T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MULTI CHANNEL ARINC 429 CONTROLLER FPGA IMPLEMENTATION AND PERFORMANCE ANALYSIS

SERHAT TÜRK

A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE
ELECTRONICS ENGINEERING DEPARTMENT

GEBZE
2023

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MULTI CHANNEL ARINC 429 CONTROLLER FPGA IMPLEMENTATION AND PERFORMANCE ANALYSIS

SERHAT TÜRK

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF SCIENCE

ELECTRONICS ENGINEERING DEPARTMENT

THESIS SUPERVISOR

ASSOC. PROF. DR. SERDAR SÜER ERDEM

GEBZE

2023

T.C.

GEBZE TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK KANALLI ARINC 429 KONTROLCÜSÜ FPGA GERÇEKLEMESİ VE PERFORMANS ANALİZİ

SERHAT TÜRK

YÜKSEK LİSANS TEZİ

ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI

DOÇ. DR. SERDAR SÜER ERDEM

GEBZE

2023

GEBZE TEKNİK ÜNİVERSİTESİ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 25/05/2023 tarih ve 2023/30 sayılı kararıyla oluşturulan jüri tarafından 20/06/2023 tarihinde tez savunma sınavı yapılan Serhat TÜRK'ün tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) : Doç. Dr. Serdar Süer Erdem

ÜYE : Dr. Öğr. Üyesi İhsan Çiçek

ÜYE : Prof. Dr. Erkay Savaş

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...…/…../……… tarih ve ……./…..... sayılı kararı.

İMZA/MÜHÜR

SUMMARY

In this work, Ethernet based multi-channel Arinc-429 controller is implemented and analyzed on Xilinx FPGAs. Communication interface is one of the most important part for an electronic equipment in an avionic vehicle especially commercial (civil) ones. Data buses like Arinc-429 and Arinc-664 are solves this part. A fully functional digital design for Arinc-429 data bus controller is developed. It converts Arinc-429 bus to Ethernet bus. Avionics LRU manufacturers who need to develop a multi-channel Arinc-429 converter can benefit from this design approach and methodology. Xilinx Artix-7 Field Programmable Gate Array (FPGA) device is used for the implementation. Arty A7 development board and HI-3593 ARINC Dual Receiver/Single Transmitter Demo Board was used in the experiment. The design controls 6 receiver channels and 3 transmit channels. Each receiver channel has dedicated buffer to store bus data. The hole design uses 9279 slice LUTs, 7233 slice registers, 4 RAMB36s, 92 RAMB18s and 46 IOB. So, a low cost and portable (generic) implementation is achieved. A dedicated GUI (graphical user interface) is written to visualize the streaming data on the bus.

Key Words: Arinc-429, Ethernet, FPGA, Portable, Bus, Avionic.

ÖZET

Bu çalışmada, Ethernet tabanlı çok kanallı Arinc-429 kontrolcüsü Xilinx FPGA'lar üzerinde gerçeklenmiş ve analiz edilmiştir. Bir hava aracında, özellikle sivil hava araçlarında elektronik bir ekipman için iletişim arayüzü en önemli parçalardan biridir. Arinc-429 ve Arinc-664 gibi veri yolları bu kısmı çözmektedir. Arinc-429 veri yolu kontrolcüsü için tamamen işlevsel bir dijital tasarım geliştirilmiştir. Arinc-429 veri yolunu Ethernet veri yoluna çevirir. Çok kanallı bir Arinc-429 dönüştürücü geliştirmesi gereken Aviyonik LRU üreticileri, bu tasarım yaklaşımı ve metodolojisinden yararlanabilir. Gerçekleme için Xilinx Artix-7 Sahada Programlanabilir Kapı Dizisi (FPGA) cihazı kullanılmıştır. Deneyde Arty A7 geliştirme kartı ve HI-3593 ARINC Çift Alıcı/Tek Verici Demo Kartı kullanıldı. Tasarım 6 alıcı kanalını ve 3 iletim kanalını kontrol eder. Her alıcı kanalı, veri yolu verilerini depolamak için ayrılmış ara belleğe sahiptir. Tüm tasarım 9279 dilim LUT, 7233 slice, 4 RAMB36, 92 RAMB18 ve 46 IOB kullanmaktadır. Böylece düşük maliyetli ve başka cihaza taşınabilir (jenerik) bir gerçekleme elde edilir. Veri yolundaki akan verileri görselleştirmek için özel bir GUI (grafiksel kullanıcı arabirimi) yazılmıştır.

Anahtar Kelimeler: Arınc-429, Ethernet, FPGA, Taşınabilir, Hat, Aviyonik.

ACKNOWLEDGEMENTS

Thanks to my advisor, who supports and helps for my graduate education and academic life, and who leads the formation of this study with his knowledge. I would like to express my sincere thanks to Assoc. Prof. Dr. Serdar Süer ERDEM.

I am grateful to my wife Tuğba Nur TÜRK and my son Akif Toprak TÜRK for their love and support.

TABLE of CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

Abbreviations and Acronyms		**Explanations**
ARINC	:	Aeronautical Radio Incorporated
BRAM	:	Block Random Access Memory
FIFO	:	First-in First-out
FPGA	:	Field Programmable Gate Array
Hz	:	Hertz
LRU	:	Line Replaceble Unit
SBC	:	Single Board Computer
SPI	:	Serial Peripheral Interface
UART	:	Universal Asynchronous Receiver Transmitter

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

ARINC Specification 429 defines the air transport industry standards for the transfer of digital information between avionics system elements [1]. Avionic computers may give information such as flight status via ARINC 429 output busses. Arinc-429 data bus works as one-way point to point communication protocol with two speed option as 100Kbps and 12.5Kbps. A single Arinc-429 frame consists of 32 bits. Standard MCUs and CPUs do not have interface for Arinc-429 bus. There is need for converting Arinc-429 interface to one of common interface like USB, UART and Ethernet. Arinc-429 to UART and Arinc-429 to Ethernet converter were designed and compared with each other in this paper. Ethernet protocol is the most complicated, fastest and most widely used one between mentioned interfaces. Manufacturers like Xilinx presents their own IP as a black box for FPGA customers. This Ethernet IP can be customized according to project needs as permitted by the manufacturer. But IP modules not generic and not portable. So, Ethernet module is designed as portable in this work. 100 Mbps (fast Ethernet) Ethernet MAC is implemented in the design.

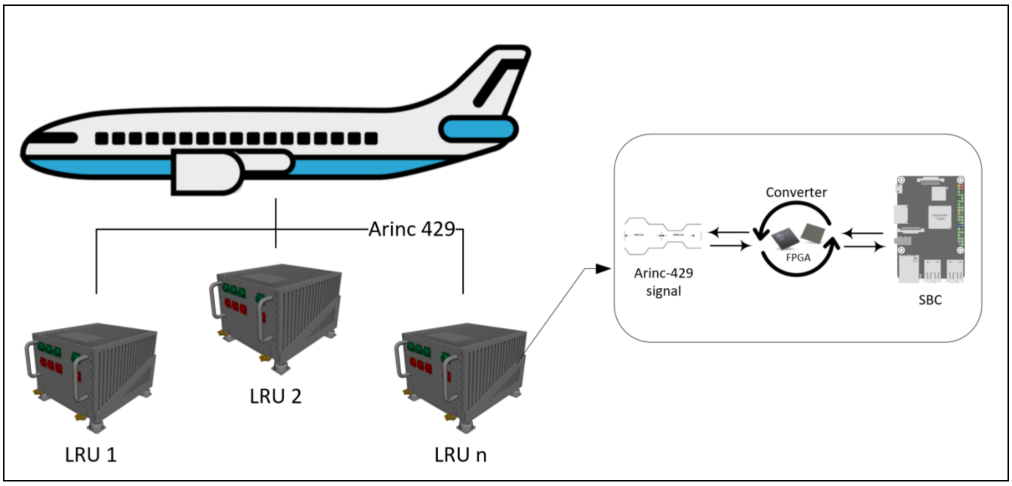

Figure 1.1: Avionics system.

This work presents high performance solution in terms of area, functionality and portability for FPGA implementation of multi-channel Arinc-429 controller. This design has also an optional mode for Arinc-429 communication between In-

Flight Entertainment (IFE) system and A/C interfaces. The organization of the paper is as follows. Section 2 mentions about communication protocols and section 3 explains the previous works about FPGA implementations of Arinc-429. In section 4, FPGA implementation of multi-channel Arinc-429 controller is presented. Results of the proposal and comparison with previous works are shown in section 5. Finally, the conclusion of the proposed design is stated in section 6. In future of work, PCIe based converter may be designed.

2. COMMUNICATION PROTOCOLS

2.1. Arinc-429 Standard

Aeronautical Radio, Inc. (ARINC) was incorporated in 1929 by four fledgling airlines in the United States as a privately-owned company dedicated to serving the communications needs of the air transport industry [13]. ARINC 429 is a two wire, point-to-point data bus that is application specific for commercial and transport aircraft and this standard was published in 1979 and called mark33 (Digital Information Transfer System or DITS) [6]. 1553B communication standard has bus controller, bus terminal and bus monitor but Arinc-429 communication standard has not this kind of elements. Arinc-429 is a simplex communication type. It has not master or slave topology to communicate with multiple nodes interactively on the bus.

Avionics systems are safety critical systems. That means each sub-part of this system also safety critical. A typical ARINC 429 interface should have an unflagged error rate in the magnitude order of 10^{-6} [14]. The enhancing of data transmission reliability means the increase of overall safety, decreasing the probability of each system processing improper inputs and avoiding faults [14].

2.1.1. Physical Layer

A general definition of digital communication is data transfer between transmitter and receiver. An Arinc-429 transmitter may “talk only” to a number of receivers on the bus, up to 20 on one wire pair, with each receiver continually monitoring for its applicable data, but does not acknowledge receipt of the data [3]. Physical layer of communication line is shown in Figure 2.1. Data lines are twisted shielded differential pair. Figure 2.1 gives information about how idle state and 0-1 logic levels works to drive the line. Differential voltage on the pair is ~10V. Each line of differential pair has voltage between +5V and -5V. They work as return-tozero prensible. Electrical characteristic like impedance and maximum ratings are learned from datasheet of the transceiver.

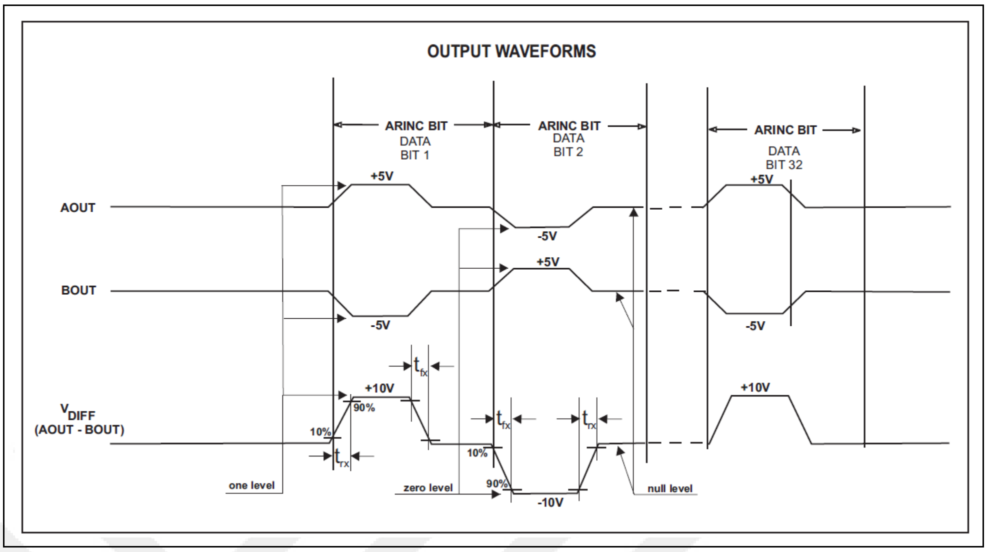

Figure 2.1: Arinc-429 physical layer.

Frame structure and time line is shown in Figure 2.2. The communication is asynchronous and full duplex. Bit rates on the standard are 100Kbps and 12.5Kbps. Figure 2.2 presents MSB-LSB data bit sequence. Frame structure shows the data part location on the frame. So, data is parsed according to this information in the design.

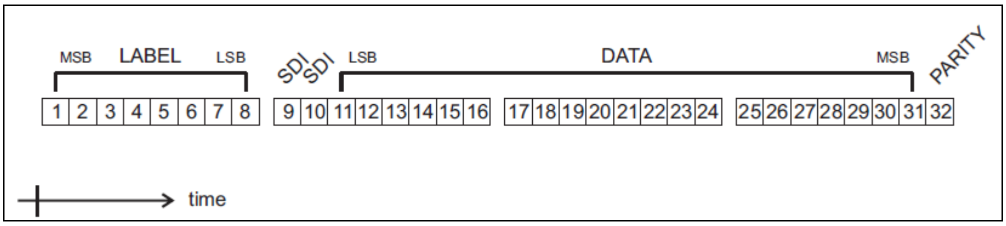

Figure 2.2: Arinc-429 message.

2.1.2. Arinc-429 Message Parcing

Binary coded decimal (BCD) word format is also gives information about data parsing like in Figure 2.3. The first eight bits of each word are assigned to a label function so that the data contained in the word may be identified [1]. A coded label example with a decimal 172 value is shown in Figure 2.4.

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
P	**SSM**		**Digit 1**			**Digit 2**				**Digit 3**				**Digit 4**				**Digit 5**				**SDI**		**Label**							

Figure 2.3: Arinc-429 BCD word format.

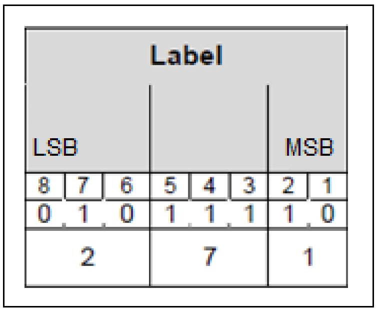

Figure 2.4: BCD label example.

Bit-9 and bit-10 is used for source/destination identification (SDI). SDI function may find application when specific words need to be directed to a specific system of a multi-system installation or when the source system of a multi-system installation needs to be recognizable from the word content [1]. The source/destination identifier encoding is as in Table 2.1.

Table 2.1: SDI encoding.

Bit Number		**System**
10	**9**	
0	0	1
0	1	2
1	0	3
1	1	4

From bit-11 to bit-29 is used for data part. For example; it consists of functional test results in discrete data format. Data formats are explained in section 2.1.3.

Bit-30 and bit-31 represents sign/status matrix (SSM) data. They indicate sign, directions and status information as shown in Table 2.2. When is appears in a word identified by its label as a system output, the "functional test" code should be interpreted as advice that the data in the word results from the execution of a functional test [1].

Table 2.2: SSM encoding.

Bit Number		Description
31	30	
0	0	Plus, North, East, Right, To
0	1	No Computed Data
1	0	Functional Test
1	1	Minus, South, West, Left, From

Sign/status matrix (SSM) and data parts give information about the label identifier. 32-th bit is used as parity bit to check data integrity. When any error occurred on the transmission etc. is detected, it may be ignored optionally. The transceiver IC (integrated circuit) has configuration for automatic parity control. So, data buffer usage may be optimized by setting parity control.

2.1.3. Data Formats

The data type on Arinc-429 bus may be binary coded decimal (BCD), two's complement binary notation (BNR) and discrete data [12].

In BCD, each decimal value represented with 4-bit grouping in data part and with 3-bit grouping in label part. If length of data part is not multiple of four, most significant bits (MSB) left as smaller than 4-bit. Similar lenght condition is applied for label part for 3-bit grouping.

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
1	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1
6			9				4				2				1			

Figure 2.5: BCD data part.

In BNR, data part represents a 19-bit (29 to 11) signed number. Bit 29 is the sign Bit and bit 28 is the most significant bit of the data field, which represents one half of the maximum value of the parameter being defined [12]. As an example; BNR translation of data in Figure 2.5 is shown in Figure 2.6.

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
1	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1
SIGN	MSB																	LSB
-93151																		

Figure 2.6: BNR data part.

In discrete, each bit of the data part has uniq meaning. For example; Communication data link status may be tracked over each bit such as bit-11 Link Available, bit-12 SATCOM OK, bit-13 VHF Link etc.

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
1	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1

Bit-11	Link Available	1
Bit-12	SATCOM OK	0
Bit-13	VHF Link	0
...		

Figure 2.7: Discrete data part.

2.2. SPI Protocol

Serial Peripheral Interface (SPI) is a synchronous and full-duplex communication protocol. SPI physical layer has four single ended wires on Transistor-Transistor Logic (TTL) 3.3V level. Each SPI Frame consists of eight bits, Figure 2.8. SPI devices talk to each other in master-slave relationship, Figure 2.9.

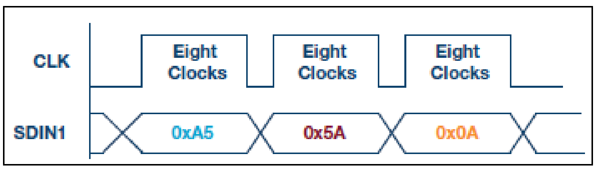

Figure 2.8: SPI frame.

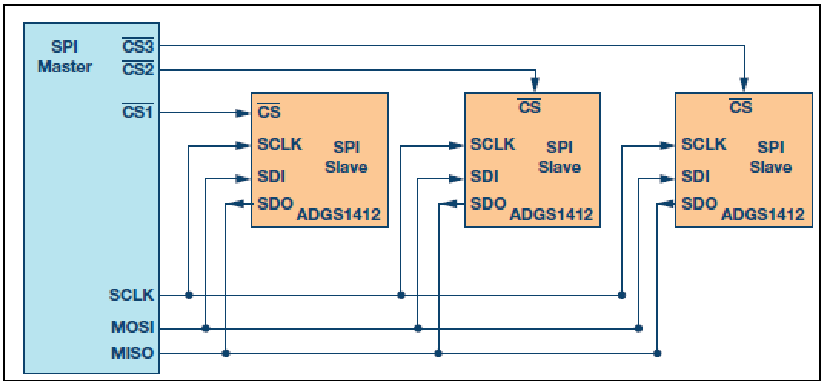

Figure 2.9: Master-Slave relationship.

In regular mode of SPI, an individual chip select for each slave is required from the master and once the chip select signal is enabled (pulled low) by the master, the clock and data on the MOSI/MISO lines are available for the selected slave [4]. CPOL (clock polarity) and CPHA (clock phase) are changes according to application. CS pin works as active low. MISO pin shall be High-Z on idle state. Data is sampled on rising edge and shifted on falling edge [4].

2.3. UART Protocol

The Universal Asynchronous Receiver/Transmitter (UART) performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU [5]. UART data bus works as one-way point to point communication protocol. It is full duplex asynchronous serial bus. The bus commonly supports 11 different data rate between 600 Baud – 115200 Baud (600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200). A single UART frame consists of 8 bits.

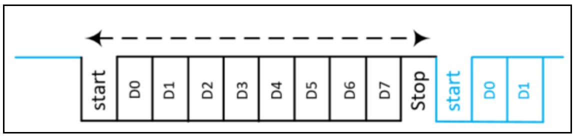

Figure 2.10: UART frame.

2.3.1. RS-232

UART line voltage is on TTL and it should be translated for media side physical layer voltage as shown in Figure 2.11. TTL signals are inversed via the transceiver and sent as inversed on the bus. RS-232 standard defines the signals which connect a DTE (data terminal equipment) and a DCE (data circuit-terminating equipment) in terms of the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors [6].

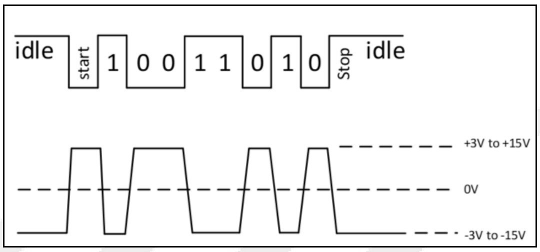

Figure 2.11: RS-232 physical layer.

2.4. Ethernet Protocol

Ethernet is used to build networks from the smallest to the largest, and from the simplest to the most complex: it connects home computers and other household devices, but it also connects the building networks that support servers and wired desktop computers, as well as the wireless access points that support smartphones, laptops, and tablets [7]. Ethernet interface supports from 10 Mbps to 100 Gbps speed. Ethernet frame has not a standard length. Frame length varies according to type of protocol such as UDP, DHCP, ICMP etc.

The Ethernet system includes four building blocks that, when combined, make a working Ethernet [7]:

• The frame, a standardized set of bits used to carry data over the system.

• The Media Access Control protocol, consisting of a set of rules embedded in each Ethernet interface that allow Ethernet stations to access the Ethernet channel, in either half- or full-duplex mode.

• The signaling components, standardized electronic devices that send and receive signals over an Ethernet channel.

• The physical medium, the cables and other hardware used to carry the digital Ethernet signals between computers attached to the network.

64 bits	48 bits	48 bits	16 bits	46 to 1500 bytes	32 bits
Preamble	Destination Address	Source Address	Type/ Length	Data	Frame Check Sequence (CRC)

Figure 2.12: Ethernet frame.

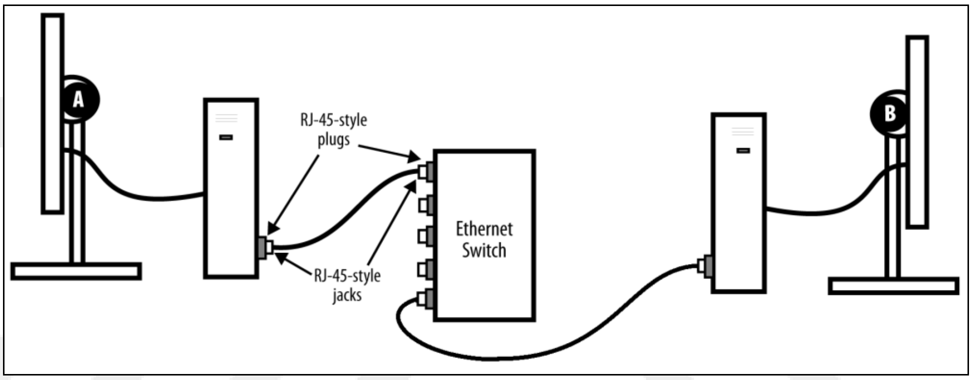

Figure 2.13: Signaling components.

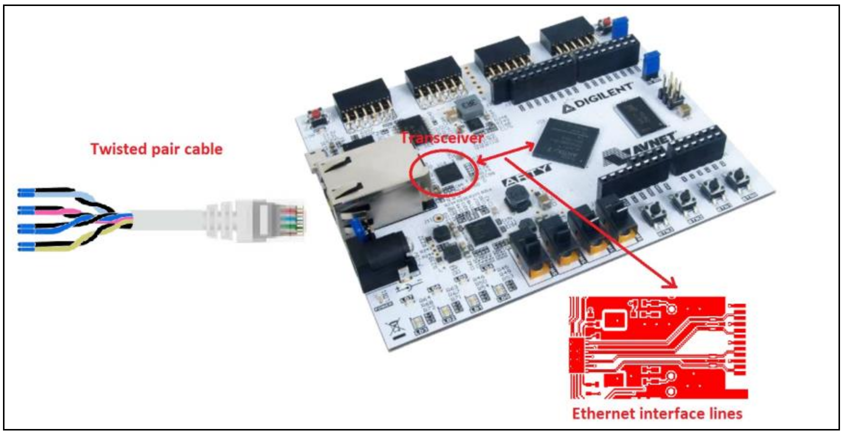

Figure 2.14: Physical medium.

Data that is being sent between computers is carried in the data field of the Ethernet frame and is structured according to higher-level network protocols [7]. Media Access Controller(MAC) talks via MII (Media Independent Interface), RMII

(Reduced MII), GMII (Gigabit MII) etc. An Ethernet interface whose 48-bit MAC address matches the unicast and any multicast destination address in the frame will continue to read the entire frame and deliver it to the networking software running on that computer [7]. The used network protocols in the design are Dynamic Host Configuration Protocol (DHCP), Address Resolution Protocol (ARP) and User Datagram Protocol (UDP). High-level network protocols have their own system of addresses, such as the 32-bit address used by IPv4, which is currently the most widely used version of the Internet Protocol (IP) [7]. UDP protocol is widely used in the design. Because, the communication between host computer and FPGA is based on UDP.

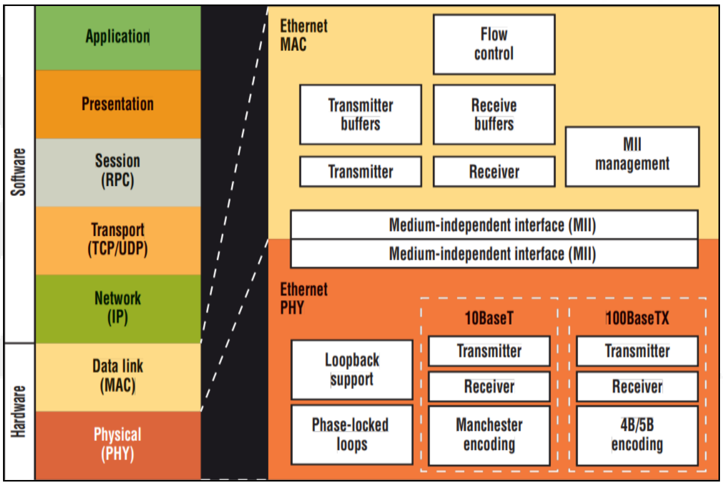

Figure 2.15: 10/100 Mbps Ethernet PHY & MAC demonstration.

2.4.1. UDP

UDP is at position which is located on transport layer of TCP/IP section in network communication protocols map. It has not a connection establishment, thus it is more simple and faster than the other protocols. A sender sends a request and receiver of this request gets this message. The receiver processes the request and sends response to the sender. The protocol just uses source identification by checking IP address and port number.

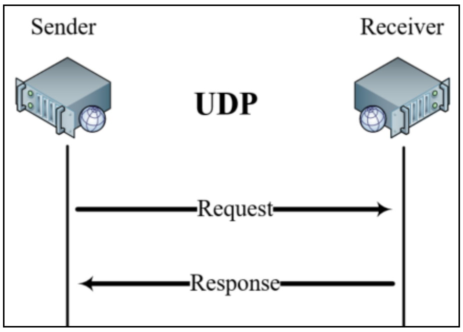

Figure 2.16: UDP.

The UDP message structure is mainly composed of two parts, namely the UDP header and the UDP data area [15]. UDP header has four type data. Source port number is port number of the sender. Destination port number is port number of the receiver. Total length is sum of payload (data area) length and header length. Checksum is checksum of payload and header. It is calculated by a method which is defined in [RFC 768](). It is used for data integrity and error checking. The checksum calculation is optional in the UDP protocol and if the source host chooses not to perform the checksum calculation, this field can be simply filled with 0 [15].

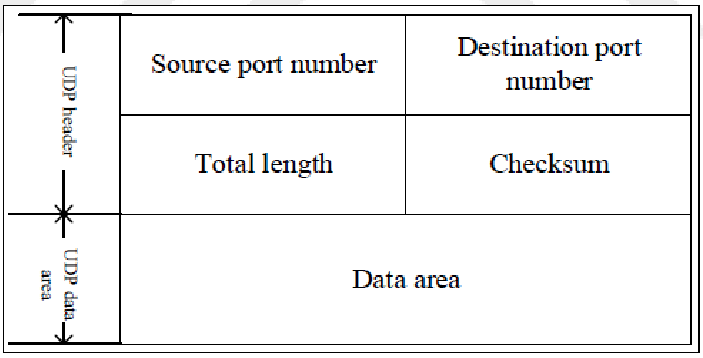

Figure 2.17: UDP message structure.

3. PREVIOUS WORKS

After Arinc-429 standard has been published, many designs were made and issued. These designs were focused on conversion of protocol for custom or generic usage. In [6], a device that enables an avionic equipment to communicate with the ARINC 429 bus over the standard RS-232 port was developed. This design tested only 9600 baud rate and it supports single transmit and single receive channel only. In [9], 16 channel transmitter controller was designed. It has only transmitter part for Arinc-429 communication. It does not serve as converter but it serves as part of a transceiver. In [10], researchers designed and implemented ARINC429 interface module based on PC/104 using XILINX company Spartan3A series FPGA devices, XC3S200A. They claim that this module supports various ways of working, less components and high reliability. This design also supports single transmit and two receive channel only, but it is faster than the design in [6]. In [11], the design and implementation of Ethernet based analyzer for analyzing dual speed avionics serial protocol was presented. It supports multichannel and it uses Ethernet protocol for host communication. But, The TEMAC Intellectual Property (IP) core from Xilinx was used in the design to perform the operation of Ethernet. Using black box IP in the design limits the capability of design in aspect of portability. Also, there is no dedicated GUI for visualization in the work.

4. FPGA IMPLEMENTATION

At the beginning, the design is started as RS-232 based multi-channel Arinc-429 converter. RS-232 based design was implemented but it was seen that it is not fast enough. So, a faster protocol was searched and Ethernet was found as fast enough. RS-232 based design was left as optional design. The final design continued over Ethernet based one. FPGA implementation scheme is mainly divided into three representative blocks: Selected ICs, Ethernet Controller block, Arinc-429 Controller block, and Algorithm Executive block. A top module serves as a junction point and connects those modules to each other and outside interfaces. Ethernet Converter module solves Data Link Layer. Packet Controller and Packet Sender modules solve Network Layer. DHCP Packets, Packet Controller and Packet Sender modules solve Transport Layer. Command controller solves Application Layer. The whole structure diagram is shown in Figure 4.1.

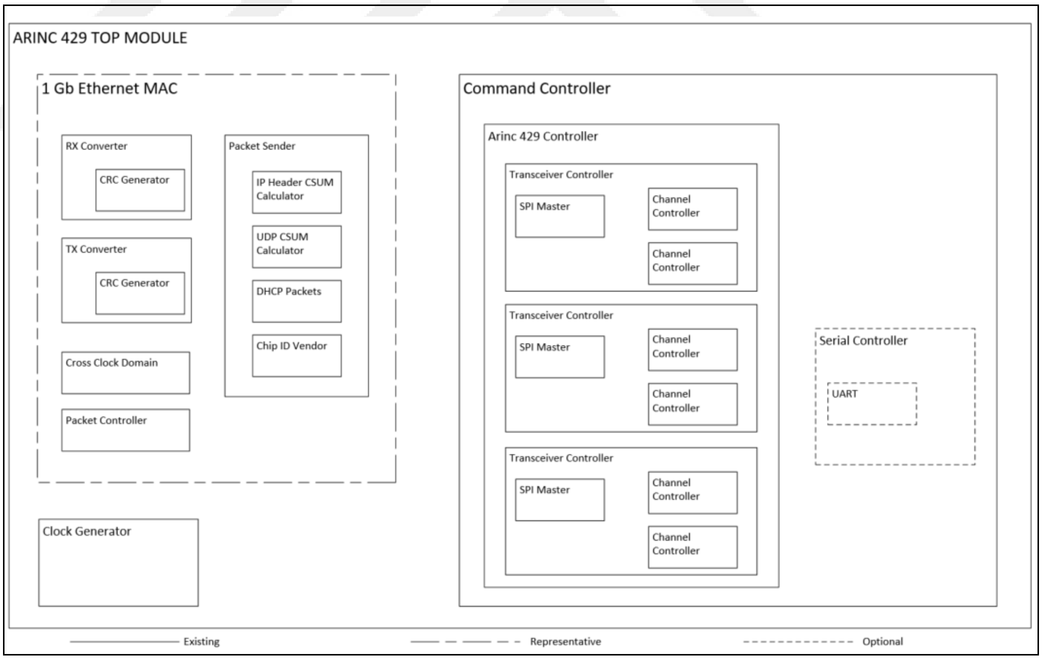

Figure 4.1: Block diagram.

Xilinx Vivado v2020.2 design tool is used for development. In “Project Manager” section of tool project and sources can be reviewed as hierarchical representation or as library files. Source files can be classified by creating sets. Two sets were created for Artix-7 and Kintex-7 implementations. Design runs were also created by relating them with specified sets. Project view on the tool is as below.

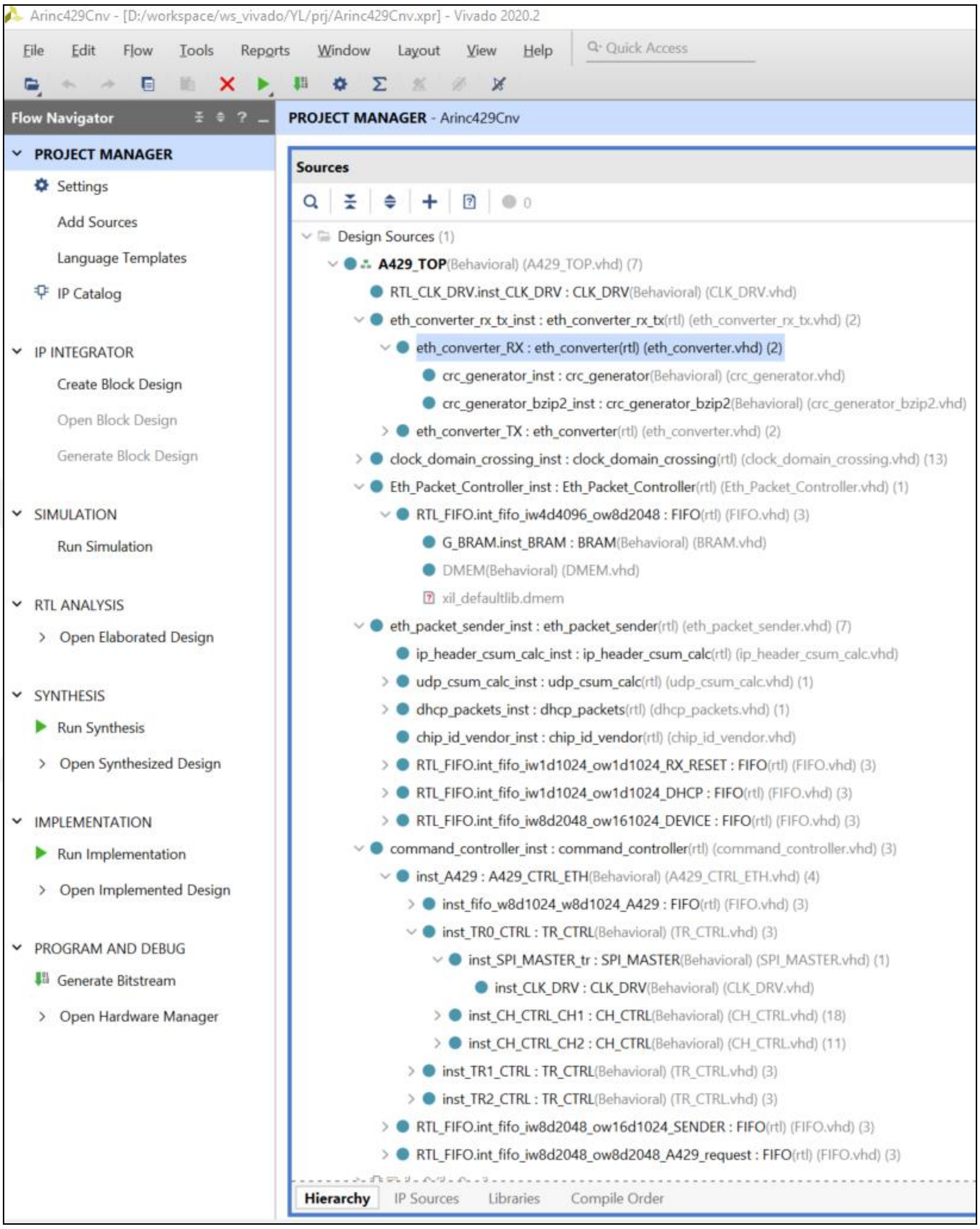

Figure 4.2: Project hierarchy.

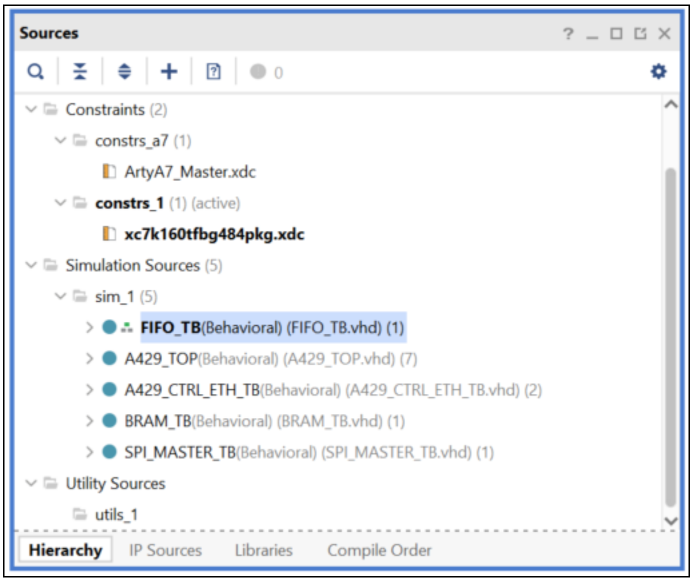

Figure 4.3: Constraint & simulation sources.

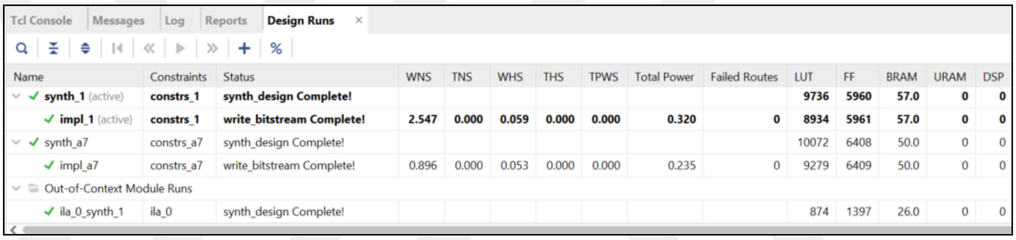

Figure 4.4: Design runs.

4.1. Selected ICs

The design is implemented on a 7 series Artix-7 FPGA chip. Vivado Default Strategies are used for both synthesis and implementation steps. Arty A7 ready-to-use development platform which contains XC7A35TICSG324-1L chip was used as test board. It has 33280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops) and 1800 Kbits of fast block RAM. HI-3593 transceiver is used for Arinc-429 interface physical layer conversion. It has two receive channel and one transmit channel. It has internal FIFOs and SPI interface. FPGA communicates with Arinc-429 transceiver via SPI interface. Main clock (MRCC) of logical blocks is driven by 100Mhz external oscillator.

DP83848J Ethernet PHY is used for Ethernet interface physical layer conversion. FPGA communicates with Ethernet PHY via 100Mbps MII mode. MAC and Logical link control (LLC) part of Data link layer is implemented in sub-modules of Ethernet Controller Module. MAC address is taken from Arty A7 Board. The manufacturer of Arty A7 board has assigned a MAC address. It is written on a sticker on RJ45 connector.

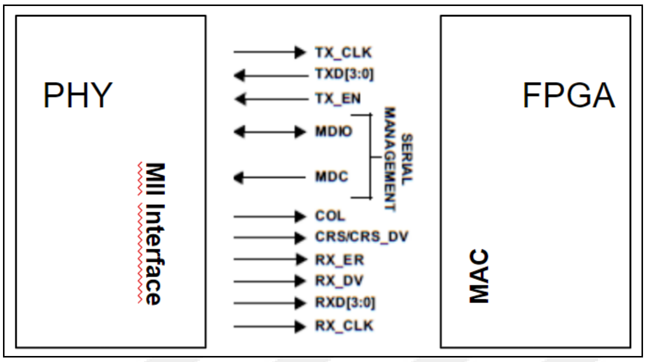

Figure 4.5: MII mode.

4.2. Ethernet Controller Block

Ethernet controller block consist of Ethernet protocol related modules; Ethernet Converter, RX Converter, CRC Generator, Cross Clock Domain, Packet Controller, Packet Sender, IP Header CSUM Calculator, UDP CSUM Calculator, DHCP Packets. Those modules implement Ethernet layers and operates the high-level network protocols such as DCHP, ARP and UDP.

Ethernet controller block has an initial IP address 0.0.0.0 on start-up. After start-up, the block broadcasts discover packets which consists of only FPGA MAC address until it gets an offer packet. The offer packet is sent as broadcast packet by an active DHCP server on the network. It consists offered client IP address. FPGA accepts the offered IP address and sends request packet as broadcast to get this IP address. DCHP server sends ACK packet as broadcast. FPGA sends broadcast ARP packet to ensure no one using the given IP address. After confirmation of the IP address is not used by any one on the network, FPGA sends broadcast ARP packet to announce that it uses the given IP address from now on. As last step of IP handshaking, FPGA replaces initial IP address with offered one when FPGA gets ACK packet it.

DHCP Network Protocol support can be configured as inactive and a static IP address can be used on the design. Bit stream code should be regenerated when this configuration changed. Data payload is transferred via UDP packets between FPGA and related client such as a computer etc. UDP data payload is structured as shown in Figure 4.7. Data section of it is also have a structure in aspect of Arinc-429 transceiver operation. It will be explained in section 4.3.

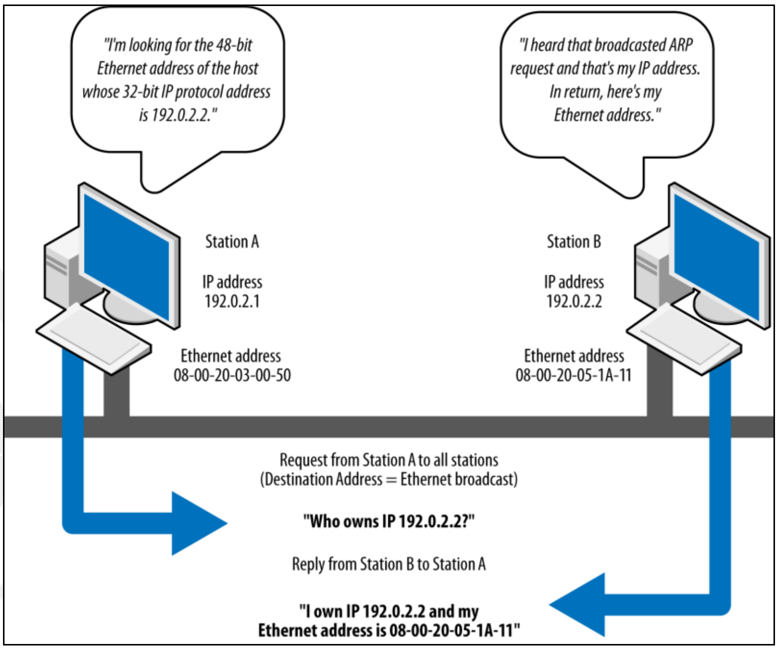

Figure 4.6: Using ARP over an Ethernet.

Start Char <1 byte>	Com Type <1 byte>	Data Type <1 byte>	Data <1 to n byte>	EOF <1 byte>

Figure 4.7: UDP data payload.

4.3. Arinc-429 Controller Block

In this block, Arinc-429 data packet is created. Its body form is shown in Figure 4.9. SPI Master, Channel Controller and Transceiver Controller modules are designed to source this body. The opcode list is taken from the datasheet of transceiver. SPI interface is written with VHDL concerning source & speed optimization and generic structure. It works on 1Mhz frequency. The design should be fast as possible as because it has multichannel capability. So, separate SPI controllers are used for each transceiver.

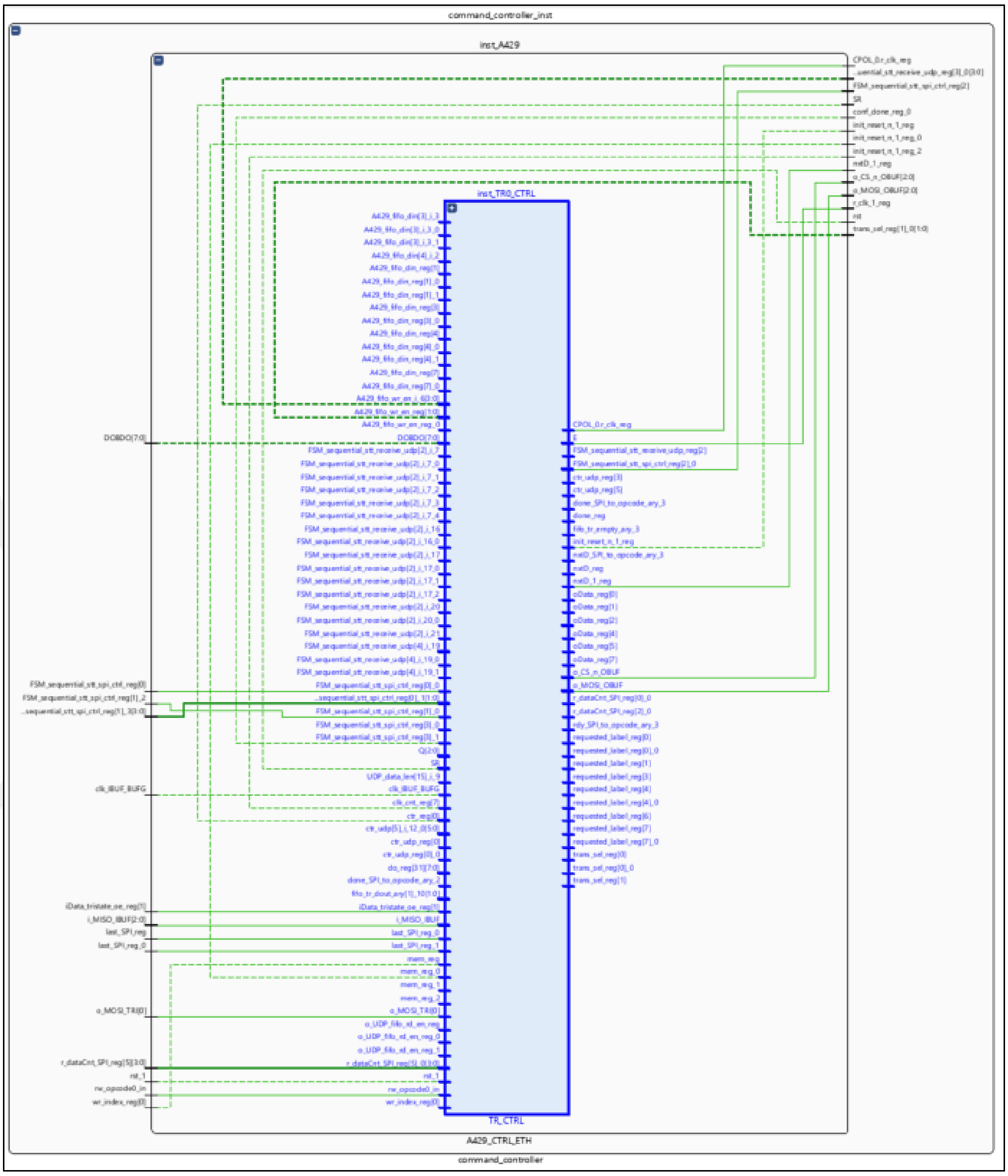

Figure 4.8: Schematic view of transceiver controller block.

Three transceivers are used in the design. Requested transceiver is selected by setting MSB 4-bit of instruction byte. A receive channel controller is written with VHDL to control each channel of each transceiver. Each channel has own FIFOs on both FPGA side and transceiver side. On FPGA side, FIFOs are specialized according to label values. Also, each Arinc-429 transceiver has features such as programmable label recognition for 256 labels. Label recognition configuration is done by using opcode mechanism. Besides, wider and more specialized (for IFE system) label recognition is implemented on FPGA side. Label filtering on FPGA side is done by using instruction mechanism. A special instruction is added to read a specific label because IFE system mostly operates in aspect of reading channel and rarely transmitting. So, receive channels are controlled over instruction mechanism. Transmit channels are controlled over opcode mechanism. Independent data rate can be selected for transmit and receive by configuring the transceiver via running related opcode.

Instruction <1 byte>	Opcode <0-1 byte>	Value <0 to m byte>

Figure 4.9: Arinc-429 controller packet body.

4.4. Memory Implementation

All message data on Arinc-429 bus and Ethernet bus was stored in memory elements. FIFO (first-in first-out) structure was used mostly. FIFO structure is based on BRAM. FPGA architecture has memory elements such as Block RAM (BRAM) and Distributed RAM (LUT RAM) for large data size. Registers and LUTs are also used as memory element but they are used for smaller data size. Actually Distributed RAM is also LUT based memory element but it is more capable for addressing memory data. It is faster than BRAM in aspect of reaching data. It has no addressing latency but DRAM amount is more limited. So generally BRAM was used in the design. BRAM or FIFO were not used as IP block but they were used by inferring method instead. BRAM code is shown in Figure 4.10.

```
----------------------------------------------------------------------------------
-- Engineer:  STURK
--
-- Module Name: BRAM
-- Description:
-- Block RAM module for memory
-- Dual port, independent clock
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--    modul read first olarak calismaktadir.
--    rst sinyaline gerek yoksa sabit set edilmelidir.
```

```
--    rst sinyalinin sabit set edilmesi LUT ve register kullanimini sifirlamaktadir.
--    clock'lar birbirinden bagimsiz yapilsa da bram'a donusuyor
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.math_real.all;

entity BRAM is
  Generic(
    BRAM_WIDTH : integer := 32;
    BRAM_DEPTH : integer := 1024
  );
  Port (
    rst       : in     std_logic;
    i_wr_clk  : in     std_logic;
    i_rd_clk  : in     std_logic;
    i_we      : in     std_logic;
    i_re      : in     std_logic;
    i_di      : in     std_logic_vector(BRAM_WIDTH-1 downto 0);
    i_addr_wr : in     std_logic_vector(integer(ceil(log2(real(BRAM_DEPTH))))-1 downto 0);
    i_addr_rd : in     std_logic_vector(integer(ceil(log2(real(BRAM_DEPTH))))-1 downto 0);
    o_do      : out    std_logic_vector(BRAM_WIDTH-1 downto 0)
  );
end BRAM;

architecture Rtl of BRAM is
  constant BRAM_DEPTH_SZ : integer := integer(ceil(log2(real(BRAM_DEPTH))));
  constant WEA : std_logic := '1';

  type VECTOR_ARRAY is array (integer range <>) of std_logic_vector(BRAM_WIDTH-1 downto 0);

  signal mem : VECTOR_ARRAY(0 to BRAM_DEPTH-1);
  signal addr_rd : unsigned(BRAM_DEPTH_SZ-1 downto 0) := (others => '0');
  signal addr_wr : unsigned(BRAM_DEPTH_SZ-1 downto 0) := (others => '0');
  signal ctr : unsigned(BRAM_DEPTH_SZ-1 downto 0) := (others => '0');
  signal di : std_logic_vector(BRAM_WIDTH-1 downto 0);
  signal do : std_logic_vector(BRAM_WIDTH-1 downto 0);
  signal we : std_logic;
begin

  -- 'read first'
  process (i_wr_clk)
  begin
    if (rising_edge(i_wr_clk)) then
      if we = '1' then
        mem(to_integer(unsigned(addr_wr))) <= di;
      end if;
    end if;
  end process;
  process (i_rd_clk)
  begin
    if (rising_edge(i_rd_clk)) then
      if i_re = '1' then
        do <= mem(to_integer(unsigned(addr_rd)));
      end if;
    end if;
  end process;

  we <= i_we when rst = '0' else '1';
  di <= i_di when rst = '0' else (others => '0');
  o_do <= do when rst = '0' else (others => '0');
  addr_wr <= unsigned(i_addr_wr) when rst = '0' else unsigned(ctr);
  addr_rd <= unsigned(i_addr_rd);
```

```
    process (i_wr_clk)
    begin
      if (rising_edge(i_wr_clk)) then
        if rst = '1' then
          ctr <= ctr + 1;
        else
          ctr <= (others => '0');
        end if;
      end if;
    end process;

  end Rtl;
```

Figure 4.10: BRAM code.

4.5. Algorithm Executive

Algorithm Executive module drives LED indicators of FIFOs state in aspect of being empty or not empty. It listens requests coming from Ethernet interface and manages them. Configuration should be set by host computer before transmit and read requests because FPGA does not know channel speed etc. at the start up. After configuration is set, FPGA goes to read-in-loop state and reads all received Arinc-429 messages from each receive channel. Internal FIFOs are filled with incoming messages from Arinc-429 channels. When an opcode operation is requested by host computer, FPGA exits from read-in-loop state and executes opcode. The flow chart of Arinc-429 algorithm is shown in Figure 4.11.

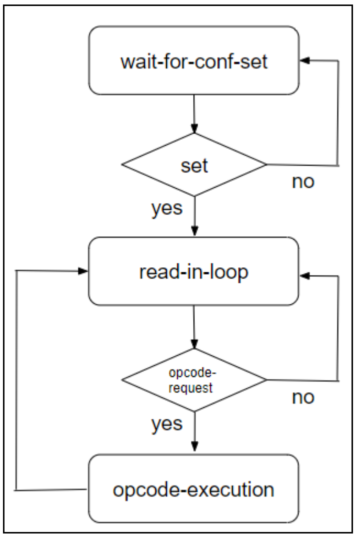

Figure 4.11: Algorithm flow diagram.

5. THE EXPERIMENT RESULTS AND ANALYSIS

This design was implemented in VHDL and tested in Xilinx Vivado v2020.2. Xilinx Aritx-7 XC7A35T was selected as target device. In this section, the experimental results were analyzed in three aspects: resource usage, running performance and accuracy of conversion. In Figure 5.1, simulation result of Arinc-429 Controller module design was illustrated. Arinc-429 Controller testbench was simulated using Vivado Simulator. The module decodes incoming UDP packet and executes requested instruction. Total operation time of module for reading a message and executing an opcode takes approximately 16,5 us. Reading a message takes ~0,4 us while executing an opcode ~16.1 us. Total operation time of module varies according to instructions.

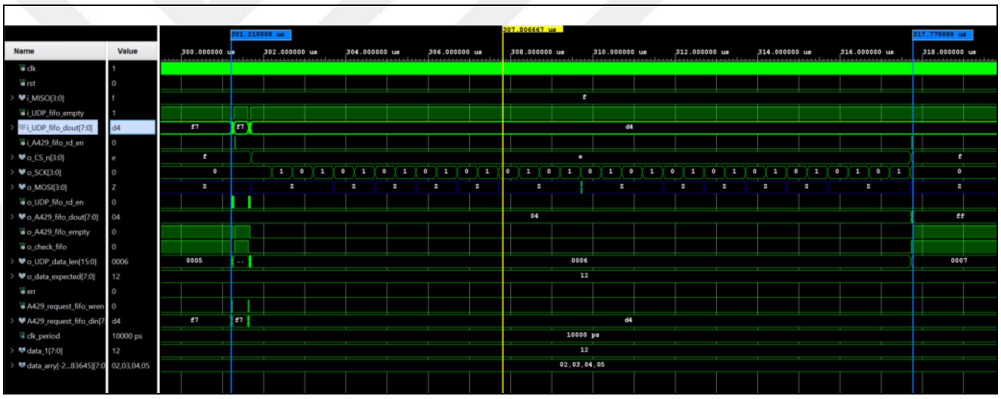

Figure 5.1: Arinc-429 controller simulation.

Purpose of tests which is conducted via Vivado simulator is to observe the signals in the designed module. So, whether signals are behaving as expected or not can be seen. Debugging with Vivado simulator on each step and each key module makes easier hole design integration. Because, if you eliminate mistakes on modules which have lower hierarchic level, possible mistakes on the modules which have higher hierarchic level can be traced and eliminated easier. Otherwise, debugging gets more complex. If an error observed on target tests, debugging is more beyond complex. Because, a logic analyzer is used in such cases and it has limits to track signals such as memory and speed. So, simulation and testbenches is critical at module designing steps.

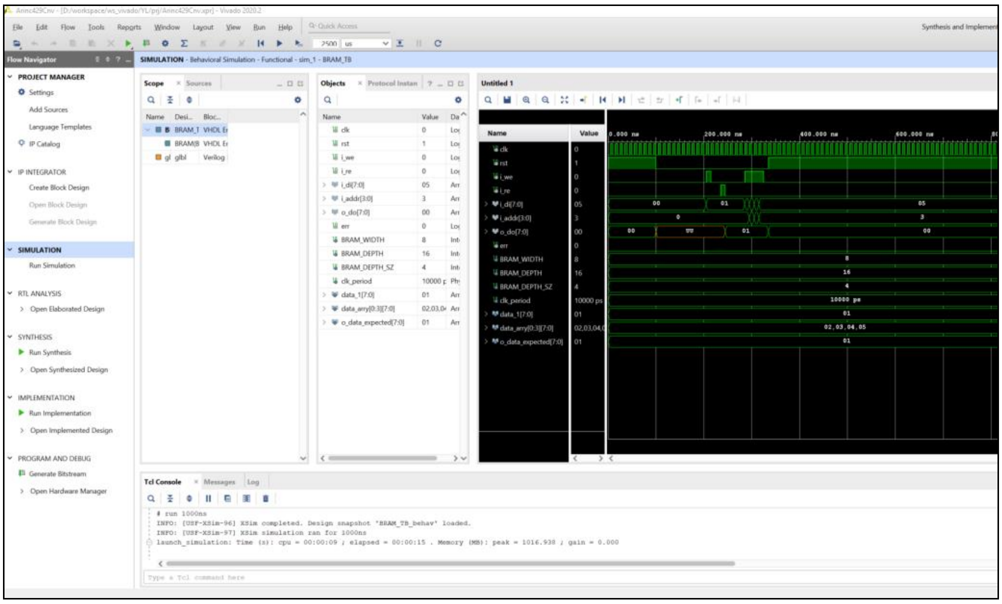

Figure 5.2: BRAM simulation.

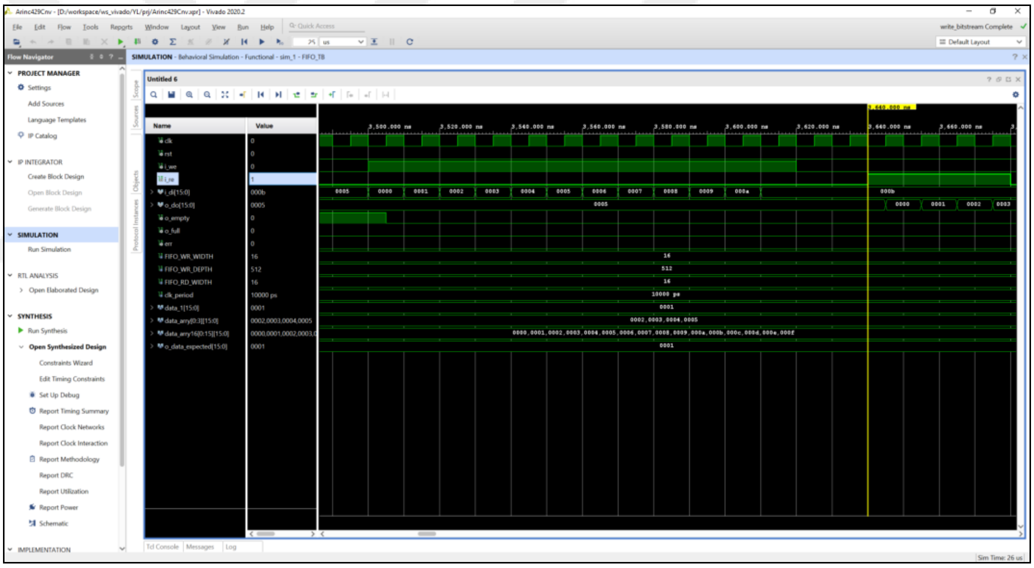

Figure 5.3: FIFO simulation.

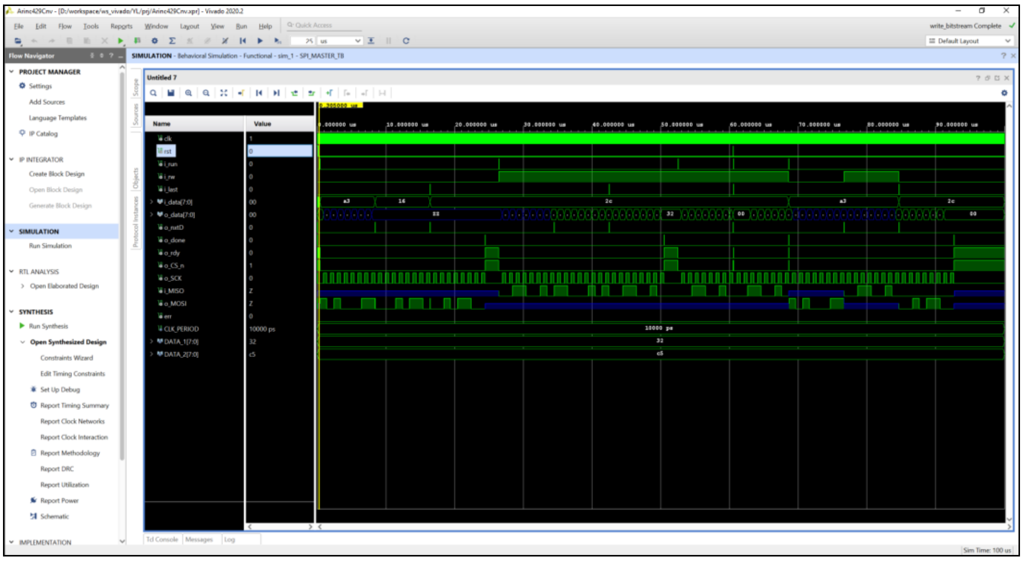

Figure 5.4: SPI master simulation.

5.1. Resource Usage

The implemented system resource usage is shown in Table 5.1. There is a hierarchical listing on the table. It means that A429_TOP module usage statistics are sum of its sub-module statistics. Note that the Transceiver Controller modules consume more resource than most of other modules, because of the repetition usage of the logical elements in subsystems. Post-implementation utilization rates of XC7A35TICSG324-1L chip is also shown in Figure 5.5.

Table 5.1: Resource usage of the system.

Module Name	Slice LUTs	Slice Registers	F7 Muxes	F8 Muxes	Slice	LUT as Logic	LUT as Memory	Block RAM Tile	Bonded IOB	BUFGCTRL
A429_TOP	9275	7233	130	33	3073	8572	703	50	46	3
eth_packet_sender_inst (eth_packet_sender)	1965	1669	126	33	739	1611	354	0.5	0	0
udp_csum_calc_inst (udp_csum_calc)	677	191	49	16	221	325	352	0	0	0
ip_header_csum_calc_inst (ip_header_csum_calc)	123	170	1	0	62	123	0	0	0	0
dhcp_packets_inst (dhcp_packets)	247	137	0	0	80	247	0	0.5	0	0
chip_id_vendor_inst (chip_id_vendor)	9	10	0	0	3	9	0	0	0	0
eth_converter_rx_tx_inst (eth_converter_rx_tx)	165	170	0	0	70	160	5	0	0	0
eth_converter_TX (eth_converter_1)	128	149	0	0	46	123	5	0	0	0
crc_generator_bzip2_inst (crc_generator_bzip2)	82	65	0	0	26	82	0	0	0	0
eth_converter_RX (eth_converter)	37	21	0	0	24	37	0	0	0	0
command_controller_inst (command_controller)	6367	4172	1	0	2025	6031	336	43	0	0
inst_A429 (A429_CTRL_ETH)	6107	4043	1	0	1973	5771	336	42	0	0
inst_TR2_CTRL (TR_CTRL__parameterized1)	1628	1027	1	0	553	1628	0	14	0	0
inst_SPI_MASTER_tr (SPI_MASTER)	98	40	0	0	50	98	0	0	0	0
inst_CLK_DRV (CLK_DRV__parameterized1)	17	10	0	0	8	17	0	0	0	0
inst_CH_CTRL_CH2 (CH_CTRL__parameterized4)	732	456	0	0	256	732	0	7	0	0
inst_CH_CTRL_CH1 (CH_CTRL__parameterized3)	762	456	1	0	267	762	0	7	0	0
inst_TR1_CTRL (TR_CTRL__parameterized0)	2305	1411	0	0	808	2305	0	20	0	0
inst_SPI_MASTER_tr (SPI_MASTER_56)	86	40	0	0	46	86	0	0	0	0
inst_CLK_DRV (CLK_DRV__parameterized1_57)	17	10	0	0	8	17	0	0	0	0
inst_CH_CTRL_CH2 (CH_CTRL__parameterized2)	989	584	0	0	355	989	0	9	0	0
inst_CH_CTRL_CH1 (CH_CTRL__parameterized1)	1195	712	0	0	453	1195	0	11	0	0
inst_TR0_CTRL (TR_CTRL)	1989	1507	0	0	607	1653	336	7.5	0	0
inst_SPI_MASTER_tr (SPI_MASTER_138)	84	40	0	0	40	84	0	0	0	0
inst_CLK_DRV (CLK_DRV__parameterized1_139)	17	10	0	0	8	17	0	0	0	0
inst_CH_CTRL_CH2 (CH_CTRL__parameterized0)	579	360	0	0	188	579	0	5.5	0	0
inst_CH_CTRL_CH1 (CH_CTRL)	1293	1032	0	0	407	957	336	2	0	0
clock_domain_crossing_inst (clock_domain_crossing)	282	171	0	0	100	282	0	6	0	0
RTL_CLK_DRV.inst_CLK_DRV (CLK_DRV)	3	3	0	0	3	3	0	0	0	0
Eth_Packet_Controller_inst (Eth_Packet_Controller)	476	978	3	0	332	468	8	0.5	0	0

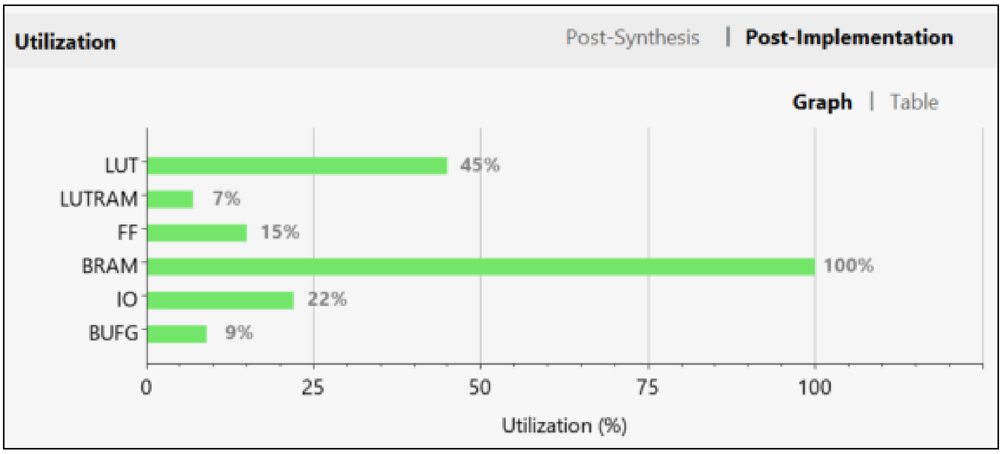

Figure 5.5: Utilization rates for Artix-7.

The design was also implemented for a Kintex-7 FPGA which is optimized family for highest system performance and capacity. XC7K160TFBG484-1 chip was selected. So, the implementation could be compared in aspect of being implemented on different chip. Resource usage of the system for XC7K160TFBG484-1 chip is 8934 slice LUTs, 6785 slice registers, 4 RAMB36s, 92 RAMB18s and 46 IOB. It is also shown that the design can be easily transferred to another device. Kintex-7 implementation used slightly less resource but the main point is BRAM usage was not saturated at Kintex-7 implementation unlike Artix-7 as expected.

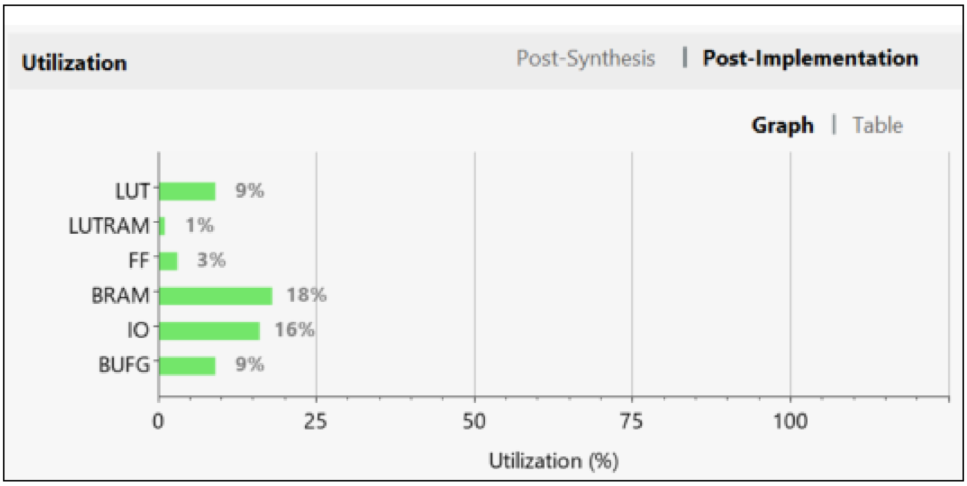

Figure 5.6: Utilization rates for Kintex-7.

5.2. Running Performance

Normally, 115200 baud rate and faster rates on host side (converted interface) is enough for Arinc-429 because Arinc-429 has maximum 100KHz bus speed. But, when multi-channel communication is needed faster rates are needed on host side. 100Mbps Ethernet supports faster rates needed to catch all messages flowing on the bus. The proposed design with 100Mbps data rate is faster from the design in [6]. It is more generic and easy to connect any host via Ethernet interface comparing to the design in [10]. In aspect of using IP core, transferring the presented implementation to an another chip is easier than the design in [11].

1MHz SPI and 100KHz Arinc-429 signals was debugged using external Logic Analyzer device as shown in Figure 5.7. Integrated Logic Analyzer(ILA) tool of Vivado is also used for another internal signal debug operations.

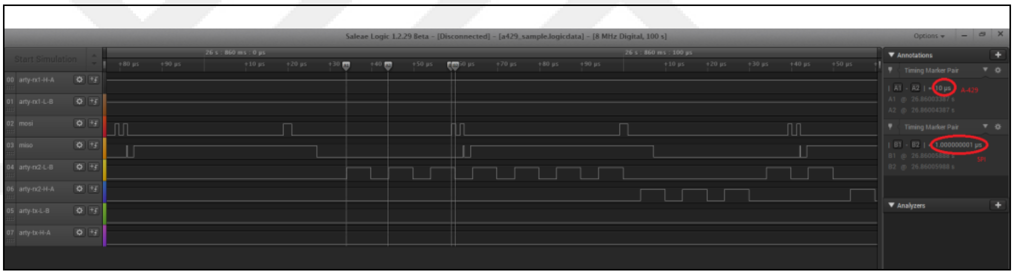

Figure 5.7: Debugging.

The design test setup was watched and recorded via Wireshark network protocol analyzer application. DHCP server application was used for DHCP server need. Some of DCHP packets and UDP packets on the network is shown in figures below. 192.168.1.1 is IP address of host computer. DHCP server assigned 192.168.1.2 IP address to FPGA board.

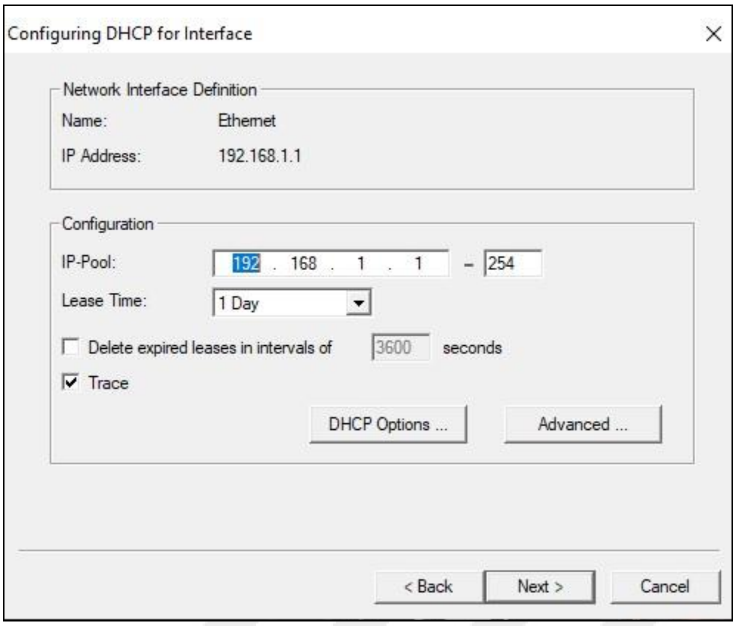

Figure 5.8: DHCP server application.

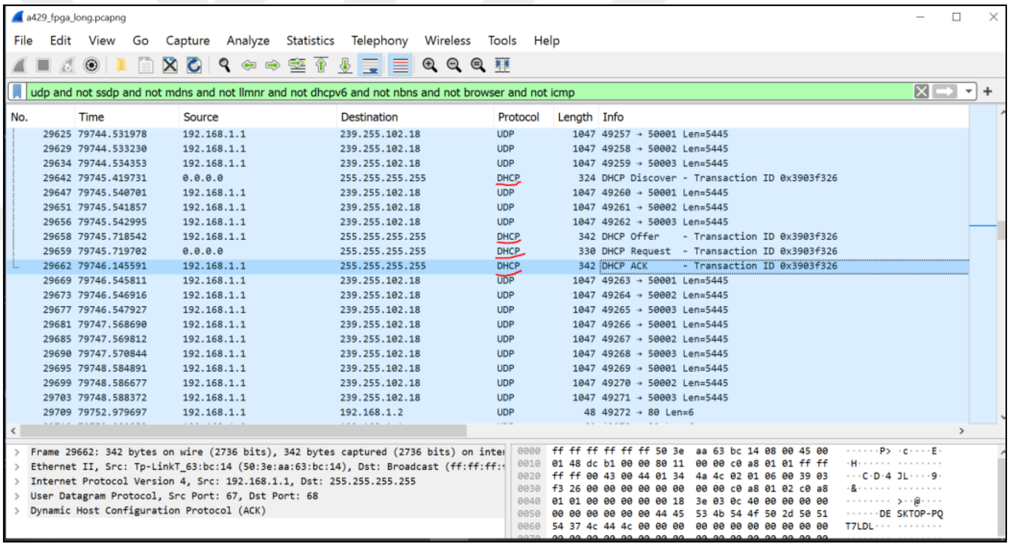

Figure 5.9: DCHP packets.

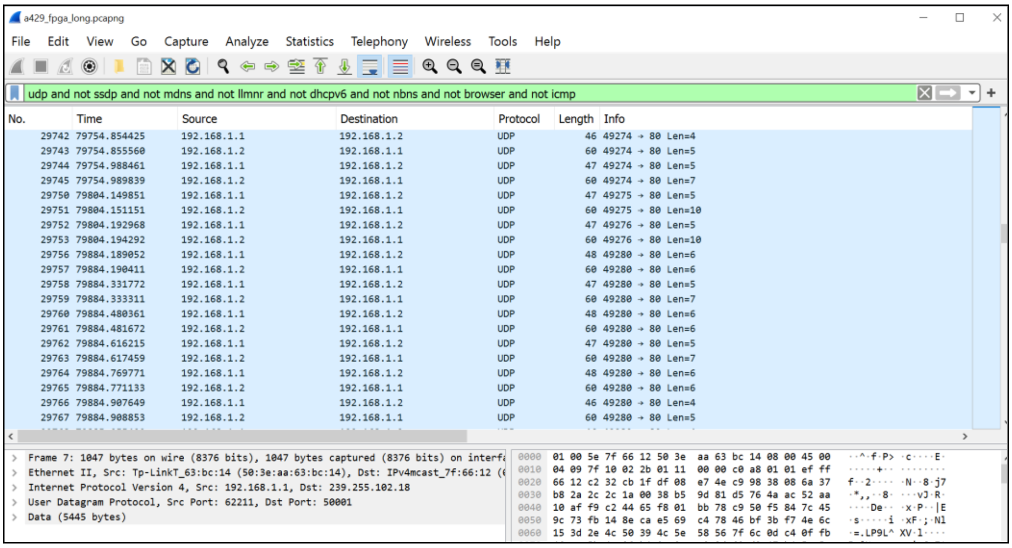

Figure 5.10: UDP packets.

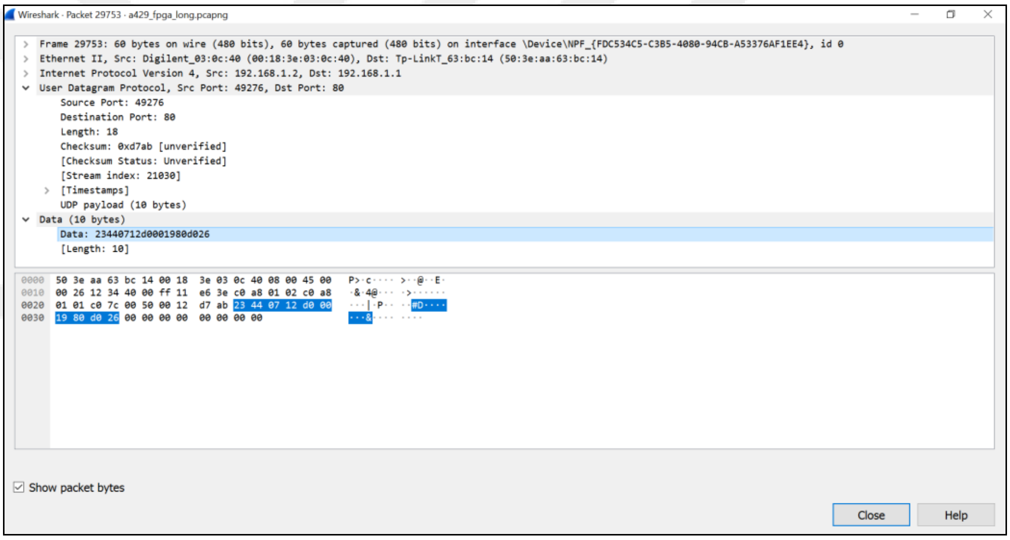

Figure 5.11: Single UDP packet.

5.3. Accuracy of Conversion

TechSAT's A429-USB-NT simulator device was used on tests. Test setup is shown in Figure 5.12. Recorded data which belongs to a commercial A/C was used on tests. Speed and altitude data was loaded to A429-USB-NT data manipulation applications. A dedicated test application was written for host computer. Arinc-429 A/C speed data is shown as speedometer gauge. A/C altitude data is shown as slider bar. A/C cabin communication is shown as table. Each letter of "pxso" text means sequentially passenger call, passenger call reset, no smoke and reading light. The letters represent the status lights above each seat in A/C. If letter “p” turns upper case “P” that means passenger call occurred and similarly for the other letters. The test application GUI is shown in Figure 5.13. While a dedicated test GUI was written for the presented implementation, the design in [11] has no dedicated test GUI. Conversion accuracy was evaluated by comparing Simulator GUI data and Host Test GUI data. Because, Simulator GUI shows transmitted data and Host Test GUI shows received data and vice-versa. So, the design was verified.

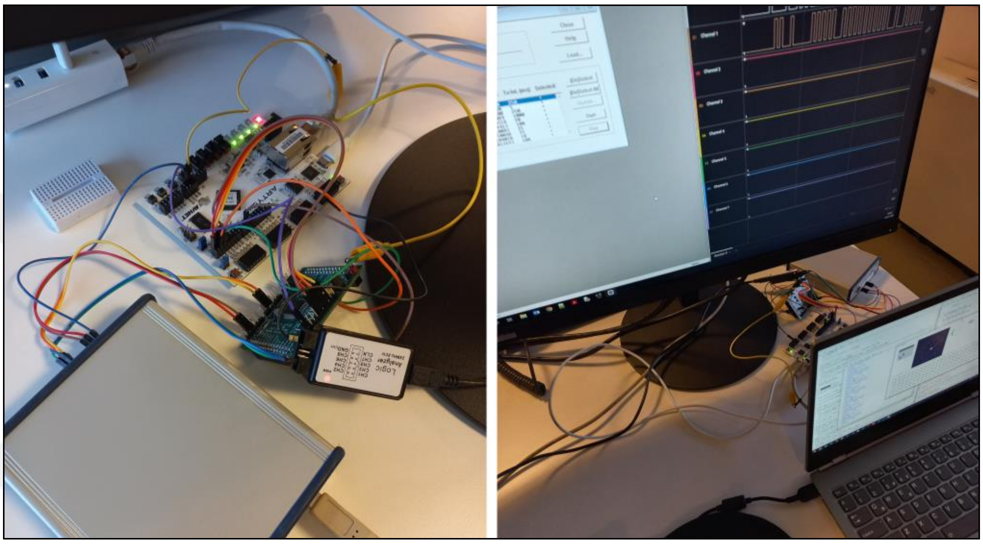

Figure 5.12: Test setup.

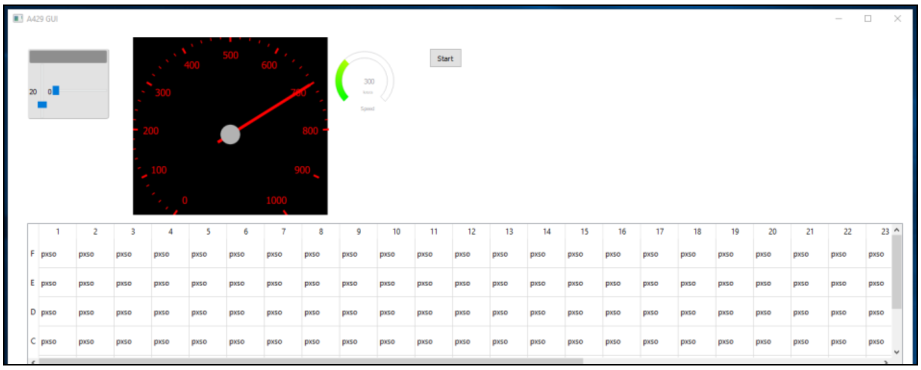

Figure 5.13: Test application on host computer.

6. CONCULUSION

Network protocols were researched and applied. Serial communication protocols such as Ethernet, Arinc-429 and UART were searched and compared in aspect of speed. Physical layers of them were examined. Size and complexity of the protocols were reviewed. Client application designs were studied for host computer GUI design. Different FPGA chip architectures were reviewed. Similar studies were researched.

Ethernet based multi-channel Arinc-429 converter has been implemented without using any IP Core. Source usage of the design was minimized. So, a compact, capable and easily transferrable implementation was achieved with the proposed Ethernet based multi-channel Arinc-429 converter design. It also supports IFE characteristics.

REFERENCES

[1] Aeronautical Radio Inc., (2012), “Digital Information Transfer System (DITS) Part 4 Archive of ARINC 429 Supplements”, Specification, ARINC Specification 429P4.

[2] Holt Integrated Circuits Inc., (2020), “HI-3593”, Datasheet, DS3593 Rev. G.

[3] Web 1, 2022, https://mitindia.edu/images/pdf/avionics_ppt/ARINC_429.pdf, (Erişim Tarihi: 10/11/2022).

[4] Dhaker P., (2018), “Introduction to SPI Interface”, Analog Dialogue.

[5] Web 2, 2010, https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf, (Erişim Tarihi: 17/11/2022).

[6] Vyas R., Bhattacharjee A., Khandelwal A., (2016), “Development of SoC for Interfacing Avionics Units with ARINC 429 Bus”, International Conference on Communication and Electronics Systems, 1-5, Coimbatore, India, 21-22 October.

[7] Spurgeon C. E., Zimmerman J., (2014), “Ethernet The Definitive Guide”, 2nd Edition, O’REILLY Media.

[8] Web 3, (2021), https://resources.altium.com/p/gigabit-ethernet-101-basics-implementation, (Erişim Tarihi: 15/01/2023).

[9] Mukherjee D., Kumar N., Singh K., Mondal H., Reddy B. V. R., (2011), “Implementation of ARINC 429 16 Channel Transmitter Controller on FPGA”, International Conference on Advances in Information Technology and Mobile Communication, 245-247, Maharashtra, India, 21-22 April.

[10] Hao L. Cao G., Shi-guo M., Xin L., Gang H., (2015), “ARINC - 429 airborne communications transceiver system based on FPGA implementation”, International Conference on Information Sciences, Machinery, Materials and Energy, 42-46, Chongqing, China, 11-13 April.

[11] Tamrakar I., Ananda C. M., Kumar B. P., Ramani P. S., (2016), “Design and Implementation of Ethernet based Analyzer for Avionics Serial Bus on FPGA”, IEEE International Conference On Recent Trends In Electronics Information Communication Technology, 1637-1641, Bangalore, India, 20-21 May.

[12] Sabihi A., (2008), “A Novel Electronic Circuit Design for ARINC 429 Transmitter Applied on Civil Aircrafts”, 7th International Conference on Mathematical Problems in Engineering, Aerospace, and Sciences, 179-189 Genoa, Italy, 24-27 June.

[13] Aeronautical Radio Inc., (2004), “Mark 33 Digital Information Transfer System (DITS) Part 2 Discrete Word Data Standards”, Specification, ARINC Specification 429 Part 2-16.

[14] Santos M. S., d'Amore R. and Saotome O., (2021), "Reliability Increase for Avionics Data Communication", IEEE Aerospace and Electronic Systems Magazine, 36 (10), 32-41.

[15] Li H., Zhao Y. and Wu R., (2021), “Optimal Design of UDP Protocol in Embedded Real-Time OS”, The 13th International Conference on Advanced Infocomm Technology , 180-184, Yanji, China, 15-18 October.

BIOGRAPHY

He successfully completed the Electronics and Communication Engineering Program of Electrical and Electronics Faculty at İstanbul Technical University at 2015. He worked at Vestek Electronics R&D in years betweem 2015-2016. He worked at TUBİTAK BİLGEM in years betweem 2016-2019. He has been working as a Senior Avionics Design Engineer at Turkish Cabin Interior since 2020. He is married and he has a child.