IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 July 2024, accepted 25 July 2024, date of publication 1 August 2024, date of current version 12 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3436564

RESEARCH ARTICLE

Ethernet–USB Bridge Application-Specific Integrated Circuit Incorporating the User Datagram Protocol and Address Resolution Protocol

GUO-MING SUNG, (Member, IEEE), ZI-YU LI, AND CHIH-PING YU
Department of Electrical Engineering, National Taipei University of Technology, Taipei City 10608, Taiwan

Corresponding author: Guo-Ming Sung (gmsung@ntut.edu.tw)

This work was supported by the National Science and Technology Council (NSTC), Taiwan, under Contract NSTC 112-2221-E-027-016-MY2.

ABSTRACT An application-specific integrated circuit (ASIC) bridge for translating and transmitting data between Ethernet and USB was developed in this study. The Ethernet packets of this device are in the IEEE 802.3 standard frame format and transmitted over a 1-Gbit media-independent interface. The device supports the user datagram and address resolution protocols with high transfer speeds. The proposed hardware solution achieves a higher data throughput (682.13 Mbps) than conventional software-based solutions. Its design was validated on a field-programmable gate array (FPGA), and the ASIC was then fabricated using the Taiwan Semiconductor Manufacturing Company 0.18-μm CMOS process. The fabricated ASIC's power consumption, gate count, and chip area were 74.68 mW, 50 100, and 1.199 $\times$ 1.196 mm^2, respectively. Moreover, its operating frequency was 125 MHz under a power supply voltage of 1.8 V. The proposed bridge had high throughput and low latency on the FPGA, and the ASIC package was robust, convenient to use, and energy efficient.

INDEX TERMS Ethernet packet, USB data, UDP, ARP, FPGA board, ASIC.

I. INTRODUCTION

Communication networks are the backbone of all industrial automation systems. They interconnect individual components to improve system efficiency and performance [1]. Ethernet is a reliable communications protocol that has powered the rapid development of Industry 4.0, Internet of Things applications, artificial intelligence communication, and automation systems. Ethernet and USB have rapidly replaced previously common communication standards, such as RS-232 and RS-422, in the consumer market. Ethernet technology has evolved to meet new bandwidth and market requirements and is now used in the harshest industrial environments and to interconnect countless devices [2]. Ethernet has replaced many legacy and proprietary data transmission systems [3], [4]. An Ethernet interface can be implemented with a DSP and controller chip, such as the KSZ8851 chip, in most embedded systems [5]. In particular, application-specific integrated circuit (ASIC) Ethernet implementations can achieve high energy efficiency as data transceivers, thus facilitating network management.

The USB standard defines cables, connectors, communication protocols, and power transmissions between devices. USB has largely replaced earlier interfaces, especially for low-power and portable devices. USB data transfer has numerous modes with differing data transfer rates and hardware and cabling requirements [1]. In particular, the USB

The associate editor coordinating the review of this manuscript and approving it for publication was Ludovico Minati.

2.0 standard (high-speed) was released in April 2000 with a maximum data transfer speed of 480 Mbit/s; however, bus access constraints limited its effective throughput to 280 or 35 Mbit/s [6]. In late 2008, the USB 3.0 specification was released, supporting a data transfer speed of up to 5.0 Gbit/s in the full-duplex transfer mode but typically achieving a data transfer speed of approximately 3.2 Gbit/s in practice. Versions 3.1 and 3.2 achieved data rates of 1 GB/s and 10 Gbit/s, respectively, and version 3.2 had enhanced data encoding efficiency [7]. Because most handheld devices use microprocessors or microcontrollers to control the battery charging process, the use of USB has a minimal effect on the cost of existing products [6].

Most high-speed, high-throughput protocols for modern data transmission are based on the user datagram protocol (UDP) instead of the transmission control protocol (TCP) [8], [9], [10]. During UDP communications, congestion resulting in interference can be resolved through simultaneous multi-path communication (SMPC), which allows the transmission to be distributed over multiple paths. This feature improves communication performance and reduces the load on network resources [11], [12]. SMPC methods range from approaches based on simple moving average to stabilizers implemented in neural network models, such as multilayer perceptron, recurrent neural network, and long short-term memory models; these methods can reduce fluctuations and improve average throughput [13], [14]. UDP congestion control is limited by the receiver's buffer size, which specifies the total amount of data to be transmitted [15]. Permutation-based encapsulation has been used to improve the goodput gain, latency ratio, physical-layer throughput, and secrecy rate of UDP by reducing the payload [16].

In LAN communications, the address resolution protocol (ARP) is used to convert dynamic Internet protocol (IP) addresses to fixed physical media access control (MAC) addresses. In a previous study, ARP broadcast messages were used to communicate covertly over a LAN through a technique that was robust against traffic normalization and resilient to frequency-analysis-based decoding techniques [17]. In another study, a binding server was developed to register client applications to nodes, thereby improving network performance [18].

One standard or protocol often must be converted to another to achieve interoperability. Protocol conversion requires the conversion of data packet contents, including messages, events, commands, and time synchronization. Key challenges in protocol translation include communication delays, processing latency, and overall processing time [1]. Conventional software-based translation stacks are CPU-intensive when operating at full transmission rates, thus resulting in high latency and low throughput; this problem can be overcome using appropriate hardware implementations [19], such as a field-programmable gate array (FPGA). FPGAs can be used in customized Ethernet-based embedded systems [20]. In the present study, a cost-effective FPGA-based Ethernet controller interface was developed as an alternative to TCP/UDP software stacks as a bridge for Ethernet and USB messages.

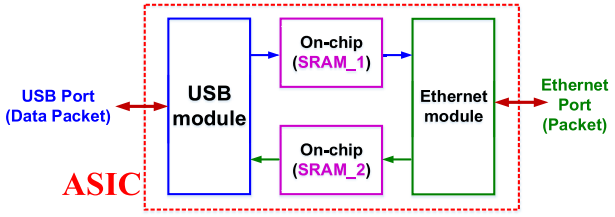


FIGURE 1. Proposed communication architecture of the designed bridge ASIC between Ethernet and USB modules with on-chip SRAM cells.

Fig. 1 illustrates the proposed communication architecture of the designed bridge ASIC. This architecture comprises Ethernet and USB modules with on-chip static RAM (SRAM) cells. A USB data packet received at the USB port is sent to the USB module and written to the first SRAM (SRAM_1), which calculates the length of this data packet. This length is then sent to the Ethernet module, which upon receiving another signal, sends the Ethernet preamble in sequence, followed by the source and destination MAC addresses. It then calculates and validates the IP checksum before adding the IP header. Subsequently, the stored complete Ethernet packet is sequentially read from the on-chip SRAM (SRAM_1) and transmitted from the first on-chip SRAM to the Ethernet port, which is a 1-Gbit media-independent interface (GMII) module. Two on-chip SRAMs are used because of the frequency incompatibility between Ethernet (125 Mbps) [12] and USB (480 Mbps). The remainder of this paper is organized as follows. Section II presents the system architecture of the Ethernet and RS485 modules. Sections III and IV discuss the simulation and measurement results, respectively, obtained for the proposed bridge ASIC with TCP and SRAM. Finally, Section V presents the conclusions.

II. COMMUNICATION ARCHITECTURE OF THE PROPOSED BRIDGE ASIC

Fig. 2 displays the system architecture of the Ethernet module, which comprises a GMII module, a MAC module, an ARP module, a UDP module, and a receiver (RX) module. The GMII module transmits Ethernet packets in the GMII interface standard format to an Ethernet port physical-layer (PHY) chip, and the RX module receives and parses the Ethernet packets. Ethernet UDP packets are stored in SRAM-2. The ARP module responds to correctly formed ARP requests by returning an ARP reply packet to the MAC module. The MAC module produces the preamble, MAC address, EtherType, and frame check sequence (FCS) for transmitted Ethernet packets. The UDP module handles IP and UDP headers and reads data from SRAM-1. The ARP module controls the ARP header for the packets. The on-chip SRAMs were implemented in an ARM 0.18-μm process and were 128 bytes each.

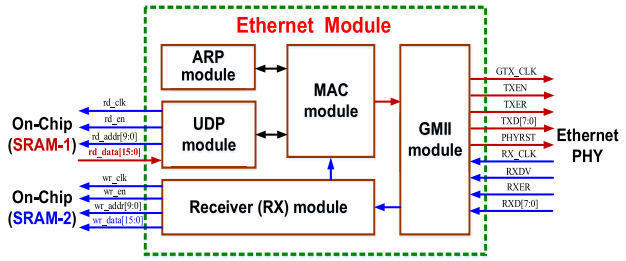


FIGURE 2. System architecture of the Ethernet module.

Fig. 3 displays the architecture of the USB module, which was designed in accordance with the CY7C68013A specification [21]. CLK_48 is a 48-MHz input clock signal, and IFCLK is an output clock signal for CLK_48. PKTEND is an end signal for a transmitted packet (PKT), which is high for the ''preset'' function and low for the ''action'' function. PKTEND must be activated simultaneously with SLWR, which is the write signal of the USB module. FlagA and FlagD are input status signals, which are low for the ''preset'' function and high when the system is ready to receive data packets. SLOE is an output enable signal of the USB module; this signal is high and low when the system is receiving and transmitting data packets, respectively. SLRD is the read signal of the USB module, which is high for the ''preset'' function and low when the system is receiving data packets. SLWR is the write signal of the USB module, which is high for the ''preset'' function and low when the system is transmitting a data packet. FIFO_Addr [1:0] is the selected address for receiving or transmitting a data packet; 2'b00 and 2'b10 are the addresses of the received and transmitted data packets, respectively. DATA[15:0] is a 16-bit bidirectional data transmission port.

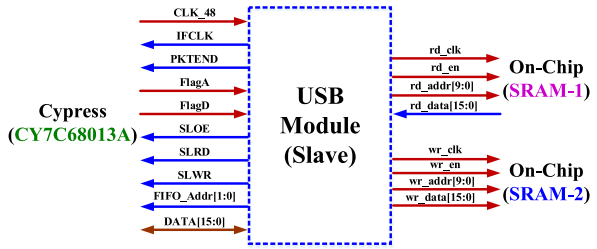


FIGURE 3. System architecture of the USB module.

Fig. 4 depicts a flowchart of the designed bridge. The Ethernet module is idle until it receives an Ethernet packet. First, the packet's preamble and MAC address are validated; any invalid packets are discarded, and the Ethernet module returns to the idle state. The received packets are then classified as ARP and UDP packets. For UDP packets, the received status is changed to ''IP_header,'' and the IP and UDP headers are validated. Subsequently, the received current status is changed to ''UDP_header.'' After the UDP port is validated, the received current status is changed to ''UDP_data,'' and the payload of the packet is stored in SRAM_2. Information on the amount of UDP data is immediately stored in the register. Finally, the FCS stored in the register is validated; if invalid, the packet data are cleared from the SRAM_2 module and register. Otherwise, upon reception of a transmission request signal (high, flag_d = 1) from the USB module, the stored UDP data are read and transmitted to the USB port (interface).

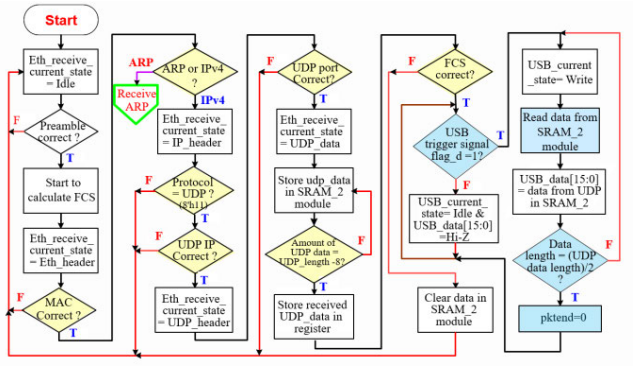


FIGURE 4. Operational flowchart of the Ethernet module, which is used to transform the Ethernet packet at the USB port (interface).

The flowchart for ARP packet reception is illustrated in Fig. 5. The formats of the ARP request packet (16'h0001) and ARP reply packet (16'h0002) are validated, including the source IP, destination IP, and FCS. For ARP reply packets, the destination MAC address is stored in a register.

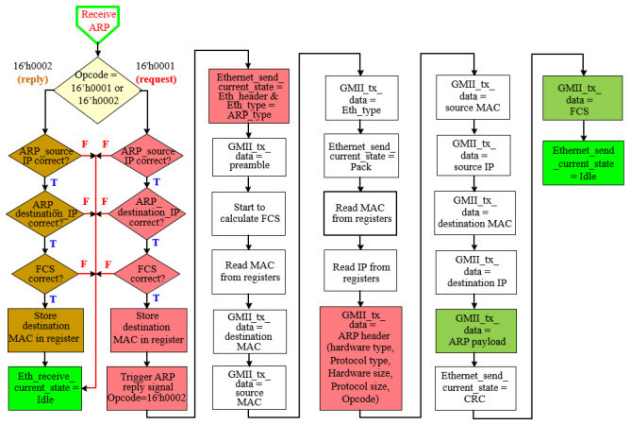


FIGURE 5. Flowchart of ARP packet reception by the Ethernet module.

For ARP request packets, an ARP reply signal is triggered with an opcode of ''16'h0002.'' The current transmission state of the Ethernet module is set to Ethernet header, and the preamble, destination MAC, source MAC, and Ethernet type, are generated and readied to send to the requesting device. Next, ARP header signals, including the hardware type, protocol type, hardware size, protocol size, and opcode, are sent to the requesting device in accordance with the destination MAC and IP. Subsequently, the ARP payload is sent. Finally, the current transmission state of the Ethernet

module is set to cyclic redundancy check (CRC), and the FCS is sent.

When the ARP request packet is ready to be transmitted (Idle) and the trigger signal (S2_in = 1) of ARP transmission is received, the ARP opcode is changed to 16'h0001, and the EtherType is changed to ARP (16'h0806). The preamble is then transmitted, and the FCS calculation begins. The source MAC is read from the register, and the destination MAC address is changed to 48'hFFFFFFFF in accordance with the ARP protocol. An Ethernet header (data), including the MAC address and EtherType, is then parsed and transmitted. Subsequently, an ARP header is transmitted with an opcode of "16'h0001." The transmitted ARP header comprises information on the hardware type, protocol type, hardware size, protocol size, and opcode. Next, the sender hardware address (source MAC), sender protocol address (source IP), destination hardware address (destination MAC), and destination protocol address (destination IP) are sent; the remaining data are the payload. After the CRC has been completed, the calculated FCS is sent to complete the transmission of the Ethernet ARP request packet. Fig. 6 illustrates the flowchart for the transmission of ARP request packets by the Ethernet module.

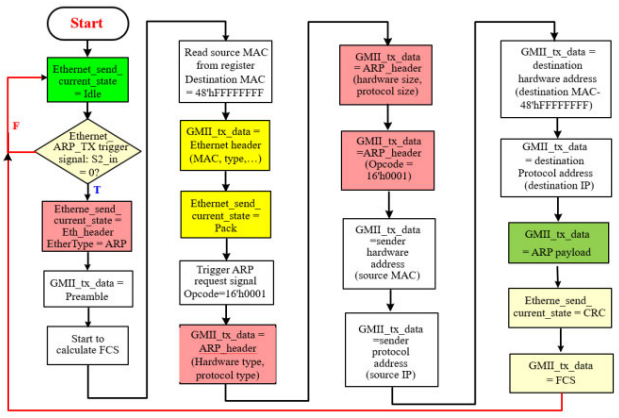


FIGURE 6. Flowchart of ARP request packet transmission by the Ethernet module.

To transmit a USB data packet from the USB port to the Ethernet port, the USB module receives the data packet and transforms it into an Ethernet packet as illustrated in the flowchart in Fig. 7. If the USB module receives a request trigger signal (flag_a = 1), the USB module reads the received USB data and stores them in SRAM_1. The received USB data are counted and stored in a register. When an Ethernet request signal (S3_in = 0) is received, Ethernet UDP packet transmission begins. After the synchronization signal (preamble) has been received, the FCS is calculated. The Ethernet header (MAC address and Ethernet type) and amount of USB data are read from the registers. Subsequently, the UDP packet length and IP packet length are calculated. UDP packets with payloads of less than 18 bytes are padded with zeros. The IP checksum is then calculated, and the IP header, including the version, length, protocol, checksum, and IP, are transmitted. The UDP checksum is calculated, and the UDP header, including the port, length, and checksum, are transmitted. The USB data are then read from the SRAM_1 module and transmitted to the GMII interface until all USB data have been transmitted correctly. Finally, the calculated FCS is transmitted, thus completing Ethernet UDP packet transmission. Note that the proposed bridge system incorporates UDP and ARP protocols, which operate within the Transport and Network layers of the OSI model. These protocols ensure data privacy and handling in accordance with ethical standards for data communication. For instance, the protocol type serves as a critical standard in Network layer operations for effective data management. Additionally, CRC (Cyclic Redundancy Check), IP (ARP) checksum, and UDP checksum mechanisms are employed to maintain data privacy and ensure data integrity in data communication.

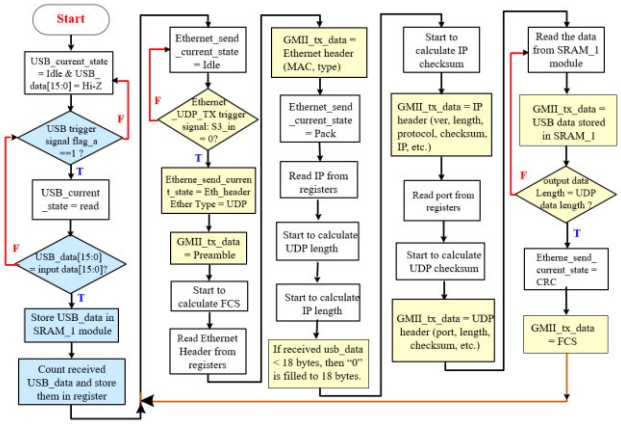


FIGURE 7. Flowchart of translation from a USB data packet to an Ethernet UDP packet.

III. FUNCTIONAL SIMULATION AND VERIFICATION

Two transmission functions of the proposed Ethernet–USB bridge ASIC had to be validated: the paths from the Ethernet port to the USB port and the reverse path. Fig. 8 depicts the simulated results for ARP request packet reception and ARP reply packet transmission. The trigger signal gmii_rx_dv is set to high (1), indicating that the preamble has been received and confirmed. The ARP request data, namely the source MAC, destination MAC, EtherType, ARP header, opcode, sender hardware address, source IP, target hardware address, destination IP, ARP payload, and FCS, are then sent from the GMII module to the ARP module. Next, if the acknowledge signal arp_ack_tx is high (1), the preamble has been transmitted and validated. The ARP reply data, including the destination MAC, source MAC, EtherType, ARP header, operation code, sender hardware address, source IP, target hardware address, destination IP, and ARP payload, are subsequently transmitted to the GMII module.

Fig. 9 shows the simulated data transformation between the UDP packet and USB data. As illustrated in Fig. 9(a),

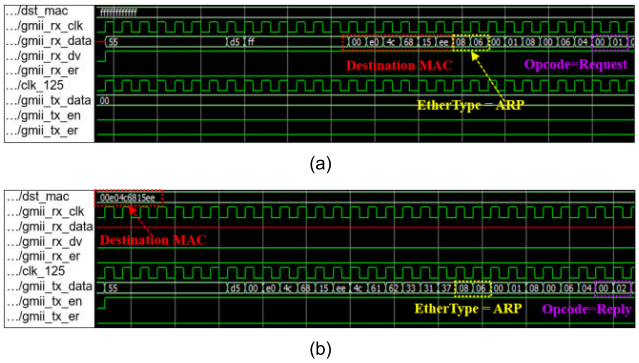


FIGURE 8. Simulated ARP packets. (a) Received ARP request packet and (b) transmitted ARP reply packet.

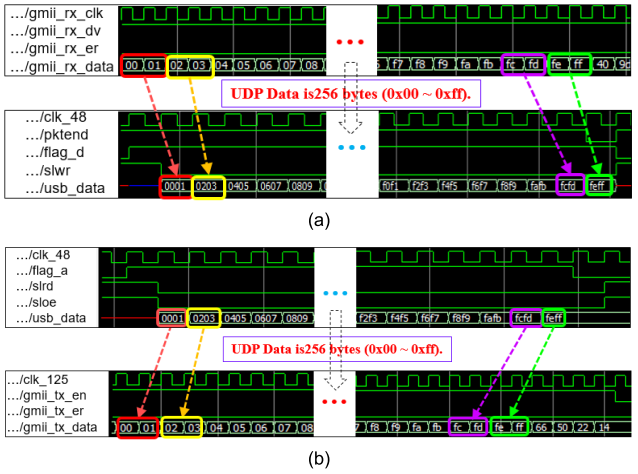


FIGURE 9. Simulated data transformation between a UDP packet and USB data. (a) From a UDP packet to USB data and (b) from USB data to a UDP packet.

if the trigger signal gmii_rx_dv is high (1) and the error signal gmii_rx_er is low (0), the received Ethernet UDP packet is correct. The UDP packet data (Payload) are then stored in the second on-chip SRAM (SRAM_2) sequentially and are ready for reading by the USB module. After the transfer trigger signal of transmission (flag_d) has been set to high (1), the enable signal and the address of the selected data are transmitted by setting SLOE = 1, SLWR = 0, and FIFO_Addr [1:0] = 2'b10 (2). The selected Ethernet UDP packet data (usb_data) stored in SRAM_2 are read and transmitted to the following USB module. However, if the trigger signal for receiving (flag_a) is high (1), the enable signal (SLOE) is low (0), and the read signal (slrd) is low (0), the selected USB data are received and stored in SRAM_1 sequentially. Next, if the trigger signal for transmission (gmii_tx_en) is received and is set to low (1), the control signals and data, including the preamble, CRC, source MAC, destination MAC, IP header, source IP, destination IP, UDP header, UDP data, CRC, and FCS, are transmitted to the GMII module. As displayed in Fig. 9(b), the USB data, which are stored in SRAM_1, are a 16-bit signal; however, the data of the GMII interface are an 8-bit signal. The received USB data must therefore be divided into two 8-bit signals that are transmitted sequentially.

Fig. 10 presents the validation setup for UDP–USB translation. This setup comprises two computers (PC1 and PC2), a USB controller, an FPGA development board (Intel DE-10 Standard, Terasic Inc., Taiwan), Ethernet PHY, an optical–electrical converter, and optical fiber. USB data were generated at PC1 by using the USB monitoring software program Device Monitoring Studio and then transmitted to PC2 through the FPGA development board, which included a USB controller daughter board (CY7C68013A). The resulting Ethernet packet was then sent from the FPGA board to the Ethernet PHY daughter board (RTL8211EG). The optical–electrical converter (STC-G3S20-11) was used to transmit the received packet over optical fiber, thus extending the transmission distance to 20 km. This cable was connected to a second converter, which was connected to PC2 with an RJ45 network cable that transmitted the transformed Ethernet packet.

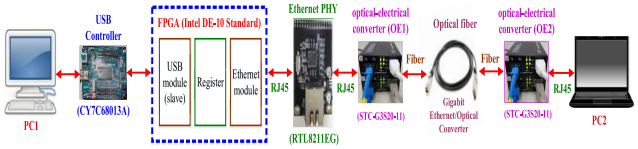


FIGURE 10. Validation setup comprising the computers, USB controller, FPGA development board, Ethernet PHY, optical-electrical converters, and optical fiber for data transformation between a UDP packet and USB data.

Fig. 11 displays the validation environment for data transformation in USB–UDP packet translation. Device Monitoring Studio was used to analyze the transmitted USB data at the USB terminal of PC1, and network packet generation software (Colasoft Packet Builder) was installed on PC2 to produce the Ethernet ARP and UDP packets. PC2 also had network packet analysis software (Wireshark) to parse and monitor all transmitted network packets.

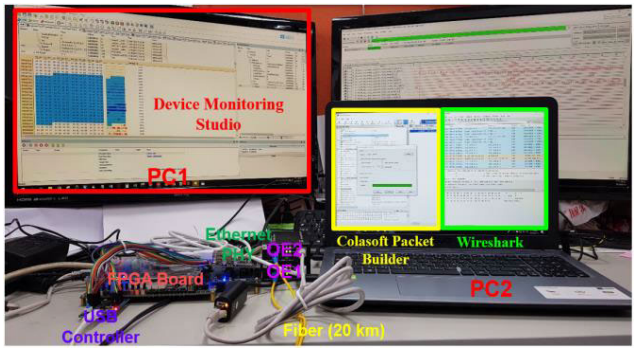


FIGURE 11. Validation environment for translation from USB data (PC1) to UDP packets (PC2).

Fig. 12 presents the content of an ARP request packet in Wireshark. Colasoft Packet Builder generated an ARP packet and sent it to the FPGA development board. Wireshark was used to validate the ARP packet type (0 × 0806), and

its opcode was requested (1). The contents of the ARP reply packet received from the FPGA board are displayed in Fig. 13. The packet type was ARP (0×0806), and its opcode was reply (2).

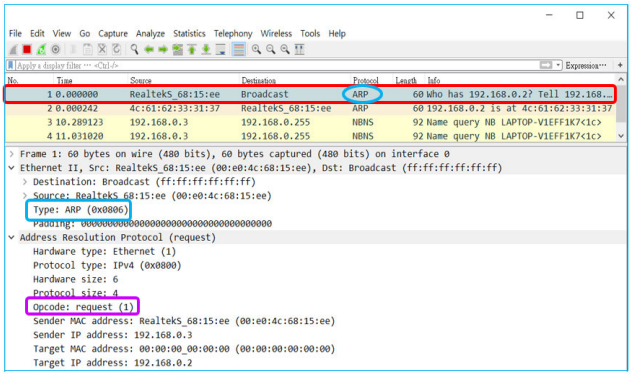


FIGURE 12. Content of ARP request packets sent from PC2 to the FPGA board in Wireshark software.

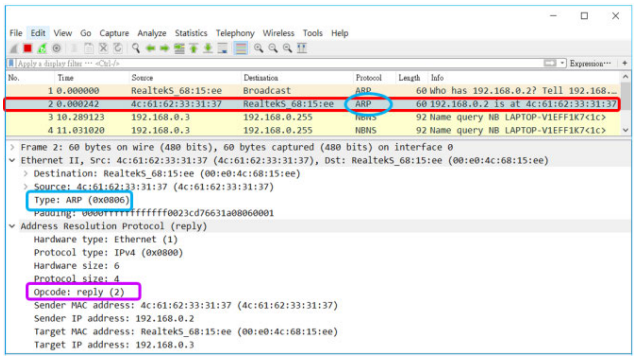


FIGURE 13. Content of ARP reply packets sent from the FPGA board to PC2 in Wireshark software.

A UDP packet was also generated with Colasoft Packet Builder, and this packet was sent to the FPGA board from PC2. Fig. 14 depicts the contents of the UDP packet (roughly 256 bytes) in Wireshark. Finally, the FPGA board transmitted the received Ethernet packet to PC1 through the USB controller. Device Monitoring Studio was used on PC1 to inspect the USB data (Fig. 15); the data were consistent with the original UDP packet.

The maximum packet length was 1526 bytes, including the 8-byte preamble, 14-byte Ethernet header, 20-byte IP header, 8-byte UDP header, 1472-byte UDP data, and 4-byte FCS. The maximum payload of a single UDP packet was 1472 bytes, and the processing time for a maximum-length packet was 12 245 149 ps [or approximately 1530 clock cycles (12 245.149 ns/8 ns)] under a clock frequency of 125 MHz. The throughput for the UDP packet was therefore 961.69 Mbps. This value was obtained by dividing the maximum UDP data length (1472 bytes) by the processing time (12 245 149 ps). Roughly 27 563.758 ns were required for transformation between Ethernet UDP packets and USB

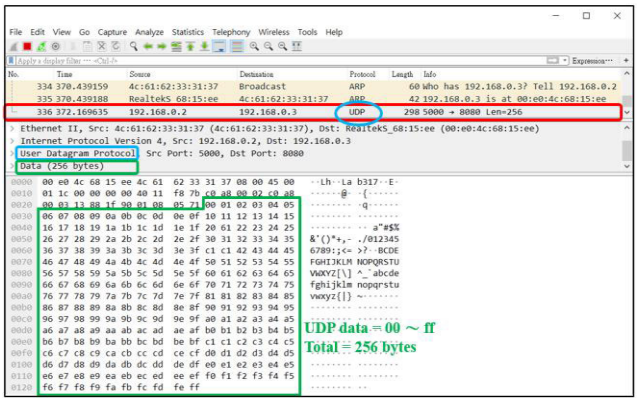


FIGURE 14. Content of the UDP packet sent from PC2 (Ethernet terminal) to the FPGA board in Wireshark software.

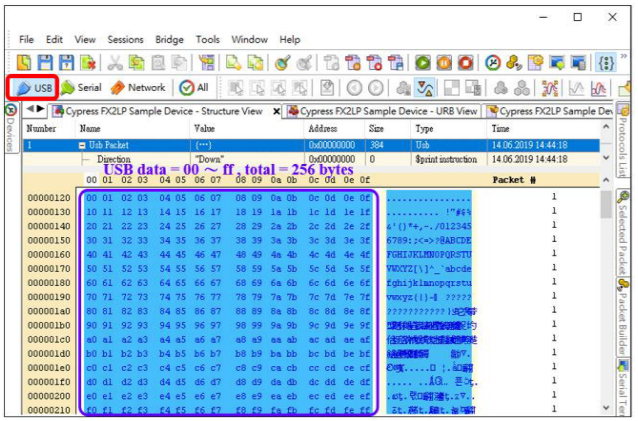


FIGURE 15. Content of the USB data sent from the FPGA board to PC1 (USB terminal) in Device Monitoring Studio.

data and subsequent transmission. The data throughput of the FPGA board was therefore approximately 427.69 Mbps (11 776 bits/27 563.758 ns). This throughput was increased to approximately 682.13 Mbps with the proposed bridge ASIC [22].

IV. BRIDGE ASIC IMPLEMENTATION AND MEASURED RESULTS

After the functions of the proposed bridge were validated on the FPGA board, circuit synthesis, automatic placement and routing, design rule check, and layout-versus-schematic checking were completed for the Taiwan Semiconductor Manufacturing Company (TSMC) 0.18-μm process. The performance of the implemented bridge ASIC was then evaluated. Debugging and validation of the ASIC were conducted using the NC-Verilog simulator and Verdi/nWave waveform viewer. Fig. 16 shows the logic gate model of the designed bridge ASIC, which is synthesized from the RTL source code. It is composed of an Ethernet module, a USB module, an SRAM_1 cell, and an SRAM_2 cell. Fig. 17 displays the layout and a photograph of the Ethernet-to-USB data transfer ASIC. After the ASIC was packaged in the CLCC84 package, measurements were performed with

a mixed-mode signal test machine (ADVANTEST V93000 PS1600). The measurement results are listed in Table 1.

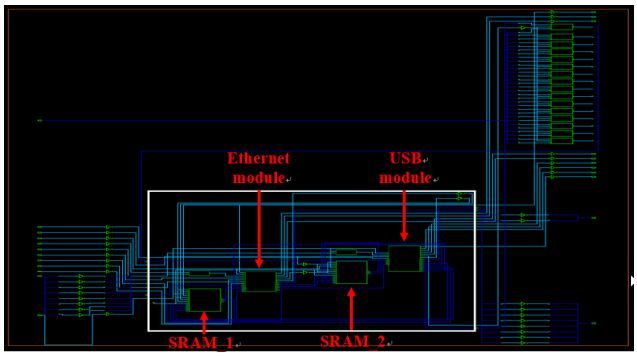
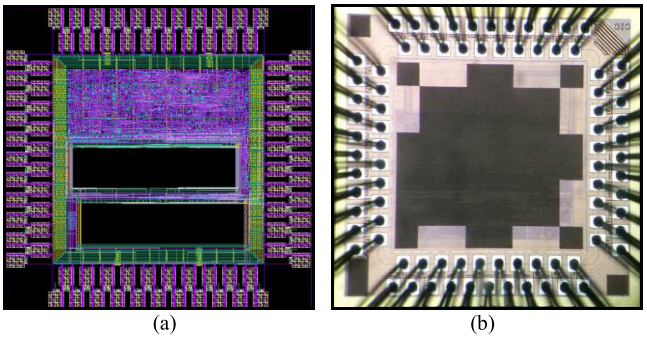


FIGURE 16. Logic gate model of the designed bridge ASIC, which is synthesized from the RTL source code.



(a) (b)

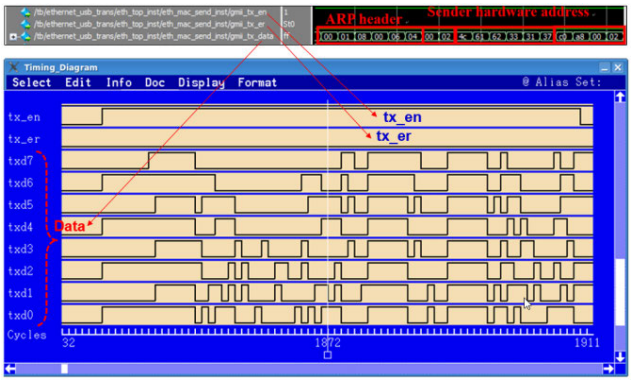
FIGURE 17. Proposed Ethernet-USB data transfer bridge ASIC. (a) Chip layout and (b) chip photograph.

TABLE 1. Measurements for the developed ethernet-to-USB ASIC.

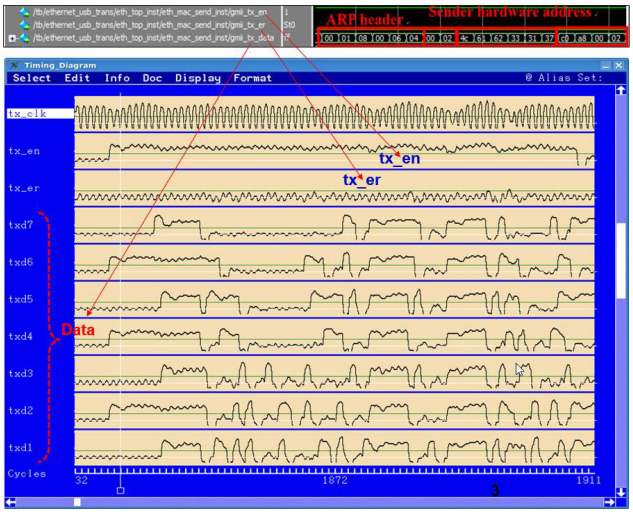
Specification	Value
Process	TSMC 0.18-μm CMOS
Operation frequency (MHz)	125 / 48
Supply Voltage (V)	+1.8
Memory (bytes)	128 × 2
Power Consumption (mW)	74.6758
Chip Size (mm^2)	1.19884 × 1.1956
Total Pins	84

Fig. 18 displays data from the sent and received ARP request and reply packets obtained with the ADVANTEST device. ARP request data transmitted when the enable signal (tx_en) was high (1) and the error signal (tx_er) was low (0) are plotted in Fig. 18(a). These ARP request data were sent from the GMII module to the ARP module, which generated ARP reply data and returned them to the GMII module; these reply packets were measured and are displayed in Fig. 18(b). The measured results displayed in Fig. 18 are similar to the simulated results shown in Fig. 8.

Fig. 19 depicts the results obtained for the transformation of Ethernet UDP packets into USB data. Fig. 19(a) presents measurements of Ethernet UDP packets, which were received



(a)



(b)

FIGURE 18. Measurement results for (a) sent ARP request packets and (b) received ARP reply packets.

if the trigger signal gmii_rx_dv was high (1) and the error signal gmii_rx_er was low (0). The UDP packet data (payload) were then stored sequentially in SRAM_2. The UDP packet data (usb_data) were measured after the transmission trigger (flag_d) and write signal SLWR were set to high (1) and low (0), respectively; the corresponding results are displayed in Fig. 19(b). The measured results displayed in Fig. 19 are similar to the simulation results shown in Fig. 9(a).

The ADVANTEST device was used to validate the data transformation from the USB terminal to the Ethernet UDP terminal. Fig. 20(a) displays the measured USB data received when flag_a was high and SLRD was low. These data were stored sequentially in SRAM_1. When the trigger signal (tx_en) was high and the error signal (tx_er) was low, the Ethernet UDP packets were transmitted to the GMII module; the measured packets are shown in Fig. 20(b). The measured results shown in Fig. 20 are similar to the simulation results displayed in Fig. 9(b).

Table 2 summarizes the performance of the proposed bridge ASIC and other similar Ethernet and USB communication systems. The proposed bridge ASIC operates

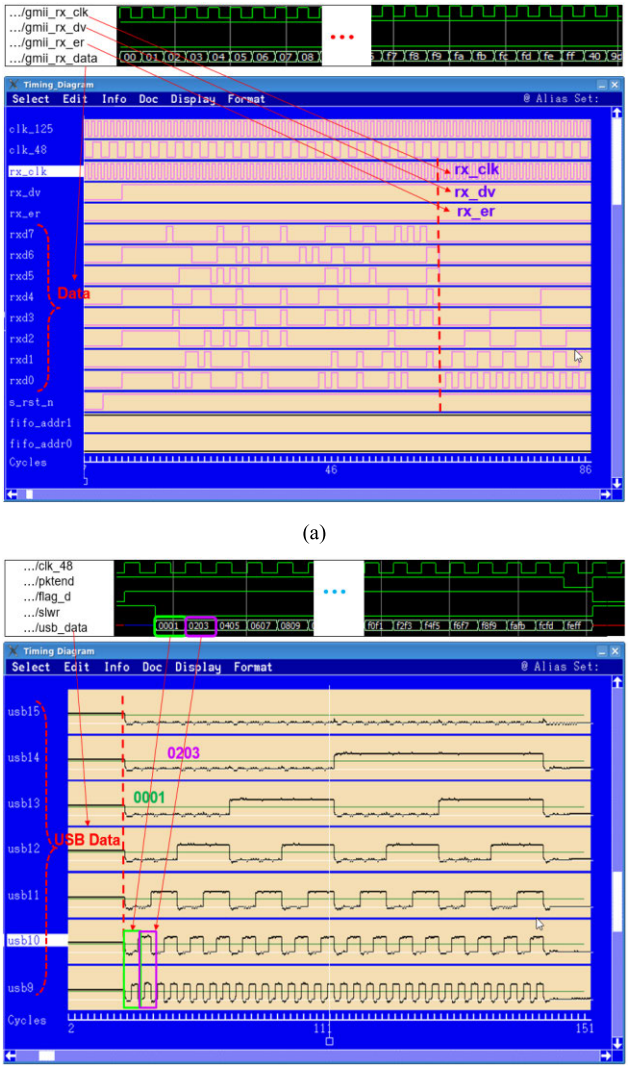


FIGURE 19. Measured transformation from UDP packets to USB data: (a) measured Ethernet UDP packet and (b) measured USB data.

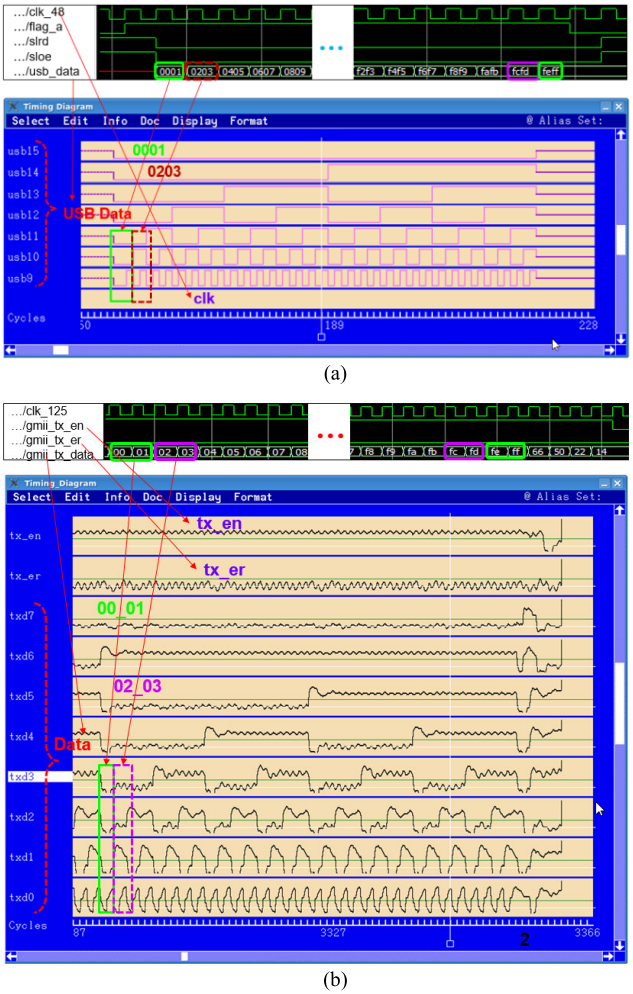


FIGURE 20. Transformation from USB data to UDP packets: (a) measured USB data and (b) measured UDP packets.

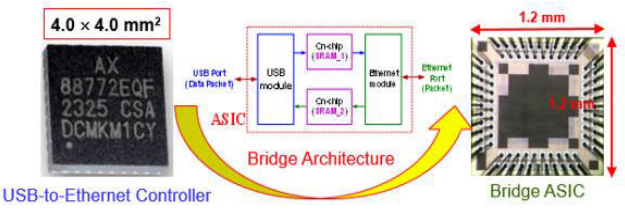


FIGURE 21. The proposed ASIC achieved a potential size reduction in a USB-to-Ethernet controller.

at 125 MHz and can achieve gigabit Ethernet performance. Two 128-byte on-chip SRAMs were used to reconcile speed disparities between the Ethernet and USB modules. The maximum throughput for 256-byte UDP packets was 961.69 Mbps with the FPGA board; this value is higher than those achieved in [26], [27], and [29]. The proposed Ethernet-USB bridge ASIC is a USB to Gigabit Ethernet controller with an integrated 10/100/1000 Mbps Gigabit Ethernet PHY. It is suitable for various applications such as notebook/laptop LAN, USB Ethernet dongles, docking stations, PDA cradles, game consoles, smart cameras, and IP set-top boxes (STBs). These commercial products are designed to meet current network standards, ensuring data interoperability and system robustness. However, the throughput of the designed bridge ASIC was approximately 682.13 Mbps, which represents optimal performance [23], [24], [25]. The lower throughput of the ASIC can be attributed to chip area constraints that limited the on-chip SRAM to 128 bytes. A significant limitation of the designed bridge ASIC is the memory capacities of RAM and FIFO. Increasing their capacity reduces computational overload and enhances accuracy but at the expense of lower computational efficiency. In practical implementations, integrating the bridge ASIC with an Ethernet PHY (Physical Layer) is expected to enhance its performance.

However, the chip area (1.2×1.2 mm^2) and power consumption (74.68 mW) of the proposed bridge ASIC were superior to those of the designs proposed in [23],

TABLE 2. Performance comparisons of the proposed bridge ASIC and other similar ethernet and USB communication systems.

Parameters	This study	[23] (2014)	[24] (2014)	[25] (2018)	[26] (2015)	[27] (2017)	[28] (2021)	[29] (2022)
Development process	FPGA /ASIC	ASIC	ASIC	FPGA /ASIC	FPGA	FPGA	Embedded Platform	FPGA
Operating frequency (MHz)	125	20	20	125	125	50	–	125
Ethernet (Mbps)	1,000	10/100	10/100/ 1000	1,000	1,000	1,000	2500/5000	1,000
FIFO (bytes)	–	–	–	1,000	9,000	–	–	–
SRAM (bytes)	128 × 2	–	–	–	–	2,000	–	–
Throughput (Mbps) (FPGA)	961.69	–	–	–	912.8	795.8	2000	747.45
Throughput (Mbps) (ASIC)	682.13	–	–	–	–	–	–	–
Protocols	IP/UDP	w/CDC-ECM /NCM	w/CDC-ECM /NCM	IP/UDP	IP/UDP /ICMP/ARP /PTP	IP/TCP /UDP/ARP /ICMP	IP/UDP	Modbus/ TCP
ASIC chip area (mm^2)	1.20 × 1.20	4.0 × 4.0	5.0 × 5.0	1.27 × 1.27	–	–	–	–
Power consumption (mW)	74.68	–	–	137	–	–	–	–

[24], and [25]. The proposed architecture was successfully implemented in an ASIC and validated, and the proposed ASIC achieved not only low power consumption and a small chip area but also high throughput. Fig. 21 shows a diagram demonstrating how the proposed bridge ASIC can reduce the module size of a USB-to-Ethernet controller from 4.0 × 4.0 mm^2 to 1.2 × 1.2 mm^2.

V. CONCLUSION

In this study, a bridge ASIC with USB and ARP functionality was designed for transformation between Ethernet packets and USB data. An FPGA development board was used to validate the functions of this ASIC, and the proposed bridge ASIC was fabricated using the TSMC 0.18-μm CMOS cell-based process. The Ethernet terminal of the ASIC generated packets and parsed ARP and UDP data. The PHY chip was integrated with a GMII interface, connected to a 1-Gbps high-speed Ethernet network and an optical fiber network. At the USB terminal, the CY7C68013A chip operated at 48 MHz. Measurements of the designed ASIC with the ADVANTEST device validated its operation characteristics and functions. The FPGA board's maximum throughput of 256-byte UDP packets was 961.69 Mbps. The developed ASIC has a smaller chip area (1.2 × 1.2 mm^2), lower power consumption (74.68 mW) than previous designs, and high throughput. This ASIC can reduce the size and improve the performance of the USB-to-Ethernet controller. Because of these advantages, the proposed bridge ASIC has high utility in industrial applications. Future research could extend to standard protocols in the Network layer, such as IPv6, ICMP, TCP, etc. To achieve a complete co-design between hardware and software, an ARM Cortex-M0 CPU could be integrated with the proposed bridge ASIC. This integration aims not only to enhance computing capabilities but also to support higher-layer network protocols such as SQL, SSL, HTTP, FTP, MQTT, etc. Furthermore, future work could enhance applications including web pages, file transfer, Internet of Things (IoT), quantum computing, AI, and other areas.

ACKNOWLEDGMENT

The authors would like to thank Taiwan Semiconductor Research Institute (TSRI) for fabricating the test chip. This manuscript was edited by Wallace Academic Editing.

REFERENCES

[1] J. A. Kay, D. C. Mazur, and R. A. Entzminger, "Basics of communication networks for electrical engineers in the forest products industries," in *Proc. IEEE IAS Pulp, Paper Forest Industries Conf. (PPFIC)*, Appleton, WI, USA, Jun. 2018, pp. 1–5.

[2] *Standard for Ethernet*, IEEE Standard 802.3-2015, IEEE, New York, NY, USA, 2015, doi: 10.1109/IEEESTD.2016.7428776.

[3] *IEEE Standard for Information Technology-Local and Metropolitan Area Networks*, IEEE Standard 802.3, IEEE, New York, NY, USA, 2009.

[4] Y.-D. Wang and X.-M. Dai, "A Ethernet interface solution based on TCP/IP protocol," in *Proc. IEEE 11th Int. Conf. Signal Process.*, vol. 2, Beijing, China, Oct. 2012, pp. 1521–1525.

[5] T. Jia and W. Dong, "An Ethernet interface solution of space science experiment payloads," in *Proc. 12th Int. Conf. Signal Process. (ICSP)*, Hangzhou, China, Oct. 2014, pp. 408–412.

[6] F. He, "USB port and power delivery: An overview of USB port interoperabiliy," in *Proc. IEEE Symp. Product Compliance Eng. (ISPCE)*, Chicago, IL, USA, May 2015, pp. 1–5.

[7] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich, "The evolution of factory and building automation," *IEEE Ind. Electron. Mag.*, vol. 5, no. 3, pp. 35–48, Sep. 2011.

[8] M. Masirap, M. H. Amaran, Y. M. Yussoff, R. A. Rahman, and H. Hashim, "Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT)," in *Proc. IEEE Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, Penang, Malaysia, May 2016, pp. 200–205.

[9] M.-H. Wang, L.-W. Chen, P.-W. Chi, and C.-L. Lei, "SDUDP: A reliable UDP-based transmission protocol over SDN," *IEEE Access*, vol. 5, pp. 5904–5916, 2017.

[10] D. Syzov, D. Kachan, K. Karpov, N. Mareev, and E. Siemens, "Custom UDP-based transport protocol implementation over DPDK," in *Proc. 7th Int. Conf. Appl. Innov. IT (ICAIIT*, Köthen, Germany, Mar. 2019, pp. 13–17.
[11] A. Tanaka and M. Harayama, "Path priority control method for simultaneous multi-path communication," *IEICE Tech. Rep.*, vol. 113, no. 472, pp. 143–148, Mar. 2014.
[12] W. Mano and M. Harayama, "Improved communication fairness of SMPC," *IEICE Tech. Rep.*, vol. 119, no. 342, pp. 2018–2134, Dec. 2019.
[13] A. Tealab, "Time series forecasting using artificial neural networks methodologies: A systematic review," *Future Comput. Informat. J.*, vol. 3, no. 2, pp. 334–340, Dec. 2018.
[14] M. Harayama and N. Miyagawa, "Intelligent throughput stabilizer for UDP-based rate-control communication system," *Intell. Converged Netw.*, vol. 2, no. 3, pp. 205–212, Sep. 2021.
[15] J. Postel, *User Datagram Protocol*, document RFC 768, Internet Eng. Task Force (IETF), Aug. 1980.
[16] Y. Yang and L. Hanzo, "Permutation-based TCP and UDP transmissions to improve goodput and latency in the Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14276–14286, Sep. 2021.
[17] A. Dua, V. Jindal, and P. Bedi, "Covert communication using address resolution protocol broadcast request messages," in *Proc. 9th Int. Conf. Rel., INFOCOM Technol. Optim. (Trends Future Directions) (ICRITO)*, Noida, India, Sep. 2021, pp. 1–6.
[18] K. Kimmatkar and U. Shrawankar, "Modified address resolution protocol for delay improvement in distributed environment," in *Proc. Students Conf. Eng. Syst. (SCES)*, Allahabad, India, Apr. 2013, pp. 1–5.
[19] P. Gour, R. S. Mishra, S. Khan, and R. Nema, "Design and optimization of medium access control protocol of IEEE 802.3 transmitter with VHDL," *Int. J. Comput. Appl.*, vol. 13, no. 1, pp. 8–12, Jan. 2011.
[20] A. Choudhary, D. Porwal, and A. Parmar, "FPGA based solution for Ethernet controller as alternative for TCP/UDP software stack," in *Proc. 6th, Ed., Int. Conf. Wireless Netw. Embedded Syst. (WECON)*, Rajpura, India, Nov. 2018, pp. 63–66.
[21] Z. Pengyu, "Radio frequency controlling and transmitting based on the USB chip CY7C68013A," in *Proc. Int. Conf. Electron. Mech. Eng. Inf. Technol.*, vol. 8, Harbin, China, Aug. 2011, pp. 4239–4242.
[22] K. Kim, J. Kim, and A. Deep, "Throughput improvement for Ethernet over USB," in *Proc. 18th IEEE Int. Symp. Consum. Electron. (ISCE)*, Jun. 2014, pp. 1–2.
[23] ASIX Electron. Corp. *USB 2.0 to 10/100 M Fast Ethernet Controller (AX88772E)*. Accessed: Jul. 25, 2024. [Online]. Available: https://www.asix.com.tw/AQ:4en/product/USBEthernet/High-Speed_USB_Ethernet/AX88772E
[24] ASIX Electron. Corp. *USB 3.2 Gen1 to Gigabit Ethernet Con548 Troller (AX88179B)*. Accessed: Jul. 25, 2024. [Online]. Available: https://www.asix.com.tw/en/product/USBEthernet/Super-Speed_USB_Ethernet/AX88179B
[25] H. K. Wang, C. P. Yu, G. M. Sung, and M. W. Li, "Intelligent packet transformation and transmission between Ethernet and optical fiber systems based on a field-programmable gate array board," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2018, pp. 4071–4076.
[26] P. Födisch, B. Lange, J. Sandmann, A. Büchner, W. Enghardt, and P. Kaever, "A synchronous gigabit Ethernet protocol stack for high-throughput UDP/IP applications," *J. Instrum.*, vol. 11, no. 1, Jan. 2016, Art. no. P01010.
[27] Q. Liu, Z. Xu, and Z. Li, "Implementation of hardware TCP/IP stack for DAQ systems with flexible data channel," *Electron. Lett.*, vol. 53, no. 8, pp. 530–532, Apr. 2017.
[28] R. K. Kiran, R. M. Chikodi, R. Soni, and V. K. Roy, "Embedded real time media streaming over Ethernet via USB-OTG," in *Proc. IEEE Mysore Sub Sect. Int. Conf. (MysuruCon)*, Hassan, India, Oct. 2021, pp. 734–737.
[29] G.-M. Sung, Z.-Y. Tan, C.-Y. Lee, C.-L. Tseng, C.-R. Chen, C.-P. Yu, C.-C. Hsiao, and R.-G. Lee, "Ethernet packet transformation and transmission between Modbus/TCP and USB 3.0 with field-programmable gate array development board," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Prague, Czech Republic, Oct. 2022, pp. 1677–1681.

GUO-MING SUNG (Member, IEEE) was born in Zhanghua, Taiwan, in 1963. He received the B.S. and M.S. degrees in biomedical engineering from Chung Yuan Christian University, Taoyuan, in 1987 and 1989, respectively, and the Ph.D. degree in electrical engineering from National Taiwan University, Taipei City, Taiwan, in 2001.

In 1992, he joined the Division of Engineering and Applied Sciences, National Science and Technology Council (NSTC), Taiwan, where he became an Associate Researcher, in 1996. Since 2001, he has been with the Department of Electrical Engineering, National Taipei University of Technology, where he is currently a Professor and the Chairperson. His research interests include magnetic sensors, USB/RS485 communication, analog-to-digital converter ICs, motor control ICs, radio-frequency harvester ICs, AI chip design, and mixed-mode ICs for use by the Internet of Things (IoT).

ZI-YU LI was born in Taiwan, in 1995. He received the B.S. and M.S. degrees in electrical engineering from the National Taipei University of Technology, Taipei City, Taiwan, in 2017 and 2019, respectively. He has been an IC Designer, since 2017. His research interests include digital ICs, Ethernet ICs, and USB ICs.

CHIH-PING YU was born in Keelung, Taiwan. He received the B.S. degree in electrical engineering from the National Taiwan University of Science and Technology, Taiwan, and the M.S. degree in electrical engineering from Washington University, USA, in 1989 and 1995, respectively. He is currently an Associate Professor with the Department of Electrical Engineering, National Taipei University of Technology. His research interests include magnetic sensor applications, analog filter circuits, and mixed-mode circuit design.