A Gigabit Ethernet Media Access Controller for TCP/UDP Radar Data Streaming and Visualization

Vukan D. Damnjanović, *Student Member, IEEE*, and Vladimir M. Milovanović, *Senior Member, IEEE*

***Abstract*—A design of a gigabit Ethernet media access controller implemented using Verilog hardware description language is depicted in this paper. The proposed digital hardware module can be utilized for establishing client-server connections over a computer network with a PC, an FPGA-based board or some other separate piece of hardware. It allows users to perform network data transfers using either TCP or UDP communication protocols, in both directions. Data is transmitted to or received from a predefined Internet Protocol address utilizing packets of predefined size, in a format suitable for the corresponding protocol, with a packet header providing the receiving end with the information about the packet itself. The described design is able to achieve network throughput rates that exceed 110 MB/s making it suitable for systems and applications that require high-speed data streaming, such as the system for radar data streaming and PC visualization depicted in the latter part of the paper. Besides that, it can be used in a wide range of applications developed on systems containing boards and devices with the Ethernet 8P8C port as an integral part. The implemented design has been thoroughly tested using a combination of a commercial FPGA development kit and the PC-run Python applications. It was verified and confirmed that the design meets the expectations regarding both the specified functionality and performance.**

***Index Terms*—Gigabit Ethernet MAC, data streaming, UDP and TCP protocols, PC data visualization, Verilog hardware description language.**

I. Introduction

During the last couple of decades, there is a growing trend for the amount of different devices used in the systems and applications in practically all the spheres of the IT industry. The same also applies for the systems used for collecting data, such as some sensor-based systems or radar systems. They are becoming more complex, consisting of more devices with more information needed to be carried. Whether it is because of the insufficient available resources, some environmental limitations, inappropriate system topology or something else, the data-collecting devices are often unable to perform the complete cycle of information extraction, processing and utilization relying only on themselves. Therefore, at some point of the cycle, it is necessary that the data is transferred to another device (or devices) so the system can work properly and fulfill its purpose. In most cases, those devices are PCs, due to their abilities and versatility.

Vukan D. Damnjanović is with the School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia and also with NOVELIC d.o.o., Veljka Dugoševića 54/B5, 11060 Belgrade, Serbia (e-mail: vukan.damnjanovic@novelic.com).

Vladimir M. Milovanović is with the Department of Electrical Engineering, Faculty of Engineering, University of Kragujevac, Sestre Janjić 6, 34000 Kragujevac, Serbia (e-mail: vlada@kg.ac.rs).

The need for these data transfers is especially emphasized on the distinguished group of systems - real-time systems, where the information acquirement, extraction and/or presentation is performed constantly at relatively high rate [1]. That implies that the data transfers from one platform to the other needs to be performed in the same manner. The amount of data that needs to be transferred and the transfer rate differ from system to system and can vary significantly, which includes some large numbers as well.

In order to achieve the ability of performing these transfers, a vast number of mechanisms have been developed during the years. They can rely on different technologies and all of them have their advantages and disadvantages. Among the most common and popular ones is definitely Ethernet [2].

Ethernet is, technically speaking, a family of wired computer networking technologies, but it usually refers to the most common type of Local Area Networks (LANs) - a connected network of computers (or, to be more precise, devices) in a small area[1]. Devices possessing the Ethernet port and connected through it to the network are able to perform data transfers with other connected devices by following the series of standardized protocols and rules [3]. Ethernet has been developing and improving during the years. It is currently one of the fastest communication technologies.

Computer networks using Ethernet consists of several abstraction layers [4]. In order for the whole mechanism to work correctly, rules for each one of them have to be applied. Following those rules is often managed by some kind of the processing unit, in devices that poses one, but in ones that do not, such as an FPGA-based board, it might be challenging to achieve the flawless operation of the system. The gigabit Ethernet media access controller from this paper's topic is created so that the Ethernet ports can be utilized for preforming data transfers without engaging any kind of processing unit.

This paper, in its first part, gives the quick overview of the data streaming protocols used in the gigabit Ethernet media access controller module, as well as the detailed description of the module's design, along with the description of its implementation using Verilog Hardware Description Language (HDL). In the second part of the paper, obtained testing results and performances are provided with the example of one of the systems which the module was tailored for in the first place.

[1]Computer networks is a large field in network sciences and a lot could be written about it, but the information provided is sufficient for the comprehension of the content of this paper.

II. A Gigabit Ethernet Media Access Controller and Data Transfer Protocols

A gigabit Ethernet media access controller is a digital component which allows the user to perform data transfers between itself and some other module or device. Basically, the implemented module allows the user to send and receive the data to and from the specified address on the network belonging to some other device. In order for it to work properly, the module requires that the Ethernet port along with the gigabit Ethernet transceiver exist on the device. It is used to set up the transceiver for working in the appropriate mode at the beginning of the application and then to send (or receive) data through the Gigabit Media-independent interface (GMII) to the transceiver [5] and through the port to the network and the rest of the system. Data is fragmented and transferred in packets, where every packet is of the same length and consists of a header and data itself.

A. Ethernet Abstraction Layers

Currently, two different versions of this module exist: one that supports transfers (receive and transmit) using Transmission Control Protocol (TCP) [6] and another that supports transfers using User Datagram Protocol (UDP) [7]. Those two protocols are parts of the Ethernet transport layer, one of the abstraction layers mentioned in the previous section. This layer provides the end-to-end communication services for applications. This module also secures that the device is working in accordance with two other abstraction layers: link layer, which provides the link to a physical connection of the host, and internet layer, which serves as a bridge between link and transport layer [4]. The fourth and final layer - application layer, can be implemented on the PC or on some other device.

The controller module implements the network link layer by applying the Address Resolution Protocol (ARP) [8]. It sends a message in the appropriate format that provides the physical MAC address of the Ethernet port when another device on the network asks for it.

The internet layer regulates that every message or packet in the network end up at the appropriate destination. The primary protocols for the internet layer are the Internet Protocol (IP). This protocol assigns an IP address to every device on the network, which allows a packet to find its way to the destination. The implemented module utilizes the IP protocol version 4 (IPv4).

The Ethernet transport layer, as mentioned before, is implemented using either TCP or UDP protocol. This layer provides services to the network, such as connection-oriented communication, reliability, flow control etc.

B. TCP and UDP Protocols

TCP is a more complex protocol than UDP. It is connection-oriented with built-in systems checking for errors and guaranteeing that data will be delivered in the order it was sent. The connection firstly needs to be established, then maintained, and finally terminated, making it a more reliable protocol. All of this, however, requires larger overheads in data packets, which reduces its speed and efficiency. On the other hand, UDP is a simpler, connectionless protocol, faster and more efficient, but it does not provide any recovery options when a data error occurs or when a packet is lost [1].

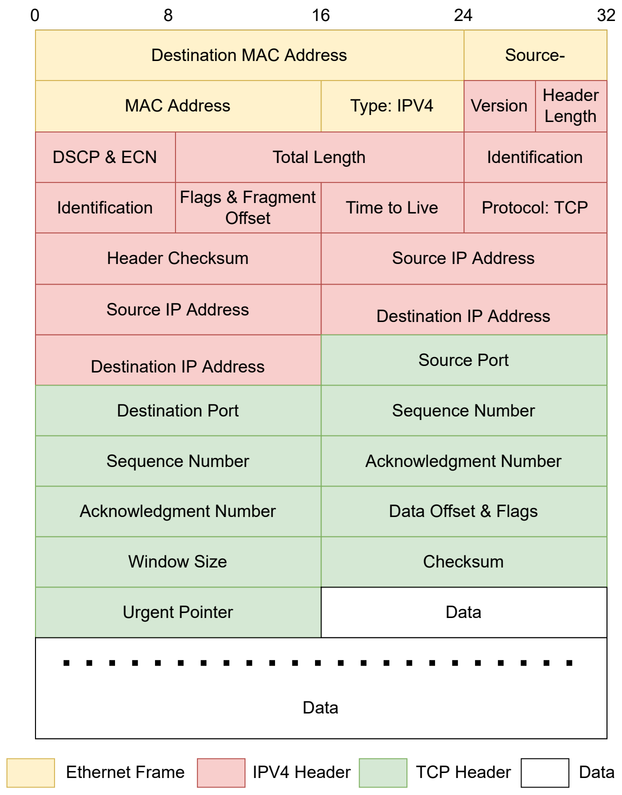

Fig. 1. A structure of data packets of the TCP protocol used in the design.

A structure of data packets of the TCP protocol used in the design is shown in Fig. 1. This type of format allows three network layers to be implemented: the link layer implementation is marked in yellow, the IPv4 header representing the internet layer is marked in red and the TCP header, as a part of the transport layer is marked in green [9]. The link layer carries the information about MAC addresses and the used IP protocol. The IPv4 header has the fields for various information, such as the IP addresses, packet identification number, packet length, header length, used transport layer protocol, the IPv4 header checksum value etc. The TCP header, besides the port numbers, holds the information necessary for establishing, maintaining and terminating the TCP connections, detecting and recovering from errors, flow controlling etc.

The main fields for enabling the TCP to have the connection-oriented communication are sequence number, acknowledgment number, flags and checksum. The sequence number is a 32-bit wide field that carries the number that identifies the first data octet in the packet. The acknowledgment number is also a 32-bit wide field, and it represents a response from the receiving end. It has the value of the next expected sequence number. If the first packet that has arrived had the sequence number of 1 and the N bytes of data have arrived, then the acknowledgment number in the response would be $1 + N$. This mechanism ensures that the order of the received packets is preserved and that there is no packet or data missing. The flag fields have the purpose to indicate that some functionality is being used. There are six flags in total indicating different things: URG - the urgent pointer field is significant, ACK - the acknowledgment number field is significant, PSH - push functionality, RST - reset the connection, SYN - synchronize sequence numbers, FIN - no more data from sender. Basically, to run a TCP connection, only three flags are needed: SYN flag for establishing it, FIN flag for terminating it and ACK flag for acknowledging every received data packet during the connection's life. The checksum field is used for detecting if there is an error in the received data. It has a width of 16 bits, and it is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text (only the TCP header and some field of the IPv4 header are included).

In Fig. 2, a diagram that depicts establishing, maintaining and terminating a TCP connection between a client and a server is shown. A client is called a device that initiates the connection and a server is a device that accepts it. Even though it is more usual for the server to send data and for the client to accepts it, it is the other way around in the example shown in Fig. 2. The client initiates the connection by sending a packet with the SYN flag active. The server responds with SYN and ACK flag, which the client acknowledges. At that point, the connection is established. The client then sends N bytes of data in each one of X sent packets, and the server responds for every packet received. Note that it is not necessary for the server to respond to every packet individually. It could also wait for all the packets to arrive and then to acknowledge the reception of them by sending the final acknowledgment number along with the ACK flag. After all the packets are sent, the client expresses the wish to end the communication, which the server accepts and the connection is then terminated.

The UDP data packet structure used in the design is similar to the one used for the TCP protocol. In fact, the only thing that differs is the transport layer protocol header. As mentioned before, the UDP does not provide the possibility of connection-oriented communication, flow control etc. so the UDP header has fewer fields than the TCP header. It only carries the information on the port numbers, packet length and the checksum value. The UDP protocol does not support or require the acknowledgement of the received packets. It straight-forwardly goes to the formation of the following packet, after the previous one has been sent.

III. The Implementation of the Controller

Previously depicted gigabit Ethernet media access controller have been implemented using Verilog HDL. Its design has been thoroughly tested using standard verification and implementation paths for FPGA design flow. The design is made available [10] by the authors for public use as a free and open-source hardware library.

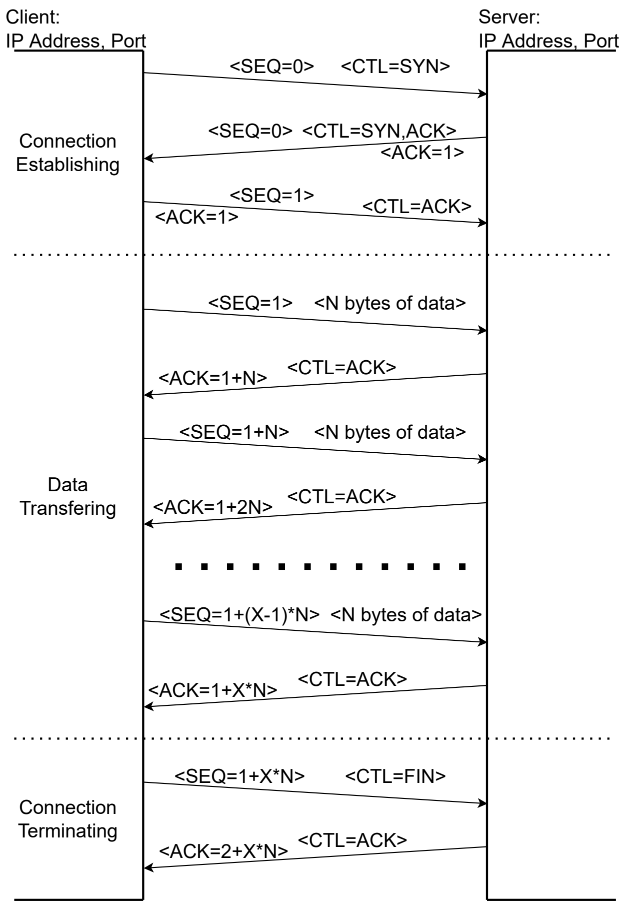

Fig. 2. A diagram that depicts establishing, maintaining and terminating a TCP connection between a client and a server.

The implemented design of the gigabit Ethernet media access controller is relatively complex. It can be divided into several mutually connected submodules, with some of them communicating with the outer world as well, through one of the module interfaces. A block diagram of the module with its submodules and interfaces is depicted in Fig. 3. In this section, descriptions of every individual submodule, implemented interfaces to the outer world and the way the module communicates with other devices will be provided.

A. *Design Interfaces*

As it can be seen from Fig. 3, the implemented gigabit Ethernet media access controller has four interfaces. The first one is the AXI Stream interface. The purpose of this interface is to continuously collect data needed to be sent between the devices on the network. The direction of the interface can be both input and output, depending on the fact whether the module is receiving data from some other device, or sending it to the network. The second interface of the implemented module is the AXI4 memory-mapped interface. This interface is used to write values to the memory-mapped configuration registers of the controller, as well as to read status values from it. The next interface is the Reduced

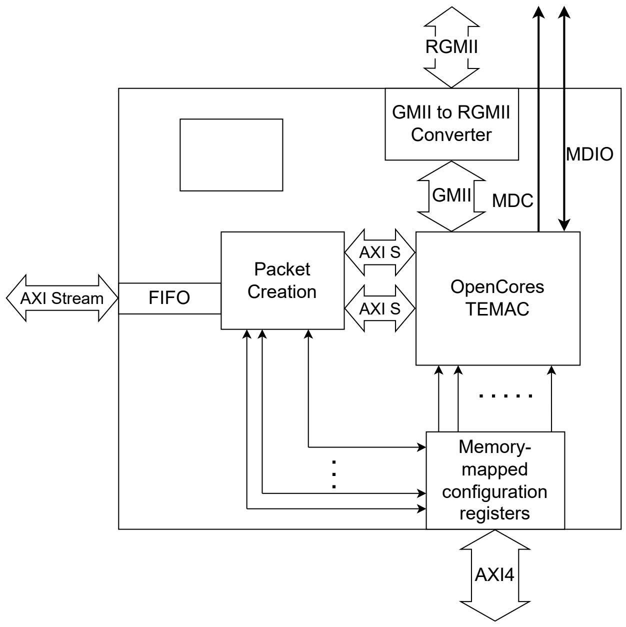

Fig. 3. A block diagram of the implemented module with its submodules and interfaces.

Gigabit Media-independent interface (RGMII), a version of the already mentioned GMII interface. Its role is to communicate with the Ethernet physical layer transceiver that controls the Ethernet port. The last interface is the Serial Management Interface (SMI), also known as Media-Independent Interface Management (MIIM). It is a serial interface used for configuration of the Ethernet physical layer transceivers. In the following paragraphs, the functionality and implementation of every submodule of the controller design will be described.

B. Design Clock Domains

The whole design can be divided into four clock domains. The frequency of the input clock equals 100 MHz, and it drives the clock generator block that creates all other clocks in the module. These four clock domains are the user clock domain, the RGMII physical layer clock domain, the MIIM management interface clock domain and the AXI4 memory-mapped clock domain. The RGMII physical layer clock domain operates at the frequency of 125 MHz, and it is the only mandatory value for all the clock frequencies in the design. The frequency of the MIIM clock domain clock is run-time configurable. Its value depends on the value stored in one of the memory-mapped registers, and it equals the value of the AXI4 memory-mapped clock frequency divided by the register value. The MIIM clock frequency must not exceed 2.5 MHz. The AXI4 memory-mapped clock frequency is the frequency on which the memory-mapped configuration registers and the AXI4 bus operate. It has to be equal to the frequencies of the other memory-mapped devices connected to the bus, and in this version of the design it equals 10 MHz.

C. Design Submodules

The OpenCores Tri-mode Ethernet Media Access Controller (TEMAC) is one of the most important submodules in the design. It is a modified version of an open-source controller downloaded from the OpenCores website [11]. The module has a ”Tri-mode” phrase in its name because, apart from being a gigabit controller, it can also operate as a 100-megabit or 10-megabit. However, in the proposed design, it is utilized solely as a gigabit version. The TEMAC module has several functionalities and five interfaces: an output and an input AXI Stream interfaces, a GMII interface, a MIIM interface and an input interface that provides the module with the memory-mapped register data. Its main function is to convert streaming data-to-be-sent from the input streaming native interface to the output GMII interface and data-to-be-accepted from the input GMII interface to the output streaming native interface. These two native interfaces are converted to the AXI Stream interfaces using the submodule wrapper. For these GMII-to-Stream conversion processes, the module instantiates two dual port block RAMs to serve as asynchronous FIFOs for getting the data from both sides. The finite-state machines (FSMs) control the flow for both directions, from 32-bit streaming data synchronized on the user clock, to 8-bit GMII data synchronized on the GMII clock (or vice versa). The submodule and the exact way it will operate can be set up by reading the input values from the memory-mapped configuration registers. Depending on some of those values, it also generates the MIIM signals for the Ethernet physical layer transceiver configuration.

The GMII to RGMII converter adapts the GMII interface signal to the needed RGMII interface signal. Basically, the RGMII signals are used instead of the GMII signals in order to reduce the number of the occupied output pins. Total number of utilized output pins is halved (12 instead of 24). It is achieved by running half as many data lines at a double speed, time multiplexing the signals and by eliminating non-essential signals. Output pins operate with double data rate (DDR) instead of single data rate (SDR), with the same clocking frequency of 125 MHz. Receiving pins are synchronized to the external clock provided by the Ethernet physical layer transceiver, while transmitting pins are synchronized to the internally generated clock. The clock generating block creates two different 125 MHz clocks with the 90 degrees phase difference, one for the output clock pin itself and the other for the synchronizing of the data and control lines, so that the setup and hold times of the output DDR pins are as large as possible.

The memory-mapped configuration registers module is a submodule whose function is to be accessible from the AXI4 interconnect bus and the rest of the system through the AXI4 memory-mapped interface and to provide the rest of the submodules inside the controller with the written values. It also has a task to inform the OpenCores TEMAC submodule to generate the signals in the MIIM interface. It has numerous registers and here are the most important ones:

- `Physical address` - Value of the address of the Ethernet physical layer transceiver.
- `No preamble` - Indicator whether the transmitting packets will have the preamble to precede them.
- `Clock divider` - Value used to calculate the frequency of the MIIM interface clock.
- `Packet size` - Number of bytes in one data packet, can be up to 1500.
- `PHY data` - Data value to be written to one of the Ethernet transceiver registers.
- `PHY register address` - Register address inside the Ethernet transceiver to which data will be written.
- `PHY write enable` - Indicator that a write operation should be performed through the MIIM interface.

The packet creation submodule is responsible for implementing all the network abstraction layers and protocols in the design. This submodule has several tasks in its jurisdiction. It wraps the data arrived from the streaming interfaces with the appropriate header and calculates all the values for the header fields. It also accepts data packages arrived from the network and checks if they are addressed to this module and creates and sends the response if it is needed. IP addresses, MAC addresses and port numbers for both the client and the server side in this design are hard-coded. For both TCP and UDP versions of the module, this submodule always checks if there is an ARP request sent to the network. If the asked IP address is the one belonging to this module and the ARP format of the message is correct, an ARP response packet is created and sent providing the information about this module's MAC address. Creation of the UDP packets is not too complicated, considering that every field of the header except the identification field is a constant value. The packet creation submodule receives the streaming data and forwards it to the OpenCores TEMAC submodule, except for the occasions when the previous packet has ended and when the header fields are needed to be sent. The end of packet is indicated by sending the active high value for the AXI Stream data last signal.

The situation for the TCP version of the design is a bit more complex. The creation of the header is not as straight-forward as in the UDP version, and the communication between the devices on the network is more complicated. The packet creation submodule calculates the value for several fields for every sent packet, such as the sequence and acknowledgment number, checksum value, flags etc. When the application starts, it sends the synchronization request packet, as depicted in Fig. 2. Then it waits for the response and acknowledges it if the response has the appropriate form and values, and starts creating data packets and streaming data. At the end of the application, it waits for the data acknowledgment message, and then it terminates the connection as described in the previous section.

IV. Testing Results and Streaming Radar Data Visualization Example

In this section, the design testing flow will be presented. During these tests, the functionality of the design was verified, performances and resource utilization were measured, and the design validity is shown as it is used in the example system for radar data visualization. The first step in the design testing flow were the software simulations in the form of the testbench files written in Verilog and VHDL languages.

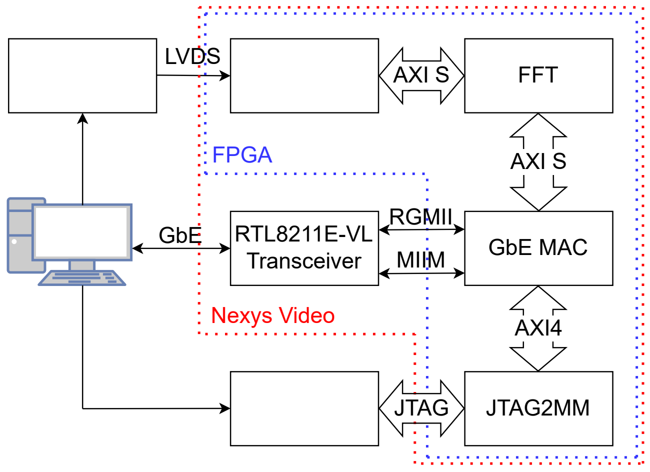

Fig. 4. A simplified block diagram of the complete radar data PC visualization system.

The next step is the implementation and verification of the design on an FPGA-based development board. A Digilent's Nexys Video board with Xilinx Artix-7 FPGA family is used for it. Nexys Video development board has the Realtek RTL8211E-VL Gigabit Ethernet Transceiver [5] as an integral part of it, making it suitable for the depicted design. For the design testing, some additional features were needed. An open-source JTAG-to-memory-mapped bus master bridge [12] for accessing the memory-mapped register space was used. An alternative for it can be Xilinx's JTAG-to-AXI4 Master module [13]. Write data transactions through AXI4 memory-mapped interface are initiated from the PC using the Python PyFTDI library and the FTDI's cable containing the FT232H chip [14]. Also, a server was run on the PC in order to generate responses for the arrived packets and to send data to the board. For running the server, Socket Python library was used. The arrived packets can be verified by utilizing software for the network packet monitoring, such as Wireshark [15]. Moreover, packets with previously defined data were sent from the board and checked on the PC using Pyhton scripts, therefore proving the correctness of the proposed design. All the examples and testing systems presented in this paper use 1066-byte long packets (52 bytes for the header and 1024 data bytes).

During the hardware implementation testing, data throughput measurements were done. It was proven that the design meets the performance expectations with the maximal data throughput of around 110 MB/s for both TCP and UDP design versions, making it around 90% of theoretically ideal value of 125 MB/s or 1 Gb/s. The resource utilization is moderate, but with less than 10% utilization for all the resource types and with the possibility to reduce it even more.

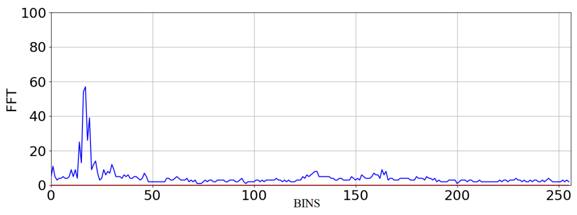

Fig. 5. An example of the radar data plot using Python libraries.

A. Streaming Radar Data Visualization Example

The implemented design of the gigabit Ethernet media access controller, due to the extensive usage of computer networks and its functionality, could find its way into a wide range of different systems. In this subsection, a system for the TCP radar data streaming and PC visualization, one of the mentioned example systems, will be presented.

A simplified block diagram of the radar data visualization system is given in Fig. 4. The system is used to continuously collect data from the radar board, have the incoming streaming data processed (fast Fourier transformation) on the FPGA-based development board, and send it using the gigabit Ethernet to the PC, where the arrived data is plotted. The PC is showing the live display of the distance between the radar board and the detected targets. An example of such a display can be seen in Fig. 5. It should be emphasized that all the processing and data transferring is realized completely in hardware, without involving any kind of processing unit.

For the radar board, Texas Instruments' AWR2243 BOOST [16], along with the MMWAVE-DEVPACK and FMC-ADC-ADAPTER, is chosen. The output is in the form of the Low-Voltage Differential Signaling (LVDS) lines [17] containing radar data, clock and frame clock. Those LVDS lines are connected to the FPGA pins on the Nexys Video board, and they are received and converted to the 32-bit AXI Stream interface. The logic behind this conversion is not relevant for the matter and therefore not elaborated. The AXI Stream data is then driven to the input of the open-source fast Fourier Transformation processor module [18] available to simultaneously perform the processing and stream the output data to the gigabit Ethernet media access controller from the topic of this paper. The TCP packets are created there and sent to the PC through the gigabit Ethernet, where there is a Python script running the server and receiving and plotting data acquired from the Ethernet port using the Python's Matplotlib library. The radar board is previously configured using the MMWAVE Studio software [17] for the PC and the USB interface, as is the gigabit Ethernet MAC module using the JTAG-to-memory-mapped bus master bridge, depicted in the previous section.

V. Conclusion

In this paper, a design of the gigabit Ethernet media access controller for the UDP and TCP data streaming implemented using Verilog HDL is proposed. This module can be used in a wide range of different systems, due to the nowadays' constant presence of the computer networks in many industrial spheres. One of those systems, or the system for the processed radar data PC visualization to be more precise, is depicted in this paper as well.

The generated instances of the JTAG to memory-mapped bus master bridge were tested and verified by both using software simulations and mapping onto a commercial FPGA development board, proving the correct functionality of the design. The hardware implementation also proved the competitiveness of the design in terms of performances, having the data throughput of over 110 MB/s.

It should be noted that this is only the first version of the design, and there is still a lot of space for improvement and for broadening the functionality of the module. Making it more parameterizable, completely run-time configurable, having better mechanisms to recover from data loss or error etc. are just some of the things that could be and hopefully will be improved in some future versions.

Acknowledgements

The authors would like to thank NOVELIC d.o.o. for financially and logistically supporting the work on this project.

References

[1] S. Tibor, P. Dukán, B. Odadžíc, and O. Péter, "Realization of reliable high speed data transfer over udp with continuous storage," in *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2010, pp. 307–310.
[2] *The Ethernet*. Digital Equipment Corporation, Intel Corporation, Xerox Corporation, 1982, a Local Area Network, Data Link Layer and Physical Layer Specifications.
[3] V. Cerf and R. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, 1974.
[4] I. E. T. Force, "Requirements for internet hosts – communication layers," in *RFC*, October 1989.
[5] *Integrated 10/100/1000M Ethernet Transceiver*, version 1.6 ed., Realtek, April 2016, track ID: JATR-3375-16.
[6] "Transmission control protocol," in *RFC*. Information Sciences Institute, University of Southern California, 1981, no. 793.
[7] J. Postel, "User datagram protocol," in *RFC*, 1980, no. 768.
[8] D. C. Plummer, "An ethernet adress resolution protocol," in *RFC*, 1982, no. 826.
[9] W. Zhang, Z. Wei, X. He, P. Qiao, and G. Liang, "The design of high speed image acquisition system over gigabit ethernet," in *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*, 2010, pp. 111–115.
[10] V. D. Damnjanović and V. M. Milovanović, "Gigabit ethernet mac," www.github.com/milovanovic/gbemac, accessed: 2022/04/15.
[11] OpenCores, "Tri-mode ethernet mac," www.opencores.org/projects/ethernet_tri_mode/, accessed: 2022/04/15.
[12] V. D. Damnjanović and V. M. Milovanović, "A chisel generator of jtag to memory-mapped bus master bridge for agile slave peripherals configuration, testing and validation," in *2021 IcETRAN Proceedings*. ETRAN Society, Belgrade, 2021, pp. 239–244.
[13] *JTAG to AXI Master v1.2*, Pg174 ed., Xilinx, February 2021.
[14] *FT232H*, version 2.0 ed., FTDI, document No.: FT000288 Clearance No.: FTDI 199.
[15] U. L. R. Sharpe, E. Warnicke, *Wireshark User's Guide*, (version 3.7) ed.
[16] *AWR2243 Single-Chip 76- to 81-GHz FMCW Transceiver*, Texas Instruments, February 2020.
[17] *DCA1000EVM Data Capture Card*, Texas Instruments, May 2018.
[18] V. M. Milovanović and M. L. Petrović, "A highly parametrizable chisel hcl generator of single-path delay feedback fft processors," in *2019 IEEE 31st International Conference on Microelectronics (MIEL)*, 2019, pp. 247–250.

ANN model for one day ahead Covid-19 prediction

Jelena Milojković, Miljana Milić, and Vančo Litovski

Abstract— **One-day-ahead prediction of number of COVID-19 infected patients is presented in this paper. The study is relying on the data available in [1]. A model of artificial neural network (ANN) was developed and used with only the most recent data taken into account. We believe that only a few data from the near past is important for this type of prediction. ANNs have been proven as a very reliable method for the real time prediction systems. In our previous work in prediction electricity consumption [2] and traffic prediction [3], we obtained small prediction error. That encouraged us to conduct the research described in this work. The absence of the trend and the seasonal component in the given time series, made the prediction task more difficult. However, we have obtained good results, which could encourage the application of the model in health management to make better decision in control and prevention of the occurrence of a pandemic.**

Index Terms—**Covid-19, number of infected, artificial neural network, short-term prediction.**

I. Introduction

Global pandemic, named COVID-19, created his first wave of infection in China in the Wuhan province [4]. It has started in December 19 and continued to the present days. By the World Health Organization (WHO), the virus has affected populations worldwide, and its rapid spread is a universal concern. The high rate of spread as well as the high chance of transmission is still not effective, even with engaging all recommended prevention and implemented control strategies (isolation, detection tests and prophylactic measures). They still have limited effect in preventing or stopping the spread of the virus worldwide [5]. Since its first reporting at the end of December 2019. until 28.04.2022, over 508 million people have been infected, around 500 million people recovered, and 6 227 291 people died due to pandemic [6]. Basic and most important fact of COVID-19 is that it is spreading rapidly by a human-to-human transmission; where about 20% infected subjects are without symptom. The main characteristics of COVID-19 pandemics are high infection rate, incubation period, patients to be contagious during the incubation period, and symptomatic infection [7]. The elderly people and those who have weakened immune systems as well as people with special health conditions such as cancer, hypertension, severe asthma, cardiovascular disease, lung conditions, heart disease, diabetes, neurological conditions, HIV/AIDS infection, pregnancy and high weight are more vulnerable to the serious effects of this pandemic [8]. Based on this we can conclude that a global pandemic like Covid-19 has a high negative impact on the population health, social–cultural activities and global economy [9]. For this reason, it is necessary to develop models to predict the course of events during a pandemic outbreak. In our paper, we used ANN adapted to predict the number of infected on a daily basis. The developed model will help decision-makers, doctors and medical assistants to prepare and understand the magnitude of the risk and take appropriate measures to prevent major leaps. Forecasting tools can also help to assess the extent of risk in a timely manner and make the necessary preparations.

According to the research in the field of Covid-19 prediction by statistical methods, in order to achieve a satisfactory prediction, a basic prediction period of several hundred samples must be used [10, 11]. In the case when we have a set of data of several dozen samples, then time series is presented as a set of trends, random and seasonal components; these models also have a very limited number of parameters. In some cases, even some time series with a striking trend and seasonal component can be predicted with a smaller base period [12]. Actually, the amount of data available in this case is large enough to apply any other prediction method [13, 14, 15], but looking at a diagram curve representing the number of infected patients in one year, we easily recognize that past values of the infected patients are not very helpful when prediction is considered. Accordingly, we propose the problem of prediction of the infected case number in the next day to be performed as a deterministic prediction based on very short time series.

II. Previous related work

The research in Covid -19 pandemic related with infected case number we describe here is based on our previous results in development and the application of ANN.

In our paper [16] we can see the evaluation of the idea about ANN structures dedicated to short term prediction. First, we will here briefly illustrate the development of two complex ANN structures, which started with a simple one-input-one-output feed-forward ANN.

We first got involved in the ANN based prediction when solving a problem of electronic waste management in Serbia [17]. It came out that there was no systematic way of forecasting the amount of electronic waste to be found in the literature. The main reason for that was the lack of data for a longer period in the past. That inspired us to start with the

Jelena Milojković, Miljana Milić and Vančo Litovski are with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: jelena.milojkovic@elfak.ni.ac.rs, miljana.milic@elfak.ni.ac.rs, vanco.litovski@elfak.ni.ac.rs).

implementation of ANNs that are known as universal approximators. Namely, by using the ANN for approximation of a function represented by a set of equidistantly taken samples one automatically solves one of the biggest problems in approximation: the choice of the approximating function. Furthermore, ANNs are known as very successful interpolators which is frequently defined as a generalization property of ANNs. One had to investigate if ANNs could also extrapolate.

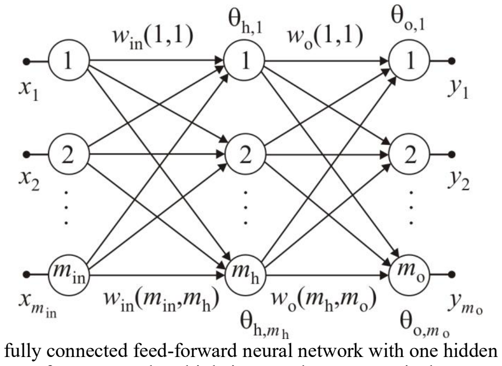

Fig.1. A fully connected feed-forward neural network with one hidden layer of neurons and multiple input and output terminals

This research was conducted in [18]. One fully connected feed-forward neural network is depicted in Fig. 1. To predict the amounts of electronic waste we have implemented a feed-forward ANN, named feed-forward accommodated for prediction - FFAP. The efficiency and good accuracy of the FFAP network inspired us to enter the problem of prediction for consumption of electrical power. There we were confronted with two types of periodicity (daily and weekly) where we have created new structure named Extended FFAP - EFFAP [19].

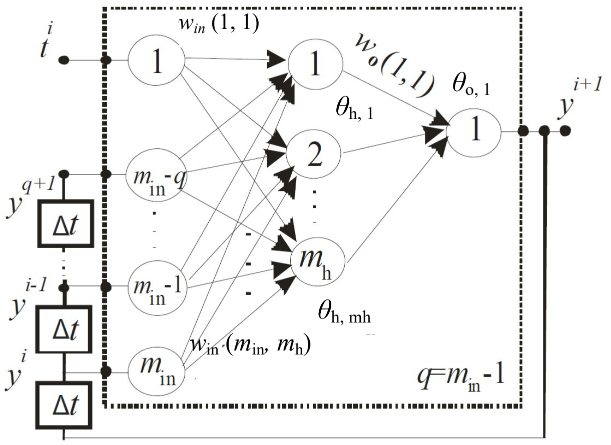

Fig. 2. Time controlled recurrent ANN - TCR

To improve the performance of the ordinary feed-forward ANN, in [20], we examined the capacities of time delayed ANN and evolved to a time controlled recurrent – (TCR) neural network depicted in Fig. 2. The prediction results obtained by the TCR ANN were equally good as those obtained by the FFAP ANN. That was confirmed in its application of prediction in microelectronics [21]. Using similar procedure to FFAP, we have formed a new structure named ETCR (Fig. 3). Two such ANN models we have also applied for prediction of electric power consumption and traffic [2, 3, 22].

III The Method

The basic neural network structure is shown in Fig. 1. It was proven that only one hidden can be sufficient for prediction problem [23] that is the subject of this research. In this figure input layer is denoted with “in”, hidden layer with “h”, and output layer with “o”. The set of weights, $w(k, l)$, connects the input and the hidden layer, where we have: $k=1,2,..., m_{in}$, $l=1,2,..., m_h$, while for the set that connects the hidden and output layer we have: $k=1,2,...m_h$, $l=1,2,..., m_o$. The threshold levels θ, are here designated with $\theta_{x,r}$, ($r= 1, 2, \dots, m_h$ or $r= 1, 2, \dots, m_o$), with x standing for “h” for hidden or “o” for the output layer. The input layer neurons are only delivering the signals, and the hidden layer neurons are activated by a sigmoidal activation (logistic) function. At the end, the output layer neurons have a linear activation function. A variant of the steepest-descent minimization algorithm is applied during the ANN training [24].

To obtain the number of hidden neurons, m_h, a procedure based on proceedings given in [25] is applied. In prediction of time series, in that case, a samples dataset is available (acquired in every two hours) which means that only one input signal is enough, and that is the discretized time. According to equation (1) only one output value is predicted at a time, which means that only one output is required, too. Network’s output signals are consumed average power for a period of two hours.

For the implementation of the architecture in Fig. 1, (one input and one output terminal), the following time series would have to be learned: $(t_i, f(t_i))$, $i=1,..., m$.

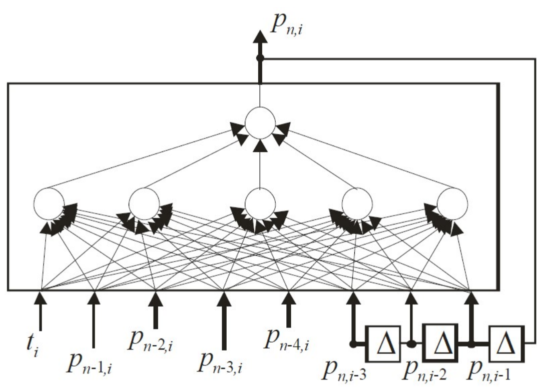

Fig. 3. ETCR. Extended time controlled recurrent ANN

To solve this problem, two new architectures were suggested as the possible solutions. They appeared to be the most convenient for the forecasting problem that is based on the short prediction base period [20]. However, these architectures had to be properly accommodated, due to the availability of data related to previous weeks.

The first network, referred to as a *time controlled recurrent* - TCR, Fig. 2 was derived from the basic time delayed recurrent ANN [26]. The structure has a recurrent architecture where time is the input variable, and it controls the predicted value. This structure is then extended, in order to allow that the values for the power consumption at a given time per day, and the values for the same days in three previous weeks, control the output. Consequently, the word *extended* had to be appended. The final architecture is depicted in Fig. 3, and is referred to as the Extended Time Controlled Recurrent (ETCR) architecture. It would be very useful to use the advantages of the ANNs' generalization property and the efficiency of the recurrent structure. This network learns a set in which the output value is controlled by the present time and its own previous instances of the average power consumption for a two hour period in a given day of the week:

$$p_{n,i}=f(t_i, p_{n,i-1}, p_{n,i-2}, p_{n,i-3}, p_{n-1,i}, p_{n-2,i}, p_{n-3,i}), \quad i=3, ..., m. \qquad (1)$$

where n stand for the number of the week (in the month or in the year). In that way the values designated with n are from the current week, while the values indexed n-j, j=1,2,3, are from the previous weeks. The designation "i" stands for the i-th sample in the selected day. The actual value $p_{n,i}$ is unknown and should be predicted.

The second architecture is referred to as a *feed forward accommodated for prediction* (FFAP) and is shown in Fig. 4. The idea here was to push the neural network to learning the same data window several times simultaneously but shifted in time. It is expected that the previous responses of the function will have larger impact to the $f(t)$ mapping. The architecture has one input terminal - t_i. The approximation y_{i+1} is obtained at the *future* terminal *Output*3. For multiple-step ahead predictions the future terminal can be considered as a vector. The *present* value y_i is represented at the terminal *Output*2. *Output*1 has to learn the *past* value i.e. y_{i-1}. *Output*1 may also be considered as a vector if we need to control the mapping using a *set* of previous values. The functionality of the network could be expressed as

$$\{y_{i+1}, y_i, y_{i-1}, y_{i-2}\} = \mathbf{f}(t_i), \qquad i=3, ...,m, \qquad (2)$$

where *Output*1={ y_{i-1}, y_{i-2}}. This indicates that one future, one present and two previous responses are to be learned.

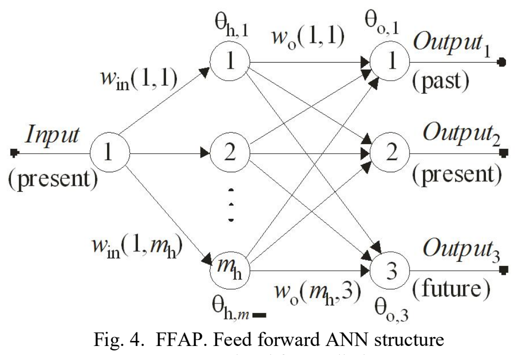

Fig. 4. FFAP. Feed forward ANN structure accommodated for prediction

According to our experience, the FFAP architectures produce more accurate forecasts than the TCR. However, it is a common practice to implement both of them for each forecasting problem and use the results obtained as a reference to each other when choosing the forecast that makes most sense. In this way, we could easily detect and avoid those solutions that represent local minima in the optimization process during the training of the ANN.

In the case of power consumption we have extended the FFAP architecture exactly in the same way as for the TCR architecture. The approximation function could then be written as

$$\{p_{n,i+1}, p_{n,i}, p_{n,i-1}, p_{n,i-2}, p_{n,i-3}\}= f(t_i, p_{n-1,i}, p_{n-2,i}, p_{n-3,i}, p_{n-4,i}), \quad i=4, ...,m. \qquad (3)$$

The obtained network can estimate the future (unknown) values $p_{n,i+1}$, using the data for:

- the actual time t_i,
- the actual consumption $p_{n,i}$,
- the past consumption values for the given day in n–th week ($p_{n,i-k}$, k=1,2,3),
- and the past consumption values for the same day and actual time of the previous weeks ($p_{n-j,i}$, j=1,2,3,4).

The new architecture is referred to as an *extended feed forward accommodated for prediction* (EFFAP), and is shown Fig. 5.

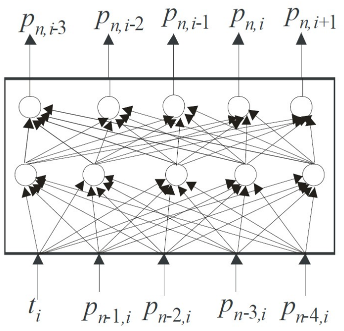

Fig. 5. EFFAP. Extended feed-forward accommodated for prediction ANN

IV. MAIN RESULTS

Having in mind the nature of the available data we have decided to implement the ETCR structure. A network with 6 hidden neurons was used while 8 previous samples were exploited for prediction.

The procedure could be described with the following steps. Having in mind the random choice of the initial values of the ANN's parameter for training, and the fact that for every such a choice local minima are reached after convergence, we have decided to repeat the prediction for every new day ten times. In that way 10 potential predictions were produced. Then, in order to make a better choice, those with a value above 80% and below 20% of the average were discarded. The final accepted prediction was the average of the rest.

Fig. 6 depicts the prediction results for a 50 day period in the summer of 2021. As can be seen errors not larger than 5% were obtained. This is in accordance with our previous results and, of course, with our expectations.

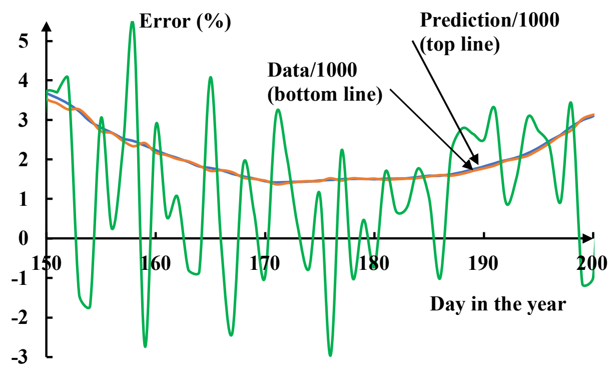

Fig. 6. Prediction results for a 50 day period in the summer of 2021.

V. Conclusion

Based on our 30 years long experience in implementation of ANN in various aspects of technological and social life, we have implemented ANNs for prediction of COVID-19. The results obtained are, in our opinion satisfactory and encouraging for further improvement. That means implementation of other structures as described in the bulk of the paper.

Acknowledgment

This work has been supported by The Ministry of Education, Science and Technological Development of the Republic of Serbia under the contract No. 451-03-68/2022-14/ 200102 of February, 04. 2022.

References

[1] https://www.worldometers.info

[2] J. Milojković, , V. Litovski, "One Month Ahead Prediction of Suburban Average Electricity Load", Proceedings of 2nd International Conference IcETRAN, Srebrno Jezero, Jun, 2015., ELI2.2, ISBN 978-86-80509-71-6.

[3] J. Milojković, D. Topisirović, M. Milić, M. Stanojević, "Short term local road traffic forecast using feed-forward and recurrent ANN", Facta Universitates, Working and Living Environmental Protection, Vol. 13, No 1, 2016, pp.1-12.

[4] C. Wang, P.W.Horby, F.G. Hayden, G.F. Gao, "A novel coronavirus outbreak of global health concern", The Lancet, Volume 395, Issue 10223,15–21 February 2020, Pages 470-473.

[5] Forecast of the outbreak of COVID-19 using artificial neural network: Case study Qatar, Spain, and Italy, 2021 Aug; 27:104484. doi: 10.1016/j.rinp.2021.104484. Epub 2021 Jun 21.

[6] https://covid19.who.int/

[7] P. Wang, J. A. Lu, Y. Jin, M. Zhu, L. Wang, S. Chen, "Statistical and network analysis of 1212 COVID-19 patients in Henan", China, Int. J. Infect. Dis. (2020). Published online 2020 Apr 24. doi: 10.1016/j.ijid.2020.04.051

[8] M. A. Turk, S. D. Landes, M. K. Formica, K. D. Goss, "Intellectual and developmental disability and COVID-19 case-fatality trends: TriNetX analysis", Disability and Health Journal, Volume 13, Issue 3, July 2020, 100942

[9] Y. Kuvvetly, M. Deveci, T. Paksoy, H. Garg, "Predictive analytics model for COVID-19 pandemic using artificial neural networks", Decision Analytics Journal, Volume 1,November 2021, 100007

[10] A. S. Ahmar, E. B. del Maj, "SutteARIMA: Short-term forecasting method, a case: Covid-19 and stock market in Spain", Science of the total environment, Vol. 729, No.10, August 2020, 138883.

[11] M. A. A. Al-qanes at all, "Optimization Method for Forecasting Confirmed Cases of COVID-19 in China", Journal of clinical Medicine, Vol. 9, no. 3, doi. 10.3390/jcm9030674.

[12] A. S., Mandel', "Method of Analogs in Prediction of Short Time Series: An Expert-statistical Approach", Automation and Remote Control, Vol. 65, No. 4, April 2004, pp. 634-641

[13] A.L.Bertozi, E.Franco, G.Mohler, D. Sledge, The challenges of modeling and forecasting the spread of COVID-19, PNAS, July 2, 2020, Vol.117 No.29, 16732-16738

[14] N. Balak at all., A simple mathematical tool to forecast COVID-19 cumulative case numbers, Clinical Epidemiology and Global Health, Vol. 12, October–December 2021, 100853

[15] Long-term forecasting of the COVID-19 epidemic, Dynamic Causal Modelling, UCL, UK, https://www.fil.ion.ucl.ac.uk/spm/covid-19/

[16] J. Milojković, and V. B. Litovski, "On the method development for electricity load forecasting", Proceedings of 1st International Conference on Electrical, Electronic and Computing Engineering, IcETRAN 2014, Vrnjačka Banja, Serbia, June 2 – 5, 2014, ISBN 978-86-80509-70-9

[17] J. Milojković, and V. B. Litovski, "Procedures of prediction of quantities of electronic computer waste", Tehnika (Elektrotehnika), Vol.56, No. 1, pp. E.7-E.16.(In Serbian), 2007.

[18] J. Milojković, and V. B. Litovski, "New procedures of prediction for sustainable development", 51th Conference of ETRAN, Herceg Novi, 04-08 Jun, 2007, Proc. on CD, EL1.8. (In Serbian).

[19] J. Milojković, and V. B. Litovski, "Short-term forecasting of electricity load using recurrent ANNs", 15th International Symposium On Power Electronics – Ee2009, Novi Sad, Serbia, ISSN Paper No. T1-1.7, 2009.

[20] J. Milojković, and V. B. Litovski, "Comparison of Some ANN Based Forecasting Methods Implemented on Short Time Series", 9th Symp. on Neural Network Applications in Electrical Eng., NEUREL-2008, pp. 179-179, Belgrade, 2008

[21] J. Milojković, and V. B. Litovski, "Prediction in Electronics based on limited information", Proc. of the 8th WSEAS Int. Conf. on Electronics, Hardware, Wireless And Optical Communications, EHAC'09, Cambridge, UK, pp. 33-38, February 2009

[22] M. Milić, J. Milojković, I. Marković, P. Nikolić, "Concurrent, Performance-Based Methodology for Increasing the Accuracy and Certainty of Short-Term Neural Prediction Systems", Computational Intelligence and Neuroscience, Vol. 2019, 1687-5265, doi:10.1155/2019/9323482, April, 2019.

[23] T. Masters, "Practical Neural Network Recipes in C++", *Academic Press,* San Diego, 1993

[24] Z. Zografski, "A novel machine learning algorithm and its use in modeling and simulation of dynamicalb systems", in Proc. of 5th Annual European Computer Conference, COMPEURO '91, Hamburg, Germany, 1991, pp. 860-864.

[25] E. B. Baum and D. Haussler, "What size net gives valid generalization", Neural Computing, 1989, Vol. 1, pp. 151-160.

[26] J. Milojković, and V. B. Litovski, "Short-term forecasting of electricity load using recurrent ANNs" 15th International Symposium On Power Electronics - Ee2009, Novi Sad, Paper No. T1-1.7, pp. 1-5, October 28th – 30th 2009

Equivalent Electromechanical Model of a Composite Ultrasonic Transducer

Igor Jovanović and Dragan Mančić

Abstract— **This paper presents an original one-dimensional model of a high-power composite ultrasonic transducer with a new structure. The equivalent circuit method is used for a model that can accurately depict the characteristics of the composite ultrasonic transducer and enable its efficient performance evaluation. The proposed model is verified by comparing the modeled dependencies of input electrical impedance vs. frequency with the experimental results. The equivalent circuit developed in this work can facilitate the design and analysis of complex composite transducer structures.**

Index Terms— **High-power ultrasound, Composite ultrasonic transducer, One-dimensional modeling.**

I. Introduction

The piezoelectric ultrasonic transducer is a device that converts desired electrical signals to ultrasonic waves. The applications of high-intensity ultrasonic waves are based on the adequate exploitation of the non-linear effects associated with high amplitudes, such as the radiation pressure, streaming, cavitation, dislocation in solids, etc. [1].

An ultrasonic transducer is a widely used high-power electromechanical transducer for ultrasonic cleaning, ultrasonic liquid processing, and ultrasonic sonochemistry. There are increasingly needed high-power ultrasonic radiators with large amounts of power radiating surfaces. The development of various power ultrasound applications requires ultrasonic transducers with more significant maximum vibration velocity, energy efficiency, and lower temperature rise [2]. Ultrasonic transducer, which consists of piezoceramic and metal rings, has a low resonant frequency (considering the size of the transducer) and a high-quality factor.

Recent research in the field of powerful ultrasound aims to optimize the design of ultrasonic transducers by numerical and analytical modeling methods and the use of precise devices for measuring vibration, mechanical displacement, and stress [3].

The finite element method (FEM) has been a commonly used numerical modeling method to analyze acoustic characteristics of ultrasonic transducers. As a representative work on the use of the FEM, Kagawa and Yambuchi used this method to assess the effect of dimension and material on the resonance frequency of an ultrasonic transducer [4]. In [5], the finite element technique is used for polymer characterization. Wang et al. used FEM to evaluate the output displacement directions of a composite transducer [6]. In addition, FEM is used to evaluate the effect of structural parameters on the output displacement of an ultrasonic transducer [7]. Lin et al. used FEM to evaluate the composite transducer's radial radiation acoustic field distribution [8].

The need for extensive computing resources and long analysis time constitutes the main disadvantage of the FEM [9]. The FEM typically requires a long analysis time and considerable computational resources despite its widespread usage. Therefore, there is a strong need for a more efficient method for analyzing the performance characteristics of the ultrasonic transducer with high accuracy.

The most widely used analytical modeling approach for ultrasonic transducers found in literature is an application of one-dimensional theory using equivalent electromechanical circuits [10]. The equivalent circuit is a method that can analyze the acoustic characteristics of transducers more simply and efficiently than the FEM [9]. It has been utilized to design and analyze various transducers [11].

In their simplest form, ultrasonic transducers are represented by one-dimensional models that represent networks with one electrical and two mechanical approaches. However, when the modeling considers the influence of other parameters (influence of bolt, electrodes, insulators, various electrical connections, prestress, loads, power, etc.) of the transducer, there is an increase in the number of electrical and mechanical approaches in the electromechanical equivalent circuit [10]. Additionally, in the [12], it has been confirmed that using equivalent electromechanical circuits is still possible to model more complex transducer constructions with reasonable accuracy.

Therefore, the composite transducer with a new structure, analyzed in this paper, is presented in the simplest form as a network with two electrical and two mechanical approaches.

II. Analytical One-Dimensional Modelling of Composite Transducer

A new structure of the composite transducer is shown in Fig. 1(a). The composite transducer contains a central mass (2) placed between the two active layers of the transducer ($PZT_{1,2}$ and $PZT_{3,4}$) and two metal endings (1 and 3) connected to the central mass by two central bolts.

Igor Jovanović is with the University of Niš, Faculty of Electronic Engineering, 14 Aleksandra Medvedeva, 18000 Niš, Serbia (e-mail: igor.jovanovic@elfak.ni.ac.rs), (https://orcid.org/ 0000-0001-7912-9154).

Dragan Mančić is with the University of Niš, Faculty of Electronic Engineering, 14 Aleksandra Medvedeva, 18000 Niš, Serbia (e-mail: dragan.mancic@elfak.ni.ac.rs).

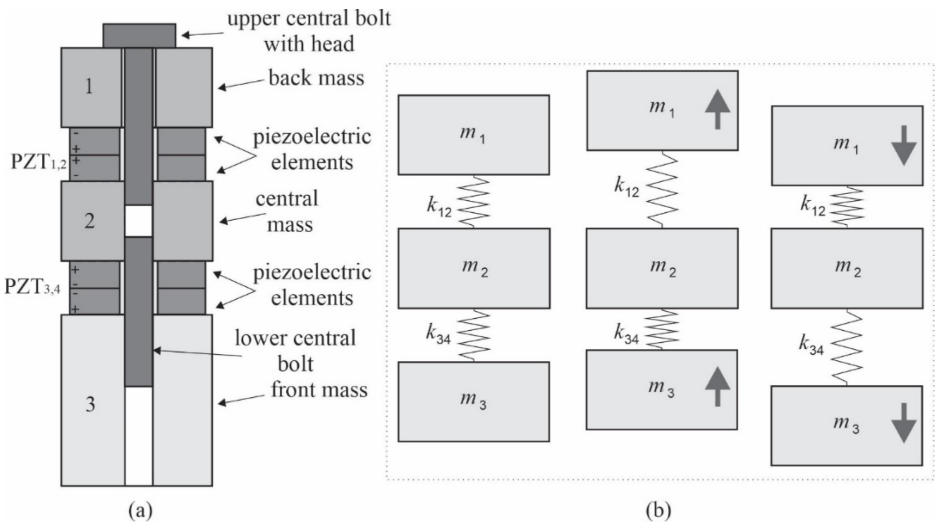

Fig. 1. The composite transducer with a new structure (a), represents of the simplest oscillatory structure of composite transducer (b).

Due to the mutually opposite polarization of the active piezoelectric elements connected to the same power supply, the masses in such a construction oscillates in the manner shown in Fig. 1(b). The three masses constituting this system are m_1, m_2, and m_3 (it is assumed in Fig. 1(b) that the masses are equal to each other), while k_{12} and k_{13} are the stiffness constant.

In its simplest form, the proposed composite transducer is a simple mechanical combination of two half-wave ultrasonic transducers with a sandwich structure that oscillates in the thickness direction (two Langevin-type transducers) [13].

Since the metal endings in the proposed structure are not of the same material, the composite transducer is not bidirectional. The proposed composite transducer has greater flexibility in operation than conventional transducers, which is reflected, among other things, in the possibility of independent excitation of the upper and lower active layer with different signals.

In this paper, modelling of the realized composite transducer with new structure, which represents a special unidirectional composite ultrasonic transducer, is performed. Prestressing the structure is achieved using two central bolts that are in contact with the central mass. The proposed model was adapted based on the structures of the composite transducer and shown as an equivalent electromechanical circuit shown in Fig. 2.

Elements of electromechanical circuits corresponding to isotropic and asymmetric metal parts made of different materials are calculated as:

$$Z_{i1} = jZ_{ci} tg \frac{k_i l_i}{2} \tag{1}$$

$$Z_{i2} = \frac{-jZ_{ci}}{\sin(k_i l_i)} \tag{2}$$

wherein $Z_{ci}=\rho_i v_i P_i$ and $k_i=\omega/v_i$ (for i=1, 2, and 3) are characteristic impedances and the corresponding wave numbers. ρ_i are densities, l_i and P_i are lengths and surface areas of the cross-sections, and v_i are the velocities of longitudinal ultrasonic waves propagation through the corresponding elements.

Elements of the circuit shown in Fig. 2 correspond to the piezoceramic rings in the upper active layer (PZT_{12}), and the piezoceramic rings in the lower active layer (PZT_{34}). These elements are determined as:

$$Z_{p1} = jZ_{cp} tg \frac{nk_p l_p}{2} \tag{3}$$

$$Z_{p2} = \frac{-jZ_{cp}}{\sin(nk_p l_p)} \tag{4}$$

wherein $Z_{cp}=\rho_p v_p P_p$ and $k_p=\omega/v_p$ are characteristic impedances and corresponding wave numbers, respectively. ρ_p, l_p, P_p are densities, lengths, and surface areas of the piezoceramic cross-sections, v_p are velocities of longitudinal ultrasonic waves propagation, respectively. The input electric voltages and currents are marked as V, I_{12}, and I_{34}.

The piezoceramic models consist of capacitance $C_0=n\varepsilon_{33}^S P_p/l_p$, and ideal transformers with transmission ratios $N=h_{33}C_0/n$, wherein n is the number of piezoceramic rings per active layer (n=2). The piezoelectric properties of the transducer active layers are represented by the piezoelectric constant h_{33} and the relative dielectric constant of the pressed ceramic ε_{33}^S.

Piezoceramic rings are mechanically connected in series with central mass, back and front endings. Back and front endings are closed with acoustic impedances Z_R and Z_E, which are in this case negligible because experimental measurements were conducted with unloaded transducers oscillating in the air.

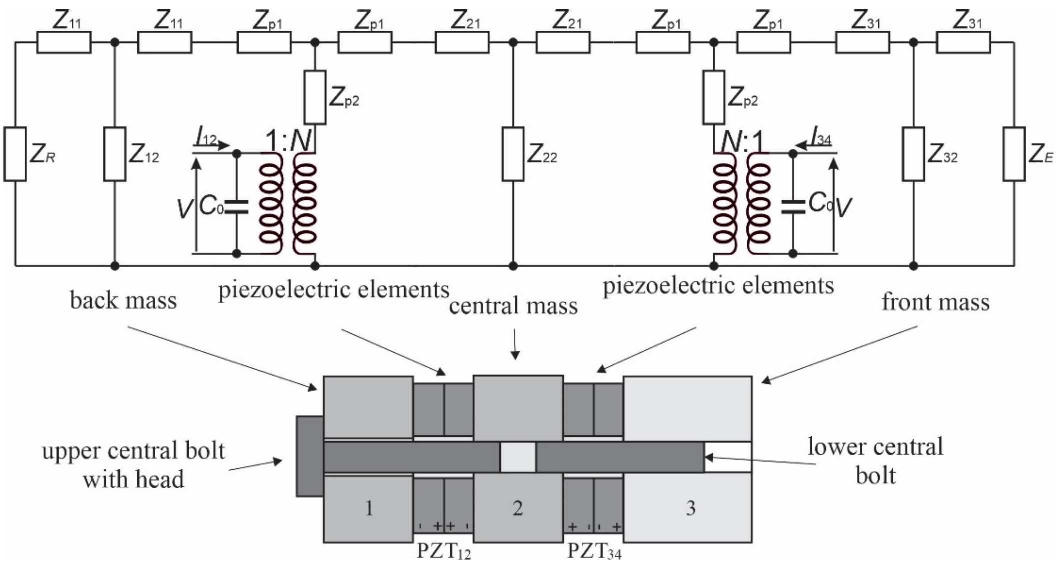

Fig. 2. One-dimensional model of the composite transducer.

Based on Eqs. (1-4) it is obvious that the transducer frequency response depends on the material characteristics of its constituting parts and their geometric dimensions.

In the proposed transducer model, it is assumed that the circuit elements are ideal, i.e. they do not have losses. Losses can be included if piezoelectric constants and constants of elasticity of the transducer metal parts are in the form of complex numbers, in which the imaginary parts represent losses.

III. Simulation and Experimental Results

Table 1 shows the dimensions of the individual composite transducer. Dimensions of the exciting piezoceramic rings are Ø38/Ø13/6.35 mm, and rings are made of PZT8 piezoceramic equivalent material [14]. L_i is the length, a_i and b_i are the outer and inner diameters of the corresponding i-th element. The front ending is made of a dural, while the back ending and the central mass are made of steel with the standard material properties. The electrical impedance measurements are conducted using a Microtest 6366 Precision LCR Meter.

TABLE I
DIMENSIONS OF COMPOSITE TRANSDUCER USED IN EXPERIMENTAL ANALYSIS

Dimension [mm]	Composite transducer
L_1=L_2	11
L_3	37
a_1=a_2=a_3	40
b_1=b_3	9
b_2	8

There is a similarity between the modeled and experimental dependences, as shown in Fig. 2. Since it is a composite transducer with a larger ratio of length and transverse dimensions, the proposed one-dimensional model gives satisfactory results during transducer analysis in the first resonant mode.

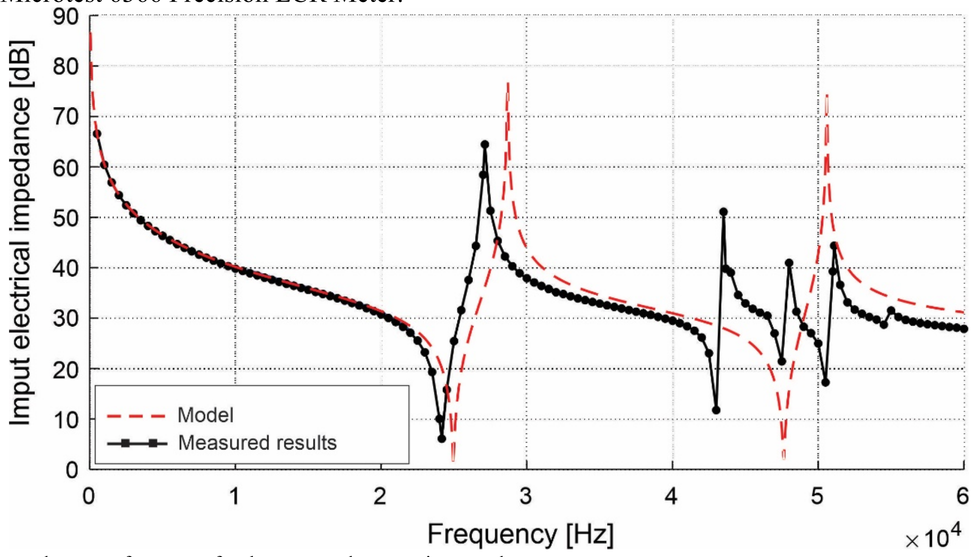

Fig. 3. Input electrical impedance vs. frequency for the proposed composite transducer

The measured resonant frequency of the fundamental resonant mode is 24.16 kHz. The calculated resonant frequency using the proposed model is 24.95 kHz, where the error made by the one-dimensional model in determining this resonant frequency is 3.27%. When it comes to the antiresonant frequency, the measured value is 27.12 kHz, while the model calculated 28.7 kHz, i.e., the error made by the model is 5.83%. The proposed model can predict the general shape of the second resonant mode but with significant error. The measured resonant frequency of the second resonant mode is 43 kHz, while the resonant frequency obtained by the model is 47.65 kHz (the error is 10.81%).

This model allows only the thickness resonant modes to be predicted and, therefore, does not consider the inevitable radial resonant modes. One-dimensional models are generally not suitable for determining resonant frequencies of thickness oscillations that are close to resonant frequencies of radial oscillations. In the case shown, when the model does not predict the third and fourth modes, the calculated resonant frequencies for the first two modes are always higher than the measured ones. If a model that considers both the third and fourth modes were used, the first two modes would be moved to lower frequencies.

IV. Conclusion

In this study, an equivalent circuit was developed for accurate analysis of the acoustic characteristics of an ultrasonic transducer over a wide frequency range.

Eqs. (1-4) confirm that the frequency characteristics of transducers in one-dimensional theory depend on the material characteristics of the components of composite transducers and their geometric dimensions.

In practice, one-dimensional modeling is most often used due to the great flexibility and efficient implementation of the model. The flexibility and efficiency of one-dimensional models come to the fore in the analysis of transducers operation, which includes a large number of parameters.

The proposed one-dimensional model of composite transducer does not include mechanical and electrical losses in the material. However, losses can be analyzed if the piezoelectric constants and the elastic constants of the metal parts of the converter are represented in complex numbers, where their imaginary parts represent losses.

Acknowledgment

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, contract no. 451-03-68/2022-14/200102.

References

[1] Y. Yao, Y. Pan, S. Liu, "Power ultrasound and its applications: A state-of-the-art review," *Ultrasonics - Sonochemistry*, vol. 62, 104722, 2020.

[2] X. Lu, J. Hu, H. Peng, Y. Wang, "A new topological structure for the Langevin-type ultrasonic transducer," *Ultrasonics*, vol. 75, pp. 1–8, 2017.

[3] D. Chen, L. Wang, X. Luo, C. Fei, D. Li, G. Shan, Y. Yang, "Recent Development and Perspectives of Optimization Design Methods for Piezoelectric Ultrasonic Transducers," *Micromachines*, vol. 12, no. 7, 779. 2021.

[4] Y. Kagawa, T. Yambuchi, "Finite element simulation of a composite piezoelectric ultrasonic transducer," *IEEE Trans. Sonics Ultrasonics*, vol. 26, no. 2, pp. 81–88, 1979.

[5] F. Wolf, T. Lahmer, L. Bahr, A. Hauck, A. Sutor, R. Lerch, M. Kaltenbacher, "Finite element modeling of ultrasonic transducer by utilizing an inverse scheme for the determination of its material parameters," 2008 IEEE International Ultrasonics Symposium, Beijing, China, 2-5 November, 2008.

[6] L. Wang, J. A. Wang, J. M. Jin, L. Yang, S.W. Wu, C. Zhou, "Theoretical modelling, verification, and application study on a novel bending-bending coupled piezoelectric ultrasonic transducer," *Mechanical Systems and Signal Processing*, vol. 168, 108644, 2022.

[7] F. Wang, H. Zhang, C. Liang, Y. Tian, X. Zhao and D. Zhang, "Design of High-Frequency Ultrasonic Transducers With Flexure Decoupling Flanges for Thermosonic Bonding," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2304-2312, 2016.

[8] S. Lin, L. Xu,W. Hu, "A new type of high-power composite ultrasonic transducer," *Journal of Sound and Vibration*, vol. 330, pp. 1419–1431, 2011.

[9] H. Shim, Y. Roh, "Development of an Equivalent Circuit of a Cymbal Transducer," *IEEE Sensors Journal*, vol. 21, no. 12, pp. 13146-13155, 2021.

[10] I. Jovanović, U. Jovanović, D. Mančić, "General One-Dimensional Model of a New Composite Ultrasonic Transducer", Proceedings of the 7th Small Systems Simulation Symposium 2018, Niš, Serbia, pp. 50-54, 12-14 February 2018.

[11] D. Mančić, I. Jovanović, M. Radmanović, Z. Petrušić, "Comparison of one-dimensional models of ultrasonic sandwich transducers" – in Serbian, Proceedings of the XXII Noise and Vibration, Niš, Serbia, pp. 119-127, 20-22. October 2010.

[12] I. Jovanović, D. Mančić, U. Jovanović, M. Prokić, "A 3D model of new composite ultrasonic transducer", *Journal of Computational Electronics*, vol.16, no. 3, pp.977-986, 2017.

[13] P. Langevin, French Patent Nos: 502913 (29.5.1920); 505703 (5.8.1920); 575435 (30.7.1924).

[14] *Properties of Piezoelectricity Ceramics*, Technical Publication TP-226, Morgan Electro Ceramics.

Hardware Realization of Nearest Neighbour Search Algorithm over an In-Memory Pre-Stored k-d Tree

Aleksandar Z. Kondić, *Student Member, IEEE*, and Vladimir M. Milovanović, *Senior Member, IEEE*

Abstract—**Nearest neighbour search is a fundamental statistical classification algorithm with widespread use in artificial intelligence (AI) sub-fields such as machine learning, computer vision, and robotics. Considering the shift in host platforms running AI algorithms from general-purpose computers to specialized hardware implementations, a parameterizable design generator of special purpose hardware instances that perform nearest neighbour search is proposed, captured inside Chisel hardware construction language, and validated on an FPGA platform. Based on an algorithm of nearest neighbour search that traverses a k-dimensional tree pre-stored inside read-only memory (ROM), the generator provides parameters for configuring the structure and volume of the tree and the points stored within it.**

Index Terms—**Nearest neighbour search, hardware implementation, Chisel hardware construction language, k-dimensional tree.**

I. Introduction

Nearest neighbour search is an algorithm which, for a given input point, finds a point closest to it among a set of points. It is useful for solving classification problems, which are especially prevalent in artificial intelligence (AI) subfields such as machine learning, computer vision [1], and robotics [2].

With artificial intelligence algorithms being increasingly shifted from general-purpose computers to dedicated hardware instances as a consequence of the need for increased computational power [3], various hardware implementations of classic AI algorithms targeting different platforms have appeared. The Nearest Neighbour Search (NNS), along with its variants, the Approximate Nearest Neighbour (ANN) and k-Nearest Neighbours (k-NN) algorithms, are no exceptions.

Considering field programmable gate arrays (FPGAs) as a hardware implementation platform of choice, there are various incarnations of the above mentioned algorithms. They are usually described and implemented either in the form of pure register-transfer level (RTL) [4], [5], high-level synthesis (HLS) [6], [7], or Open Computing Language (OpenCL) [8], [9] code. An alternative to these approaches is to write the behavioral code in an RTL-like form but utilizing a higher level hardware design language instead. One such language is Chisel [10], which is embedded in the Scala programming language, enabling its users to write RTL instance generators while providing benefits of both functional and object-oriented programming paradigms. This paper proposes an implementation of the nearest neighbour search algorithm in Chisel using an agile [11] digital design methodology.

Aleksandar Z. Kondić was with the Faculty of Engineering, University of Kragujevac, Sestre Janjić 6, Kragujevac, Serbia (e-mail: konda@uni.kg.ac.rs).
Vladimir M. Milovanović is with the Department of Electrical Engineering, Faculty of Engineering, University of Kragujevac, Sestre Janjić 6, 34000 Kragujevac, Serbia (e-mail: vlada@kg.ac.rs).

An effective implementation of the nearest neighbour search algorithm should presumably work for a large number of predefined points as potential output points for a given input point. For such an implementation to be efficient in terms of resource utilization for a target hardware platform such as FPGA, the points need to be stored inside a memory module. This naturally implies that the digital logic may have access only to a limited number of pre-stored points per clock cycle. Therefore, it is desirable to minimize the number of memory accesses for a given input point while obtaining the correct solution.

This is the same problem that a purely software implementation of a nearest neighbour search algorithm on a processor would have. To minimize the amount of time needed to process an input point, an efficient algorithm with a desirable run-time complexity needs to be chosen. While the simplest solution would be to run an exhaustive search of the entire memory containing pre-defined points to find a point with the minimal distance from the input point—yielding a linear run-time complexity—more efficient algorithms exist.

Similar problems were encountered in the field of computer graphics. In order to ensure the rendering of a scene in a timely manner, it was necessary to retrieve relevant spatial data of the scene efficiently. A technique named *binary space partitioning* (BSP) was developed to solve this problem, mainly implemented through a tree data structure [12]. The technique entails recursively subdividing space into two parts along a hyperplane. When a given point or polygon is queried, the search is performed only in the sub-spaces where it could possibly be located, thus reducing the search domain.

Space partitioning is a general method of subdividing space in a defined manner until a certain condition is satisfied. There are multiple implementations of this method in the form of different tree structures with specific criteria on how a space is divided into sub-spaces and under which conditions. Examples of some tree structures that perform space partitioning are k-d trees, quadtrees, and octrees. Concerning the nearest neighbour search problem, some of the appropriate data structures that can be used are R-trees and k-dimensional trees.

The principal data structure driving this particular implementation of the nearest neighbour search algorithm is the k-dimensional tree, or k-d tree for short. A k-d tree is essentially a binary search tree that contains multi-dimensional points and is traversed based on the value of one of the coordinates of the input point at each node in the tree.

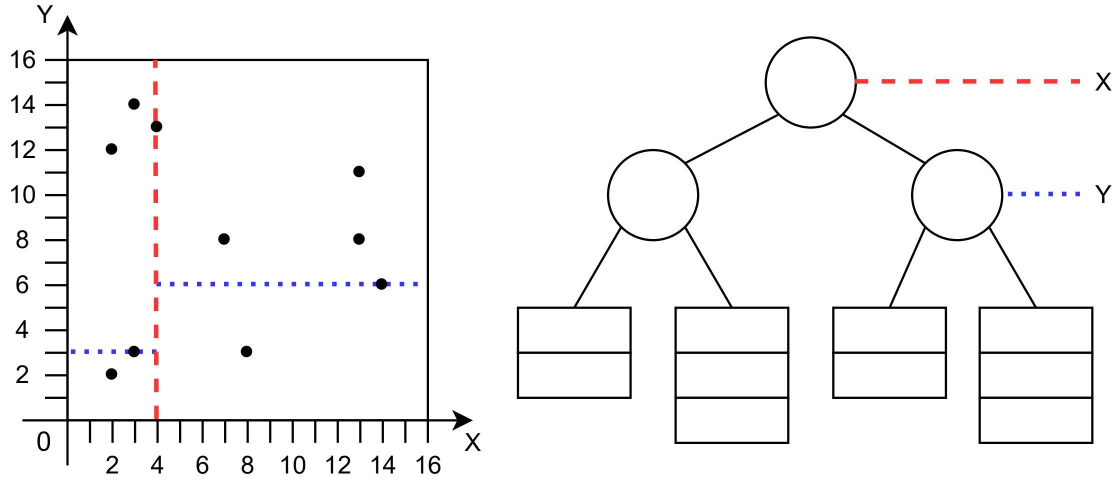

Fig. 1. An illustrative example of a k-dimensional tree (with $k = 2$ to simplify the drawing) and the two-dimensional space partitioning it performs.

Each node of the k-d tree contains a value by which the hyperspace it belongs to is split into two. The dimension in which the split is performed corresponds to the node's depth in the tree, which repetitively cycles from the last dimension to the first when the depth of the node becomes greater than the dimensionality of the points stored inside the tree. All child points that have the value of the coordinate in the corresponding dimension less than the node's stored value are part of the left sub-tree, while the child points with the corresponding coordinate's value greater than the value in the node are part of the right sub-tree. In the case of a child point having an equal corresponding coordinate value to the value of the node—due to the nature of the nearest neighbour search algorithm—it may belong to either of the sub-trees.

The average run-time complexity of the nearest neighbour search algorithm over a k-dimensional tree is $\mathcal{O}(m + \log_2 n)$, where n is the number of nodes in the tree and m is the average number of points contained in a leaf node.

II. A k-d Tree-Based Hardware Implementation

The primary purpose and the use scenario of the proposed implementation is to execute the nearest neighbour search algorithm over a k-dimensional tree. The tree structure, along with the points it contains is assumed to be constructed and stored beforehand inside some form of a read-only memory (ROM). In the case of an FPGA platform the ROM is in the form of a single-port block RAM and mimics the static RAM.

This implementation uses a variant of the k-dimensional tree in which the number of nodes in the tree is not necessarily equal to the number of points. The points are, after a proper traversal through the k-d tree, stored in the leaf nodes. A leaf node may contain more than one point. An example k-dimensional tree of this kind is shown in Fig. 1, along with an illustration of how the tree partitions a two-dimensional space (but in general it can be an arbitrary k-dimensional hyperspace).

The nearest neighbour search algorithm finds the closest point to the query point by first performing a traversal of the k-d tree until reaching a leaf node. When visiting a node, the value of the query point's coordinate in the dimension corresponding to the node's depth is checked against the value in the node. If the value is smaller, traversal proceeds to the left sub-tree. Otherwise, traversal proceeds to the right sub-tree. When reaching a leaf node, all of the points in the leaf node are checked, calculating the distances between them and the query point. The current closest point, along with its distance to the query point, are stored inside dedicated registers which are updated when a closer point is found.

After exhausting all of the points in a leaf node, the search algorithm traverses backwards, that is up the tree and checks if the hypersphere around the current closest point with the radius equal to its distance from the query point intersects the node's splitting hyperplane. If so, a closer point to the query point may exist on the other side of the splitting hyperplane, so the search algorithm proceeds by traversing down the sub-tree contained in the node's unvisited child, until reaching a leaf node again. This process is repeated until the algorithm terminates when it is guaranteed to yield a point stored within the k-dimensional tree with the minimal distance from the query point.

For the purposes of this work, the structure of the k-dimensional tree and the points it contains are stored in two separate memories (or two non-overlapping memory segments). The memory used to store information about the points contains coordinates of each point. The points inside this particular memory (segment) are arranged in such a way that the points belonging to the same leaf node of the k-dimensional tree occupy consecutive memory locations.

Memory containing the tree structure stores the properties of each node. The following properties are stored: an indicator bit of whether the node is a leaf node, the discriminating value stored inside the node for tree traversal (valid only for non-leaf nodes), the starting address in the points ROM and the

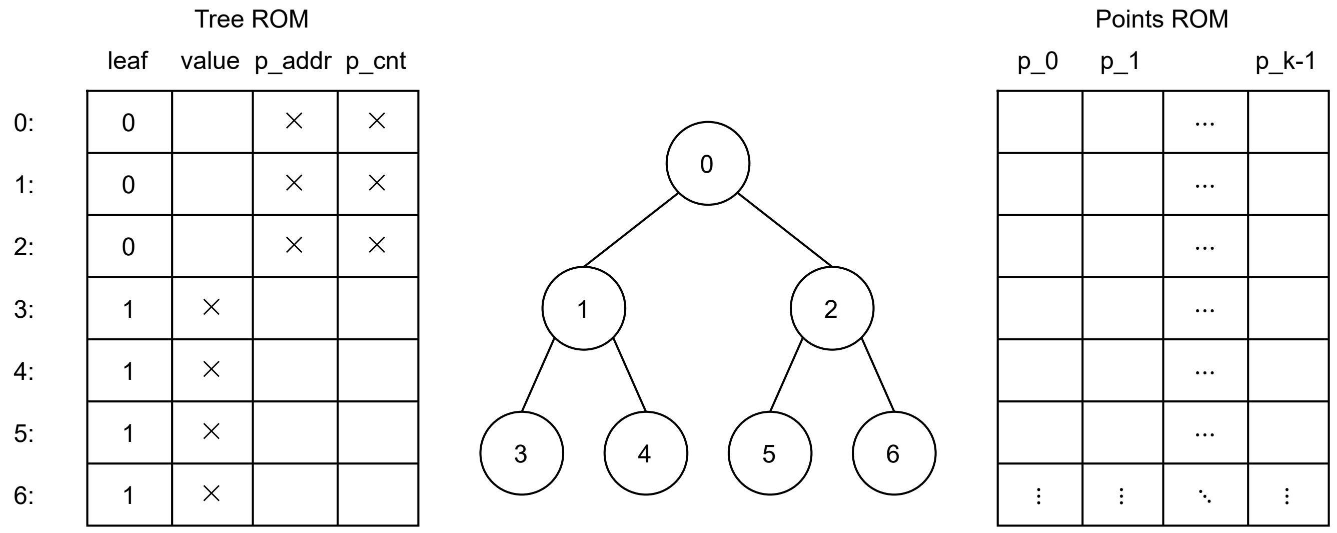

Fig. 2. Memory layout of a k-dimensional tree showing the associated Tree ROM and Points ROM structures over which the NNS realization operates.

number of points contained in the node (valid only for leaf nodes). A node at location n in the tree ROM has its left and right children at locations $2n + 1$ and $2n + 2$, respectively. With this it is assumed that the stored k-d tree is balanced. It is possible to construct a balanced k-d tree from an arbitrary set of points as long as the points with the same coordinate value as the discriminating value in the node may be stored in either of the node's child sub-trees. In particular use cases of interest this discriminating value is actually the median value of relevant coordinates of the points being considered during k-d tree construction. The described memory layout is illustrated in Fig. 2.

The tree traversal algorithm is recursive. Traversal is performed in a depth-first manner, that is similar to the depth-first search algorithm (DFS), which is also recursive. The DFS algorithm, starting at the root of the tree, visits its child nodes in a pre-defined order. When visiting one of the child nodes, another instance of the DFS algorithm is started on the node, running more instances of the DFS algorithm on its children if it has any. Once an instance of the DFS algorithm for one child node terminates, the same process is repeated for the other. Therefore, by the time DFS starts visiting the root node's second child, the entirety of the sub-tree rooted in its first child will have already been explored.

An example of the order of traversal of binary tree nodes in the depth-first search algorithm is shown in Fig. 3. Non-leaf tree nodes are each visited a total of three times in order to visit the subtree rooted in their second child after visiting the first, and to potentially traverse back up to the parent node.

Each non-leaf node's left child is first explored, followed by the right child. The primary characteristic of DFS is that after visiting the leaves, it traverses back up the tree in order to traverse down unvisited sub-trees, repeating this process until the entire tree is explored.

A k-d tree traversal is essentially a variation on DFS tree traversal. The difference with k-d tree traversal is that the order of the children visited depends on the query point for which the closest point is to be found. The left child is first visited if the query point is on the "left" side of the splitting hyperplane represented by the node, otherwise the right child is first visited. Also, if the first child node's closest point is at a distance shorter than or equal to the distance of the query point from the splitting hyperplane, the second child is not explored. Unlike depth-first search, with k-d tree search the entire tree may not necessarily be explored.

A tree traversal over an example k-d tree is illustrated in Fig. 4. The query point for which to find the closest point is $(5, 3)$. In this example, during the traversal three out of the four leaf nodes were visited. The metric used to calculate the distance between two points (or between the query point and a splitting hyperplane) is the squared Euclidean distance.

Software implementations of recursive algorithms may make use of recursive function calls, which are realized on a call stack, or allocate a stack structure specifically to store their data and implement the algorithm as a non-recursive function. In this case only the second option is viable, so the stack data structure is actually implemented as an array of registers. A separate dedicated register is used to store a pointer that keeps track of the position of the top of the stack in the array.

III. Design Generator of the k-d Tree-Based NNS

The previously described accelerator has been implemented as a parameterized RTL design generator in Chisel 3 hardware construction language. The generator has been extensively tested by following standard Chisel verification and implementation paths for FPGA design workflows. As a hardware library it is freely available for public use [13]. The next few paragraphs are elaborating on different generator parameters, as well as modes of operation of the module.

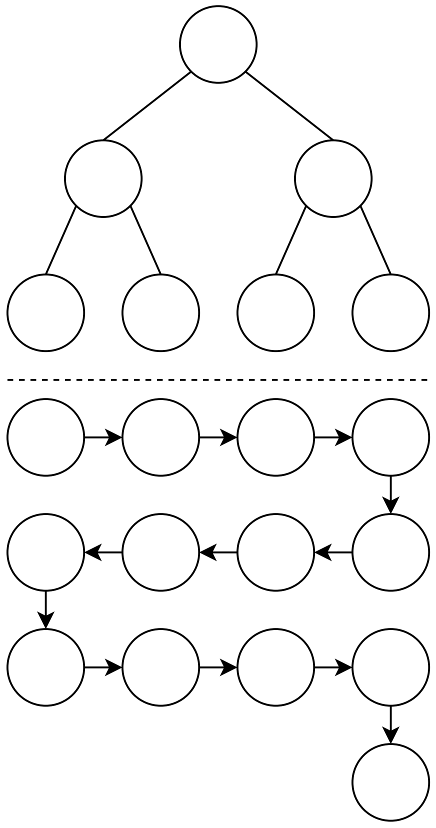

Fig. 3. An example of the order of nodes traversed in a binary tree using depth-first search.

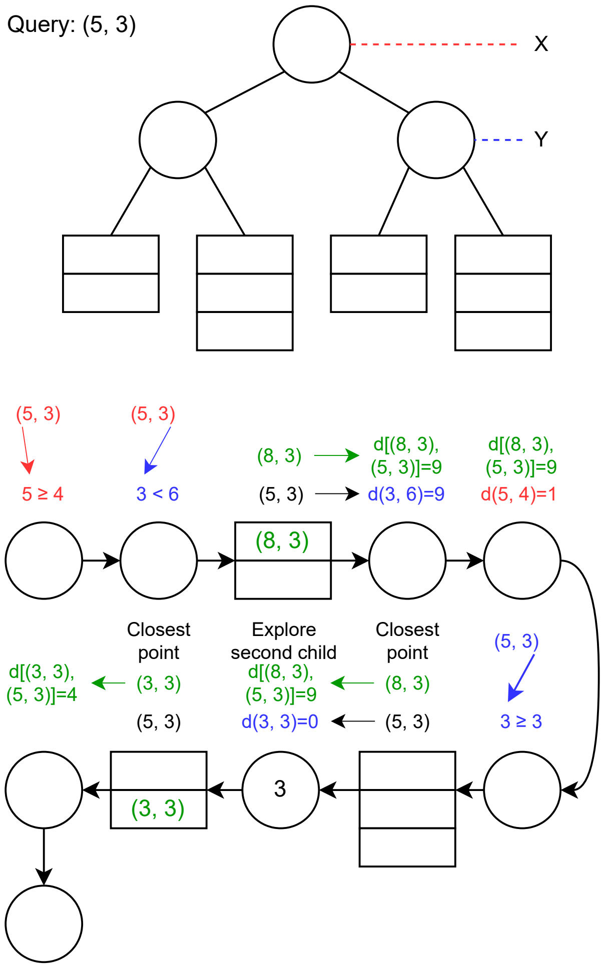

Fig. 4. An example of the order of nodes traversed in a k-d tree when finding the closest point to the query point $(5, 3)$.

A. Generator Parameters

Parameterizable properties of the design pertain mainly to the structure of the k-dimensional tree itself and its points.

The values of point coordinates are signed integers with a specified bit width, which is one of the parameters of the design generator. Another generator parameter is the number of dimensions of each point. The total size of the points ROM is inferred from the bit width of its unsigned integer addresses, which is specified as a yet another generator parameter. These three parameters make up the structure of the points ROM.

Concerning the structure of the tree ROM, each location contains one bit indicating whether the node is a leaf, a signed integer representing the discriminator value of a node (in our use cases referred to as the *median*), and two unsigned integers representing the location and count of points inside the points ROM. The bit width of the discriminator is the same as the bit width of the points' individual coordinates, while the bit width of the location and count of points is the same as the bit width of the addresses in the points ROM. The size of the tree ROM, along with the bit width of its addresses is inferred from a generator parameter specifying the maximum depth of the tree. The number of nodes in the tree may be arbitrary though, as the nearest neighbour search algorithm assumes that nodes marked as leaves in the tree ROM do not have children.

B. Modes of Operation

To keep track of the nodes visited for the purposes of traversing back up the tree after visiting a leaf node, a stack structure is implemented as an array of registers. Three distinct values are pushed onto the top of the stack to aid with the execution of k-d tree search:

- *Address of the node in tree ROM* – this is the main piece of data used to keep track which tree node to visit next.
- *The child of the node to visit next* – a single bit that determines whether to visit the first or the second child of the node during tree traversal. A value of 0 corresponds to visiting the first child, while a value of 1 corresponds to visiting the second child. When the value is 1, first the distance of the current closest point to the query point is compared to the distance of the query point to the splitting hyperplane of the node (the median value of the node in this case). If the distance of the closest point to the query point is not greater, the second child is not visited.
  Since after visiting the second child of a node there is nothing left to process, the current node visited is popped from the stack before the second child node is pushed

onto the stack. More precisely, the value at the top of the stack, which currently contains data about the current node being visited, is simply replaced with the data of its second child. In case the second child node is not to be visited, the current node is popped from the stack.

- *The depth of the node in the tree* – This information is used for calculating the distance of the current closest point from the splitting hyperplane. The value of the depth directly maps to which coordinate of the current closest point to compare to the stored node median and is also used to determine the coordinate of the query point to compare to the median value during tree traversal.
  While the depth of the node can be calculated from its index (memory address) in the tree ROM, it is simpler to just push onto the stack the current depth value of the node incremented by one when pushing child nodes. The depth value does not exceed the dimensionality of the points in the tree as it cycles between 0 and $k-1$.

Since the first node to process when a new query point is given is always the root of the tree (address 0 in tree ROM), whose depth is 0, and the next node to process is always its first child, the stack is always initialized to contain these values as the sole element of the stack before processing a new point.

At each clock cycle, the values at the top of the stack are retrieved, which mostly determine the mode of operation of the module. The relevant modes of operation are as follows:

- *Leaf processing* – this mode is active when the indicator for whether the current node is a leaf has the value 1. The values at the top of the stack are not used in this case. At each clock cycle, a counter indicating how many points were visited in the points ROM is incremented until reaching the value in tree ROM that indicates how many points a leaf node has. After that, the counter is reset to 0 and the current node is popped off the stack.
  The value of the counter is added to the starting address of the node's points in the points ROM, yielding an address of each point to be retrieved from the points ROM. The distance of each point is compared to the current minimal distance from the query point. In case it is smaller, both registers containing the closest point and its distance from the query point are updated accordingly.
  Since there is a delay of a few clock cycles due to memory access in the points ROM and distance calculations using registers to decrease the length of logic paths, once the points counter is set to 0, a "delay" counter is also initialized to 2. Each clock cycle the value of this counter is decremented until it reaches 0, regardless of the mode of operation. The module may not produce a valid result on its output while the value of this counter is greater than 0, even if there are no remaining nodes left in the tree to process for a given query point.
- *Tree traversal, first child node being next* – This mode is active when the current node is not a leaf (explained above) and the value of the child indicator at the top of the stack is 0. The appropriate query point coordinate is compared to the median value of the node to determine which of the left and right children is the first child to be visited. After that, the corresponding first child node is pushed onto the stack.
- *Tree traversal, second child node potentially being next* – This mode is active when the current node is not a leaf and the value of the child indicator is 1. The stored current minimal distance from the query point is compared to the distance of the query point's appropriate coordinate from the median value of the node. If the current minimal distance is not greater, the second child of the node will not be visited, and the current node is popped off the stack. Otherwise, values at the top of the stack are replaced with the values corresponding to the second child node.
  Since the median distance calculation also uses a register in order to decrease the length of logic paths, the result of this calculation is available in the next clock cycle. Therefore, a special one-bit register is set to signify that this mode of operation is still in progress. In the next clock cycle this register is reset, and the rest of the operations are performed as described.
- *Final phase of the algorithm* – active when the node stack is empty. While the value of the previously described "delay" counter is greater than 0, no additional operations are performed. Once the value of the counter reaches 0, the output valid signal of the module is set to 1, while the output point is simply a set of wires connected to the register storing the current closest point to the input point. The initial values of the stack are pushed onto the empty stack to prepare the processing of the next input point.

The block diagram in Fig. 5 depicts the generator's design. The inputs and outputs of this design adhere to the Ready/Valid handshaking protocol. Apart from the *ready* and *valid* signals, both the input and the output consist of a single k-dimensional point represented as a series of k signed integers depicting their respective coordinate values. The dimensionality of these points, along with the structure of the point and tree ROMs, depend on the generator's parameters.

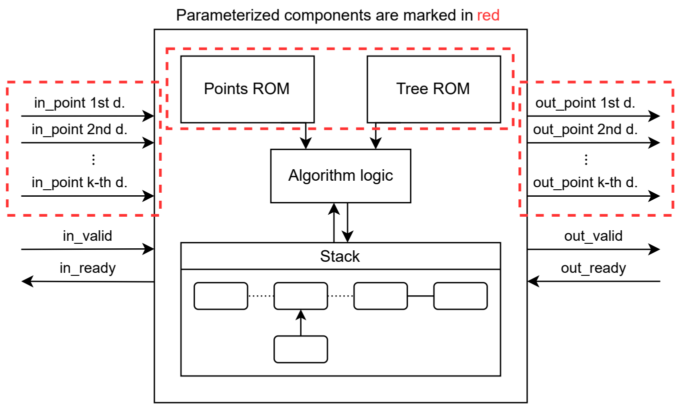

Fig. 5. Interface of the implemented Chisel design showing its input, output, and internals.

IV. Implementation and Testing Results

Testing of the generator is also performed using testing facilities provided by Chisel, i.e. *ChiselTest*. All generator parameters are randomized during testing, and the ROMs are also populated by appropriate randomly generated k-dimensional trees. The output of generated instances for random inputs is compared against the output of a k-d tree golden model written in Scala. The distances of the outputs of Chisel instances are compared with the distances of the outputs produced by the respective Scala golden model class instances.

The Scala golden model of the k-dimensional tree has also undergone rigorous testing. A list of random points is generated after selecting a random number of dimensions for the points. From the list of points and the desired number of tree nodes an instance of the golden model class is created.

This "golden model" instance is then supplied with random points as input. The output point's distance from the input point is compared to the distance of the closest point to the input point from the list of points in the tree, which is obtained by applying a simple brute-force exhaustive search algorithm. Due to the order of nodes and points traversed not being the same for the k-d tree model and the brute-force algorithm, only distances of the respective closest points to the input point are compared.

For additional testing and real in-hardware validation, various instances obtained from the design generator have been synthesized and implemented onto a commercially available FPGA development board. The board in question is Digilent's Arty A7 with Xilinx's Artix-7 FPGA family. All instances have been synthesized for a 100 MHz target clock frequency.

Resource utilization for the different generated instances is shown in Table I. Slice LUT utilization is most influenced by the bit width of the coordinates and k, the dimensionality of the points. A more minor effect on slice LUT utilization is exerted by the sizes of the point and tree ROMs. The number of slice registers seems to be mostly influenced by the bit width of the coordinates, followed by the dimensinoality of the points. Number of dimensions k has an influence on both Block RAM Tile and DSP multiplier counts, although the greatest influence on the number of DSP multipliers is exerted by point coordinate data bit width.

TABLE I
FPGA Resource Utilization for Generated Design Instances

Generator Instance Parameters				FPGA Resources			
Data Width	Nodes	Points	k	Slice LUTs	Slice Regs	BRAM Tiles	DSP muls
8 bits	31	100	3	728	272	1	3
16 bits	31	100	3	383	288	1.5	7
24 bits	31	100	3	528	444	1.5	11
32 bits	31	100	3	706	412	1	19
16 bits	7	100	3	334	287	1.5	7
16 bits	15	100	3	328	288	1.5	7
16 bits	63	100	3	339	288	1.5	7
16 bits	31	50	3	326	271	1.5	7
16 bits	31	200	3	372	311	1.5	6
16 bits	31	100	2	339	232	1	6
16 bits	31	100	4	381	337	1.5	8
16 bits	31	100	5	453	393	2	9

V. Conclusion

An approach to implementing a nearest neighbour search algorithm on an FPGA hardware platform has been explored. One of the key characteristics in this approach is in using a pre-stored k-dimensional tree to perform nearest neighbour search operations. Another is in using the Chisel hardware design language to create a generator of instances that can accommodate k-d trees with different structure parameters.

A variety of instances have undergone testing and additional verification by implementing them on a commercial FPGA development board. Apart from testing and verification, some consideration has also been given to their utilization of resources. This paper proves on an example case of the nearest neighbour search algorithm that parameterizable design generators can be used to produce instances of AI and machine learning hardware modules as an alternative to using CPU-based implementations.

References

[1] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[2] A. Bewley and B. Upcroft, "Advantages of exploiting projection structure for segmenting dense 3d point clouds," in *Australian Conference on Robotics and Automation*, vol. 2, 2013.

[3] M. A. Talib, S. Majzoub, Q. Nasir, and D. Jamal, "A systematic literature review on hardware implementation of artificial intelligence algorithms," *The Journal of Supercomputing*, vol. 77, no. 2, pp. 1897–1938, 2021.

[4] M. A. Mohsin and D. G. Perera, "An fpga-based hardware accelerator for k-nearest neighbor classification for machine learning on mobile devices," in *Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*, 2018, pp. 1–7.

[5] T. Ito, Y. Itotani, S. Wakabayashi, S. Nagayama, and M. Inagi, "A nearest neighbor search engine using distance-based hashing," in *2018 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2018, pp. 150–157.

[6] Z.-H. Li, J.-F. Jin, X.-G. Zhou, and Z.-H. Feng, "K-nearest neighbor algorithm implementation on fpga using high level synthesis," in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. IEEE, 2016, pp. 600–602.

[7] A. Lu, Z. Fang, N. Farahpour, and L. Shannon, "Chip-knn: A configurable and high-performance k-nearest neighbors accelerator on cloud fpgas," in *2020 International Conference on Field-Programmable Technology (ICFPT)*. IEEE, 2020, pp. 139–147.

[8] J. Zhang, S. Khoram, and J. Li, "Efficient large-scale approximate nearest neighbor search on opencl fpga," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4924–4932.

[9] F. B. Muslim, A. Demian, L. Ma, L. Lavagno, and A. Qamar, "Energy-efficient fpga implementation of the k-nearest neighbors algorithm using opencl." in *FedCSIS (Position Papers)*, 2016, pp. 141–145.

[10] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: Constructing hardware in a scala embedded language," in *DAC Design automation conference 2012*. IEEE, 2012, pp. 1212–1221.

[11] Y. Lee, A. Waterman, H. Cook, B. Zimmer, B. Keller, A. Puggelli, J. Kwak, R. Jevtic, S. Bailey, M. Blagojevic *et al.*, "An agile approach to building risc-v microprocessors," *ieee Micro*, vol. 36, no. 2, pp. 8–20, 2016.

[12] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," in *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, 1980, pp. 124–133.

[13] A. Kondić and V. Milovanović, "Hardware realization of nearest neighbour search algorithm over an in-memory pre-stored k-d tree," www.github.com/milovanovic/nns, accessed: April 15, 2022.