LinkBo: A Single-Wire, Low-Latency, and Robust Protocol for Variable-Distance Chip-to-Chip Communications

Bochen Ye*,‡, Gustavo Naspolini†, Kimmo Salo†, Manil Dev Gomony*
*Eindhoven University of Technology, Eindhoven, The Netherlands
†NXP Semiconductors, Nijmegen, The Netherlands
b.ye@student.tue.nl, gustavo.naspolini@nxp.com, kimmo.salo@nxp.com, m.gomony@tue.nl

2025 IEEE 38th International System-on-Chip Conference (SOCC) | 979-8-3315-9478-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/SOCC66126.2025.11235355

***Abstract*—Cost-effective embedded systems necessitate utilizing the single-wire communication protocol for inter-chip communication, thanks to its reduced pin count in comparison to the multi-wire I2C or SPI protocols. However, current single-wire protocols suffer from increased latency, restricted throughput, and lack of robustness. This paper presents LinkBo, an innovative single-wire protocol that offers reduced latency, enhanced throughput, and greater robustness with hardware-interrupt for variable-distance inter-chip communication. The LinkBo protocol-level guarantees that high-priority messages are delivered with an error detection feature in just 50.4 µs, surpassing current commercial options, 1-wire and UNI/O by at least 20X and 6.3X, respectively. In addition, we present the hardware architecture for this new protocol and its performance evaluation on a hardware platform consisting of two FPGAs. Our findings demonstrate that the protocol reliably supports wire lengths up to 15 meters with a data rate of 300 kbps, while reaching a maximum data rate of 7.5 Mbps over an 11 cm wire, providing reliable performance for varying inter-chip communication distances.**

***Index Terms*—single-wire protocol, serial interface, hardware design, FPGA, Chip-to-chip communication.**

I. Introduction

Common chip-to-chip communication methods typically rely on protocols such as Serial Peripheral Interface (SPI) or Inter-Integrated Circuit (I2C) [1]. SPI uses four wires: one for the clock signal, one for chip select, and two for data transmission. In contrast, I2C streamline communication with just two wires: one for data and one for the clock signal. However, as the number of interconnected chips increases, the total number of required pins for both SPI and I2C grows accordingly. In Fig. 1 (left), the total number of communication pins required for SPI increases sharply while I2C exhibits a more linear growth.

In embedded systems, minimizing the physical footprint, particularly the number of off-chip communication wires, is a critical design goal. Reducing the number of external pins can significantly shrink the chip's packaging area, potentially saving several square centimeters. According to the packaging information [2], [3], a 16-pin package occupies only 40% of the area of a 20-pin package and incurs just 60% of the cost (Fig. 1). These findings highlight the strong correlation between pin count and both packaging area and cost. Consequently, reducing the number of external pins is essential for area-constrained applications. Although SPI and I²C require only 2 to 4 communication pins, they may still be suboptimal for systems with stringent area or cost limitations.

‡This work was performed while an intern at NXP.

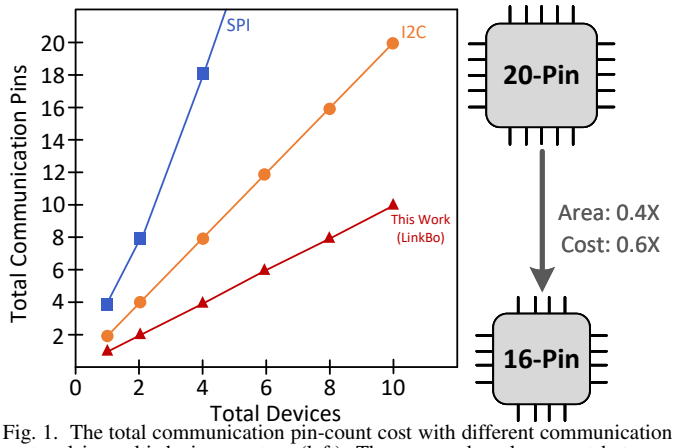

Fig. 1. The total communication pin-count cost with different communication protocol in multi devices system (left). The area and package cost between 20-pin package and 16-pin package (right).

A single-wire communication protocol offers a promising alternative by single pin design. "Single-wire" refers to a class of protocol that uses a single wire for communication. The state-of-the-art (SOTA) single-wire protocol, known as the 1-wire protocol, operates as a half-duplex serial bus using only one data line and a single pin [4]. However, the SOTA 1-wire protocol prioritizes long-distance communication, leading to extended bit-slot durations, which result in significant message latency and a low bit rate of 16.3 kbps, failing to meet the high-speed processing requirements of modern applications. Furthermore, 1-wire communication protocol is not particularly robust as it becomes susceptible to failure if the host malfunctions, and it lacks mechanisms to secure data integrity. To address these issues, this paper proposes the following contributions:

1) We propose LinkBo protocol, which is a novel chip-to-chip single-wire protocol that can achieve lower latency, higher throughput, and improved robustness. It also contain a system-level model to analysis performance.
2) We propose a novel hardware architecture for different speed and variable-distance LinkBo protocol on two FPGAs with only single wire.
3) Performance comparison of LinkBo with SOTA single-wire protocols in terms of latency and bit rate. In addition, the performance of the LinkBo protocol is analyzed under different parasitic parameters, wire lengths, and bit rate (clock frequencies) to assess robustness.

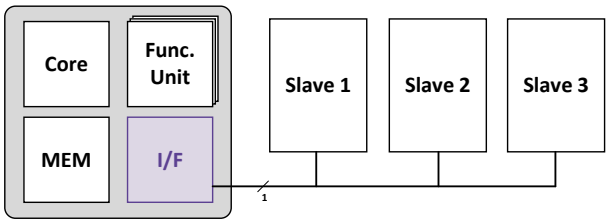

Fig. 2. Current 1-wire bus architecture. The host contain processing core, function units, memory and 1-wire protocol. The host use 1-bit wire to send message to multiple slaves and receive data from slaves.

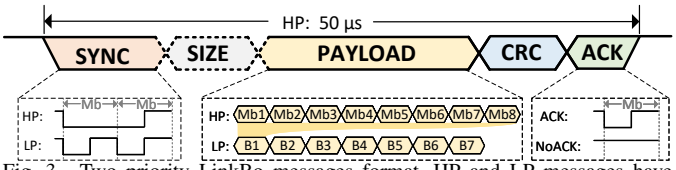

Fig. 3. Two priority LinkBo messages format. HP and LP messages have different SYNC and PAYLOAD fields, but share the same CRC and ACK fields. Only LP messages include a SIZE field. HP messages are always completed within 50 µs, whereas the duration of LP messages depends on the payload size. In PAYLOAD, one byte in LP message contain 8 Manchester bits.

The rest of the paper is organized as follows. Section II introduces the foundational concepts and examines related work on the single-wire protocol. Section III outlines the functionality of the Linkbo protocol. Section IV offers a high-level overview of the channel model. Section V details the hardware architecture of our transmitter, receiver, and driver. Section VI presents the FPGA setup and experimental results with respect to bit rate, latency, physical parameters, wire length, and clock frequency. Finally, the paper concludes with a summary of our findings and suggestions for future research in Section VII.

II. Background and Related work

1-wire protocol overview. The 1-Wire bus [4], [5], a low-speed single-wire protocol for long-distance communication (up to 100 m), is illustrated in Fig. 2. It employs a single host to control multiple slaves. Due to differing or misaligned clock signals, chip-to-chip communication is typically asynchronous, risking receiver errors. This necessitates encoding or clock synchronization to ensure reliable data transfer.

Asynchronous communication. The traditional single-wire protocol utilizes Non-Return-to-Zero (NRZ) coding, in which a high signal denotes a 1 and a low signal signifies a 0. This method, though, introduces a direct current (DC) component, complicating clock synchronization and heightening susceptibility to interference. Conversely, Manchester encoding removes the DC component. This study adopts IEEE 802.3 as the Manchester encoding standard, where a rising edge represents bit 1 and a falling edge denotes bit 0. Manchester encoding inherently self-synchronizes and is resilient to interference, with each bit incorporating a transition that aids in error detection and clock alignment [6].

Related work. 1-wire is a protocol first invented by Dallas Semiconductor (now part of ADI) in 1995 and is still used by TI and ADI today [4], [5]. Microchip developed a proprietary single-wire protocol known as the UNI/O bus [7], which finds extensive application in embedded systems. Using Manchester encoding for synchronization, the UNI/O bus supports messages of arbitrary byte length and incorporates MAK/SAK acknowledgment. However, both bus systems experience inefficiencies due to extended standby pulses and an excessive number of address bits. Moreover, UNI/O does not include mechanisms for error checking or conflict resolution, deficiencies that our protocol addresses. There are also single-wire protocols designed for debugging systems rather than data transmission, such as SWIM [8] and debugWIRE. In addition, prior academia efforts to enable multi-node communication over a single wire explored ring topologies, clock-data merging, FDMA, and CDMA techniques [9], [10], [11], [12], [13], [14], but faced limitations in speed, scalability, error rates, or hardware practicality. Some paper optimize single-wire interface in analog circuit while our work only use digital technology [15], [16], [17].

III. LinkBo protocol definition

Protocol overview. The proposed LinkBo protocol uses Manchester encoding (IEEE802.3) to enable bidirectional data transmission between two asynchronous chips over a single wire. Fig. 3 illustrates the message format and the specific timing details of the LinkBo protocol. Each message is composed of multiple fields: synchronization field, size field, payload field, CRC field, acknowledgment field.

Priority message. In the LinkBo protocol, messages are categorized into two types based on the structure of their synchronization fields, as illustrated in Fig. 3. The first type is the high-priority (HP) message, which uses only 15 Manchester bits (Mb): 2 bit slots for synchronization, 8 bits for the payload, 4 bits for CRC, and 1 bit for acknowledgment (ACK). Each HP message can carry only one byte (B) of payload and does not include a size field. The second type is the low-priority (LP) message, which can support 1 to 7 bytes of payload. It uses a 3-bit size field to represent this. Consequently, a LP message contains from 18 to 66 Mb: 2 bits for synchronization, 3 bits for size, 8 to 56 bits for the payload, 4 bits for CRC, and 1 bits for ACK.

HP messages are designed to support hardware interrupts, providing minimal latency for time-critical or emergency data transmission but carries only a small amount of information. To balance throughput, LP messages are introduced to transmit larger volumes of data, albeit with higher latency. LinkBo allows HP messages to interrupt LP messages transmission when necessary to ensure low latency. This hardware-level interrupt enables faster response times compared to software-based interrupt handling (e.g., I2C), ensuring timely delivery of high-priority data.

Synchronization and re-synchronization. In single-wire communication, synchronization must rely solely on signal edges due to the absence of a dedicated clock line. To differentiate between HP and LP messages during synchronization, the LinkBo protocol utilizes two distinct synchronization field patterns. As illustrated in Fig. 3, the synchronization field includes two Mb slots. For HP messages, the initial slot is consistently held low to differentiate it from LP messages. This

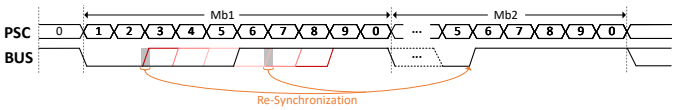

Fig. 4. The re-synchronization process for Manchester code during the transmission. The depth of the red edge represents the severity of the error.

extended low signal also facilitates the creation of an interrupt signal on the bus, enhancing its detectability relative to other signals. In contrast, for LP messages, the first slot signifies a 1, marked by a transition from low to high. The second Mb in both types of message is the same and represents a 1.

Upon detecting the first falling edge, the prescaler counter (PSC) starts to measure the cycle duration. If the low signal lasts longer than one Manchester slot, the message is identified as HP; otherwise, it's LP. For HP messages, the counter stops at the first rising edge; for LP messages, at the second. The count reflects 1.5 Mb slots, so dividing by 1.5 yields the Mb slot duration in the receiver's clock domain. This process is known as synchronization.

Manchester coding requires a transition in the middle of every Mb slot, which can be used for error checking. In Fig. 4, black signal shows a correct Manchester code with a transition in the middle, indicating that there are no errors. The red signal shows early or late transitions, which are not correct but still acceptable. If edge in other place, it represents a error case for the receiver, where it cannot determine whether the Manchester bit is 1 or 0. Resynchronization is the process of adjusting the next transitions to ensure that they occur in the middle of each Mb slot. This mechanism ensures that each Manchester bit is correctly aligned, helping to maintain accurate data transmission.

CRC and acknowledgment Cyclic redundancy check (CRC) is a method used to detect errors in data messages or memory storage. By performing a polynomial division on the data to generate an N-bit checksum, the CRC can effectively detect and identify common errors such as bit flips, ensuring the integrity and reliability of the data [18]. In this work, we use a 4-bit CRC with the polynomial X^4+X+1 because it can detect any single-bit error and can detect most two-bit error. In Eq. (1), the b represents the number of burst error bits, and k means the number of CRC bits. It shows that a 4-bit CRC can detect burst errors of up to 4 bits with 100% accuracy and burst errors longer than 4 bits with 93.75% accuracy.

$$P_{crc}(b) = \begin{cases} 1, & \text{if } b \leq k \\ 1 - \frac{1}{2^k}, & \text{if } b > k \end{cases} \quad (1)$$

The receiver will then send an ACK signal back to the transmitter. If the CRC check is correct, it will send Manchester code 1 (low to high). If the CRC check is incorrect, it will not edge. The ACK signal is shown in Fig. 3 right.

Scalability for multi-device The 1-Wire bus operates with a single master and multiple slaves, which limits scalability and places a heavier load on the host. If the host device fails, the entire system stops working. To address this, the LinkBo protocol uses a peer-to-peer bus, ensuring better scalability and fault tolerance. In LinkBo, each module can communicate

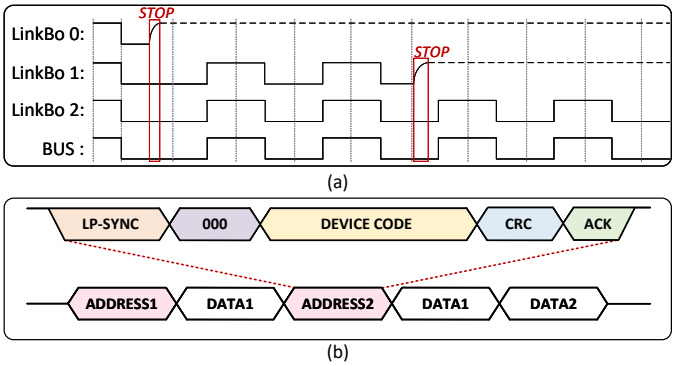

Fig. 5. (a) Multi-device arbitration with Wired-AND logic. (b) The address message for configuration.

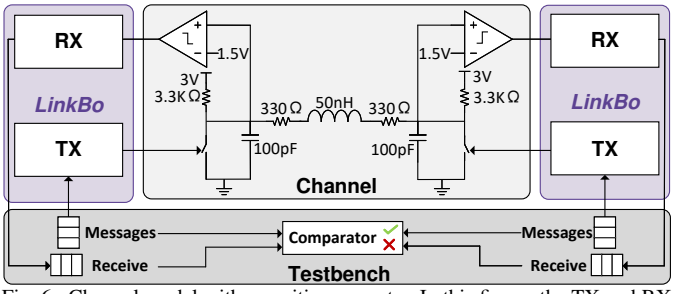

Fig. 6. Channel model with parasitic parameter. In this figure, the TX and RX serve as interface between LinkBo and channel. The inductance, resistance and capacitance simulate the real wire characteristics.

independently with others, allowing the system to continue functioning even if one device fails. In Fig. 5 (a), The bus uses wired-AND arbitration during transmission. LinkBo 0 sends a low-priority message, while LinkBo 1 and 2 send high-priority ones. From the bus's perspective, the first high signal loses arbitration, so LinkBo 1 and 2 are excluded successively. To support address in our protocol, we can repurpose the reserved '000' SIZE field to define a new message type that carries an address in Fig. 5 (b). Each device responds only to its unique address. This approach adds latency but minimizes hardware changes to the original design.

IV. System Model for Performance Evaluation

Hardware inherently possesses numerous parasitic parameters that can influence voltage and current delivery, resulting in performance variations in actual chips. To investigate the performance of the LinkBo protocol, we developed a high-level model in Simulink, depicted in Fig. 6. This model features two identical LinkBo modules, each with a transmitter (TX) and a receiver (RX). Both the TX and RX leverage the driver to transmit digital signals to control the single-wire bus. For testing our model, we set up a test bench for the ideal LinkBo model, designed to deliver stimuli to the TX of one module and receive data from the RX of the other.

In Fig. 6, the TX is connected to a switch, while the RX is connected to a 1.5V comparator. When the driver is high (bit 1), it opens the switch, allowing current to flow through the channel. When the digital output is low (bit 0), the switch closes, directing the current to the ground. If the voltage exceeds 1.5V, the input of RX registers as binary level 1; otherwise, it registers as binary level 0.

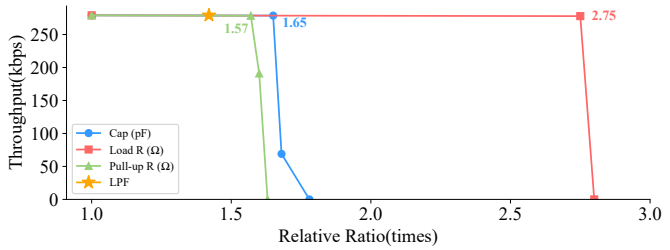

Fig. 7. Throughput vs Parameter (Capacitance, Load resistor, Pull-up resistor). All three lines drop sharply at a certain point, but they can still reach the LPF condition without any performance degradation.

To ensure that our protocol can achieve the same throughput or bit rate in a real chip, the parasitic parameters of the channel model can be adjusted to evaluate the relationship between parasitic parameters and throughput. These parameters are the same as those used on the PCB that holds the FPGA. Fig. 7 presents the simulation results, with the X-axis indicating the parameter variation ratio. On the Y-axis, $Throughput$ is depicted, defined as the count of RX bits received successfully per unit time. In Eq. (2), B_{re} denotes the bits received by RX, and T represents the duration of one transmission.

$$Throughput = \frac{B_{re}}{T} \tag{2}$$

The blue capacitance line signifies a tolerance for a 1.65X variation; however, its efficiency drops to 25% at a 1.68X variation before gradually declining to zero. The red line for the load resistance shows that it can withstand a 2.75X variation before rapidly falling. This is because a change in the load resistor alters the current while the capacitor impacts the circuit's rise and fall times, delaying voltage stabilization. The green pull-up resistor line demonstrates a sharp performance decline following a variation of 1.57X. Due to the size of the baseline pull-up resistor, slight changes significantly affect resistance. Inductance, not depicted in Fig. 7, can support up to 167.5 µH. In summary, the component that affects the throughput is mainly the capacitor, followed by the pull-up resistor, then the load resistor, whereas the inductance has the least influence.

The maximum resistance and capacitance for the low-pass filter to filter out high-frequency noise can be calculated by Eq. (3). F_c is cut off frequency set at 3.3 MHz. As a result, the load resistor needs 470 Ω, while the capacitance needs 100 pF. The 470 Ω is only 1.42X of the baseline value (star marker in Fig. 7), so it can easily meet the requirement.

$$F_c = \frac{1}{2\pi RC} \tag{3}$$

V. Hardware Architecture and Implementation

Top-level architecture. In Fig. 8 (1), the top-level hardware architecture of LinkBo is shown. The core elements within the LinkBo module include the transmitter (TX), receiver (RX), prescaler counter (PSC), TOP FSM, synchronizer, and driver. The TX module sends messages, while the RX module is in charge of receiving them. The driver encodes information into Manchester bits and controls the bus. The PSC divides the clock frequency and produces the Manchester mask signal.

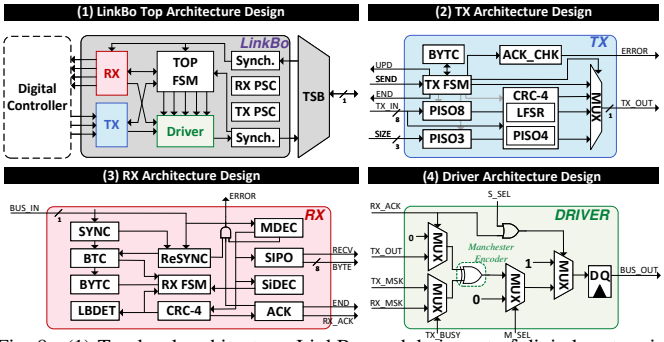

Fig. 8. (1) Top-level architecture. LinkBo module is part of digital system in each chip, consisting of synchronizer, TOP FSM, RX, TX, driver, and PSC(for both TX and RX). (2) Transmitter architecture. TX is responsible for sending any priority message. (3) Receiver architecture. The RX is responsible for receiving messages, synchronizing, and sending an acknowledgment (ACK) signal. (4) Driver architecture. Driver consists of one Manchester encoder (XOR gate) and 4 MUXes.

The synchronizer resolves timing discrepancies in digital circuits when signals transition between distinct clock domains. The TOP FSM, a state machine, manages the activation and deactivation of TX/RX operations during interruptions. There is a tri-state buffer out of LinkBo module, which is used to control the input and output of bus. The digital controller can provide the LinkBo module with different clock frequencies to support various speeds or bit rates for communication over different distances.

Transmitter architecture. The transmitter is controlled by TX FSM. The UPD signal confirms that the BUS_IN has been transmitted. Two shift registers, an 8-bit and a 3-bit, convert parallel input data into 1-bit serial output data (PISO8, PISO3). The CRC-4 module employs 4 registers in a Linear-Feedback-Shift-Register (LFSR) configuration with the polynomial $X^4 + X + 1$ to produce a 4-bit CRC checksum, transmitted bitwise through a 4-bit shift register (PISO4). A 4-to-1 multiplexer (MUX) selects the data set for transmission. The byte counter (BYTC) can ensure to send multi-byte message. The ACK_CHK module verifies CRC errors and reports errors if found. The architecture is depicted in Fig. 8 (2).

Receiver architecture. The receiver is depicted in Fig. 8 (3) and is more sophisticated than the transmitter. The synchronization module (SYNC) first identifies the falling edges, initiating a counter to determine the Manchester bit slot; as previously explained in Section III. The Manchester decoder (MDEC) converts each Manchester bit into a standard bit, while the serial-in-parallel-out (SIPO) module delivers one byte of the message. The CRC-4 divider checks for errors in the payload. If none is found, the ACK module signals the driver. The state machine (RX FSM) utilizes bit and byte counters (BTC, BYTC) to track Manchester bits and message bytes. Sometimes, edges may not align centrally; the re-synchronization module (Re-SYNC) is designed to address this. Lastly, the low-bus detector (LBDET) watches for prolonged low-level signals on the bus, signaling a high-priority event and triggering an interrupt.

Driver architecture. The driver consists of several MUXes and a Manchester encoder, see in Fig. 8 (4). The Manchester

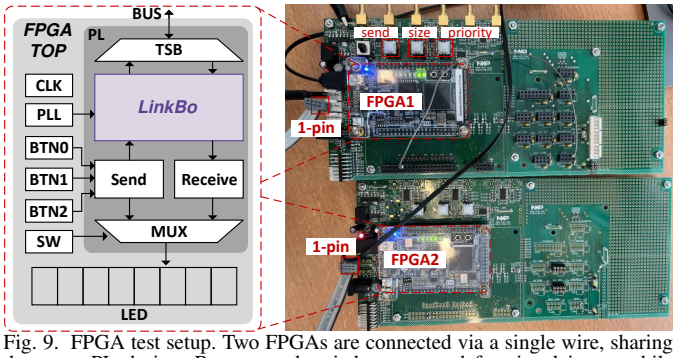

Fig. 9. FPGA test setup. Two FPGAs are connected via a single wire, sharing the same PL design. Buttons and switches are used for signal input, while LEDs display the transmitted and received data.

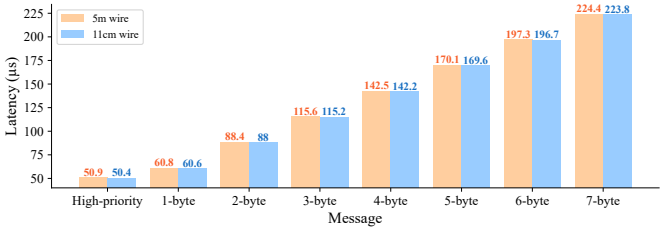

Fig. 10. Latency for different messages. The HP message is transmitted faster than the LP message, even though they carry the same payload. The wire length does not significantly affect latency.

encoder uses an XOR gate to generate Manchester code. The driver utilizes an output register to eliminate race conditions and hazards in the combinational logic.

VI. Experiments and Results

FPGA test setup. Fig. 9 shows the FPGA test setup. In our experimental setup, two Altera Cyclone-IV FPGAs are used and connected via a single pin wire. Each FPGA features three buttons (BTN) to pick input signals (send, size, and priority) and eight LEDs to indicate payload data. FPGA 1 transmits data to the LinkBo module, showing the size and priority of the LEDs, while FPGA 2 receives data from the LinkBo module, with the LEDs showing a payload byte. A Phase-Locked Loop (PLL) can control the clock frequency up to 100MHz and give it to LinkBo module (3MHz in this test). Tri-state buffer (TSB) manages the bus output or input on the shared communication line, and a pull-up resistor of PCB ensures that the bus defaults to a high voltage when unconnected.

Latency analysis. Fig. 10 shows that sending HP messages yields a minimum latency of 50.4 µs, while LP ones take 60.6 µs for the same bit number. In addition, LP message latency increases linearly with payload size, peaking at 224.4 µs for a 7-byte message. Comparison of wire lengths shows that they have a minimal effect on latency.

Sensitivity analysis. To assess performance, the length of the cable between two FPGAs is varied, starting with a 5 meter wire for data transfer. Fig. 11 reveals that LP message performance declines rapidly between 5.6 and 9 meters, eventually reaching zero. However, HP messages begin to decline at 15 meters, reaching zero at 23 meters. This discrepancy arises because the synchronization pulse for HP messages is three times longer than that for LP ones, increasing robustness. This also accounts for the 1-wire (ADI) protocol's requirement for longer reset times and bit slots. When the message length

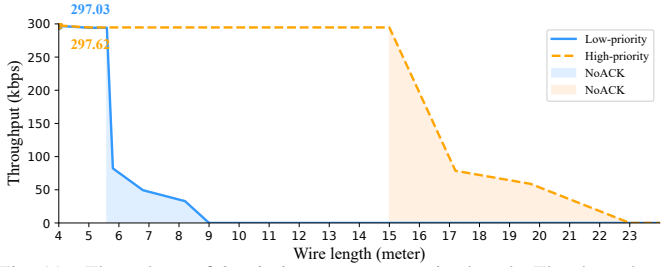

Fig. 11. Throughput of 2 priority message vs wire length. The throughput remains stable up to a certain threshold length, after which it suddenly drops to a low level and then gradually approaches zero. The shadow means RX not give back ACK signal.

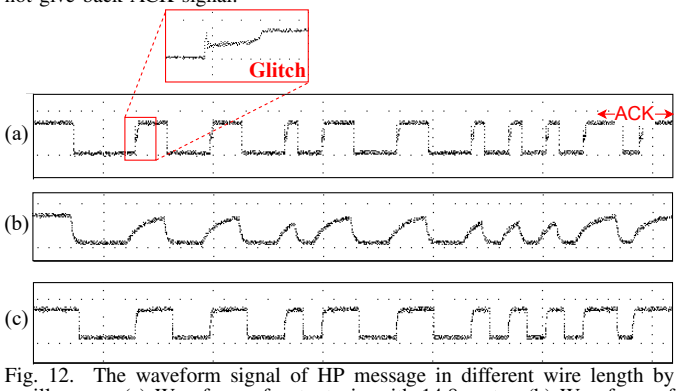

Fig. 12. The waveform signal of HP message in different wire length by oscilloscope. (a) Waveform of output pin with 14.8 meter. (b) Waveform of 14.8 meter bus wire. (c) waveform of 11 centimeter bus wire.

exceeds a threshold, signal degradation prevents the receiver from reconstructing a full byte, thereby no ACK on the bus. The curve in Fig. 11 is similar to that in Fig. 7, showcasing a sharp drop beyond a certain point. Unlike the previous high-level test that altered only one parameter, varying the length of the wire in this experiment can affect the inductance, the load resistor, and the load capacitance concurrently.

In Fig. 12, (a) presents the output pin waveform when two FPGAs are linked by a 14.8-meter cable, highlighting glitches on the rising edge. (b) shows the waveform of a bus signal over 14.8 meters, while (c) illustrates it over an 11-centimeter distance. Findings show that longer wires shorten high-level duration, extend rise time, and eventually hinder edge detection beyond a threshold length.

Our experiments demonstrate that LinkBo supports distances up to 5.6 meters for LP messages and 15 meters for HP messages, which is noticeably shorter than what 1-Wire (ADI) offers. However, LinkBo is optimized for chip-to-chip communication on a single PCB or multiple PCB rather than for extended distances. As chip-to-chip communication generally occurs over centimeters, LinkBo reduces bit slot duration and elevates clock frequency, thus achieving lower latency and a higher bit rate. As detailed in fig. 13, the highest bit rate reaches about 300 kbps (3 MHz) for low-priority messages and 710 kbps (7.1 MHz) for high-priority messages over a 5-meter wire, although such lengths are impractical for PCBs. At an 11 cm scale, low-priority messages can attain up to 2.3 Mbps, while high-priority messages can reach 7.5 Mbps. This indicates that our protocol provides very high bit rates over short distances and maintains strong performance over 10–20 meters.

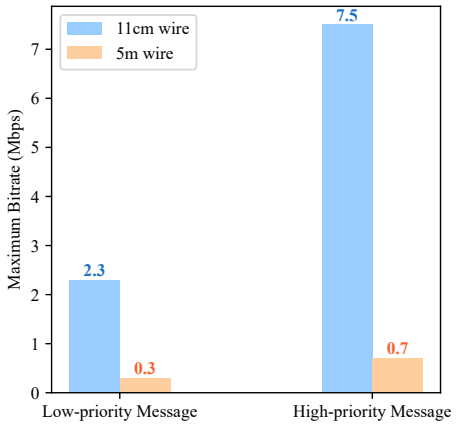

Fig. 13. Maximum bit rate of different priority message for 11cm and 5m wire length. The longer the wire length, the lower the supported bit rate. High-priority messages can support a bit rate that is 2 to 3 times higher than that of low-priority messages.

TABLE I
COMPARISON BETWEEN 1-WIRE (ADI), UNI/O AND LINKBO.

Protocol	1-wire (ADI)	UNI/O	**LinkBo**
Type	Asynchronous	Asynchronous	Asynchronous
Duplex	Half	Half	Half
Bit rate (kbps)	8.33 - 111	10 - 100	**294.8 - 297.6**
EBR (kbps)	5.8 - 77.2	7.97 - 79.7	**158.72 - 252.5**
Latency (µs)	1520 - 4880	810 - 1410	**50.4 - 223.8**
Interrupt	No	No	**Yes**
Multi-byte	No	Yes	**Yes**
CRC	Yes, 8 bits	No	**Yes, 4 bits**
Acknowledge	No	Yes	**Yes**
Distance (m)	20-100	N/A	**5.6-15**

Protocol comparison. Table I shows a comparison of this work with the SOTA protocol in terms of latency and robustness. Half-duplex refers to devices that transmit in one direction at a time. Bit rate specifies the bits sent per second. In Eq. (4), Bit rate R_b represents the kilobits per second, B_{total} is the total bits per transmission, and T is the total duration of a transmission.

$$R_b = \frac{B_{total}}{T} \tag{4}$$

The effective bit rate (EBR) denotes the upper limit for payload data bits transferred per second. In Eq. (5), B_{data} signifies the total number of data bits within a single transmission. Latency is defined as the minimum duration for a single message. The distance represents the maximum range for transmission between two devices. All data is derived from protocol specifications operating under a 3 MHz clock frequency.

$$EBR = \frac{B_{data}}{T} \tag{5}$$

Table I shows that LinkBo outperforms the other protocols in bit rate and throughput. With two devices, it delivers HP messages in 50.4 µs and incurs a maximum delay of 223.8 µs for LP 7-byte messages. The highest EBR occurs with LP messages, and the lowest with HP. While 1-Wire (ADI) supports long distances (up to 50 m), it requires longer resets and bit slots. UNI/O reduces reset overhead but uses 8-bit synchronization and dual ACKs per byte, increasing latency and lowering EBR. In contrast, LinkBo uses only 2 sync bits and shorter bit slots, reducing message size and latency.

VII. CONCLUSION

In this paper, we presented LinkBo, a novel single-wire, low-latency, robust and high-throughput protocol for variable-distance chip-to-chip communication, include its protocol definition and its hardware architecture. LinkBo can achieve a transmission with different priority and error detection method, using Manchester encoding. High priority and interrupt can reduce minimum delay of transmission. We designed a novel hardware architecture for LinkBo protocol and implemented on a hardware demonstrator comprising two FPGA boards. Our protocol achieves a low latency of just 50.4 µs, which is lower by at least 20X and 6.3X compared to the SOTA 1-wire (ADI) and UNI/O protocol, respectively. Furthermore, we investigated robustness by changing the wire length and bit rate on FPGA performance, demonstrating that our hardware facilitates communication over distances up to 15 meters at around 300 kbps, and 11 cm at 7.5 Mbps. This capability ensures robustness suitable for different distance from chip-to-chip communication to standard PCB-level communication.

REFERENCES

[1] F. Leens, "An introduction to i 2 c and spi protocols," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.
[2] N. S. B.V, "Sot109-1," 2016.
[3] N. S. B.V, "Sot163-1," 2016.
[4] A. D. Inc., "Overview of 1-wire technology and its use," 2008.
[5] T. I. Inc., "Implementing 1-wire enumeration for tmp1826 with tm4c129x microcontrollers," 2018.
[6] A. Al-Sammak, "Encoder circuit for inverse differential manchester code operating at any frequency," *Electronics Letters*, vol. 38, no. 12, p. 1, 2002.
[7] M. T. Inc., "Uni/o bus sepecification," 2009.
[8] S. NV, "Stm8 swim communication protocol and debug module," 2016.
[9] C. A. dos Reis Filho, E. Da Silva, E. d. L. Azevedo, J. A. Seminário, and L. Dibb, "Monolithic data circuit-terminating unit (dcu) for a one-wire vehicle network," in *Proceedings of the 24th European Solid-State Circuits Conference*, pp. 228–231, IEEE, 1998.
[10] H. Rahman and M. T. Arefin, "Design and analysis of an experimental data and clock multiplexing technique for generating faster single wire synchronous data bus," in *2019 2nd International Conference on Innovation in Engineering and Technology (ICIET)*, pp. 1–5, IEEE, 2019.
[11] H. F. Rezaei and A. Kruger, "Wired-fm, a novel distributed digital single-wire field bus," in *2012 IEEE International Conference on Electro/Information Technology*, pp. 1–4, IEEE, 2012.
[12] B. Peiffer and A. Kruger, "Physical layer architecture for 1-wire sensor communication bus: Binary channel code division multiple access," in *2011 IEEE Sensors Applications Symposium*, pp. 100–105, IEEE, 2011.
[13] T. Nikolic, G. Djordjevic, and M. Stojcev, "Simultaneous data transfers over peripheral bus using cdma technique," in *2008 26th International Conference on Microelectronics*, pp. 437–440, IEEE, 2008.
[14] H. F. Rezaei and A. Kruger, "Low weight double layer coded cdma as a novel physical layer for onewire bus communication in sensor networks," in *2013 IEEE Sensors Applications Symposium Proceedings*, pp. 15–20, IEEE, 2013.
[15] S. Ahasan, A. Binaie, A. Dascurcu, M. B. Dastjerdi, R. Garg, M. Johnson, A. Galioglu, A. Natarajan, and H. Krishnaswamy, "Frequency-domain-multiplexing single-wire interface and harmonic-rejection-based if data de-multiplexing in millimeter-wave mimo arrays," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 5, pp. 1360–1373, 2021.
[16] R. Garg, G. Sharma, A. Binaie, S. Jain, S. Ahasan, A. Dascurcu, H. Krishnaswamy, and A. S. Natarajan, "A 28-ghz beam-space mimo rx with spatial filtering and frequency-division multiplexing-based single-wire if interface," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 8, pp. 2295–2307, 2021.
[17] D. Simic, K. Guo, and P. Reynaert, "A 420-ghz sub-5-m range resolution tx–rx phase imaging system in 40-nm cmos technology," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 12, pp. 3827–3839, 2021.
[18] M. Sprachmann, "Automatic generation of parallel crc circuits," *IEEE Design & Test of Computers*, vol. 18, no. 3, pp. 108–114, 2001.