BUS GUARDIAN DESIGN FOR AUTOMOBILE NETWORKING ECU NODES COMPLIANT WITH FLEXRAY STANDARDS

Gang-Neng Sung, Chun-Ying Juan, and Chua-Chin Wang, Senior Member, IEEE

Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan 80424.
Email: ccwang@ee.nsysu.edu.tw

ABSTRACT

This paper presents a Bus Guardian Design used in an in-car network compliant with FlexRay standards. FlexRay is a new standard for data/signal communication among electronic devices installed in a vehicle. An 8051-compatible microcontroller was used to implement the system controller. Most important of all is that the Bus Guardian (BG) in charge of security and safty is proposed and implemented. This work was implemented by hardware description language (HDL) and verified by Xilinx field-programmable gate array (FPGA).

Keywords—**FlexRay, baseband, bus guardian, automobile electronic, in-car networking, 8051.**

1. INTRODUCTION

Car electronics has been deemed as the 4th ”C” right after Computer, Communication and Consumer electronics. The car electronics include power train, chassis safety, peripheral electronics control system, telematics communication system, in-car networking, etc. Many novel electronic devices have been introduced and installed in recently publicized cars, e.g., car TVs. Therefore, an in-car network has been proposed to control and supervise all of the automobile electronics. Prior in-car networks were mainly composed of CAN (controller area network) or LIN (Local Interconnect Network) which emphasized that safety and reliability. However, the limited 1 Mbps bandwidth is not sufficient for rapid growth of the data/signal in an in-car networking. By contrast, MOST (Media Oriented Systems Transport) [1] network provided a very efficient mechanism for transporting high volumes of media information, but lack of control capability.

FlexRay is a new communication protocol proposed by several automobile power houses, including BMW, Daimler-Chrysler, General Motos, Freescale, Philips, Robert Bosch, Volkswagen, etc. It is designed to provide message and data exchange among electronic devices installed in a vehicle. It will not replace the existing network, but it can combine and integrate with existing network systems, including CAN, LIN, MOST and J1850 protocol, etc. FlexRay requires 10 Mbps data rate in either one of the two channel of an ECU (electronic control unit). If a single channel is used alone, the speed of the total data rate is expected to be 20 Mbps. Therefore, even the video signals, multimedia and control signals can communicate via the FlexRay system in such a high bandwidth. The ultimate goal is that the automobile is X-by-wire (X = steer, break, accelerate, A/V, safety, etc.). Fig. 1 shows that the explosive view of a FlexRay network. Fig. 2 shows the block diagram of ECU nodes connected in a FlexRay system. Each ECU node contains at least a Host, a Communication Controller [2], two Bus Drivers and two Bus Guardians.

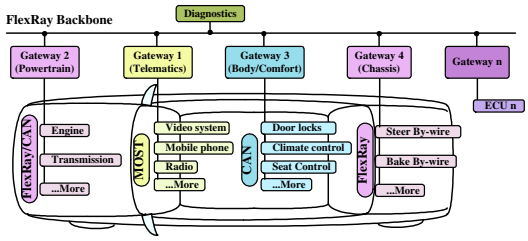

Fig. 1. Explosive view of a FlexRay network

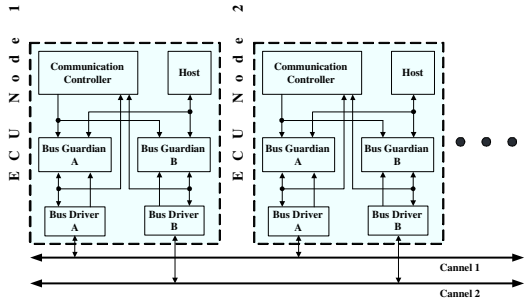

Fig. 2. ECU node on the FlexRay bus

This research was partially supported by National Science Council under grant NSC96-2628-E-110-018-MY3.

2. BUS GUARDIAN DESIGN

The proposed Bus Guardian is in charge of security and safety for FlexRay communications systems. It protects a channel from interference caused by any message that is not aligned with the communication schedule. The function of the Bus Guardian is required to supervise the communication controller during different protocol operating modes like wakeup, communication startup, and synchronized operation.

2.1. Bus Guardian Operation Modes

Fig. 3 gives an overview of the Bus Guardian operation modes as well as mode transitions. There are four modes which are described as follows:

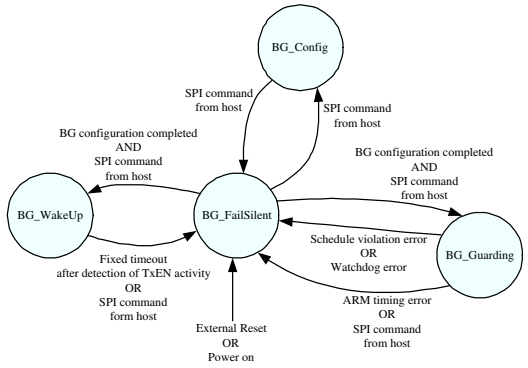

Fig. 3. Bus Guardian functional operation diagram

2.1.1. BG_Config mode

Configuration registers in the Bus Guardian can only be accessed in *BG_Config mode*, which can only be entered from *BG_FailSilent mode*. The configuration register starts from the address which is programmed and increments automatically in every SPI (serial peripheral interface) communication period until the stop instruction (FFFFh) is encountered. The stop instruction is written by the Host. After writing the stop instruction, the Host shall start verification of the configuration data via read back. Then, the Bus Guardian sets the configuration complete flag, while the operation mode is switched back into *BG_FailSilent mode*.

2.1.2. BG_FailSilent mode

In *BG_FailSilent mode*, the bus can not be accessed. It is the initial mode of the Bus Guardian when an error occurs. As soon as an error occurs, an interrupt signal is generated and the corresponding error flags are set. Besides, *BG_FailSilent mode* can be entered from any other operation mode by an SPI instruction from the Host. *BG_FailSilent mode* can only be left by an SPI instruction from the Host.

2.1.3. BG_WakeUp mode

In *BG_WakeUp mode*, the communication controller can transmit a wakeup message to the Bus Guardian and give the permission to wake up the whole set of ECU nodes in the FlexRay communication system. *BG_WakeUp* mode can only be entered from the *BG_FailSilent mode* by an SPI instruction from the Host.

2.1.4. BG_Guarding mode

When Bus Guardian is configured (made earlier in *BG_Config mode*), the scheduler is initialized. Then, the bus guarding function is active in *BG_Guarding mode*. It provides the standard operation of the ECU node and can only be entered from *BG_FailSilent mode* by an SPI instruction form Host after the configuration of the Bus Guardian is completed. In the *BG_Guarding mode*, the Bus Guardian dominates the communication bus and detects errors.

2.2. Bus Guardian Implementation

The proposed Bus Guardian implementation is shown in Fig. 4. Functional blocks are connected with 16-bit buses. The detailed functionality of each functional block is described in the following text.

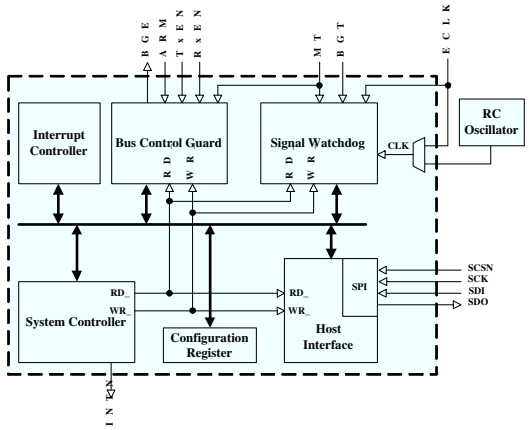

Fig. 4. Block diagram of Bus Guardian

2.2.1. System Controller

The System Controller of the Bus Guardian is implemented by an 8051-compatible microcontroller (μC). It is in charge of triggering all internal Bus Guardian states and operating modes, decoding and executing SPI commands, and error handling. The System Controller communicates with the Host via the SPI interface to select the operating mode. If an error is reported by the Signal Watchdog or by the Bus Control Guard,

the System Controller writes the corresponding status information to the registers in SPI and sends an interrupt request to the Host.

2.2.2. Bus Control Guard

The bus guardian scheduler is contained in the Bus Control Guard. The Bus Control Guard enables transmitting with Communication Controller via the bus only according to the configured bus guardian scheduler. The schedule includes active slots in the static segment, the dynamic segment, and the symbol window. The static segment is a portion of the communication cycle where the media access is controlled via a static Time Division Multiple Access (TDMA) scheme. During the static segment, the access of the media is determined solely by the progression of time. The dynamic segment and the symbol window are optional elements of the communication cycle. In addition, the Bus Guardian may be configured to enable or disable transmit access during the dynamic segment. If the communication controller attempts to transmit to the bus when it is not allowed, the Bus Control Guard disables transmit access to the medium and the Bus Guardian reports an error to the Host. The Bus Guardian stays in the *BG_FailSilent mode* after detecting an error. The Host must initiate a mode transition to the *BG_Guarding mode* in order to recover from such an error scenario.

The scheduler inside the Bus Control Guard is clocked directly with the MT (microtick) signal from the Communication Controller such that it can synchronize every ECU in the FlexRay communication system. The ARM signal is generated by the Communication Controller to indicate the start of the communication cycle to the Bus Guardian. The TxEN signal is a transmission enable signal issued by Communication Controller. If the TxEN signal is enabled out of the communication schedule, the scheduler denies the bus access and the BGE (bus guardian enable) signal is not enabled to stop the signal transmitting to the bus.

2.2.3. Serial Peripheral Interface (SPI)

The serial peripheral interface provides the communication path between Host and Bus Guardian. It is used for Bus Guardian configuration and for exchange of status and control information. The SPI interface supports full duplex operation at a data rate of at least 1 Mbit/s.

The SPI interface provides four interface signals, including chip select. Bit sampling is performed at the falling clock edge and the data is shifted at the rising clock edge. SPI interface signals are implemented as follows:

- **SCSN:** The SPI chip select, active "Low". The Host enables this signal to start one SPI communication period.
- **SCK:** The SPI clock, default voltage level is "Low" due to low-power concept and save the power consumption.
- **SDI:** The SPI data input. The Bus Guardian receives data from the Host through this signal.
- **SDO:** The SPI data output, floating during "High" level of pin SCSN. The Bus Guardian transmits data to the Host through this signal.

Within one SCSN cycle, there are exactly 16 clock periods. Any deviation in the number clock periods results in an SPI Failure Interrupt. The access, thus, is ignored by Bus Guardian. The data of SPI could be the instructions of the Host or the parameters of the communication configuration. Fig. 5 shows the timing diagram in one SPI communication period. A 16-bit instruction can be split into H8 and L8, which length are 8 bits each. In the H8 segment, the 8th bit denotes to read or write, the 5th to 7th bits are the register type, and the 1st to 4th bits are the Bus Guardian operation mode or register bit selector. In the L8 segment, the entire 8 bits denote the start memory address in the burst mode when the registers are accessed. When switching the operating mode, the lower 4 bits are ECC code which can ensure that the instructions are correct and the higher 4 bits are ignored. The SPI instruction frame is shown in Fig. 6.

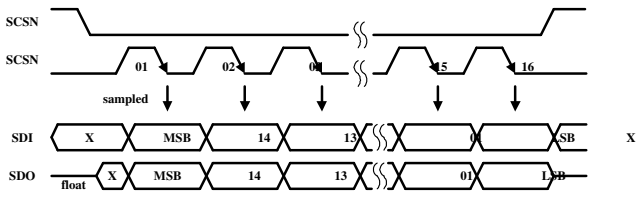

Fig. 5. SPI timing diagram

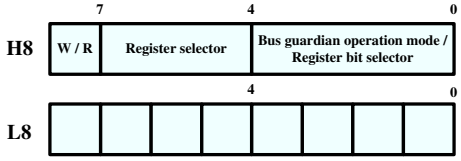

Fig. 6. A SPI instruction frame

2.2.4. Signal Watchdog

The Bus Guardian contains watchdog circuit to moritor the clock frequency and Bus Guardian clock signals delivered from the Communication Controller. Detection of the macrotick signal (MT) is performed by means of the bus guardian tick signal (BGT), which is also provided by the Communication Controller. The Signal Watchdog checks the ratio between the

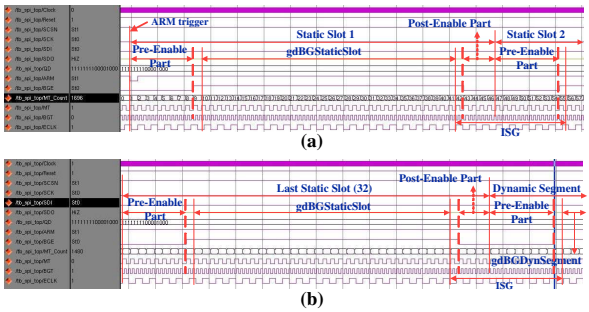

Fig. 7. Bus Guardian timing within the static segment

MT period and the BGT period to detect if there is any deviation from the nominal ratio exceeding a pre-defined configurable limit. The Signal Watchdog also supervises the BGT period using the Bus Guardians local clock signal CLK as a reference which is generated by an RC oscillator circuit or an external clock (ECLK). The Bus Guardian supervises all of the signals from the Communication Controller that are used to synchronize the bus guardian. Additional fail detection circuits are required to detect missing clock signals at the MT and BGT inputs and at the local clock source.

The BGT signal is directly derived from the clock oscillator of the Communication Controller by a frequency divider and its period is not influenced by the MT clock calibration. Supervision of the BGT signal is performed by the internal clock signal CLK of the Bus Guardian. In short, the Bus Guardian contains clock signal fail detectors in order to detect missing signals at the Bus Guardian inputs MT, BGT, and ECLK.

2.2.5. Interrupt Controller

The interrupt controller is responsible for allocating the bus in the Bus Guardian and transmitting the interrupt signal to the Host. The priority of the bus privilege form high to low is assigned to Signal Watchdog, SPI, and Bus Control Guard.

3. IMPLEMENTATION AND VERIFICATION

This design is carried out by Verilog HDL code and verified by Xilinx Virtex-4 MB Development Board. The total equivalent gate count of the proposed design is 46,166 including the 8051-compatible microcontroller. The system clock is 80 MHz. In this design, we implement the system controller with the 8051-compatible microcontroller to make the Bus Guardian more flexible. Different engineers can enhance or add different operation modes based on the standard FlexRay specification to achieve higher safety or more powerful operation. Table 1 gives an comparison with [3]. Fig. 7 (a) shows the Bus Guardian timing within the static segment with the MT period is 600 ns, the BGT period is 300 ns, the ECLK period is 1 us, and the RC_CLK period is 1 us. Fig. 7 (b) shows the Bus Guardian timing at the beginning of the dynamic segment. All of the simulations are verified to be compliant with the FlexRay specification.

	Gate Count	Main Clock
[3]	60,000	47 MHz
ours	46,166 (with a μC)	80 MHz

Table 1. Comparison in gate count and clock rate

4. CONCLUSION

This paper proposed a hardware implementation of Bus Guardian in the FlexRay automotive communication system using CMOS cell-based design flow. In the system controller design, we use the 8051-compatible microcontroller which can provide more flexibility. The implementation on Xilinx FPGA justifies the superiority of the proposed design.

Acknowledgment

The authors would like to express their deepest gratefulness to CIC (Chip Implementation Center) of NAPL (National Applied Research Laboratories), Taiwan, for their thoughtful chip fabrication service. The authors also like to thank “Aim for Top University Plan” project of NSYSU and MOE, Taiwan, for partially supporting this investigation.

5. REFERENCES

[1] H. Schopp, and D. Teichner, “Video and Audio applications in vehicles enabled by networked systems,” *International Conference on Consumer Electronics,* pp. 218-219, June 1999.

[2] A. Techmer, and P. Leteinturier, “Implementing FlexRay on Silicon,” *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006 (ICNICONSMCL ’06),* pp. 34-40, April 2006.

[3] P. M. Szecowka, and M. A. Swiderski, “On hardware implementation of flexray bus guardian module,” *International Conference on Mixed Design of Integrated Circuits and Systems 2007 (MIXDES ’07),* pp. 309-312, June 2007.