Advances in Computer Science Research, volume 89
2019 International Conference on Wireless Communication, Network and Multimedia Engineering (WCNME 2019)

Design method of Ethernet based on SGMII

Xi Hu[1] and Weigong Zhang [1,*]
[1]College of Information Engineering, Capital Normal University, China
[*]Corresponding author

***Abstract*—In order to solve the problem of remote image data transmission in AI development platform based on image analysis, an Ethernet controller based on SGMII was developed. Based on the open source MAC, the AXI bus interface is designed according to the requirements. The adapter converter between MII and the SGMII is designed. It has been implemented on the Virtex Ultrascale+ hardware development platform. The accuracy and effectiveness of the data transmission of the Ethernet controller are verified by the transmission of ARP and UDP network packets. The results show that the designed Ethernet controller can achieve reliable transmission of data at 100Mbps.**

Keywords—AI; AXI Bus; FPGA; Ethernet; SGMII

I. INTRODUCTION

With the development of artificial intelligence (AI), more and more AI accelerators have emerged, such as the Loihi chip introduced by Intel Corporation, the MLU smart chip developed by Zhongke Cambrian, and the Thinker chip designed by the Institute of Microelectronics of Tsinghua University [1, 2]. To support the development of AI chips, an AI development platform based on image analysis as shown in Figure 1 is proposed. The platform uses LEON2 processor as the system processor, AXI bus as the system bus, DDR4 as the storage module, and PCIe as the short-distance transmission channel. Compared with other communication interfaces, Ethernet has a faster transmission rate, and the speed is up to 100 Gbps; furthermore, Ethernet has a longer transmission distance and a wider transmission range as a widely used LAN technology[3].Therefore, Ethernet can bring high efficiency and reliability to the transmission of remote image data in the AI platform system.

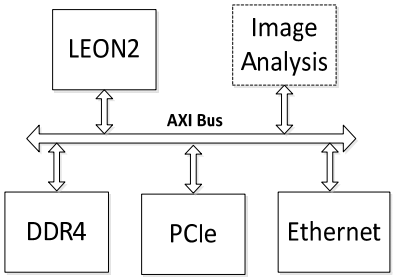

FIGURE I. AI DEVELOPMENT PLATFORM STRUCTURE

At present, most Ethernet design based on system-on-chip use Wishbone, AHB and other buses as system buses, such as the Ethernet IP core design implemented [4], and the dual-channel Ethernet media access control (MAC) [5]. As a new generation of system-on-chip bus developed by ARM, the AXI bus can further enhance the transmission performance of Ethernet and the AI platform. Xilinx's Virtex Ultrascale+ hardware development platform which supports SGMII was selected for this design. Compared with the parallel transmission interface, the SGMII protocol has stronger stability, and simplify the interface design [6]. In the design of Ethernet, many open source MAC simplify the design work, save the design cost, and most of the current open source MAC code is reliable. So this paper proposes an SGMII-based Ethernet design method using an open source MAC on an FPGA developed by Virtex Ultrascale+ hardware.

II. OPEN SOURCE MAC

In the design of the Ethernet controller, MAC is an indispensable part. This paper uses COBHAM's open source 10/100 Mbps MAC as the design basis. Its address channel and data channel are both 32 bits. The overall structure is shown in Figure 2[7].

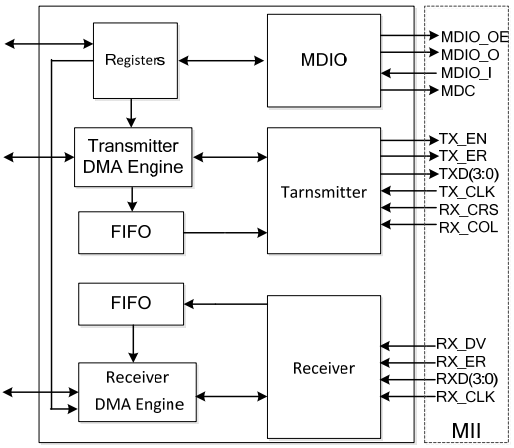

FIGURE II. OPEN SOURCE MAC STRUCTURE

The functions of each module in the figure are as follows:

- Direct memory access (DMA) controller: The DMA controller includes transmitter DMA engine and receiver DMA engine; it provides direct access to the system storage for the Ethernet module without CPU scheduling. Therefore, it greatly improves the storage access efficiency and improves the overall performance of the system[8, 9].

- First In First Out (FIFO): The FIFO module includes a transmitter FIFO and a receiver FIFO; it provides a space for temporarily storing data between the DMA controller and the MAC protocol processing module, and solves problems such as data path congestion caused by inconsistent rates.

- MAC protocol processing module: The MAC protocol processing module includes transmitter and receiver; the

transmitter provides functions such as encapsulation of a frame, and the receiver provides functions such as decapsulation of a frame.

- Registers: The registers provide the setting function of the MAC. Before starting the transmitting and receiving operations, the MAC must be initialized through the registers.

- Management data input/output (MDIO): The MDIO module provides the access logic of the physical (PHY) layer chip register through which the PHY can be set.

- Media independent interface (MII): The MII is one of the Ethernet interface protocol developed by the IEEE802.3 working group. It implements data transmission between MAC and PHY with 4-bits parallel data.

This design uses Xilinx's Virtex Ultrascale+ hardware development platform as the hardware foundation of the AI development platform. In order to make the open source MAC meet the design requirements, it is necessary to carry out the following transformation designs, which are the key works of this paper:

- The AI development platform selects the 64-bits AXI bus as the system bus. Therefore, this design will add the necessary AXI bus interfaces to the MAC.

- The selected Virtex Ultrascale+ hardware development platform provides a PHY chip based on the SGMII protocol, and the MAC provides the MII protocol. Therefore, to realize the data transmission between the MAC and the PHY, it is necessary to adapt the MII and SGMII.

III. ETHERNET DESIGN AND IMPLEMENTATION

A. Ethernet Overall Structure Design

According to the above analysis of the transformation design of the open source MAC, this paper proposes the overall structure diagram of the Ethernet controller as shown in Figure 3.

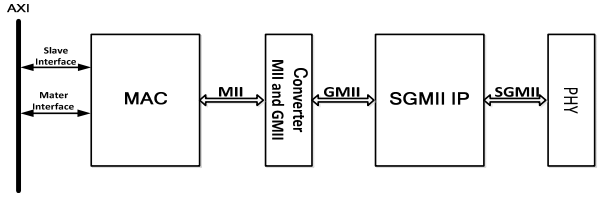

FIGURE III. ETHERNET OVERALL STRUCTURE

In the overall structure of the Ethernet controller, the DMA controller will act as the master device on the AXI bus, and the register module will act as the slave device on the AXI bus; in order to adapt the MII to the SGMII, this design selects the SGMII IP core provided by Xilinx for the Virtex Ultrascale+ hardware development platform and sets it accordingly. The selected SGMII IP core only provides Gigabit Media Independent Interface (GMII) on the MAC side. So this paper designed the converter between MII and GMII, to realize the data transmission between MAC and SGMII IP core. The other side of SGMII IP core directly connects with PHY chip to realize data interaction.

B. Design and Implementation of AXI Bus Slave Interface

The Ethernet registers are 32-bits, so the AXI bus always uses the lower 32 bits of the data and address to access the registers. According to the different access modes of the registers to the bus, this design divides the design of the AXI bus slave interface into reading timing design and writing timing design.

The Ethernet AXI bus slave device interface reading timing design is shown in Figure 4. In the reading and writing timing design of the slave interface, the writing operation always takes precedence over the reading operation, and the writing operation can interrupt the reading operation, so the design of the reading timing is divided into no waiting and waiting.

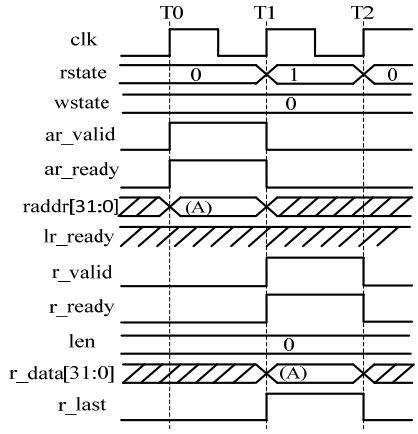

FIGURE IV. ETHERNET AXI BUS SLAVE DEVICE INTERFACE READING TIMING DESIGN

As shown in Figure 4, at time T0, the reading timing state machine (rstate) is in the idle state, at which time a reading operation request comes, ar_valid and ar_ready are set to 1 to indicate that the read channel handshake operation is started, and the address corresponding to the read data A is sent to the read address bus (raddr[31:0]). At time T1, since the register reading data ready signal (lr_ready) is kept high, r_valid and r_ready are set directly to indicate that the reading data channel handshake begins, the state machine enters the read data state, and the data A is sent to the read data bus (r_data[31:0]). At the same time, since only one 32-bits data is transmitted in this reading operation, the read last signal r_last is set to 1 to indicate that the current data is the last data of the current transfer. At time T2, the read operation is completed, and the state machine returns to the idle state.

Different from the reading timing design in the case of no waiting, there is a case where the reading timing design has a writing operation request when the reading operation is performed. Therefore, the corresponding writing operation is completed first, and the reading operation state machine continues the read data state, the read address channel and the read data channel extend the corresponding clock cycle; until the writing operation enters the write response channel handshake phase, the reading operation will continue.

Figure 5 shows the timing design of the AXI bus slave interface. At time T0, the state machine (wstate) is in the idle state. At this time, the writing operation request comes,

aw_valid and aw_ready are set to 1 to indicate that the write address channel starts the handshake operation, and the write data address is sent to the address bus (waddr[31:0]). At time T1, the state machine shifts to the write data state. Since the register write data ready signal lw_ready remains valid, w_valid and w_ready are directly set to 1 to indicate that the write data channel starts to handshake, and data A is transferred to the write data bus (w_data[31:0]). The current transmission only transmits a 32-bits word, so the last writing signal w_last is set to 1 to indicate that the current data is the last data. At time T2, the data transmission is completed, b_valid and b_ready are set to 1 to indicate the write response channel handshake start. At time T3, the write response channel has handshaked, and the data is successfully written.

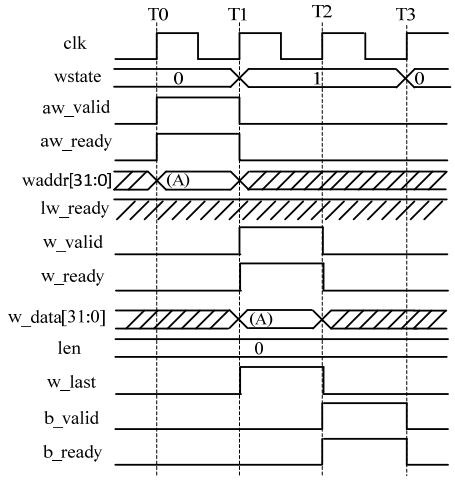

FIGURE V. ETHERNET AXI BUS SLAVE DEVICE INTERFACE WRITING TIMING DESIGN

C. Design and Implementation of AXI Bus Master Interface

According to the characteristics of the MAC, this paper will design the master interface between the DMA controller and the AXI bus. The DMA controller accesses system storage in a descriptor-based manner. It can access a series of non-contiguous addresses, which not only improves device efficiency but also reduces storage device requirements. The DMA obtains the first address of the data and the corresponding control information through the descriptors.

The DMA controller's access request to the bus is divided into TX and RX requests. The processing of the RX request of the master device interface always takes precedence over the processing of the TX request, but whether it is a TX or RX request both contain reading and writing access operations to the bus. The reading and writing accesses of the Ethernet AXI bus master interface are implemented by the same state machine. The states and descriptions of the state machine are shown in Table 1.

TABLE I. ETHERNET MASTER INTERFACE STATE MACHINE DESCRIPTION TABLE

State name	Status number	Description
IDLE	0	Idle state, bus interface has no read or write request
W	1	The first state of the write operation
R_WAIT_RESP	2	The first state of the read operation, waiting for data from the slave
R_WAIT_NEXT	3	Waiting for the last transmission from the device
B	4	Write end state

According to the characteristics of reading access, this design divides the AXI bus master reading operation timing design into no waiting and waiting. Taking the RX request as an example, the DMA controller's reading operation timing design for the AXI bus is shown in Figure 6.

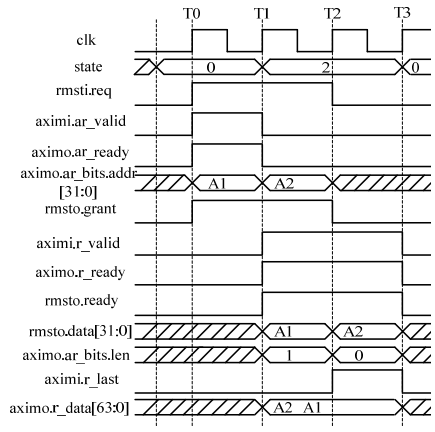

FIGURE VI. ETHERNET AXI BUS MASTER INTERFACE READING TIMING DESIGN

In order to comply with the access bandwidth of the DMA controller, the read address bus of AXI uses its lower 32 bits for address interaction, and the 64-bits read data bus of AXI sequentially transfers data to the 32-bits data bus of the DMA controller in accordance with the order in which the data arrives at the bus.

Figure 6 shows the reading timing design without waiting. The transmitted data is two 32-bits data A1 and A2. At time T0, the state machine is in the idle state, rmsti.req is set to 1 to indicate that the request in the receive direction is coming, and the request is a reading bus request, so aximi.ar_valid and aximo.ar_ready are set, and read address channel starts handshake operation. At the same time, the bus is authorized to the RX channel of the DMA, and the address corresponding to the data A1 is sent to the address bus (aximo.ar_bits.addr[31:0]). At time T1, aximi.r_valid and aximo.r_ready are set to 1, the bus read data channel performs a handshake operation, and rmsto.ready is set to 1 to indicate that the RX channel data is ready. At the same time, the state machine is transferred to the read data state, and the data (A1) on the lower 32 bits of aximo.r_data[63:0] is sent to the RX data channel rmsto.data[31:0]. At time T2, aximi.r_last is set to

1 to indicate that the current data is the last data of this transmission. At time T3, the transmission ends and the transmission is successful, and the state machine switches back to the idle state.

Different from the reading timing design without waiting, the bus reading operation on the RX channel has a request for a sudden withdrawal. Therefore, the address bus and the data bus must wait for the corresponding cycles, and the state machine enters the R_WAIT_NEXT state; until the RX channel resumes the bus reading operation request, the address bus and data bus continue to transmit until the transfer is completed.

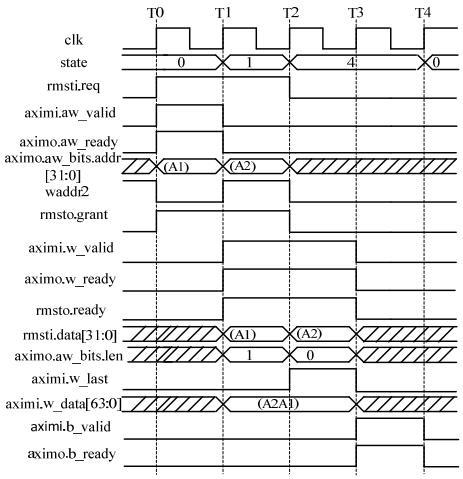

FIGURE VII. ETHERNET AXI BUS MASTER DEVICE INTERFACE WRITING TIMING DESIGN

Same as reading timing, the write address bus uses its lower 32 bits for address interaction in the writing timing design. The 64-bit write data bus determines whether the data from the DMA controller's 32-bits data bus channel is sent to the lower 32 bits or the upper 32 bits according to the data address 2.

Figure 7 shows the bus master device interface writing timing design on the RX channel. The data transferred is two 32-bits data A1 and A2. At time T0, the state machine is in the idle state, at which time rmsti.req is set to 1 to indicate that the request in the receive direction is coming, and the request is a write bus request, so aximi.aw_valid and aximo.aw_ready are set, and the bus write address channel starts the handshake operation. The bus is granted to the RX channel of the DMA. The address corresponding to the data A1 is sent to the write address bus (aximo.aw_bits.addr[31:0]). At time T1, aximi.w_valid and aximo.w_ready set to 1 to indicate the bus write data channel for handshake operation, and rmsto.ready is set to 1 to indicate that the RX channel data is ready, the state machine is transferred to the write data state. At the same time, the data A1 on rmsti.data[31:0] is sent to the lower 32 bits of the 64-bits data bus aximi.w_data[63:0]. At time T2, aximi.w_last is set to 1 to indicate that the current data is the last data of the transmission, and the state machine is transferred to the end write operation state. At time T3, the current transfer is completed, aximi.b_valid and aximo.b_ready are set to 1 to indicate that the write response channel to start the handshake operation. At time T4, the write response channel handshake is completed indicating that the data has been successfully written.

D. Design and Implementation of SGMII

The implementation of the SGMII relies on serdes, high-speed transceiving and other functions in the physical coding sub-layer (PCS) [10]. In the usual design, the designer will use the high-speed transceiver resources of the development platform to achieve high-speed transceiver function. The high-speed transceiver IP in the Virtex Ultrascale+ hardware development platform does not provide an interface to the PHY. Therefore, based on the detailed understanding of the hardware development platform, the design uses the SGMII IP resource to implement the corresponding functions of the PCS. Since the SGMII IP core only provides the GMII on the MAC side, this design also implements the converter between the MII and the GMII, and finally realizes the adaptation of the MII to the SGMII..

1) SGMII IP core settings

The settings of the SGMII IP core include mode, rate and its basic function selection. According to the design requirements, this paper selects the Gigabit SGMII IP core based on asynchronous LVDS, which supports 10/100/1000Mbps. In addition, the design adds extended MDIO and auto-negotiation.

The extended MDIO is an internal register group of the SGMII IP core. This register group implements several basic registers of the PHY, which facilitates the access of common registers and improves the working efficiency of the Ethernet.

At present, most network devices use Gigabit Ethernet, but the Ethernet controller implemented in this design is 100Mbps. Therefore, to achieve communication between network devices at different rates, an auto-negotiation function is required. The auto-negotiation can coordinate the devices at both ends of the communication to a unified working mode and working speed through the detection of the fast pulse link, which saves the trouble of manual setting and improves the intelligence of the design.

2) Design and implementation of converter between MII and GMII

The difference between the MII interface and the GMII interface is the difference in data bit width. The data bandwidth defined by the MII protocol is 4 bits, and the data bandwidth defined by the GMII protocol is 8 bits. Therefore, the main work of the converter is the conversion of data width. This design utilizes the difference in the number of counter counts to achieve data bit width conversion at 10Mbps and 100Mbpss respectively. According to the different data transmission directions, the converter design is divided into TX and RX. The conversion timing design at 100Mbps is shown in Figure 8.

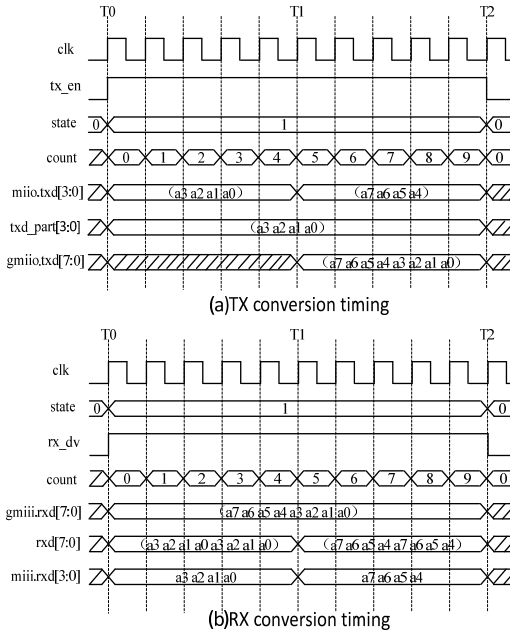

FIGURE VIII. MII AND GMII CONVERSION TIMING DESIGN

Figure 8(a) shows the conversion timing design in the TX direction. The transmitted data are two binary data a3a2a1a0 and a7a6a5a4. At time T0, tx_en changes from 0 to 1 to indicate that the transmission is valid, the state machine (state) transitions from the idle state to the transition state, and the data a3a2a1a0 is sent from miio.txd[3:0] to txd_part[3:0]. The counter (count) counts from 0. At time T1, the counter's value reaches 5, and the data a7a6a5a4 in miio.txd[3:0] is spliced together with the data a3a2a1a0 in txd_part[3:0] to gmiio.txd[7:0]. The two sets of 4-bits parallel data a3a2a1a0 and a7a6a5a4 are successfully converted into 8-bits parallel data a7a6a5a4a3a2a1a0. At time T2, the counter is cleared, and the state machine returns to the idle state when the current transfer ends.

Figure 8(b) shows the conversion timing design in the RX direction. The transmitted data is a binary data (a7a6a5a4a3a2a1a0). At time T0, the received data valid signal rx_dv is set to 1 to indicate that the data transfer is valid, the state machine enters the transition state from the idle state, the counter starts counting from 0, and the lower 4 bits of data gmiii.txd[7:0] (a3a2a1a0) are repeatedly transferred to the lower 4 bits and the upper 4 bits of rxd[7:0]. Then the lower 4 bits data (a3a2a1a0) of rxd[7:0] are transferred to miii.rxd[3:0]. At time T1, the counter's value reaches 5, the high 4-bits data (a7a6a5a4) of gmiii.txd[7:0] is repeatedly transferred to the lower 4 bits and the upper 4 bits of rxd[7:0], and the lower four bits of rxd[7:0] (a7a6a5a4) are dumped to miii.rxd[3:0]. At time T2, the conversion of the a7a6a5a4a3a2a1a0 from the GMII to the MII is completed, the counter is cleared, and the state machine is turned back to idle when the RX transmission is completed.

IV. TESTING RESULTS

The Virtex Ultrascale+ hardware development platform is shown in Figure 9. The network port is connected to the PC network port through a network cable. In this paper, the test of the Ethernet controller is divided into two parts: functional testing and performance testing.

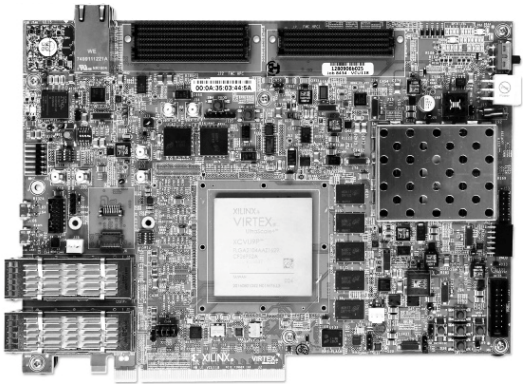

FIGURE IX. VIRTEX ULTRASCALE+ HARDWARE DEVELOPMENT PLATFORM

A. Function Testing

The purpose of functional testing is to verify whether the designed can successfully exchange data with other network devices. Therefore, according to the data transmission direction, this paper divides the functional testing into transmitting test and receiving test. Functional testing is accomplished using the transmission of ARP packets.

In the transmitting test, the logic data analyzer in the Vivado is used to observe the relevant data signals, and the Wireshark is used to capture the ARP packet which sent to the target PC. The observed data is compared to the data in the acquired network packet. After comparative analysis, the data on the data path is consistent with the packet data obtained in Wireshark., and the information such as the MAC address in the ARP packet conforms to the transmission the MAC address and other information in the ARP packet are consistent with the transmission environment and ARP protocol transmission rules. Therefore, the designed Ethernet controller can transmit data correctly.

In the receiving test, the logic data analyzer in the Vivado is also used to observe the relevant data signals, and the Wireshark is used to capture the ARP packet on the PC which sent to the Virtex Ultrascale+ hardware development platform. By comparing and analyzing, the data on the data path is exactly the same as the content of the data packet. Therefore, the designed Ethernet controller can receive data correctly.

B. Performance Testing

In order to verify the performance of the designed Ethernet controller, this paper uses the transmission of UDP data packets to test the actual transmission rate and packet loss rate. Set the length of the transmitted data packets to 1500 bytes. The test results are shown in Figure 10.

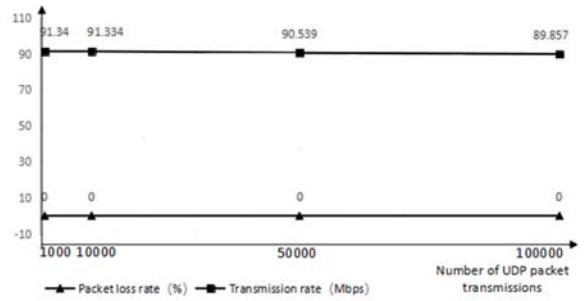

FIGURE X. ETHERNET CONTROLLER PERFORMANCE TEST LINE CHART

As shown in the line chart, the transmission error rate is always 0%, and the transmission rate will fluctuate to some extent when the number of transmitted UDP packets is different, but the basic transmission rate remains above and below 90Mbps. The rate obtained by the test is within a reasonable range, so the designed Ethernet controller performance meets the design expectations.

V. SUMMARY

This design uses the open source 10/100Mbps MAC on Xilinx's new Virtex Ultrascale+ hardware development platform to complete the interface design between Ethernet and AXI bus, and realizes the adaptation between MII of MAC and SGMII of PHY, and finally realizes the overall function of the Ethernet controller. It is verified that the data can be stably and correctly transmitted at a hundred-megabit rate through Ethernet controller. More importantly, the design of the Ethernet controller as the basis provides strong support for the implementation of the AI development platform. However, this design also has certain shortcomings. The use of the IP core provided by Xilinx has limited the reusability of the design. Therefore, in the future research, the related functions of the SGMII IP core should be further independently designed to improve the reusability of the design.

ACKNOWLEDGMENT

This research was financially supported by the National Natural Science Foundation of China, the Common Information System Equipment Advance Research Project, the Beijing Science and Technology Star Program, the State Key Laboratory of Architecture, the Beijing Future Chip Technology High-tech Innovation Center Research Fund Project and the Science and technology Innovation Service Capacity Building.

REFERENCES

[1] YIN Shougang,GUO Heng,GUO Shaojun. The Current Situation and Trend of the Development of Artificial Intelligence Chips[J]. Science and Technology Review,2018,36(17):45-51.

[2] Yin S, Ouyang P, Tang S, et al. A high energy efficient recon□figurable hybrid neural network processor for deep learningapplications[J]. IEEE Journal of Solid-State Circuits, 2018, 53(4): 968-982.

[3] ZHAI Dahai,YANGSHE Rongyuan.The development status and trend of high-speed Ethernet technology[J].Modern Transport, 2018 (01): 58-63(in chinese).

[4] JI Maosheng,ZUO Guohui,ZHANG Lina.Gigabit Ethernet Access Design Based on SoC FPGA[J].Computers and Networks, 2017,43 (Z1): 94-96(in chinese).

[5] WANG Yaqi.Design and Implementation of Time Triggered Ethernet Node Card Based on FPGA [D]. University of Electronic Science and Technology, 2017(in chinese).

[6] DAI Yulong,LI Jinbiao.Gigabit Ethernet Switching Design Based on 88E6185[J].Industrial Control Computer, 2017,30(08): 24-25+27(in chinese).

[7] GRLIB IP Core User's Manual Version 2017.3[S].

[8] ZHAO Qiang,CHEN Lan.Design of DMA Controller Based on ABB Bus Protocol [J]. Microelectronics and Computer, 2014, 31 (02): 129-131+136(in chinese).

[9] SHI Wenxia,WU Longsheng,SHENG Tingyi,AI Diao,CHEN Qingyu.Design of a Multi-Channel DMA Controller Supporting Full Duplex Data Transmission [J].Microelectronics and Computer, 2015,32(02): 76-79+83(in chinese).

[10] Serial-GMII Specification Revision 1.8[S].