IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 March 2024, accepted 16 March 2024, date of publication 25 March 2024, date of current version 9 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3380477

RESEARCH ARTICLE

A Lightweight and High-Throughput Asynchronous Message Bus for Communication in Multi-Core Heterogeneous Systems

QINGYANG ZENG[1,2], (Student Member, IEEE), JINGYU WANG[1,2], (Student Member, IEEE), JIAXU CONG[1,2], (Student Member, IEEE), AND DELONG SHANG[1,2,3]

[1]Institute of Microelectronics of the Chinese Academy of Sciences, Beijing 100029, China
[2]School of Integrated Circuits, University of Chinese Academy of Sciences, Beijing 100049, China
[3]Nanjing Institute of Intelligent Technology, Nanjing 211100, China

Corresponding author: Delong Shang (shangdelong@ime.ac.cn)

This work was supported in part by the Scientific and Technological Innovation 2030-Major Project under Grant 2021ZD0200300.

ABSTRACT In multi-core heterogeneous systems, communication on the data bus/NoC (Network-on-Chip) is complex. To ensure low-latency transmission of high-level messages, such as control messages, configuration instructions, and status information, while maintaining the efficiency of data bus/NoC transmission, we propose the concept of a message bus. In this paper, we present a lightweight and high-throughput asynchronous message bus capable of receiving and forwarding messages from different synchronous domains. A Quasi-Synchronous communication mechanism is proposed, where flits from the transmitter can be transmitted at fixed intervals without waiting for the ready signal from the receiver. This resolves the additional latency introduced by asynchronous handshaking, particularly in long-wire transmissions. Instead of flit-level handshaking, we introduce a packet-level flow control mechanism to avoid data overflow. Furthermore, we propose a novel Asynchronous Blocking Retransmission Buffer (ABRB) to address the communication congestion where subsequent packets are blocked by preceding ones. The packets can be sequentially written into the ABRB, enabling partial parallel transmission. For various benchmarks, the proposed asynchronous message bus achieves significant performance and energy consumption improvements compared to the synchronous baseline, at the cost of some additional area overhead and more design effort.

INDEX TERMS Asynchronous message bus, quasi-synchronous communication, asynchronous blocking retransmission buffer, globally asynchronous locally synchronous.

I. INTRODUCTION

In a multi-core heterogeneous system, CPUs, and accelerators inevitably need to exchange a large amount of data with the memory. The communication between CPUs and accelerators is complex, and currently, it is accomplished through data buses or NoCs. Complex data transfers can lead to increased congestion on the data bus or NoC [1]. In complex SoC systems, there is also a need to transmit high-level protocol messages between CPUs and accelerators such as control signals, status information, and configuration instructions. These messages have low latency requirements and high transmission priority. Transmitting these high-level protocol messages directly on the data bus/NoC will reduce system performance. To address this issue, the concept of a message bus is proposed in this paper. The message bus is a separate lightweight bus compared to the data bus/NoC, used for transmitting specific high-level messages. It can provide a more flexible and low latency communication mechanism.

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

As the area of the chip increases, the issues of global clock trees have become more prominent, resulting in additional system power consumption and timing overhead [2]. Different processors, accelerators, memories, and other modules operating at their optimal frequencies can achieve higher system efficiency. Therefore, it appears that GALS (Globally-Asynchronous Locally-Synchronous) systems are a more suitable solution [3], [4]. Asynchronous buses/NoCs play a crucial role in GALS systems by enabling data interaction between different synchronous domains. Compared to synchronous circuits, asynchronous circuits have inherent advantages in terms of low power consumption due to the elimination of a global clock and their event-driven characteristic [2]. They can easily connect blocks operating at different voltages and frequencies ignoring the clock alignment at the interface. Consequently, in recent years, there has been an increasing interest in asynchronous NoC (Network-on-Chip) research [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20]. Numerous asynchronous NoC architectures have been proposed, demonstrating significant reductions in area and power consumption compared to synchronous NoCs while maintaining comparable performance [10], [11].

The aforementioned NoCs typically employ traditional asynchronous communication protocols for data communication, including Bundled Data (BD) and Quasi Delay Insensitive (QDI). Some asynchronous NoCs utilize the BD transmission protocol [5], [6], [7], [8], [9], [14], where asynchronous pipelines such as Click, Mousetrap, and Latch Controller are used as control paths to generate local clock signals. To ensure functional correctness, additional timing constraints must be met between handshake signals and data. On the other hand, other asynchronous NoCs [15], [16], [17], [18], [19], [20] employ QDI transmission protocols that do not require additional timing constraints to maintain the relationship between clocks and data. Special encoding binds data and request signals together. Data validity is detected based on changes in the encoded signal, such as dual-rail encoding, 1-of-4 encoding, 2-of-7 encoding, and so on. This approach exhibits excellent robustness, but it introduces additional area and power overhead due to reduced encoding efficiency.

In traditional asynchronous transmission, the handshake signals are used to ensure the completion of data transmission, which increases the communication latency of the data. Especially for long-wire communication, each transmission incurs double the wire delay or even worse, leading to a significant decrease in system throughput. While in synchronous transmission, ignoring global clock overhead, the clock period just needs to ensure that the data can be received by the receiver. Inspired by this, an innovative Quasi-Synchronous communication mechanism is proposed in this paper to eliminate the handshake time of traditional asynchronous transmission and improve throughput.

In bus-based architectures, there are multiple initiators accessing the same target, leading to communication congestion. Many of the aforementioned asynchronous NoCs or buses lack effective congestion handling mechanisms, resulting in decreased system communication efficiency. Some NoCs [5], [6], [24], [25], [26] employ virtual channel techniques to alleviate congestion, but this approach incurs significant area and power overheads. To alleviate the issue of communication congestion, a novel Asynchronous Blocking Retransmission Buffer is proposed in this paper.

A. CONTRIBUTIONS

In this paper, we propose a lightweight message bus for transmitting high-level messages in a multi-core heterogeneous system. The message bus is designed using asynchronous techniques, and data transmission is performed in a serialized manner using 2-of-7 non-return-to-zero encoding. The message bus utilizes a Quasi-Synchronous communication protocol for data transmission, along with the Asynchronous Blocking Retransmission Buffer and a novel flow control mechanism, resulting in significantly improved throughput. In summary, the contributions of this paper are as follows:

1. We have implemented an asynchronous lightweight message bus with high throughput using traditional EDA tools in a 55nm process. The throughput of the Routing-Node reaches 6.27 Gbps when no blocking occurs, and in the random data traffic model, it achieves a throughput of 4.69 Gbps. The proposed asynchronous message bus exhibits an 85.3% system throughput improvement compared to the synchronous baseline version, with an additional 38% area overhead and the cost of more design effort.

2. Data transmission is performed in a serialized manner to reduce area overhead. We employ 2-of-7 Non-Return-to-Zero encoding, where a single link with seven wires transmits 4-bit data. Only 2 bits change during each data transmission on the link, leading to reduced power consumption.

3. To reduce the transmission cycle overhead caused by asynchronous handshake, a novel Quasi-Synchronous communication mechanism is proposed. It allows data to be transmitted at a pre-set time, eliminating the feedback of acknowledgment signals, similar to synchronous transmission, reducing the transmission time per flit, while still maintaining asynchronous event-driven characteristics. Instead of a flit-level handshake signal, this paper proposed a novel packet-level flow control mechanism to prevent packet overflow. The control information is transmitted through the data channel using a special encoding to complete the back pressure of data.

4. To address the issue of throughput decline caused by communication congestion when multiple initiators access the same target, we propose an Asynchronous Blocking Retransmission Buffer (ABRB) that can receive multiple packets and concurrently transfer them to their respective destinations. The ABRB consists of the proposed novel asynchronous FIFOs and asynchronous control logic. The asynchronous control logic significantly reduces the occurrence of back pressure, resulting in high throughput.

II. BACKGROUND

A. ASYNCHRONOUS DATA COMMUNICATIONS

Within a synchronous circuit, synchronization is global and implicit through the use of a global clock signal. Conversely, synchronization is accomplished explicitly, through handshake signaling protocols in an asynchronous circuit. Depending on the timing requirements, asynchronous circuits can be classified into ''Bounded-Data(BD)'' and ''Delay-Insensitive(DI)/Quasi-Delay-Insensitive(QDI)'' protocols. In BD protocol, the ''req'' signal is separated from the data itself. It is related to a local clock signal and the timing assumptions are necessary. On the other hand, in the DI/QDI protocol, data and ''req'' are fully correlated [22], [23], so no timing assumptions are required. Another acknowledgment (ack) signal is used for flow control and back pressure. It exhibits strong robustness to process, voltage, and temperature (PVT) variations. There are many QDI data encodings including one-hot codes (dual-rail, 1-of-4, 1-of-5), 2-of-7, 3-of-6 and so on [21].

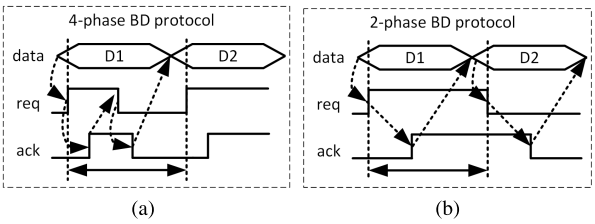


FIGURE 1. The asynchronous communication protocol - Boundled-Data. (a) 4-phase handshake protocol; (b) 2-phase handshake protocol.

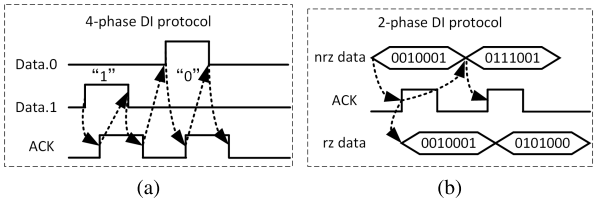


FIGURE 2. The asynchronous communication protocol - Delay-Insensitive. (a) 4-phase handshake protocol; (b) 2-phase handshake protocol.

Among various asynchronous data communication protocols, the most common handshake protocols are four-phase and two-phase handshake protocols, as illustrated in the FIG.1 and FIG.2 respectively. The four-phase handshake offers a simpler circuit structure, while the two-phase handshake provides higher communication efficiency.

FIG.1a illustrates the 4-phase BD transmission protocol. The transmitter detects the ready state of the receiver and then transmits the new data and ''req'' signal to the receiver. The receiver detects the ''req'' signal, raises the ''ack'' signal, and receives the current data. Upon detecting a level change of the ''ack'' signal, the transmitter lowers the ''req'' signal. If the receiver detects that the current data is invalid, it lowers the ''ack'' signal and returns to the ready state. Once the transmitter detects that the receiver is ready again, it can initiate the next data transmission.

FIG.1b depicts the 2-phase BD transmission protocol. It differs from the 4-phase BD protocol in that the changes in ''req'' and ''ack'' signals are represented by edges rather than levels. The two-phase transmission protocol does not require a spacer phase and can achieve higher performance, but it involves more complex circuit implementation compared to the 4-phase protocol.

FIG.2a represents the 4-phase DI transmission protocol. The data is transmitted using dual-rail encoding, where two lines represent 1-bit data. ''00'' indicates no data transmission, ''01'' indicates data '0', ''10'' represents data '1' and ''11'' denotes invalid data. Unlike 4-phase BD protocol, the DI protocol encodes the ''req'' signal in the data. Upon detecting a change in the dual-rail data, the receiver provides feedback with the ''ack'' signal following the four-phase handshaking protocol.

FIG.2b showcases the 2-phase DI protocol. The data transmission does not require a spacer phase, resulting in higher performance. The specific transmission process of 2-of-7 encoding will be further presented below.

B. DI PROTOCOL: 2-OF-7 ENCODING

The 2-of-7 encoding is a classical encoding technique in asynchronous circuit design, representing 4-bit data with 7 wires. During data transmission, only 2 bits change at a time, resulting in reduced power consumption. Due to its ease of detection and low power characteristics, 2-of-7 encoding is commonly utilized in the design of asynchronous routers and circuits [3], [21], [22], [23]. The 2-of-7 encoding has 21 valid symbols, with 16 symbols used for encoding 4-bit data. The remaining 5 symbols are used for various purposes. Usually, one symbol is dedicated to representing EOP (End of Packet). Another symbol, referred to as ''FULL'' in this paper, is used to provide a new flow control mechanism.

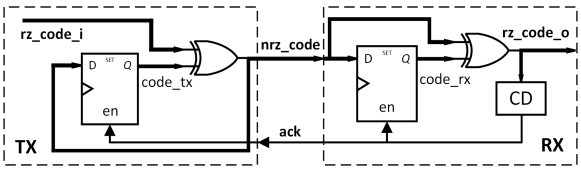


FIGURE 3. The circuit implementation of 2-of-7 DI transimission protocol.

The circuit of the 2-of-7 DI transmission protocol is shown in FIG.3. At the TX (transmitter) side, the data to be transmitted is encoded into a 2-of-7 Return-to-Zero (RZ) code, denoted as rz_code_i. This rz_code_i is then XORed with code_tx to generate the NRZ code called nrz_code. It represents the encoded data without zero crossings. The NRZ code nrz_code is transmitted to the RX (receiver) side, where it is XORed with code_rx to obtain the RZ code rz_code_o. The RX receives the rz_code_o and uses a completion detection (CD) circuit to generate an ack signal, indicating that the current data has been received by the RX

successfully. The ack signal updates code_rx to nrz_code, and this updated value is fed back to the TX side to update code_tx to nrz_code. The process is repeated for the next data to be transmitted.

III. OVERVIEW OF THE ARCHITECTURE

The proposed asynchronous message bus that supports Quasi-Synchronous communication is now introduced, including its novel flow control mechanism without flit-level handshake, a novel Asynchronous Blocking Retransmission Buffer that enables partial parallel transmission of packets, before presenting the detailed designs.

A. QUASI-SYNCHRONOUS COMMUNICATION

Chip-wide synchronous operation is becoming prohibitively difficult to achieve in large, high performance chip [2]. Possible solutions are in direction of asynchronous communication, since they rely on the clockless handshaking for inter-domain communication. Asynchronous communication comes with a number of potential advantages: self-timed, data-driven control, and no dynamic power consumption when idle. Traditional asynchronous communication uses the handshake protocol to control communication. It will cause a large overhead of cycle time due to the long-wire latency of the handshake signals.

1) QUASI-SYNCHRONOUS COMMUNICATION PROTOCOL

FIG.4 compares the asynchronous four-phase handshake communication protocol with the proposed Quasi-Synchronous communication protocol under conditions of long wire delay. In the case of the asynchronous four-phase handshake protocol, the cycle of the flit transmission includes not only the time required for the four handshake phases but also four times the wire delay, resulting in a lower communication throughput. We propose a Quasi-Synchronous communication that effectively improves the throughput of asynchronous communication. It allows data to be transmitted at a pre-set time, eliminating the feedback of acknowledgment signals, similar to synchronous transmission, reducing the transmission time per flit, while still maintaining asynchronous event-driven characteristics. The specific implementation is described as follows.

Upon the arrival of valid data, the transmitter sends the newly arrived data at a fixed interval without waiting for the receiver to be ready. However, it is assumed that the receiver can consistently receive new data at this frequency. This fixed delay is implemented through an internal circuit in the TX. On the other hand, for each newly detected data, the receiver generates a clock pulse that meets the timing requirements of the current data. The receiver does not need to interact directly with the transmitter but passively waits for new data to arrive. In the absence of new data, both the transmitter and receiver remain in an IDLE state, performing no operations, resulting in no energy consumption. This novel communication protocol ensures the event-driven characteristics of asynchronous communication maintaining the essence of low power consumption, and guarantees the throughput of data communication when transmitting continuous new data.

2) THE FLOW CONTROL OF THE QUASI-SYNCHRONOUS COMMUNICATION

A back pressure mechanism is employed to control the communication and maintain the communication correctness and data throughput. In conventional synchronous buses like ARM's AMBA bus [28], back pressure is achieved through a "READY" signal, and this signal needs to be synchronized in clock domain crossing communication. While in asynchronous communication, "req" and "ack" are used as handshake signals. As mentioned above, to reduce the transmission latency of flits, we proposed a quasi-synchronous communication that eliminates the feedback of acknowledgment signals. In the absence of acknowledgment signals, we proposed a new packet-level flow control mechanism to ensure no packet overflow. The overflow signal is transmitted through the data channel using a special encoding.

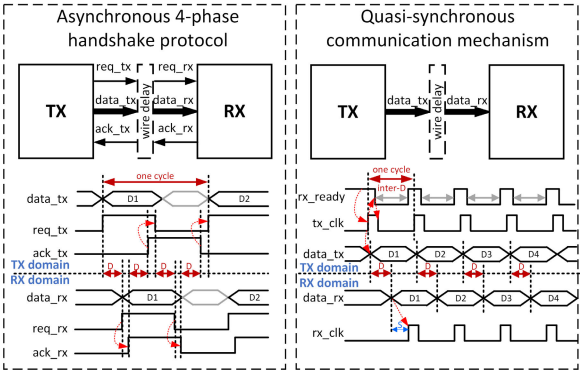


FIGURE 4. The timing diagram of asynchronous four-phase handshake communication and the proposed Quasi-Synchronous communication.

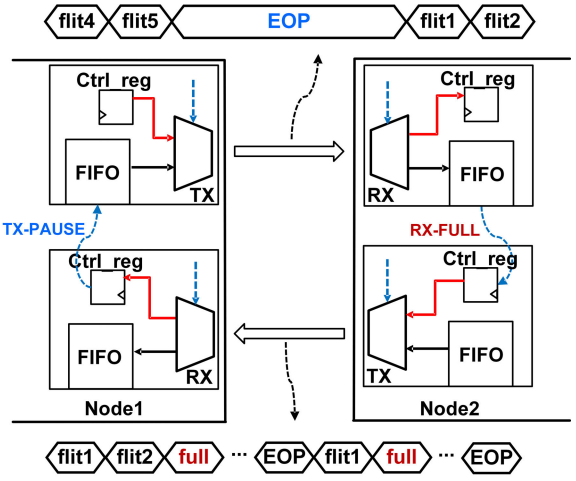


FIGURE 5. The novel flow control mechanism in the Quasi-Synchronous communication.

The specific process is depicted in the FIG.5. When a packet arrives at the RX and encounters blocking, the RX signals the TX on the same side with an RX-full signal, denoting that the RX is unable to accept a new packet (still able to receive the current packet). Upon receiving the RX-full, the TX on the same side encodes it as a full information flit, inserting it into the outgoing packet and transmitting it to the RX on the opposite side via the data channel. Upon reception, the full information flit is decoded into the TX-PAUSE signal by the RX on the opposite side, instructing the TX on the opposite side to suspend the transmission of the next packet until the current packet is completely transmitted. This flow control process is faster than a single packet transmission time, so no data packet loss will occur.

B. ASYNCHRONOUS BLOCKING RETRANSMISSION BUFFER

In the traditional crossbar structure, as shown in FIG.6, each input port has access to an output port. The system achieves maximum throughput when there is no blocking. However, when two or more input ports request the same output port, only one input port among them can establish communication with the output port. The other input ports have to pause until the destination output port is released, even if the subsequent packets request different output ports. Additionally, when multiple ports request the same output port, the other output ports remain idle. As a result, the system throughput deteriorates. As shown in FIG.6, the packet from the S direction to the N direction is being blocked for 5 packet cycles, even though there is no many contention from other sources in the N direction.

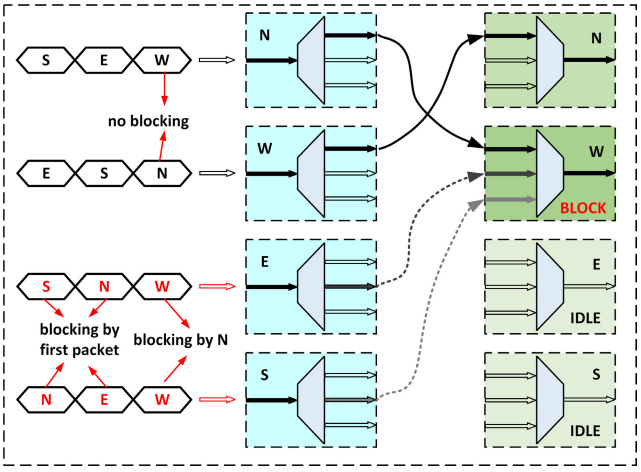


FIGURE 6. Blocking analysis of the traditional crossbar.

We introduce an Asynchronous Blocking Retransmission Buffer (ABRB) to avoid communication pauses caused by packet blocking. The structure of the ABRB is depicted in FIG.7. For a routing node, simultaneously blocking packets heading in three different directions and then restoring their transmission simultaneously yields maximum benefits for the ABRB. Therefore, there are three FIFOs in the ABRB where the FIFOs are implemented using redesigned asynchronous FIFOs and have a depth of one packet, allowing a maximum of three packets to be stored. The specific operation is described as follows:

Firstly, packets are written into different FIFOs in the ABRB. The first packet is initially written into FIFO1 and requests the destination output port. If the output port is currently transmitting packets from other input ports, the first packet will be blocked in FIFO1. The subsequent packet will be written into FIFO2. If the subsequent packet has the same destination address as the first packet, it must wait for FIFO1 to complete transmission before requesting access to the output port. Conversely, if it has a different destination address, it can independently request access to the output port corresponding to its destination address. At this point, no blocking has occurred for the current input port. As shown in FIG.7, the 12 packets from four directions are transmitted within four cycles. The packet from the S direction to the N direction is successfully transmitted in the third cycle without being blocked by previous packets.

Traditional asynchronous FIFOs use synchronized read and write pointers to generate empty and full signals, with the full signal serving as the back pressure signal. In an asynchronous system, where the write clock and read clock are asynchronous events, it is not possible to synchronize the read and write pointers. The read clock can occur at any time, and the signal length of the full signal cannot be guaranteed.

A new back pressure mechanism is proposed to ensure that the FIFO does not overflow while maintaining high throughput in the ABRB. Data from TX can continue to be transmitted until only one FIFO remains empty in the ABRB. In other words, when the first flit of the packet from TX is written into the last empty FIFO, the transmission pauses after the current packet is completed and resumes only after all the packets in the ABRB have been read. This approach is taken because when two input ports are blocked, the duration of the blockage is one packet transmission time. The blocked packet has completed transmission before the second packet is fully written into FIFO2, so when the third packet is written, there are two empty FIFOs and the input port will not be blocked.

The ABRB effectively improves the system throughput. Input ports no longer need to enter the flow control mechanism every time a packet is blocked. Instead, transmission is paused after receiving up to three blocking packets. In most cases, data transmission continues uninterrupted, and blocking only occurs in rare situations.

IV. IMPLEMENTATION

The asynchronous message bus includes the Sync-Async-Host and Routing-Node. The Sync-Async-Host is responsible for receiving and sending packets to the synchronous domain, while the Routing-Node transmits flits to other IP domains. By changing the number of routing nodes, the asynchronous message bus can connect multiple domains. N routing nodes can connect to a maximum of $2(N+1)$ domains.

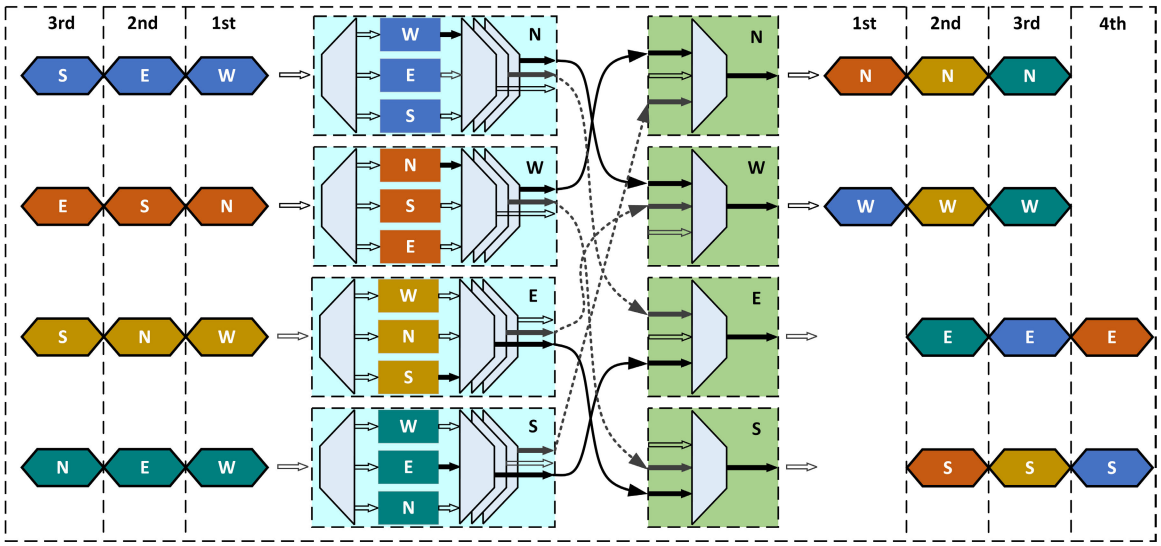


FIGURE 7. Blocking analysis of the proposed new crossbar architecture with ABRB.

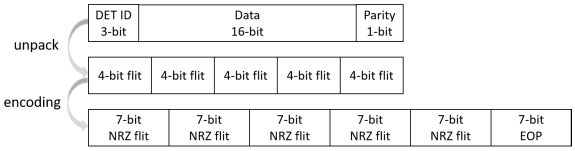


FIGURE 8. The format of the packet and the serial transmission.

A. THE FORMAT OF PACKETS

Data from a synchronous domain is transmitted in packets with a length of 20 bits. The detailed format of a data packet is shown in FIG.8. For lightweight, serial transmission is used in communication between different synchronous domains, and the 2-of-7 encoding is used to lower the power consumption. Therefore, a 20-bit packet will be unpacked to 5 4-bit flits, which, after encoding, will be transmitted within a 7-bit transmission channel.

B. THE SYNC-ASYNC-HOST

The Sync-Async-Host is an interface between the synchronous domain and the asynchronous domain. It consists of two parts, Host-TX and Host-RX. Host-TX can receive packets from synchronous domains and send flits to asynchronous domains and Host-RX receive flits from asynchronous domains and send packets to synchronous domains. This part is not the focus of this article, so it will not be described in more detail.

C. THE ROUTING-NODE

The Routing-Node receives flits and sends them to other Sync-Async-Hosts of synchronous domains. Each routing node has four directions, allowing it to be connected to form any topology, providing good scalability. It consists of Routing-Node-RX, Routing-Node-TX. The Routing-Node-RX module includes routing logic to determine the destination of the packet, an Asynchronous Blocking Retransmission Buffer to alleviate congestion, and a RX-CLK generator to convert the received Non-Return-to-Zero(NRZ) data into Return-to-Zero(RZ) data, and picks out the control data for realizing flow control. The Routing-Node-TX module receives input data from various directions and grants one of them. It detects the arrival of RZ data, generates a local clock to drive NRZ data transmission, and inserts control data into the data channel according to the requirements of flow control. The details of the above modules will be presented in the next two sections.

D. ROUTING-NODE-RX

The Routing-Node-RX mainly consists of the RX-CLK generator, the routing logic, the ABRB, and other logic. The RX-CLK generator first detects the arrival of a new flit and generates a clock pulse signal RX-CLK. The routing logic then examines the address information contained in the head of the flit and stores the address information in an address register. Simultaneously, the current flit is written to the ABRB corresponding to the current address register. The Routing-Node-RX continues this process until no new flits arrive. The routing logic is not the primary focus of this paper and will not be extensively discussed below.

1) THE GENERATION OF RX-CLK

As shown in FIG.9 and FIG.10, upon the arrival of new data, it is XORed with the value in the register. Due to the nature of the 2-of-7 encoding, if the XOR result is non-zero, it indicates that the current data is new data. After a stable delay(D in FIG.9), which should satisfy the setup and hold time requirements of the register, the clock pulse signal RX-CLK is generated. If the XOR result corresponds to the RX-FULL control information, it is decoded into the TX-

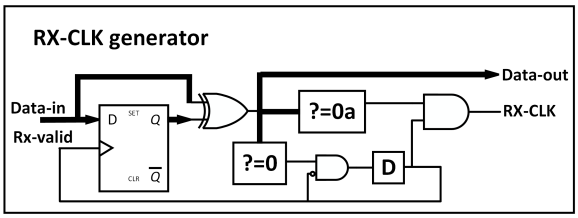


FIGURE 9. The circuit structure of the RX-CLK generator in Routing-Node-RX.

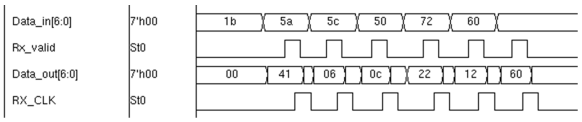


FIGURE 10. The post-layout simulation waveform of the RX-CLK generator in Routing-Node-RX.

PAUSE signal. Subsequently, the current address is stored in the address register using the following RX-CLK signal, and the current data is written into the ABRB. When there is no new data arrival, the clock generation module remains inactive, consuming no power.

2) ASYNCHRONOUS BLOCKING RETRANSMISSION BUFFER

To further increase the system's throughput, this paper proposes the asynchronous blocking retransmission buffer, as shown in the FIG.11. It consists of a redesigned asynchronous FIFO and control logic. The asynchronous FIFO is used to store data packets and separates two different clock domains, enabling data transfer between them. The control logic consists of a flow control module, a FSM, and a data channel arbiter. The control logic is responsible for controlling the read and write order of data packets in the FIFO, and implementing the back pressure mechanism.

3) A NOVEL ASYNCHRONOUS FIFO

We propose a novel asynchronous event-driven FIFO structure in our asynchronous message bus, as illustrated in FIG.12. The asynchronous FIFO includes a write pointer update module in the RX-CLK clock domain, a read pointer update module in the TX-CLK clock domain, and FIFO empty/full control logic. For each data in the FIFO, there are two registers indicating whether it has been read or written, thus representing the empty/full status of that slice. The overall empty/full status of the FIFO is determined accordingly. This approach avoids the need for synchronization operations using clocks from different domains in traditional asynchronous FIFOs.

The specific operation process is shown in FIG.13. When new data arrives, the flit is written into the FIFO under the drive of RX-CLK, and the write pointer is updated. For each slice in the FIFO, its initial state is empty. After data is written, the slice's state becomes full, and after data is read, the slice's state becomes empty again. The full state of a slice indicates that it contains valid data. This valid signal, along with the current flit, is passed to the TX, generating the TX-CLK. After the current flit is read by TX-CLK, the read pointer and the empty/full status of the current slice are updated. For an individual slice, it is undoubtedly a write-then-read operation. For the entire FIFO, read and write operations can be performed concurrently.

The proposed FIFO exhibits asynchronous data-driven characteristics, where the read and write clocks are generated based on data. All of these write or read processes do not incur power consumption when there is no data transfer.

4) THE CONTROL LOGIC IN ABRB

The read and write order of the FIFO, as well as the back pressure of the ABRB, are determined by the control logic. It has a great impact on the system performance and function correctly. The read and write order should follow the following principle. First, a FIFO follows the principle of first in, first out. Second, for the ABRB, packets with the same destination address need to be read from the destination port in strict accordance with the order in which they were written to ensure the correct basic functionality of the bus. The read order at each port strictly follows the arrival order of the data destined for that port, which is achieved through the data channel arbiter. Third, for the ABRB, packets with different destination addresses can be accessed in parallel by their respective destination ports. This effectively improves the average packet transmission cycle.

Since the writing and reading of packets occur asynchronously, determining whether the buffer needs back pressure is a relatively complex but crucial task. In synchronous circuits, D Flip Flop(DFF) is used for event detection. However, applying this method to asynchronous circuits can lead to highly dangerous metastability. Therefore, to achieve event detection in asynchronous circuits, we innovatively use MUTEX instead of DFF to detect asynchronous events, effectively avoiding the occurrence of metastability.

The order of writing data is ensured by the FSM (Finite State Machine) as shown in FIG.11, where W1, W2, and W3 represent the current packet being written into FIFO1, FIFO2, and FIFO3, respectively. The data follows the overall sequence of 1, 2, and 3, taking into account the availability of the current FIFOs. This is primarily achieved through the combination of the FSM and MUTEX. The MUTEX serves as an event detector, specifically to check whether there is an available empty FIFO to receive new data in the current state. N FIFOs require $N(N\text{-}1)$ MUTEX units to form the control unit in ABRB. This introduces additional area and power overheads, impacting packet transmission latency as well.

The back pressure mechanism of the ABRB shown in FIG.11 follows this principle: during the writing of packets, if the ABRB still has a capacity greater than one packet, it can continue to receive new data. Otherwise, a back pressure signal needs to be generated to pause transmission until the buffer is cleared. At the moment when FIFO1 becomes empty, FIFO2 is also empty, and the data is written into FIFO2. If FIFO2 is not empty but FIFO3 is empty, the data

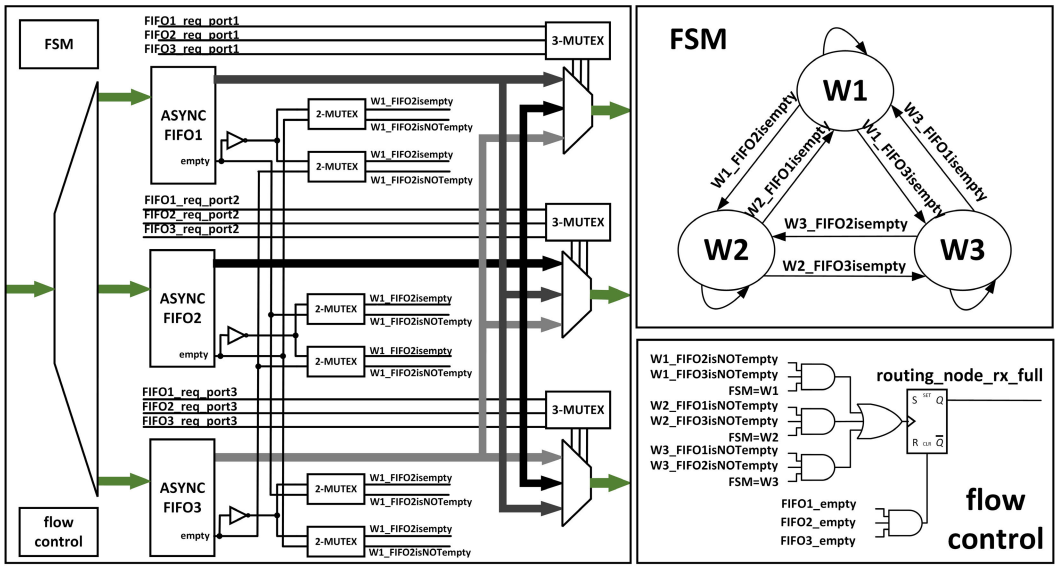


FIGURE 11. The architecture of the proposed asynchronous blocking retransmission buffer.

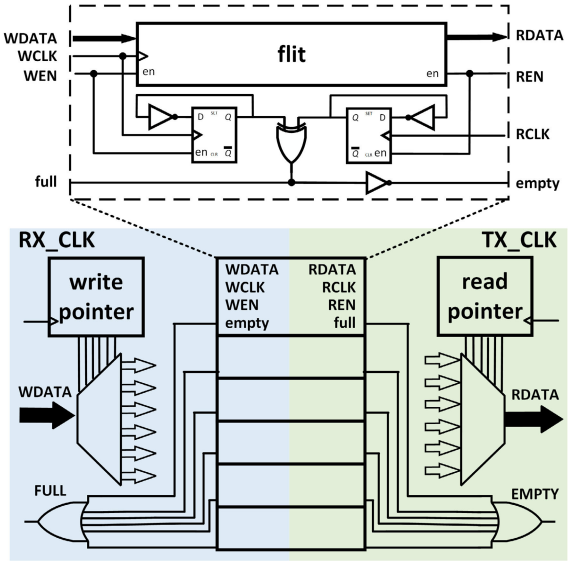


FIGURE 12. The detailed architecture of the proposed asynchronous FIFO.

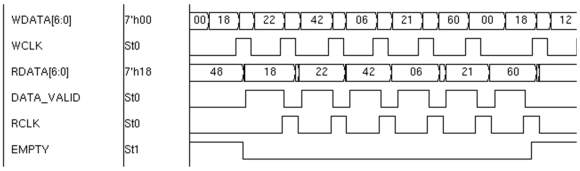


FIGURE 13. The post-layout simulation waveform of the proposed asynchronous FIFO.

is written into FIFO3. If both FIFO2 and FIFO3 are not empty, a back pressure signal RX-FULL is generated. The data writing process for FIFO2 and FIFO3 follows the same principle.

E. ROUTING-NODE-TX

The main component of Routing-Node-TX is the TX-CLK generator. As mentioned earlier, the TX exhibits both asynchronous event-driven characteristics and synchronous fixed-time sending characteristics. Another crucial function of Routing-Node-TX is to implement the flow control mechanism by inserting RX-FULL information flits into normal packets. What's more, upon receiving the TX-PAUSE signal, the transmission is paused after the current data packet is transmitted.

1) THE GENERATION OF TX-CLK

To achieve a Quasi-Synchronous communication of flits, Routing-Node-TX requires an internal delay that matches the timing interval for transmission. When valid data arrives, the valid signal handshakes with the internal ready signal and generates the TX-CLK. The flow control mechanism is implemented through a MUTEX to select between data flits and control flits, generating the data clock and control clock, respectively. When the TX-PAUSE signal is present, the data clock is inhibited, pausing the transmission while the current flit continues to be transmitted. The specific circuit structure for Routing-Node-TX is illustrated in the FIG.14.

2) THE PROCESS OF DATA TRANSMISSION

As shown in FIG.15, when the valid signal Nvalid arrives, the Nreq signal is generated through MUTEX-pause when there is no TX-PAUSE, followed by the Ngnt signal through MUTEX-full when there is no RX-FULL, indicating that new data is ready without any pause or control flit insertion required. The TX-READY signal is generated internally in the TX and is used to indicate that the receiving side (Routing-Node-RX) is ready to receive new data. The Ngnt signal represents that the current data to be sent is ready. The

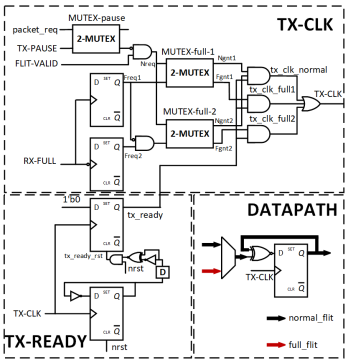


FIGURE 14. The circuit of Routing-Node-TX.

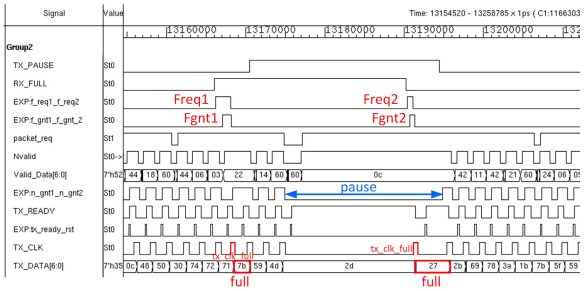


FIGURE 15. The post-layout simulation waveform of data transmission in the Routing-Node-TX.

TX-CLK signal is generated through a handshake between the Ngnt and TX-READY signals. Using the 2-of-7 encoding scheme, the new data is XORed with the current data and then transmitted. After the data is sent, the TX-READY signal is pulled low until the next-level RX successfully receives the data, and then it is pulled high again. The reassertion of the TX-READY signal is achieved through an internal delay in the TX. The T Flip Flop is triggered by TX-CLK, generating an edge. This signal, after an internal delay, is XORed with its complement to obtain a reset pulse.

3) THE PROCESS OF CONTROL FLIT INSERTION

When the RX-FULL signal wins the MUTEX-full, indicating that valid data arrives after the control flit, the Fgnt signal is generated. When Fgnt and TX-READY are both high, the TX-CLK-full signal(red TX-CLK in FIG.15) is generated, and the "full" information is inserted in the data channel using a 2-of-7 encoding scheme. The process of TX-READY signal generation is the same as described earlier.

4) THE PROCESS OF TRANSMISSION PAUSE

The packet-req represents the current packet transmission being valid. The TX-PAUSE and packet-req compete through MUTEX-pause, and the result of MUTEX-pause is only generated after packet-req goes low, i.e., after the current packet transmission is completed and before the next packet transmission. The TX-PAUSE signal stops the generation of the normal clock but does not affect the generation of the control clock since the flow control needs to be executed even during the transmission pause state.

5) THE MUTEX DESIGN

In our proposed asynchronous message bus, MUTEX plays a crucial role. In addition to serving as the arbiter for output ports, MUTEX also acts as the arbiter for data channels, determining the reading order of data in ABRB. The structures of the 2-MUTEX used in this paper are illustrated in the FIG.16. It employs a four-input NOR gate as a metastability filter [27]. Compared to previous arbiters, this type of arbiter is constructed using standard cells, making it easier to integrate into the system.

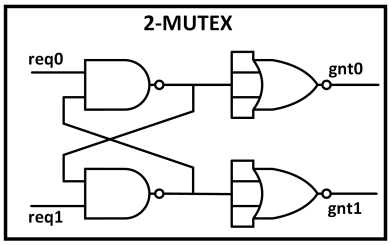


FIGURE 16. The circuit of 2-MUTEX used in this paper.

V. EXPERIMENTAL AND RESULTS

The Routing-Nodes and Sync-Async-Hosts are implemented using standard cells at the 55nm process node. The entire design involves traditional EDA tools such as Synopsys DC for circuit synthesis and Synopsys ICC for layout and routing. The layout of the asynchronous message bus in FIG.17 identifies the different network topologies and the functional blocks in the Routing Node. Due to the local clocks in this design instead of a global clock, the clock tree synthesis and timing analysis become complex. We employ a combination of static timing analysis using EDA tools and manual timing analysis to do this, which requires more design effort compared to synchronous designs. Partial timing constraints in synthesis are shown in FIG.18. The delay values required to be inserted for Quasi-synchronous communication are obtained by Static Timing Analysis. Subsequently, *set_min_delay* insert delay cells at layout and

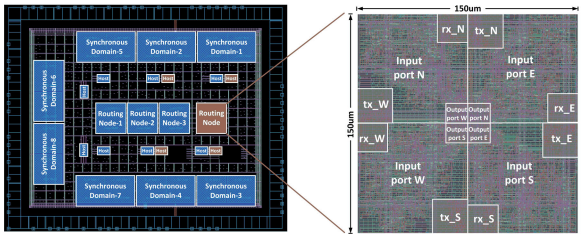


FIGURE 17. The layout of asynchronous message bus with different network topology and the functional blocks in the routing node. The orange part is the simple network and the blue part is the complex network.

Partial timing constrains in synthesis

1.create_clock # for all RX-CLK and TX-CLK
2.synthesis
3.report_timing # get delay
α = ***max_delay*** (from TX-CLK to DATA_IN)
β = ***min_delay*** (from TX-CLK to RX-CLK)
4. Ω = α - β # value of insert match delay
5.set_min_delay Ω from set_tx_ready to set_tx_ready_d
insert delay cell
6.synthesis

FIGURE 18. **Partial timing constrains in synthesis.**

routing. Post-layout simulation is performed to obtain the area, energy consumption, and system performance.

A. COMPARISON WITH STATE-OF-THE-ART

We follow a similar approach as in previous studies [8], [14], [16], measuring the static performance of the message bus and comparing it with existing works. This method provides a better reflection of the message bus's performance at the circuit level, independent of factors such as topology and routing algorithms. We obtained average latency (with longer head latency followed by shorter subsequent latency), energy consumption, and throughput by continuously transmitting a congestion-free 6-flit packet shown in FIG.19, and the results are summarized in Table 1.

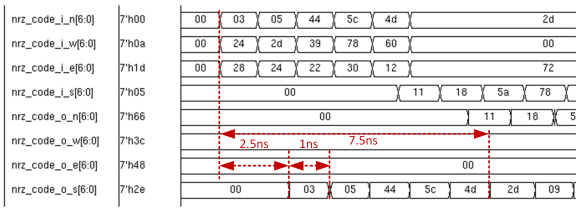


FIGURE 19. **The post-layout simulation waveform of transmitting a congestion-free 6-flit packet.**

Table 1 compares the area, performance, and energy consumption of the recent asynchronous router with different communication protocols. The data in Table 1 is from [7], [8], [14], [16], and [19] and technology and voltage are normalized to 65nm and 1.2V according to the equation in [29]. This work is implemented in 55nm and 1.2V, so compared to other designs in 65nm, latency, and energy are slightly optimistic while the area is pessimistic for this work.

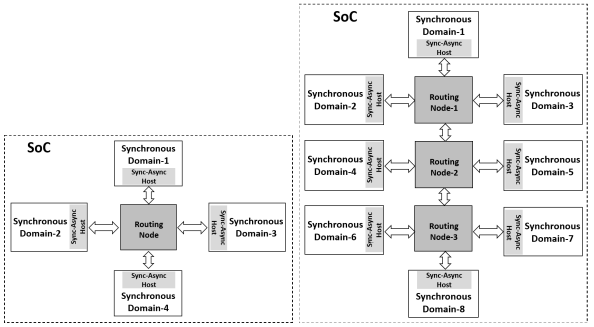
ANoC is a 4-phase QDI router, which is an influential design series from CEA-LETI. It has gone through several generations [16], [19]. Compared to the ANoC, this work performs 45.6% lower latency, 44% lower throughput, 60% lower energy, and 86.8% lower area. The ANoC uses 4-phase delay-insensitive communication, which has the advantage of robustness and variability, at the cost of the low throughput and the large numbers of wires. ANoC has 2 virtual channels, which can provide the ability of congestion-free at the cost of area and energy.

BAT-Hermes [8] is a 16-bit two-phase bundled-data router. This work has 66.2% lower latency and 2.6x equivalent speed at the cost of 45.9% higher energy per bit. BAT-Hermes just has an input port FIFO and it cannot deal with the congestion. Nonetheless, it has a 62.16% higher area than this work.

TaBuLA [14] is a two-phase bundled-data router with the Mousetrap controller, which is an asynchronous pipeline controller using a latch instead of a flip-flop. It also has gone through several generations [7]. Normalized to 65nm and 1.2V, it has comparable latency and 10% higher area than this work. Compared to the asynchronous 2-phase bundled data, the proposed Quasi-Synchronous communication has higher throughput. Despite the faster latch used in TaBuLA, this work has 23.07% higher throughput than it. TaBuLA has two level pipeline stage at the input port and output port, so the cycle is the latency of data from an input port to an output port. In this work, the cycle is the latency of data from an input port to FIFO, so it makes higher throughput. TaBuLA has 2 virtual channels using crossbar replication instead of a virtual channel buffer to gain lower area and energy.

B. SYSTEM LEVEL PERFORMANCE ANALYSIS

More detailed system level performance comparison by contrasting it with a synchronous baseline version of the same structure.



(a) (b)

FIGURE 20. **The architecture of proposed network architecture, (a) simple network; (b) complex network.**

1) EXPERIMENTAL SETUP

The Routing-Node and Sync-Async-Host form two different network architectures: simple network architecture with 4 synchronous domains and 1 Routing-Node and complex network architecture with 8 synchronous domains and 3 Routing-Node, as shown in FIG.20. The packet with 6-flit is sent from different synchronous domains to the Routing-node under test with maximum injection rate, and then the flit will be sent to other synchronous domains. According to different traffic models, each synchronous domain sends 1024 packets. Upon completion of the transmission and reception of all data packets by each synchronous domain, the completion time and the energy consumption are observed to obtain system throughput and energy per bit. The

TABLE 1. Comparison of asynchronous routers.

	ANoC [16]	TaBuLA [14]	BAT-Hermes [8]	This work
Technology	65nm,1.2V	40nm,1.2V	65nm,1V	55nm,1.2V
Communication protocol	4-phase QDI	2-phase bundled data	2-phase bundled data	Quasi-Synchronous
Flit Width	32	32	16	4
cycle(ns)	1.8	1.3	2.1	1
average latency(ns)	2.3	1.3	3.7	1.25
throughput(Mflit/s)	550	769	385	1000
area(mm^2)	0.1696	0.0249	0.0592	0.0224
energy(pj)/bit	0.93	0.2	0.2	0.37
congestion-free buffer	2 VC buffer	output port FIFO	input port FIFO	input port 3FIFO
congestion-free optimization	yes	yes	no	yes

area is directly obtained from the layout area. At different injection rates, the send and arrival times of each packet are recorded to calculate the average packet latency, thus determining the network system's performance.

For comparison, we also implemented a synchronous baseline of the asynchronous message bus. The synchronous message bus includes Hosts and Routing-Nodes. The Host is responsible for packing and serially transmitting flit, while the Routing-Node receives packets from different clock domains and forwards them to their respective destinations. Asynchronous FIFOs are added at the input port of the Routing-Node to receive data from different clock domains, incurring two additional cycles. The synchronous message bus uses traditional synchronous transmission with valid and ready handshakes for data transmission. The flow control mechanism is primarily controlled by the empty and full signals of the traditional asynchronous FIFOs.

To evaluate the impact of different numbers of FIFOs with ARBR on the performance of the Routing-Node, we also implement three versions of Routing-Nodes: ABRB-1FIFO, ABRB-2FIFO, and ABRB-3FIFO. Under the same conditions, we have obtained the results for the area, energy consumption, and system performance of the three different ARBR implementations in both simple and complex network architecture.

2) BENCHMARK

By setting the "DEST" field of the packets sent from each synchronous domain, the data flow can be controlled. To obtain performance results under different traffic models, we propose two benchmarks to simulate different traffic models.

Maximum Performance Traffic Model: Certain destinations are set for each clock domain, simulating a scenario where no blocking occurs in the message bus. This benchmark provides a more realistic reflection of the performance metrics of the basic circuit.

Random Traffic Model: Packets with random destinations are continuously sent at a saturate injection rate from each clock domain. This model represents situations with both blocking and concurrent transmissions, providing a more realistic evaluation of the performance metrics of the architecture of the message bus.

In the experimental setup, we conducted measurements on the proposed asynchronous message bus and the synchronous message bus under the two different network architectures. We obtained the area, energy consumption of the Routing-Node, and the system throughput, the latency of the message bus. Our analysis focused on the following two aspects:

(1)Comparison with synchronous baseline: We compared the area, system throughput, packet latency, and energy consumption of our proposed asynchronous message bus with synchronous baseline. This analysis aims to demonstrate that the proposed Quasi-Synchronous communication achieves comparable performance to synchronous communication while maintaining the low energy consumption characteristic.

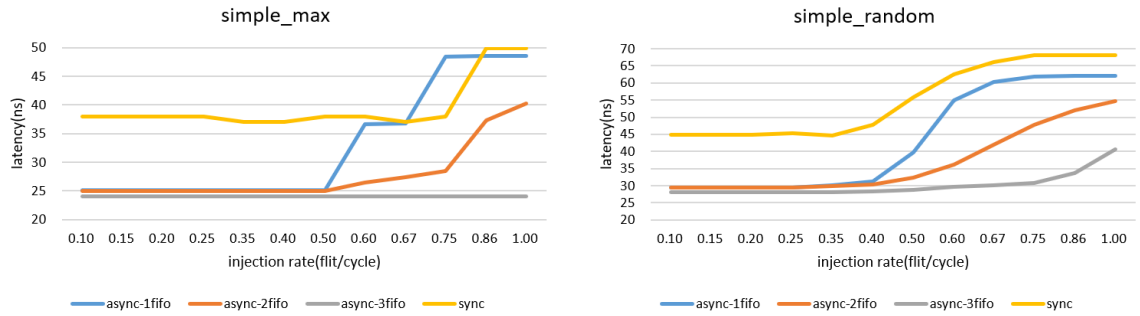
(2)Evaluation of the impact of ABRB with different FIFOs on system performance: By comparing ABRB-1FIFO, ABRB-2FIFO, and ABRB-3FIFO under the same conditions, we obtain the system throughput and latency of ARBR with different FIFOs. This analysis aimed to demonstrate that ABRB with three FIFOs achieves better performance for asynchronous message buses.

3) THE ASYNCHRONOUS MESSAGE BUS AND THE SYNCHRONOUS BASELINE

In the simple network architecture, we analyzed the area, throughput, energy consumption, and average latency for ASYNC-1FIFO and SYNC-1FIFO. The area, system throughput, and energy consumption of the simple network are shown in Table 2.

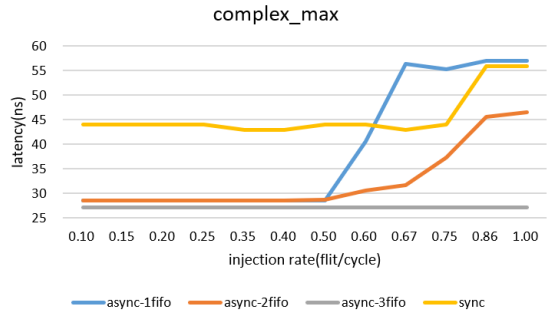
a: SYSTEM THROUGHPUT ANALYSIS

In the maximum performance traffic model, the throughput of the synchronous and asynchronous message bus is nearly the same, which demonstrates that our proposed Quasi-Synchronous communication achieves similar transmission efficiency as synchronous communication. In the

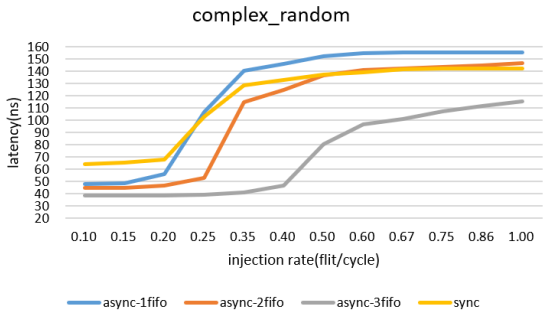


(a) simple-max

(b) simple-random



(c) complex-max



(d) complex-random

FIGURE 21. The latency of different networks at different injection rates.

TABLE 2. The analysis of the area, throughput, and energy between asynchronous message bus and synchronous baseline in different traffic models in the simple network.

Traffic Model	Metrics	ASYNC-1FIFO	SYNC-1FIFO
	Area(mm^2)	0.01	0.0072
Random Data Traffic Model	Throughput(Gbps)	3.18	2.62
	Energy(mj)	0.09	0.13
Max. Performance Traffic Model	Throughput(Gbps)	3.9	3.96
	Energy(mj)	0.09	0.11

TABLE 3. The analysis of the area, throughput, and energy between asynchronous message bus and synchronous baseline in different traffic models in the complex network.

Traffic Model	Metrics	ASYNC-1FIFO	SYNC-1FIFO
	Area(mm^2)	0.01	0.0072
Random Data Traffic Model	Throughput(Gbps)	2.5	2.59
	Energy(mj)	0.39	0.54
Max. Performance Traffic Model	Throughput(Gbps)	7.42	7.99
	Energy(mj)	0.27	0.29

random traffic model, the asynchronous message bus proves to be a better solution. The system throughput of the asynchronous message bus reaches 3.18 Gbps, showing a 21.4% improvement compared to the 2.62 Gbps of the synchronous version in the random traffic model, where congestion occurs frequently. The proposed Quasi-Synchronous communication and flow control mechanism is designed to effectively handle the congestion in the message bus.

b: ENERGY CONSUMPTION ANALYSIS

In the maximum performance traffic model, compared to the synchronous baseline, the asynchronous message bus exhibits a 12.64% reduction in energy consumption. In the random traffic model, it exhibits a 27.18% energy advantage. This mainly stems from the following aspects: Firstly, in GALS systems, asynchronous communication eliminates the synchronization overhead across clock domains and uses local clocks instead of a global clock. Secondly, our proposed asynchronous message bus features a data-driven nature, resulting in flips only when there is valid data, thus avoiding additional dynamic power consumption. Lastly, we utilize a 2-of-7 encoding method where only 2 bits flip during each data transmission, leading to lower energy consumption compared to synchronous systems.

c: AREA ANALYSIS

Compared to the synchronous message bus, the asynchronous message bus incurs an additional 38% area overhead. Unlike

TABLE 4. The area, throughput, and energy of Routing-Node with different FIFO-based ABRB in different networks and traffic models.

Network	Traffic Model	Metrics	ABRB-3FIFO	ABRB-2FIFO	ABRB-1FIFO
		Area(mm^2)	0.0225	0.0144	0.01
simple	Random Data	Throughput(Gbps)	4.69	3.8	3.18
	Traffic Model	Energy(mj)	0.12	0.11	0.09
	Max. Performance	Throughput(Gbps)	6.27	4.6	3.9
	Traffic Model	Energy(mj)	0.11	0.1	0.09
complex	Random Data	Throughput(Gbps)	4.8	3	2.5
	Traffic Model	Energy(mj)	0.5	0.43	0.39
	Max. Performance	Throughput(Gbps)	12.72	8.82	7.42
	Traffic Model	Energy(mj)	0.31	0.3	0.27

the 4-bit synchronous message bus, we've employed a 7-bit low-power encoding, significantly increasing the number of registers within the router. Additionally, our usage of local clocks instead of a global clock leads to extra area overhead from the local clock generation units.

d: SYSTEM PERFORMANCE

As shown in FIG.21a and FIG.21b, the ASYNC-1FIFO shows substantial benefits achieving 34% lower low-load latency when the injection rate is 0.1 flit/cycle. The cost is introduced due to the deeper pipeline and the synchronizers for clock domain crossing in synchronous communication. For ASYNC-1FIFO, however, it enters the saturation region at an injection rate that is earlier than SYNC-1FIFO and has the same average latency as SYNC-1FIFO when it achieves saturation. Because the flow control of the ASYNC-1FIFO is not enough to handle the higher load communication.

The results for the complex network architecture are shown in Table 3, and in the maximum performance traffic model, they exhibit similar conclusions to those in the simple network architecture. In the random traffic model, the throughput of the asynchronous message bus decreases. Due to the flow control mechanism of ASYNC-1FIFO, it pauses every transmission after receiving each packet to avoid overflow. As the network becomes more complex with an increasing number of synchronous domains, the transmission pause time also increases. This has a significant negative impact on throughput.

In the maximum performance traffic model, it also exhibits similar conclusions of latency to those in the simple network as shown in FIG.21c. But in the random traffic model under the complex network shown in FIG.21d, at low load when injection is 0.1 flit/cycle, the advantage latency of ASYNC-1FIFO narrows to 25%, and it enters the saturation region earlier. Furthermore, the latency after reaching the saturation region is also 10% lower than that of the SYNC-1FIFO. It is evident that the ASYNC-1FIFO, with its inherent low latency characteristics, achieves significant performance advantages over SYNC-1FIFO under low-load conditions. However, as the load increases, the flow control disadvantages of ASYNC-1FIFO become increasingly apparent, leading to a substantial overall performance decrease. Therefore, we propose the ASYNC-3FIFO to better address high-load situations. This will be demonstrated in the next experiment.

4) ABRB WITH DIFFERENT NUMBERS OF FIFO

As the network architecture becomes more complex, ABRB-1FIFO struggles to cope with the increased complexity of traffic patterns, resulting in a decrease in system throughput. To overcome this issue, ABRB-2FIFO and ABRB-3FIFO were introduced. These additional FIFOs in the ABRB mechanism help mitigate the impact of complex traffic models and improve system performance. Despite the potential for achieving better system throughput with more FIFOs, the tradeoff involves increased latency, power consumption, and area. Additionally, we believe that the throughput improvement it brings may also diminish. Table 4 displays the area, throughput, and energy consumption of Rouing-Node with different FIFO-based ABRB.

It is evident that as the number of FIFOs increases, the area of the Routing-Node significantly increases. However, the additional FIFOs result in a significant improvement in system throughput. In the maximum performance traffic model of the simple network, ABRB-3FIFO and ABRB-2FIFO achieve 60.8% and 17.9% higher system throughput, respectively, compared to ABRB-1FIFO. In the random data traffic model of the simple network, ABRB-3FIFO, and ABRB-2FIFO achieve 47.5% and 19.5% higher system throughput, respectively, compared to ABRB-1FIFO. The additional energy consumption incurred is negligible,

indicating that the ABRB-3FIFO structure has higher communication efficiency.

The ABRB-3FIFO structure is more effective in handling congestion, especially when the destination of subsequent packets is different. The ABRB-3FIFO can concurrently transmit subsequent data packets while the previous packet is blocked. In the case of ABRB-2FIFO, it is challenging to ensure that the second packet arrives after the first packet has been read in dense data transfers. Hence, blocking occurs after both packets are written into the two FIFOs.

The conclusions obtained in Table 4 also validate this observation. In the maximum performance traffic model of the complex network, the system throughput of ABRB-3FIFO reaches 12.72 Gbps, which represents a 44.2% and 71.42% improvement over ABRB-2FIFO and ABRB-1FIFO, respectively. It performs the best among all tested Routing-Nodes. In the random data traffic model of the complex network, the ABRB-3FIFO version of the Routing-Node achieves an 85.3% system throughput improvement compared to the synchronous version, with no additional energy consumption overhead.

In the maximum performance traffic model shown in FIG.21a and FIG.21c, the average latency remains unchanged by the increase in injection rate. This indicates that the ASYNC-3FIFO structure performs well in flow control under low-load conditions. In contrast, for several other structures including ASYNC-2FIFO, ASYNC-1FIFO, and SYNC-FIFO, with the increase in injection rate, it becomes more likely to trigger backpressure mechanisms, leading to increased latency. In a random traffic model shown in FIG.21b and FIG.21d, compared to several other asynchronous and synchronous structures, ASYNC-3FIFO, due to its stronger ability to handle congestion, enters the saturation later. Furthermore, after reaching saturation, in both the simple network and complex network, ASYNC-3FIFO exhibits an improvement of 40.16% and 18.62%, respectively, compared to the synchronous baseline.

VI. CONCLUSION

This paper introduces a lightweight asynchronous message bus with high throughput. It aims to transmit high-level protocol messages independent to the data bus/NoC, improving the overall communication efficiency of the system. To minimize the additional impact of the message bus on the system, an asynchronous manner is adopted, featuring data-driven characteristics that reduce energy consumption when no message transmission is occurring. To enhance transmission throughput, a novel Quasi-Synchronous communication is proposed, eliminating flit-level handshake signals and replacing them with packet-level flow control signals. These control signals propagate through the data channel using a special flow control mechanism, reducing additional control line overhead. To further alleviate bus congestion caused by access conflicts, an Asynchronous Blocking Retransmission Buffer (ABRB) is introduced, effectively improving the system throughput. The proposed asynchronous message bus, along with the synchronous message bus, is implemented using conventional EDA tools in the SMIC 55nm process node, and performance, area, and energy consumption metrics are obtained through post-simulation. Compared to the synchronous baseline, our message bus demonstrates a 22.8% overall throughput improvement and a 17.78% energy consumption improvement in the random data traffic model at the cost of 38% area and more design effort. Additionally, it reveals that ABRB-3FIFO achieves higher system performance and energy efficiency, along with better handling of congestion. Finally, in the complex network with the random data traffic model, the ABRB-3FIFO version of the asynchronous message bus exhibits an 85.3% system throughput improvement and 18.62% system average latency improvement compared to the synchronous baseline, with no additional energy consumption overhead.

ACKNOWLEDGMENT

The authors would like to thank the colleagues of the Department of Neuromorphic computing at NIIT and Huawei Technologies Company Ltd., for their helpful discussions.

REFERENCES

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Design Autom. Conf.*, Las Vegas, NV, USA, 2001, pp. 684–689.

[2] E. Beigne, P. Vivet, Y. Thonnart, J.-F. Christmann, and F. Clermidy, "Asynchronous circuit designs for the Internet of Everything: A methodology for ultralow-power circuits with GALS architecture," *IEEE Solid StateCircuits Mag.*, vol. 8, no. 4, pp. 39–47, Fall. 2016.

[3] L. A. Plana, J. Bainbridge, S. Furber, S. Salisbury, Y. Shi, and J. Wu, "An on-chip and inter-chip communications network for the SpiNNaker massively-parallel neural net simulator," in *Proc. 2nd ACM/IEEE Int. Symp. Networks-on-Chip (nocs)*, Tyne, U.K., Apr. 2008, pp. 215–216.

[4] A. Sheibanyrad, A. Greiner, and I. Miro-Panades, "Multisynchronous and fully asynchronous NoCs for GALS architectures," *IEEE Design Test Comput.*, vol. 25, no. 6, pp. 572–580, Nov. 2008.

[5] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar, "An asynchronous router for multiple service levels networks on chip," in *Proc. 11th IEEE Int. Symp. Asynchronous Circuits Syst.*, New York, NY, USA, 2005, pp. 44–53.

[6] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. Design, Autom. Test Eur.* Munich, Germany: IEEE, 2005, pp. 1226–1231.

[7] A. Ghiribaldi, D. Bertozzi, and S. M. Nowick, "A transition-signaling bundled data NoC switch architecture for cost-effective GALS multicore systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 332–337.

[8] M. Gibiluka, M. T. Moreira, F. G. Moraes, and N. L. Vilar Calazans, "BATHermes: A transition-signaling bundled-data NoC router," in *Proc. IEEE 6th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Feb. 2015, pp. 1–4.

[9] E. Kasapaki, M. Schoeberl, R. B. Sørensen, C. Müller, K. Goossens, and J. Sparsø, "Argo: A real-time network-on-chip architecture with an efficient GALS implementation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 479–492, Feb. 2016.

[10] M. Imai, T. Van Chu, K. Kise, and T. Yoneda, "The synchronous vs. asynchronous NoC routers: An apple-to-apple comparison between synchronous and transition signaling asynchronous designs," in *Proc. 10th IEEE/ACM Int. Symp. Networks-on-Chip (NOCS)*, Aug. 2016, pp. 1–8.

[11] W. Jiang, D. Bertozzi, G. Miorandi, S. M. Nowick, W. Burleson, and G. Sadowski, "An asynchronous NoC router in a 14 nm FinFET library: Comparison to an industrial synchronous counterpart," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 732–733.

[12] C. Effiong, G. Sassatelli, and A. Gamatie, "Scalable and power-efficient implementation of an asynchronous router with buffer sharing," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2017, pp. 171–178.
[13] A. He, G. Feng, J. Zhang, P. Li, Y. Hei, and H. Chen, "Click-based asynchronous mesh network with bounded bundled data," in *Proc. 47th Int. Conf. Parallel Process.*, Aug. 2018, pp. 1–8.
[14] D. Bertozzi, G. Miorandi, A. Ghiribaldi, W. Burleson, G. Sadowski, K. Bhardwaj, W. Jiang, and S. M. Nowick, "Cost-effective and flexible asynchronous interconnect technology for GALS systems," *IEEE Micro*, vol. 41, no. 1, pp. 69–81, Jan. 2021.
[15] J. Bainbridge and S. Furber, "Chain: A delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, no. 5, pp. 16–23, Sep. 2002.
[16] Y. Thonnart, P. Vivet, and F. Clermidy, "A fully-asynchronous low-power framework for GALS NoC integration," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Mar. 2010, pp. 33–38.
[17] N. Onizawa, A. Matsumoto, T. Funazaki, and T. Hanyu, "High-throughput compact delay-insensitive asynchronous NoC router," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 637–649, Mar. 2014.
[18] W.-G. Ho, K.-S. Chong, N. K. Z. Lwin, B.-H. Gwee, and J. S. Chang, "High robustness energy- and area-efficient dynamic-voltage-scaling 4-phase 4-rail asynchronous-logic network-on-chip (ANoC)," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 1913–1916.
[19] P. Vivet, Y. Thonnart, R. Lemaire, and C. Santos, "A 4 × 4 × 2 homogeneous scalable 3D network-on-chip circuit with 326 MFlit/s 0.66 pJ/b robust and fault tolerant asynchronous 3D links," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 33–49, Jan. 2017.
[20] W.-G. Ho, K.-S. Chong, K. Z. L. Ne, B.-H. Gwee, and J. S. Chang, "Asynchronous-logic QDI quad-rail sense-amplifier half-buffer approach for NoC router design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 1, pp. 196–200, Jan. 2018.
[21] W. J. Bainbridge, W. B. Toms, D. A. Edwards, and S. B. Furber, "Delay-insensitive, point-to-point interconnect using m-of-n codes," in *Proc. 9th Int. Symp. Asynchronous Circuits Syst.*, May 2003, pp. 132–140.
[22] Y. Shi, S. B. Furber, J. Garside, and L. A. Plana, "Fault tolerant delay insensitive inter-chip communication," in *Proc. 15th IEEE Symp. Asynchronous Circuits Syst.* Chapel Hill, NC, USA: IEEE, May 2009, pp. 77–84.
[23] G. Liu, J. Garside, S. Furber, L. A. Plana, and D. Koch, "Asynchronous interface FIFO design on FPGA for high-throughput NRZ synchronisation," in *Proc. 27th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2017, pp. 1–8.
[24] S. Rasheed, P. V. Gratz, S. Shakkottai, and J. Hu, "STORM: A simple traffic-optimized router microarchitecture for networks-on-chip," in *Proc. 8th IEEE/ACM Int. Symp. Networks-on-Chip (NoCS)*, Sep. 2014, pp. 176–177.
[25] G. Miorandi, A. Ghiribaldi, S. M. Nowick, and D. Bertozzi, "Crossbar replication vs. sharing for virtual channel flow control in asynchronous NoCs: A comparative study," in *Proc. 22nd Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2014, pp. 1–6.
[26] K. Bhardwaj and S. M. Nowick, "A continuous-time replication strategy for efficient multicast in asynchronous NoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 2, pp. 350–363, Feb. 2019.
[27] K. van Berkel, F. Huberts, and A. Peeters, "Stretching quasi delay insensitivity by means of extended isochronic forks," in *Proc. 2nd Work. Conf. Asynchronous Design Methodologies*, London, U.K., 1995, pp. 99–106.
[28] *AMBA AHB Protocol Specification*. Accessed: 2021. [Online]. Available: https://developer.arm.com/documentation/ihi0033/latest/
[29] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, Jun. 2017.

QINGYANG ZENG (Student Member, IEEE) received the B.S. degree in electronic science and technology from Jilin University, Changchun, Jilin, China, in 2019. He is currently pursuing the Ph.D. degree in microelectronics and solid state electronics with the Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China.

His research interests include asynchronous network on chip and multi-core communication.

JINGYU WANG (Student Member, IEEE) received the B.S. degree from the School of Microelectronics, Xidian University, Xi'an, China, in 2019. He is currently pursuing the Ph.D. degree in microelectronics and solid state electronics with the Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China.

His research interests include asynchronous interface and asynchronous communication.

JIAXU CONG (Student Member, IEEE) received the B.S. degree in integrated circuit design and integrated systems from Dalian University of Technology, Dalian, Liaoning, China, in 2021. He is currently pursuing the Ph.D. degree in microelectronics and solid state electronics with the Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China.

His research interests include asynchronous network on chip and the architecture of brain-like computers.

DELONG SHANG received the B.S. degree in computer architecture from Nanjing University, the M.S. degree in computer architecture from the Chinese Academy of Sciences, China, and the Ph.D. degree in microelectronics and computer architecture from Newcastle University, U.K.

He is currently a Full Professor with Nanjing Institute of Intelligent Technology, China. His research interests include computer architecture, asynchronous systems, power-efficient systems, and neuromorphic computing.