A Local Interconnect Network Controller for Resource-Constrained Automotive Devices

Kwonneung Cho, Hyun Woo Oh, Jeongeun Kim, Young Woo Jeong, and Seung Eun Lee*
Department of Electronic Engineering
Seoul National University of Science and Technology
Seoul, Republic of Korea
{chokwonneung, ohhyunwoo, kimjeongeun, jeongyoungwoo, *seung.lee}@seoultech.ac.kr

***Abstract*— As the amount of data for automotive systems is increased, a dedicated communication controller for in-vehicle networks is required. This paper proposes a local interconnect network (LIN) controller for resource-constrained devices. The designed LIN controller efficiently reduces the workload of target devices by processing the LIN frame header, data response, and protocol errors. To demonstrate the feasibility of design, a Cortex-M0 is employed as a main processor and connected to the LIN controller. We implemented a LIN node by programming the processor, and the functionality of LIN controller was verified with a LIN frame analyzer and hardware scope. In addition, we analyzed the affection of communication loads on the processor and evaluated the benefits of LIN controller.**

Keywords—LIN controller, automotive system, Cortex-M0, resource-constrained device

I. INTRODUCTION

With the development of intelligent and self-driving vehicles, a lot of electronic control units (ECUs) are mounted on a vehicle, and the traffic of in-vehicle networks is increased [1-2]. Local interconnect network (LIN) is widely employed as an in-vehicle network with controller area network (CAN) and CAN with flexible data rate (CAN-FD) [3]. As the CAN and CAN-FD protocol provide the high-speed data rate with communication reliability, CAN and CAN-FD have been applied to the networks where immediate response and stability are essential [4]. On the other hand, LIN is suitable for resource-constrained devices since LIN provides a low-cost implementation of in-vehicle network with a single communication bus and relatively slow data rate [5].

The LIN communication is implemented by processing the LIN frame header and responding to the header as shown in Fig. 1. The LIN frame is shared with all LIN nodes through the LIN bus, and the LIN nodes need to identify the validity of the frame to respond. For identifying the valid frame, two sequences are required. One is to compare the length of break field and sync field, and the other is to check the frame ID and its parity in the PID field.

As described in Fig. 1, the LIN frame header maintains dominant value for 13 nominal bit times in the break field, and repeats dominant and recessive value for every bit time in the sync field. The relation of the length between the break field and sync field is critical to ensure the validity of LIN frame header. Since the tendency of LIN signal which appears in the break field and sync field can also appear in the data field, the break field has to maintain dominant value at least 11-bit times [6]. Therefore, a LIN node needs to compare the length of break and sync field in order to identify a valid break-sync pair. In addition, as the publishers and subscribers of the LIN responses are pre-defined with the frame ID, a LIN node needs to check the frame ID and its parity when a valid break-sync pair is received.

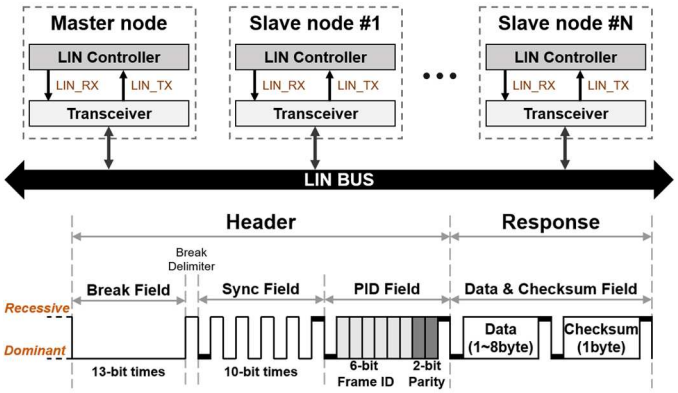

Fig. 1. LIN network and frame format

Unfortunately, processing the LIN frames with software causes a lot of workload to the resource-constrained devices. The main processors of the devices require peripheral controls such as timer or GPIO, and need to perform interrupt service routine (ISR) frequently to process the LIN frame. Since the LIN frames are transferred periodically, the loads of LIN communication lead to performance degradation. Therefore, employing a dedicated hardware controller for LIN communication is required to process increasing data in the resource-constrained automotive devices.

In this paper, a hardware LIN controller for resource-constrained automotive devices is proposed. The designed LIN controller automatically detects a valid break-sync pair, data rate of LIN frame, and all sources of communication error. As a result, the main processor is able to perform LIN communication with only read or write configuration registers in ISR. These simplified operations efficiently reduce the workload of the main processor. To evaluate the benefits of the designed LIN controller, a Cortex-M0 was employed as a main processor, and connected to the LIN controller through advanced high-performance bus (AHB). The performance of LIN controller is evaluated by measuring the benchmark score of the processor and analyzing the affection of LIN communication. The functionality of LIN controller was verified with a LIN frame analyzer tool and hardware scope. Additionally, the feasibility of LIN controller was demonstrated by communicating with an NXP automotive development board.

2022 IEEE International Conference on Consumer Electronics (ICCE) | 978-1-6654-4154-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCE53296.2022.9730493

978-1-6654-4154-4/22/$31.00 ©2022 IEEE

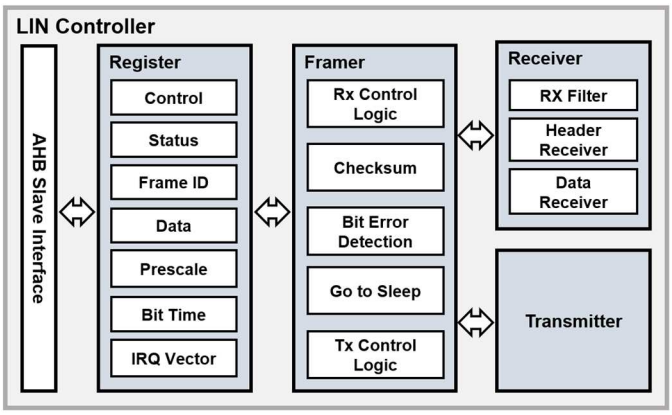

Fig. 2. Architecture of LIN controller

II. Architecture of LIN Controller

Structural design is applied to the LIN controller and the overall architecture is presented in Fig. 2. The AHB slave interface, register bank, framer, receiver, and transmitter are included in the LIN controller. The LIN controller receives the LIN bus signal through the RX filter. The RX filter prevents the noise of LIN bus by blocking rapid changes of signal. The RX filter provides flexible noise filtering by changing the filtering time according to the operating frequency of main processor. The header receiver detects a valid break-sync pair, and calculates communication data rate, and identifies a sleep state of LIN bus by checking the length of LIN bus signal. When a valid header is received, the header receiver triggers an interrupt request and informs the calculated bit time of LIN frame to the data receiver and transmitter. The data receiver and transmitter perform universal asynchronous receiver transmitter (UART) operations with the received bit time scale to response to the data field. The framer controls the receiver and transmitter for response to the data field. The frame format is constructed with the responded data. Additionally, the checksum calculation, detecting bit collision, and go-to-sleep command are processed in the framer.

A main processor accesses the register bank through the AHB slave interface, and controls the LIN controller by reading or writing the registers. The IRQ vector register informs the interrupt sources that include the information of receiving header, error occurrence and end of transfer. When a valid LIN header is received and the interrupt signal is triggered, the main processor reads the frame ID register and writes the control register to decide whether to respond or ignore the received frame. The main processor is able to monitor the LIN communication process by reading the status register and bit time register. The pre-scale register sets scale counters for the RX filter and internal counter in the header receiver. The LIN controller supports operating frequency from 2MHz to 64MHz through the pre-scale register.

III. Implementation and Verification

The proposed LIN controller was designed with Verilog HDL and simulated with Altera-Modelsim and Synopsys-VCS. In order to demonstrate the functionality of LIN controller, a system-on-chip (SoC) design which includes a Cortex-M0 core, LIN controller, on-chip memories, system bus, and peripherals

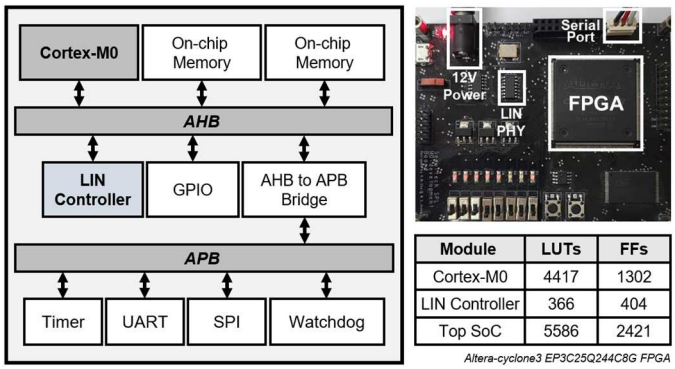

Fig. 3. Implementation of the SoC design on FPGA

was implemented on Altera-Cyclone III FPGA. A LIN node was realized with the SoC design by programming the Cortex-M0. The Cortex-M0 core accesses the designed LIN controller through AHB, and controls the LIN controller to perform LIN communication.

Fig. 3 shows the top-level architecture of SoC design and prototype PCB board. The PCB board contains LIN PHY chip and 12V power port to provide LIN physical layer. In the Altera-Cyclone III FPGA, the designed LIN controller utilizes 366 look-up tables (LUTs) and 404 flip-flops (FFs) of the FPGA resource. The amount of resources for LIN controller occupies 6.55% LUTs and 16.69% FFs of the SoC design.

In order to verify the LIN controller, we set up the verification environment as shown in Fig. 4. Vector-VN1630A and PicoScope-5000 were employed as a LIN interface and hardware scope to analyze LIN frames and physical layer. Vector-Canoe software tool provides virtual LIN nodes with a standard LIN description file (LDF) that contains information of LIN nodes and network schedules. We constructed a LIN network with the Canoe virtual nodes, the implemented LIN nodes with the FPGA board, and the test board. NXP-DEVKIT-S12VR automotive development board which supports LIN physical layer and software library was employed for the test board. The LIN controller successfully detects valid LIN headers and responses to the headers without any missing frames. The captured waveform is presented in Fig. 4.

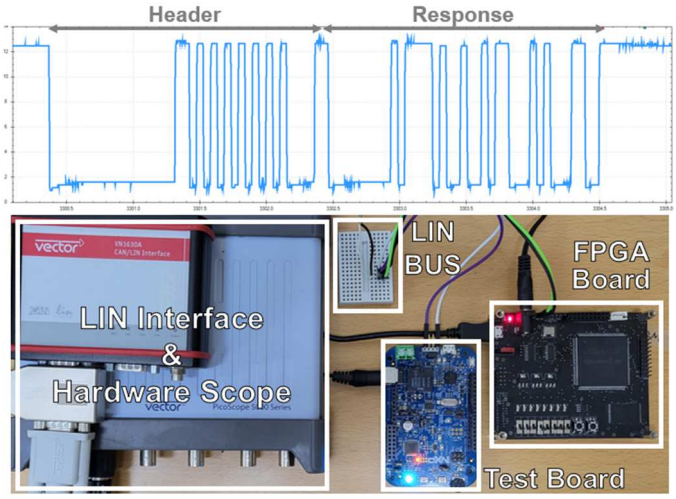

Fig. 4. Verification environment and captured waveform

The functionality of LIN controller was verified by the experiments. The LIN controller successfully provides 2Kbps to 20Kbps data rate in the operating frequency range of 2MHz to 64MHz. The data rate satisfies the specification of LIN 2.2A, and the range of operating frequency is suitable for resource-constrained devices which operate with low speed [6]. Further experiments for communication error handlings such as parity error, bit collision, frame error, and checksum error are conducted. When an error occurs, the LIN controller informs the main processor of the corresponding error source, and the main processor ignores or stops the current frame.

IV. Result and Discussion

A. Area efficiency

In order to analyze the area efficiency of LIN controller, the number of logic cells is measured by synthesizing the SoC design with Cyclone III FPGA, 130nm CMOS process and 180nm CMOS process. Table I indicates the synthesis results. The logic cells of FPGA contain resources of LUTs and FFs, and the logic cells of CMOS process are calculated based on 2-input NAND gate. Compared to the Cortex-M0 core, the LIN controller occupies 13.46% of FPGA resources, 21.05% of logic cells in 130nm process and 22.62% of logic cells in 180nm process. Considering that the Cortex-M0 core is optimized for resource-constrained devices, the proposed LIN controller enables the main processor to implement LIN communication with area efficiency. The peripherals in Table I include a UART, timer and GPIO which are generally used to implement a software LIN node. Although the LIN controller utilizes more resources than the peripherals, additional area benefits are gained since the LIN controller reduces the software complexity of main processor, achieving memory efficiency.

B. Performance analysis

To demonstrate the benefits of LIN controller, the performance of the SoC design is evaluated by running Dhrystone 2.1 benchmark with the load of LIN communication. In the benchmark test, the requests for LIN communication affect the performance of main processor. However, as the LIN controller reduces the workload of main processor, the benefit of LIN controller is confirmed by comparing the benchmark scores with the original scores which are measured without any external interrupts. Table II presents the summary of Dhrystone benchmark. The benchmark results without LIN communication only depend on the performance of main processor, and 27.736DMIPS was measured at 50MHz clock frequency. In the benchmark test with LIN communication, a LIN frame is transferred at every 5ms with 19.2kbps data rate. The LIN frame requests interrupt the operation of main processor. However, the benchmark score was measured at 27.723DMIPS. Since the LIN controller processes the LIN frame with dedicated hardware, only 0.05% performance degradation is occurred. Therefore, the proposed LIN controller demonstrated the advantage of reducing the workload of main processor for LIN communication.

TABLE I. Logic Cell Utilization

Module	Logic Cells		
	Cyclone III FPGA	*CMOS 130nm*	*CMOS 180nm*
LIN Controller	770	5,319	5,301
Cortex-M0	5,719	25,269	23,435
Peripherals	698	4,328	4,347

TABLE II. Dhrystone 2.1 Benchmark Summary

Entity	Value	
	Result without LIN	*Result with LIN*
RUN Count	151,552	151,552
Execution Time	3.109s	3.111s
DMIPS	27.736	27.723
DMIPS / MHz	0.555	0.554

V. Conclution

In this paper, a LIN controller for resource-constrained automotive devices is proposed to deal with the increasing data of in-vehicle network. As resource-constrained devices have restricted performance and resources, it is important to efficiently reduce the communication loads. The proposed LIN controller meets the goal by processing the LIN frame with compact resources. The LIN controller was designed with Verilog-HDL and implemented on Altera-Cyclone III FPGA. The functionality of LIN controller was verified by employing the Vector-Canoe software analyzer, Vector-VN1630A interface, PicoScope-5000 hardware scope, and NXP automotive development board. In order to demonstrate the benefits of LIN controller, an SoC design which includes the Cortex-M0 core, LIN controller, and peripherals is implemented. As a result, the area and performance benefits of LIN controller were demonstrated by experiments.

Acknowledgment

This paper was supported by Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOTIE) (P0017011, HRD Program for Industrial Innovation)

References

[1] C. J. Taek, J. Jun Yun, M. J. Hwan and M. Jung, "Implementation of a binary translation for improving ECU performance," IEEE Inter-national Conference on Consumer Electronics (ICCE), 2019, pp. 1-3.

[2] M. Z. MANIC, M. Z. PONOS, M. Z. BJELICA and D. SAMARDZIJA, "Proposal for graphics sharing in a mixed criticality automotive digital cockpit," IEEE International Conference on Consumer Electronics (ICCE), 2020, pp. 1-4.

[3] A. Srivastava and D. Adhikari, "CAN-LIN bridge for driver assistance and passenger comfort an optimized resource approach," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017, pp. 506-510.

[4] S. Jin, J. -G. Chung and Y. Xu, "Signature-Based Intrusion Detection System (IDS) for In-Vehicle CAN Bus Network," IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1-5.

[5] Y. Lee, E. Kang, C. Hiler, Y. Park and J. Choi, "EMI Effects on Electrical Parameters in Fiber Optic Converters for LIN (Local Interconnect Network) Communication," International Symposium on Electro-magnetic Compatibility - EMC EUROPE, 2020, pp. 1-6.

[6] LIN Specification Package Revision 2.2A, Dec, 2010.