2009 Fourth Latin-American Symposium on Dependable Computing

A Low-Cost On-Line Monitoring Mechanism for the FlexRay Communication Protocol

Yasser Sedaghat
Dependable Systems Laboratory
Sharif University of Technology
Tehran, Iran
y_sedaghat@ce.sharif.edu

Seyed Ghassem Miremadi
Dependable Systems Laboratory
Sharif University of Technology
Tehran, Iran
miremadi@sharif.edu

Abstract

Nowadays, communication protocols are used in safety-critical automotive applications. In these applications, fault tolerance is a main requirement and the existence of single points of failure is a serious threat to system failures. Among the communication protocols, FlexRay is expected to become the communication backbone for future automotive systems. In this paper, we identify single points of failure in the FlexRay protocol by injecting a total of 135,600 single-bit transient faults into all accessible registers of the FlexRay communication controller. The results showed that about 1.2% of all injected faults caused the controller to freeze immediately. Based on these results and detailed study of the FlexRay specifications, a low-cost on-line monitoring mechanism is proposed to check freeze errors. Using this mechanism, the host controller distinguishes about 99.8% all occurred freeze errors into errors due to transient faults or due to standard operation of the FlexRay protocol.

Keywords: Distributed embedded systems, FlexRay protocol, Fault injection, Transient faults, Single point of failure, On-line monitoring.

1. Introduction

Car electronics has been deemed as the 4th "C" right after Computer, Communication and Consumer electronics [1]. Electronics has been the key innovation driver for automotive systems throughout the last decade, and this situation is not going to change in the near future [2]. Nowadays, automotive electronic systems are complex distributed embedded systems which are composed of several electronic control units (ECUs) (sometimes called nodes) interconnected by a communication network. An ECU is composed of a processing unit and a set of actuators and sensors. This unit is connected to the network by a communication controller. The controller is responsible for implementing a communication protocol for transferring ECU's data.

Major automotive systems that rely on networking are relevant to chassis, air-bag, powertrain, body and comfort electronics, diagnostics, X-by-Wire, multimedia and infotainment, and wireless and telematics [3]. To interconnect these systems, there is a need for high bandwidth together with flexibility and determinism. In addition, among these systems, chassis, air-bag, X-by-Wire, and powertrain are safety-critical, and consequently fault tolerance is a main requirement for them [3].

For non safety-critical communication, a number of protocols have become popular, such as LIN, CAN, MOST, Bluetooth, and ZigBee. For safety-critical and more complex applications, several communication protocols have been introduced such as Byteflight, TTP/C, TT-CAN, and FlexRay [3]. Among the latter protocols, the FlexRay protocol is advancing as the predominant protocol and is expecting to become the de-facto industry standard for X-by-wire and other safety-critical applications [2], [4], [5], [6], [7]. As an example, in the BMW X6 "Sports Activity Coupe", the FlexRay protocol is used in steering, braking and suspension systems [8].

The FlexRay protocol were proposed by several industrial members, including BMW, Daimler-Chrysler, Freescale, Philips, General Motors, Robert Bosch, Volkswagen, etc. Three top design objectives were considered in the standardization of the FlexRay protocol: high speed transmission, deterministic communication, and fault-tolerant communication [7].

The overall reliability of the distributed embedded systems not only depends on the reliability of the nodes, but also on the reliability of the communication network which is composed of communication

978-0-7695-3760-3/09 $25.00 © 2009 IEEE
DOI 10.1109/LADC.2009.20

protocols and communication links [4], [9], [10], [11]. In these systems, the communication network is a single point of failure [12]; thus fault-tolerant operation of communication controllers and communication links must be guaranteed.

Several work have investigated fault tolerance and reliability of communication protocols. Effects of masquerade failures [13] and message missing failures [14] have been studied for the CAN protocol by the simulation-based transient single bit-flip fault injection. Also, [15] has investigated effects of simulation-based fault injection in the CAN protocol. Fault tolerance of the TTP/C protocol has been assessed, by heavy-ion fault injection [16] and physical pin-level fault-injection [17]. Moreover, fault tolerance and the occurred failures in this protocol have been studied using heavy-ion fault injection and simulation-based fault injection [18]. Fault-containment and error detection mechanisms of the TTP/C and the FlexRay protocols have been investigated by [19]; this paper also introduces some critical failures and analyzes them on these two protocols. The removal of babbling idiot failures by designing a “Bus Guardian” in CAN, TTP/C, and FlexRay based communication networks have been studied in [20], [21], and [1] respectively. Furthermore, an evaluation of the FlexRay protocol using simulation-based fault injection has been reported in [22] and [23]. In these studies, faults were injected only into about 10% of the FlexRay registers. In [10], an exhaustive evaluation of the FlexRay protocol using simulation-based single bit-flip fault injection has been reported and the occurred errors have been categorized in [11].

In this paper, single points of failure in the FlexRay communication controller have been identified. To do this, transient single bit-flip faults have been injected into all registers of the controller. Fault injection results and detailed study of the FlexRay specifications show that the occurrence of faults in some registers halts the controller immediately. A low-cost on-line monitoring mechanism is proposed to report this halting situation. Using this mechanism, the host controller can distinguish between two types of freezes: 1) freezes due to transient faults, and 2) freezes due to standard operation of the protocol.

The remainder of the paper is organized as follows. Section 2 introduces the FlexRay protocol briefly. Freeze conditions in the FlexRay protocol are presented in Section 3. In Section 4, experimental results and analyses are presented. Section 5 includes the proposed mechanism and finally, the conclusions are given in Section 6.

2. FlexRay protocol concepts

The FlexRay protocol provides flexibility, scalability, and high data rate up to 10 Mbit/sec. The protocol itself offers deterministic data transmission, guaranteed message latency and message jitter. The FlexRay supports dual and redundant transmission channels and transmission mechanism is contention free [24].

In the FlexRay protocol specifications [24] and other its documents, only a bus guardian mechanism and a fault-tolerant clock synchronization algorithm have been mentioned; and other fault-tolerant mechanisms, such as a membership service or a clique avoidance mechanism, should be implemented in software or hardware layers on top of the FlexRay. This will allow to conceive and to implement exactly the services that are needed with the drawback that correct and efficient implementations might be more difficult to achieve in a layer above the communication controller [9].

Based on the experimental results and detailed studies of the FlexRay protocol specifications, a main purpose of this paper is to demonstrate that in this protocol there are several vulnerable parts that injecting a transient single bit-flip fault into them, can lead to halt the controller immediately. It seems that in addition to existing fault-tolerant techniques in the FlexRay communication controller, other useful techniques should be applied.

2.1. Media access scheme in the FlexRay

Communications in the FlexRay protocol are based on sending messages in predetermined interval times which are named communication cycles (bus cycles). These communication cycles are executed periodically. In this protocol, a communication cycle is a concatenation of a time-triggered (or static) window, an event-triggered (or dynamic) window, a symbol window and a network idle time (NIT) window. The time-triggered window uses a Time Division Multiple Access (TDMA) [25] mechanism. In the event-triggered part of the communication cycle, the arbitration mechanism is Flexible TDMA (FTDMA) [26]. The symbol window is a communication period in which a symbol can be transmitted on the network. The NIT window is a communication-free period that concludes each communication cycle. In the FlexRay protocol, data frames are sent in the static slots or in the dynamic slots of each communication cycle. Figure 1 shows an example of a communication cycle in the FlexRay protocol. More details about the media access scheme in the FlexRay protocol have been presented in [24] and [10].

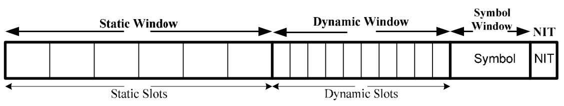


Figure 1. Communication cycle in the FlexRay protocol

2.2. Structure of the FlexRay controller

The FlexRay communication controller consists of six modules [24]: controller host interface (CHI), protocol operation control (POC), coding and decoding (CODEC), media access control (MAC), frame and symbol processing (FSP), and clock synchronization process (CSP). Figure 2 illustrates the relation between these modules.

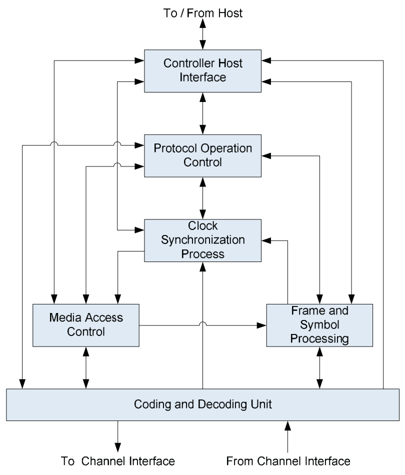

To / From Host
Controller Host Interface
Protocol Operation Control
Clock Synchronization Process
Media Access Control
Frame and Symbol Processing
Coding and Decoding Unit
To Channel Interface
From Channel Interface

Figure 2. Structure of the FlexRay Communication Controller [24]

The CHI module, manages all data and control flow between the host controller and the FlexRay communication controller within each node. The POC module adjusts operational modes of the FlexRay modules. The CODEC module is responsible for encoding the communication elements into a bit stream and for receiving communication elements, making bit streams and investigating correctness of bit streams. The MAC module controls access to the bus. In the FlexRay protocol, media access control is based on recurring communication cycles. The FSP module is responsible for checking the correct timing of received frames and symbol, applying further syntactical tests to received frames, and checking the semantic correctness of received frames. The CSP module is responsible for generating timing units in the FlexRay communication controller, e.g., communication cycles. Moreover this module uses a distributed clock synchronization mechanism in which each node individually synchronizes itself to its cluster by observing the timing of transmitted frames from other nodes [24].

2.3. Clock synchronization

In a distributed communication system every node has its own clock. Because of temperature fluctuations, voltage fluctuations, and production tolerances of the timing source (an oscillator, for example), the internal time bases of the various nodes diverge after a short time, even if all the internal time bases are initially synchronized.

The FlexRay protocol uses a distributed clock synchronization mechanism in which each node individually synchronizes itself to the cluster. The CSP module performs the initialization at cycle start, the measurement and storage of deviation values, and the calculation of the offset and the rate correction values. The offset correction values are calculated based on arrival time of a frame in a static slot and location of action point (expected arrival time of that frame) in that slot. The rate correction values are determined by comparing the corresponding measured time differences from two successive cycles.

After calculating these values and before applying the calculated correction values, they shall be checked against pre-configured limits. If both (offset and rate) correction values are inside the limits, the correction will be performed. If either value exceeds its preconfigured limit, an error is reported to the POC module. More detailed descriptions about the synchronization mechanism in the FlexRay protocol are beyond the scope of this paper and readers are referred to the FlexRay protocol specifications [24].

3. Freeze conditions in the FlexRay protocol

In order to respond to severe errors, the POC module can freeze the communication controller, immediately. More detailed study of the FlexRay specifications [24] showed that there are some conditions that trigger these severe errors.

These conditions, which are named freeze conditions, have been illustrated in Figure 3 and are described as follows: 1) Product-specific errors such as Built-In Self-Test (BIST) and sanity check errors in the communication controller, 2) Requested halts by the host controller, due to occurred errors in it, result in a FREEZE command or a CHI halt command which are sent to the POC via the CHI, 3) Fatal errors detected by the MAC or the FSP Modules, and 4) If synchronization errors persist, or if these errors are severe enough.

In the standard operation of the FlexRay controller, if any of the above conditions is fulfilled, the value of some registers, which are named freeze condition registers in this paper, are changed. After that, based on these values, the POC module freezes the other modules, and then the controller will be halted. Furthermore, in the standard operation of the FlexRay controller, the values of these freeze condition registers are changed (due to fulfill a freeze condition) only in special operational states of the FlexRay modules. For example, as shown in Figure 3, the "Fatal_Protocol_Error_FSP" single-bit register only should be set if the FSP state is "wait for transmission end".

Investigations show that the existence of single points of failure in this protocol is due to the lack of attention to fault effects on the freeze condition registers. For example, in the FlexRay protocol, if a single bit-flip fault sets the "Fatal_Protocol_Error_FSP", the POC module freezes the controller immediately, whereas it may be possible that the FSP state is not the "wait for transmission end". Whereas, if a transient single bit-flip fault affects one of these registers in one node, that node halts and loses its communication with other nodes, immediately and a system failure may occur. Consequently, these freeze condition registers can be considered as "single points of failure" in the controller.

In the next subsection, the protocol specifications have been reviewed carefully and causes of halting the controller, due to freeze conditions, in the standard operation of the protocol are introduced.

3.1. Causes of freezing FlexRay controller [24]

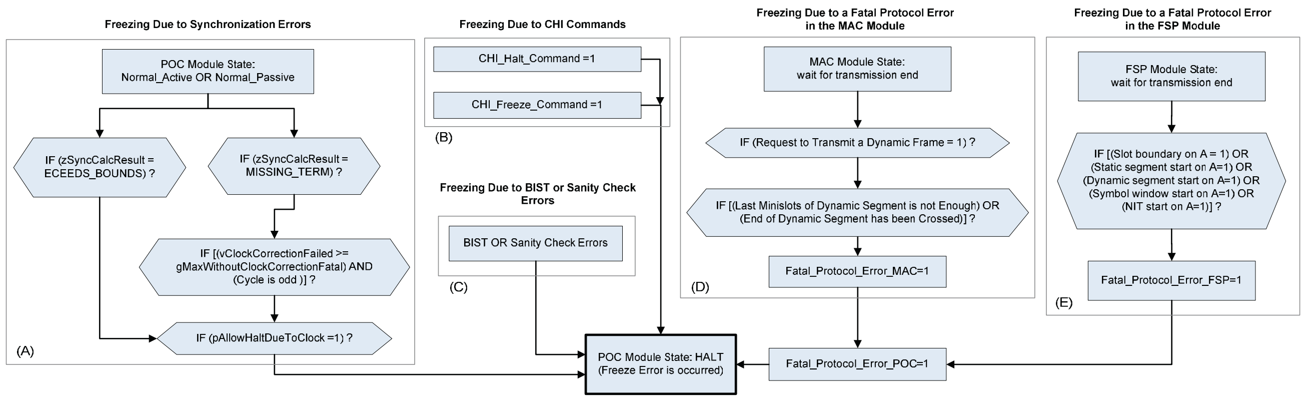
As mentioned before, there are four freeze conditions in the FlexRay protocol. Among these conditions, Product-specific errors such as Built-In Self-Test (BIST) and sanity check errors are not related to the protocol directly, hence they are not considered in this paper. It should be noticed that all registers and flags which are introduced in this paper, have been named based on the FlexRay protocol specifications.

Freezing due to synchronization

As shown in Figure 3, in the FlexRay protocol, "pAllowHaltDueToClock" is a flag in the CHI module. If the node is configured to allow communication to be halted due to severe clock calculation errors, this flag should be set true, otherwise this flag should be set false. On the other hand, if the "pAllowHaltDueToClock" is false, clock synchronization errors could not lead the controller to halt.

At the end of each communication cycle, the CSP module communicates the error consequences of the clock synchronization mechanism's rate and offset calculations [24]. In the protocol specifications, "zSyncCalcResult" assumes one of three values: 1) WITHIN_BOUNDS: indicates that the calculations resulted in no errors. 2) MISSING_TERM: indicates that either the rate or offset correction could not be calculated. 3) EXCEEDS_BOUNDS: indicates that either the rate or offset correction term calculated was deemed too large when compared to pre-configured limits.

As illustrated in Figure 3, in the POC:Normal_Active or POC:Normal_Passive states, if the "pAllowHaltDueToClock" is true and the "zSyncCalcResult" is EXCEEDS_BOUNDS, the POC



Freezing Due to Synchronization Errors
POC Module State: Normal_Active OR Normal_Passive
IF (zSyncCalcResult = ECEEDS_BOUNDS) ?
IF (zSyncCalcResult = MISSING_TERM) ?
IF [(vClockCorrectionFailed >= gMaxWithoutClockCorrectionFatal) AND (Cycle is odd)] ?
IF (pAllowHaltDueToClock =1) ?
(A)
Freezing Due to CHI Commands
CHI_Halt_Command =1
CHI_Freeze_Command =1
(B)
Freezing Due to BIST or Sanity Check Errors
BIST OR Sanity Check Errors
(C)
POC Module State: HALT (Freeze Error is occurred)
Freezing Due to a Fatal Protocol Error in the MAC Module
MAC Module State: wait for transmission end
IF (Request to Transmit a Dynamic Frame = 1) ?
IF [(Last Minislots of Dynamic Segment is not Enough) OR (End of Dynamic Segment has been Crossed)] ?
Fatal_Protocol_Error_MAC=1
(D)
Fatal_Protocol_Error_POC=1
Freezing Due to a Fatal Protocol Error in the FSP Module
FSP Module State: wait for transmission end
IF [(Slot boundary on A = 1) OR (Static segment start on A=1) OR (Dynamic segment start on A=1) OR (Symbol window start on A=1) OR (NIT start on A=1)] ?
Fatal_Protocol_Error_FSP=1
(E)

Figure 3. The Freeze conditions in the FlexRay protocol

freezes all the FlexRay modules and transits its state to POC:halt state; so the controller will be halted. Also, in the controller, there is a counter which counts how many consecutive odd cycles have yielded MISSING_TERM. This counter has been named “vClockCorrectionFailed”. If “zSyncCalcResult” is MISSING_TERM and the cycle is odd and value of “vClockCorrectionFailed” is larger than a predefined value and the “pAllowHaltDueToClock” flag is true, then the POC freezes all the FlexRay modules and transits its state to POC:halt state; and the controller will be halted.

Freezing due to fatal protocol error

In the POC module, there is an input flag which indicates the existence of a fatal protocol error in the communication controller. If the fatal protocol error flag is set, the POC module immediately freezes the other modules and then the controller is halted. The fatal protocol error flag of the POC is originated from two flags with same name in the FSP and the MAC modules.

As shown in Figure 3, the fatal protocol error flag in the FSP module is set when this module waits for end of transmission in a cycle, but either a slot boundary or one of the four segment boundaries, i.e., static segment, dynamic segment, symbol window, and NIT, is crossed. In the MAC module, if a request for transmitting a dynamic frame is received, but either last minislots of dynamic segment was not enough or end of dynamic segment had been crossed, the fatal protocol error flag of the MAC module is set.

Freezing due to CHI requests

In the FlexRay protocol, the host can freeze the communication controller by sending halt or freeze commands. After receiving one of these commands, the POC freezes other modules and enters to POC:halt state and controller is halted, immediately.

4. Experimental analyses

Safety-critical applications have to function correctly even in presence of faults. Faults can be permanent, transient, or intermittent. The transient faults are the most common, and their number is dramatically increasing due to the continuously raising level of integration in semiconductors [27]. Transient single bit-flip errors, which are considered in this paper, are more common consequences of transient faults [28].

4.1. Experimental environment

The FlexRay communication controller was implemented by hardware description language, Verilog HDL, based on State Diagram Language (SDL) descriptions of this controller in the FlexRay specifications [24]. After that, specifications of this controller, e.g. timing and configuration, were tested according to the FlexRay protocol conformance test specification [29]. This controller, according to its specifications [24], has six modules to perform its functions: controller host interface (CHI), protocol operation control (POC), clock synchronization process (CSP), frame and symbol process (FSP), media access control (MAC), and coding and decoding (CODEC).

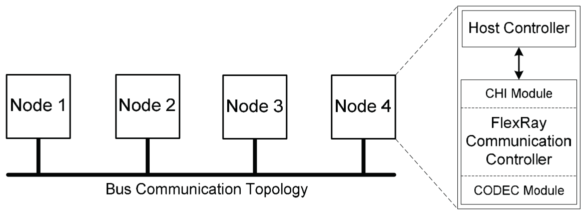


Figure 4. Experimental setup

A cluster was formed consisting of 4 nodes with single bus topology (Figure 4). In this topology, a node is composed of a host controller and a communication controller. The host controller typically is a hardware unit that generates data to exchange with other nodes through a communication channel. In the experiments, instead of a real host, a data generator was implemented to generate static frames with fixed length and dynamic frames with variable length at the start of the communication cycles. In this cluster, each node was allowed to send and receive frames on the communication channel.

Whereas the modeled FlexRay communication controller is not still synthesizable, for investigating the existence of single points of failure in the FlexRay communication controller and identifying them, transient single bit-flip faults were injected into all accessible registers of the controller modules of node 2 and their effects on that controller were investigated.

To simulate the experiments, the ModelSim 5.5 simulation environment was used. The simulation includes five communication cycles; in the second and third cycles, a single transient bit-flip fault was injected randomly, then simulation was resumed two cycles to guarantee that the injected fault shows its effects or is overwritten.

To reach a relative accurate fault sensitivity of each register, 50 transient bit-flip faults were injected to each bit of all accessible FlexRay controller registers and gathered results were investigated. The fault injection process used in this experiment, is the same as is used in [10], and [11].

4.2. Fault injection results

A total of 135,600 transient single-bit flip faults were injected into all 408 single-bit and multiple-bit registers of the FlexRay communication controller. Table 1 shows the occurrence ratio of the Freeze error due to fault injection in the FlexRay communication controller.

Table 1. Fault injection results

Modules	Injected Faults	Freeze Errors	
		#	%
CHI	32350	406	1.2
CODEC	32350	40	0.1
CSP	47750	0	0
FSP	6900	51	0.7
MAC	11050	144	1.3
POC	5200	979	18.8
Total	135600	1620	1.2

Although the presented results in this table show that only about 1.2% of 135,600 injected faults result in the freeze error, it should be noticed that this error freezes all the FlexRay modules and halts the FlexRay controller immediately.

Among the FlexRay modules, the POC module is the most vulnerable because about 18.8% of 5200 injected faults to it result in the freeze error. Moreover, the fault injection results show that the CSP module tolerates all injected faults and prevent them from halting the controller.

5. Proposed freeze error monitoring mechanism

Detailed studies of the FlexRay protocol show that this protocol has a fail silent property, because if a severe error occurs, the POC module can freeze the whole controller; but fault injection results showed that a single bit-flip fault can result in freezing the controller, whereas any severe error has not occurred. Consequently a monitoring mechanism should be used to investigate that an occurred freeze error is due to a severe error in standard operation of the FlexRay protocol or due to a transient fault.

Experimental analyses also show that if a single bit-flip fault results in a freeze error, other error types may be occurred [11]. Then if the monitor recognizes a freeze error due to a fault, it will inform the host controller about it and this controller will be responsible for recovering the communication controller from these errors.

As mentioned before, the FlexRay protocol described by SDL in the FlexRay Specifications. To design the freeze error monitoring, this property was used and the monitor was designed based on checking the freeze conditions which were shown in Figure 3.

For example, if a freeze error occurs, when the Fatal_protocol_Error_POC and Fatal_protocol_Error_ MAC single-bit registers had been set, the part (D) of Figure 3 is checked by the proposed monitor. To do this, if the MAC module's state is "wait for transmission end" and other conditions which have been shown in the part (D) of Figure 3 have been fulfilled, a flag which was named Freeze_OK is set.

When the Freeze_OK flag is set, it means that the occurred freeze error is due to a sever error in the standard operation of the FlexRay protocol; otherwise, if a freeze error occurs and the monitor, after checking the conditions, does not set the Freeze_OK flag, it means that this freeze error is due to a fault, e.g., a transient bit-flip fault in the Fatal_Protocol_Error _MAC register which is caused to set the Fatal_protocol_Error_POC register.

To check the conditions in Figure 3 by the monitor, it needs to insert some status checking flags in all six FlexRay controller modules. These flags indicate the operational state of these modules. Implementation of this controller showed that only 16 single-bit flags should be used to monitor the freeze error conditions.

Fault injection results showed that the proposed freeze error monitor is able to distinguish about 99.8% all occurred freeze errors into errors due to transient faults or due to sever errors in standard operation of the protocol and to report it to the host controller. Table 2 shows these results. Also this monitor, by checking the status flags and freeze condition registers, is able to locate the fault site in the controller and inform the host controller about it. As shown in this table, the proposed monitor did not distinguish only 6 freeze errors due to injected faults into the CHI module. Investigations show that by using more status checking flags in this module, these freeze errors are also distinguishable by the monitor.

It should be noticed that, using the TMR and the Hamming techniques for protecting freeze condition registers is not a good idea, because these techniques incur more hardware and power consumption overheads. Also, because the delay of inserted gates, it is possible that timing constraints of the communication controller are threatened. Furthermore, analyses show that freeze errors due to multiple-bit faults are also distinguishable by the proposed monitor.

Table 2. Statistics of errors reported by the monitoring mechanism

Modules	Freeze Errors due to Injected Faults	Distinguished Freeze Errors by the Monitor	
	#	#	%
CHI	406	400	98.5
CODEC	40	40	100
FSP	51	51	100
MAC	144	144	100
POC	979	979	100
Total	1620	1616	99.8

By implementing this monitor beside the FlexRay modules, the host controller is able (with probability of 99.8%) to discover that an occurred freeze error is due to a fault or due to a sever error in standard operation of the protocol. In addition, because the proposed monitor is only activated when a freeze error occurs, in the normal and freeze error-free operation of the protocol this monitor is idle and has not any power consumption. In addition to 2712 flip-flops of the FlexRay controller, this monitor adds 16 flip-flops (about 0.6%) to them. Also results of single bit-flip fault injections on the added flip-flops (16 single-bit status checking flags) showed that these flags do not affect the fault tolerance of the FlexRay controller and the proposed monitor. Figure 5 shows the structure of the FlexRay controller with freeze error monitor.

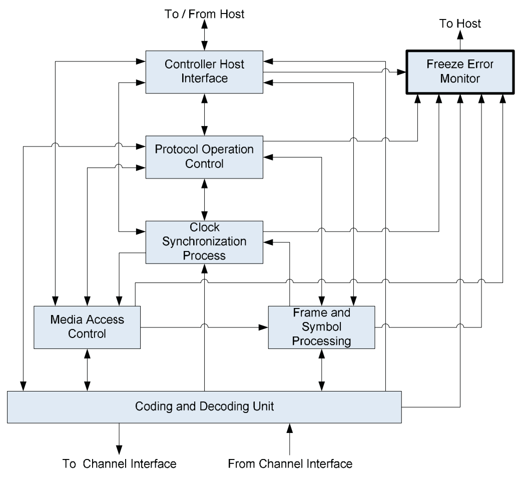


Figure 5. Structure of the FlexRay controller with freeze error monitor

6. Conclusions

Safety-critical automotive control systems are complex distributed embedded systems and the communication protocol is an essential part of them. It is now widely believed that the FlexRay protocol will emerge as the predominant protocol and will become the de-facto standard for these systems. In this paper, single points of failure in the FlexRay communication protocol have been identified by injecting a total of 135,600 single bit-flip transient faults into all accessible registers of the FlexRay communication controller. Results showed that about 1.2% of injected faults caused the controller to halt immediately. Based on detailed studies of the protocol specifications, a low-cost and on-line monitoring mechanism is proposed. This mechanism correctly distinguishes about 99.8% all occurred freeze errors into errors due to transient faults or due to sever errors in standard operation of the FlexRay protocol and reports them to the host controller. The proposed mechanism has low power consumption and has only 0.6% hardware overhead.

7. References

[1] G.N. Sung, C.Y. Juan, and C.C. Wang, "Bus Guardian Design for Automobile Networking ECU Nodes Compliant with FlexRay Standard," *Proc. of the IEEE International Symposium on Consumer Electronics (ISCE'08)*, Algarve, Portugal, 14-16 April 2008, pp. 1-4.

[2] E. Armengaud, A. Steininger, and M. Horauer, "Towards a Systematic Test for Embedded Automotive Communication Systems," *IEEE Trans. on Industrial Informatics, Vol. 4, Issue 3,* August 2008, pp. 146-155.

[3] T. Nolte, H. Hansson, and L. Bello, "Automotive Communications - Past, Current and Future," *Proc. of 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'05), Vol. 1*, Catania, Italy, 19-22 September 2005, pp. 985-992.

[4] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing Analysis of the FlexRay Communication Protocol," *Proc. of the 18th Euromicro Conference Real-Time Systems (ECRTS'06)*, Dresden, Germany, 5-7 July 2006, pp. 203-216.

[5] T. Pop, P. Pop, P. Eles, and Z. Peng, "Bus Access Optimization for FlexRay-based Distributed Embedded Systems," *Proc. of Design, Automation & Test in Europe Conference & Exhibition 2007 (DATE '07)*, Nice, France, 16-20 April 2007, pp. 1-6.

[6] A. Hagiescu, U.D. Bordoloi, and S. Chakraborty, "Performance Analysis of FlexRay-based ECU Networks," *Proc. of 44th ACM/IEEE Design Automation Conference (DAC '07)*, San Diego, USA, 4-8 June 2007, pp. 284-289.

[7] R. Makowitz and C. Temple, "FlexRay- A Communication Network for Automotive Control Systems,"

Proc. of the 6th IEEE International Workshop on Factory Communication Systems (WFCS 2006), Torino, Italy , 28-30 June 2006, pp. 207-212.

[8] J. Voelcker, "Top 10 Tech Cars," *IEEE Spectrum magazine, Vol. 45, No. 4*, April 2008, pp. 27.

[9] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in Automotive Communication Systems", Proc. *of the IEEE, Vol. 93, Issue 6*, June 2005, pp. 1204-1223.

[10] Y. Sedaghat, and S.G. Miremadi, "Investigation and Reduction of Fault Sensitivity in the FlexRay Communication Controller Registers", *Proc. of 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP'08)*, Newcastle upon Tyne, UK, 22-25 September 2008, pp. 153-166.

[11] Y. Sedaghat, and S.G. Miremadi, "Categorizing and Analysis of Activated Faults in the FlexRay Communication Controller Registers", *Proc. of 14th European Test Symposium (ETS'09)*, Seville, Spain, 25-29 May 2009.

[12] J. Rushby, "A Comparison of Bus Architectures for Safety-Critical Embedded Systems", *Technical Report NASA/CR-2003-212161 & EMSOFT*, 2001, pp. 306-323.

[13] H. Salmani and S.G. Miremadi, "Contribution of Controller Area Networks Controllers to Masquerade Failures", *Proc. of 11th Pacific Rim International Symposium on Dependable Computing (PRDC'05)*, Changsha, Hunan, China , 12-14 December 2005, pp. 310-316.

[14] H. Salmani and S.G. Miremadi "Assessment of Message Missing Failures in CAN-based Systems", *Proc. of IASTED International Conference on Parallel and Distributed Computing and Networks*, Austria, 15-17 February 2005, pp.

[15] J. Perez, M.S. Reorda, and M. Violante, "Dependability Analysis of CAN Networks: an Emulation-based Approach", *Proc. of 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03)*, Boston, MA, USA, 3-5 November 2003, pp. 537-544.

[16] H. Sivencrona, P. Johannessen, M. Persson, and J. Torin, "Heavy-ion Fault Injections in the Time-triggered Communication Protocol", *Proc. of 1st Latin American Symposium on Dependable Computing (LADC '03)*, Sao Paulo, Brazil, 21-24 October 2003, pp. 69-80.

[17] S. Blanc and P.J. Gil, "Improving the Multiple Errors Detection Coverage in Distributed Embedded Systems", *Proc. of 22nd International Symposium on Reliable Distributed Systems (SRDS'03)*, Florence, Italy, 6-18 October 2003, pp. 303-312.

[18] A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin, "Evaluation of Fault Handling of the Time-Triggered Architecture with Bus and Star Topology*", Proc. of International Conference on Dependable Systems and Networks (DSN'03)*, San Francisco, CA, USA,22-25 June 2003, pp. 123-132.

[19] H. Kopetz, "Fault Containment and Error Detection in the Time-Triggered Architecture," *Proc. of the 6th International Symposium on Autonomous Decentralized Systems (ISADS'03)*, Pisa, Italy, 9-11 April 2003, pp. 139-146.

[20] I. Broster and A. Burns, "An Analyzable Bus-Guardian for Event-Triggered Communication," *Proc. of the 24th IEEE Real-Time Systems Symposium (RTSS'03)*, Cancun, Mexico, 3-5 December 2003, pp. 410-419.

[21] G. Bauer, H. Kopetz, and W. Steiner, "The Central Guardian Approach to Enforce Fault Isolation in theTime-Triggered Architecture," *Proc. of the 6th International Symposium on Autonomous Decentralized Systems (ISADS'03),* Pisa, Italy, 9-11 April 2003, pp. 37-44.

[22] V. Lari, M. Dehbashi, S.G. Miremadi, and N. Farazmand, "Assessment of Message Missing Failures in FlexRay-Based Networks", *Proc. of 13th Pacific Rim International Symposium on Dependable Computing (PRDC'07)*, Melbourne, Australia,17-19 December 2007, pp. 191-194.

[23] V. Lari, M. Dehbashi, S. G. Miremadi, and M. Amiri, "Evaluation of Babbling Idiot Failures in FlexRay-based Networks", *Proc. of 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems (FET'07)*, Toulouse, France, 7-9 November 2007, pp. 8.

[24] FlexRay Communications System - Protocol Specification V2.1 Revision A, www.flexray.com

[25] K. Tindell, and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems", *Trans. on Microprocessing & Microprogramming, vol. 40, Issue 2-3*, April 1994, pp. 117-134.

[26] G. Cena, and A.Valenzano, "Performance Analysis of Byteflight Networks", *Proc. of 5th IEEE international Workshop Factory Communication Systems (WFCS'04)*, Vienna, Austria, 22-24 September 2004, pp. 157-166.

[27] V. Izosimov, P. Pop, P. Eles, and Z. Peng, "Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems", *Proc. of Design, Automation and Test in Europe Conference and Exhibition 2005 (DATE'05)*, vol. 2, Munich, Germany, 7-11 March 2005, pp. 864-869.

[28] E. Armengaud, F. Rothensteiner, A. Steininger, and M. Horauer, "A Method for Bit Level Test and Diagnosis of Communication Services", *Proc. of 8th International IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS'05)*, Sopron, Hungary , 13-16 April 2005, pp. 69-74.

[29] FlexRay Communications System - Protocol Conformance Test Specification V2.1, www.flexray.com