to be written in C. This C project needs to be assigned the hardware platform which is imported from the VIVADO as shown in figure 4.

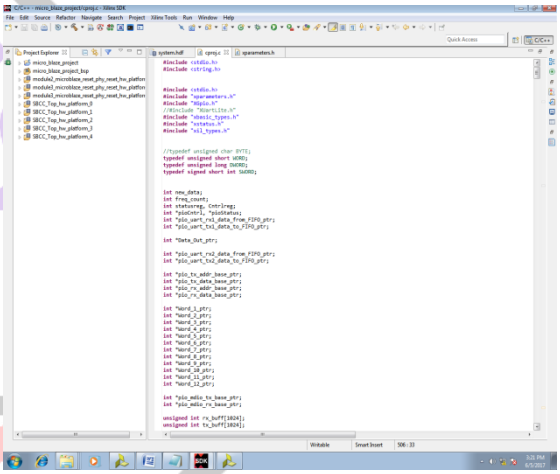

Fig. 4. C-program in SDK software

The various steps involved in the C code are listed below.

A. Initilization

In this function, various parameters are set for the processor. IP version of 4 is set. Header length of the IP is set as 20. IP time to live is set as 128. Set 0 as IP checksum. UDP length is set to 20. UDP checksum is set to 0. Source and destination MAC are set a value of 6 bytes each. UDP source and destination ports are set to a common number. Initialize the source and destination IP addresses.

B. DATA RECEIVE

The data received by the FPGA is stored into a First In First Out register (FIFO) of 32 bit width. Once the data arrives, an interrupt is generated and given to the processor. On the interrupt event, the C code starts to write its internal registers with the data present in the FIFO queue.

C. DATA INTERPRETATION

The data present in the internal registers of the microprocessor, which are 32 bit wide, is now analyzed. The LAN protocol formats have fixed defined locations. Now, the sender MAC and receive MAC addresses are read to know the valid transmitter and if the data is meant for the

module. Next, the mode is checked. This gives the information about the type of protocol being called for.

D. ADDRESS RESOLUTION PROTOCOL

The header provides the information about if the ARP protocol is called for. Next, it is checked if it is an ARP request or reply. If it is an ARP request, the IP address of the sender is recorded and it is checked if the request is intended for the module. If yes, then reply is sent to the sender in ARP reply format. If it is an ARP reply, it means sender has replied for the ARP request.

E. IP PROTOCOL

If the data does not have an ARP header, then it is checked for IP protocol header. If it matches, it can mean the data can be TCP, UDP, ICMP and many other protocols.

F. USER DATAGRAM PROTOCOL

If it's confirmed that it is an IP protocol, then it is checked UDP header. If it matches, then it means that the remaining data bits are the payload[7].

The C project is now saved and built to generate an .elf file. In the VIVADO software, the microprocessor needs to be associated with the generated elf file and the project needs to be updated to generate a bit file. This bit file now contains both the hardware section which includes the HDL code and the processor itself and the software part which in includes the elf file to instruct the microprocessor. This bit file can be converted into a .mcs file which can be loaded into a flash device mounted inside the module. Time taken to load a .mcs file is greater than the .bit file.A Graphical User Interface has been developed to transmit UDP packets to the module.

In this application, UDP packets are sent in two formats, format A and format B. Both the formats are encoded with an initial 2 bytes of data. This is to ensure that the UDP packet which is sent to the module is a valid data from the RADAR controller. The difference between the formats is that the 2 bytes are different. Two different applications run when the two formats are detected. The former store the frequency information into a temporary register, and the latter transfers the frequency information to the FPGA which performs the necessary action to change the frequency. This is done to achieve synchronization with the RADAR controller.

IV. RESULT

The PC IP address is set to RADAR controller IP address. Figure 4 depicts how the IP address can be altered.

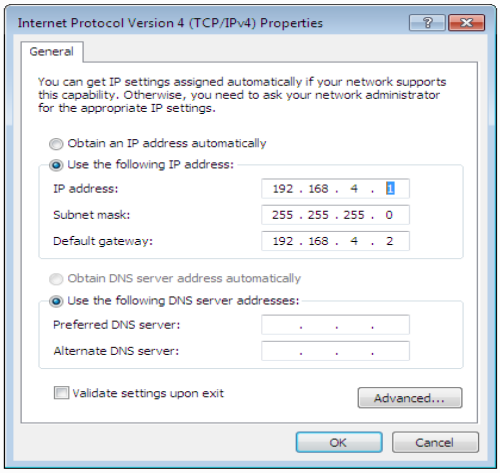

Fig. 4. IP address of computer

Once the module is powered up, the phy device is active. The LEDs on the module confirms that it is in Auto-negotiation mode. A PING operation is done to confirm the IP address of the module. The receiver IP address is sent with a ping instruction from the command prompt of the PC. If the IP address matches, the module replies to the request and result are shown in figure 5.

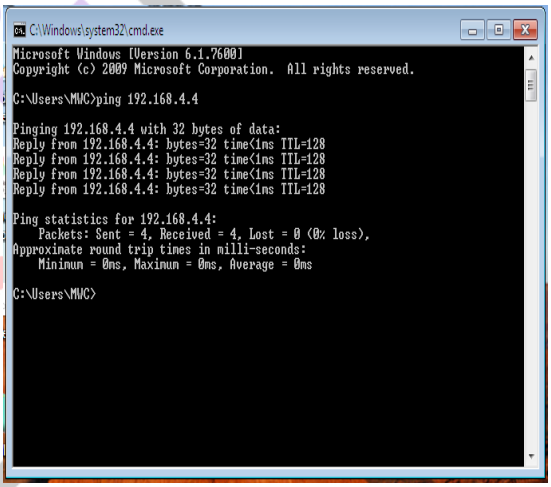

Fig. 5. PING response

Acknowledgment

I would like to thank Mr. RUCHIT.M.S Senior Engineer C-D&E (MWC) Bharat Electronics Limited, for granting permission to publish this paper. I would like to thank Mrs. J Pushpanjali, assistant professor, dept. of ECE, Bangalore institute of technology (BIT) for support in the completion of the project. I would like to thank Dr. Anitha S, professor, dept. of biomedical engineering, ACSCE, Bangalore for extending warm support in the completion of the project.

References

[1] Feng Nai, Sebastián M. Torresand and Robert D. Palmer, "Adaptive Beamspace Processing for Phased-Array Weather Radars," P IEEE Transactions on Geoscience and Remote Sensing (Volume: 54, Issue: 10, Oct. 2016).

[2] Saeed Golestan, Seyyed Yousef Mousazadeh , Josep M. Guerrero, " A Critical Examination of Frequency-Fixed Second-Order Generalized Integrator-Based

Phase-Locked Loops", IEEE Transactions on Power Electronics (Volume: 32, Issue: 9, Sept. 2017).

[3] I´ Gorka Rubio-Cidre, Alejandro Badolato, Luis Úbeda-Medina, Jesús Grajal, Beatriz Mencia-Oliva, "DDS-Based Signal-Generation Architecture Comparison for an Imaging Radar at 300 GHz," IEEE Transactions on Instrumentation and Measurement (Volume: 64, Issue: 11, Nov. 2015).

[4] Gil Kedar, Avi Mendelson, and Israel Cidon, "SPACE: Semi-Partitioned CachE for Energy Efficient, Hard Real-Time Systems," 0018-9340 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

[5] Embedded Processor Hardware data sheet, "Vivado Design Suite Tutorial".

[6] Microblaze MCS Tutorial Jim Duckworth, WPI Data sheet, "Microblaze MCS Tutorial for Xilinx Vivado 2015".

[7] Microchip AN1120 data sheet, "Ethernet theory of operation".