J Electron Test (2010) 26:535–547
DOI 10.1007/s10836-010-5164-9

# Classification of Activated Faults in the FlexRay-Based Networks

Yasser Sedaghat · Seyed Ghassem Miremadi

Received: 30 September 2009 / Accepted: 14 June 2010 / Published online: 20 July 2010

**Abstract** FlexRay communication protocol is expected to become the de-facto standard for distributed safety-critical systems. This paper classifies the effects of transient single bit-flip fault injections into the FlexRay communication controller. In this protocol, when an injected fault is activated, this may result in one or more error types, i.e.: Boundary violation, Conflict, Content, Freeze, Synchronization, Syntax, and Invalid frame. To study the activated faults, a FlexRay bus network, composed of four nodes, was modeled by Verilog HDL; and a total of 135,600 transient faults was injected in only one node, called the target node. The results show that only 9,342 of the faults (about 6.9%) were activated and their effects were observed in the network nodes. The results also show that the Synchronization error is the widespread error with the occurrence rate of 70.1%. The Invalid frame, Boundary violation, Syntax, Content, Freeze, and Conflict errors have the occurrence rates of 51.1%, 32.4%, 24.6%, 20.6%, 17.3%, and 0.0%, respectively. Among the error types, the Freeze errors are the most critical errors as they mostly result in system failures. The results also show that most of the activated faults, about 75.3%, were observed simultaneously in the target node and a neighbor node. About 11.1% of the activated faults were observed only in the target node and about 13.5% only in the neighbor node.

**Keywords** Safety-critical applications · Distributed embedded systems · FlexRay protocol · Fault injection

Responsible Editor: C. Metra

Y. Sedaghat · S. G. Miremadi (✉)
Dependable Systems Laboratory, Sharif University of Technology, Tehran, Iran
e-mail: miremadi@sharif.edu

Y. Sedaghat
e-mail: y_sedaghat@ce.sharif.edu

## 1 Introduction

With the advent of distributed embedded systems and using them in safety-critical applications such as avionics, automotive, and railway, fault tolerance has become an essential demand for these systems. A distributed embedded system is composed of several different hardware units (called nodes), e.g., processing units, sensors, and actuators, interconnected by a communication network.

It has been reported that the overall reliability of a safety-critical distributed embedded system not only depends on the reliability of its nodes, but also more on the reliability of its communication network [11, 14]. This means that if a node fails, its task can be assigned to another node, however, if a communication network fails, the overall operation cannot be succeeded. Consequently, the communication network is a critical part in the safety-critical distributed embedded systems. A network failure may occur either if a link fails or the communication protocol fails.

Communication in a distributed architecture can be triggered either dynamically, in response to an event (event-driven), or statically, at predetermined moments in time (time-driven). Examples of event-triggered protocols are Byteflight, CAN, LonWorks, and Profibus. The main drawback of event-triggered protocols is their lack of predictability [13]. Examples of time-triggered protocols are SAFEbus, SPIDER, and TTP/C. The main drawback of time-triggered protocols is their lack of flexibility [13]. To resolve the drawbacks of both event-triggered and time-triggered protocols, other protocols such as TTCAN, FTT-CAN, and FlexRay [5] are introduced. 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 applications [7, 13, 14, 19]. As an example, the next edition of the BMW X5 uses the FlexRay protocol in its electronically controlled dampers [19].

Several work have investigated the reliability and fault tolerance of communication protocols. Effects of masquerade failures [15] and message missing failures [16] have been studied for the CAN protocol by the simulation-based fault injection. Also, in [12], effects of simulation-based fault injection in the CAN protocol has been investigated. Fault tolerance of the TTP/C protocol has been assessed, by heavy-ion fault injection [20] and physical pin-level fault-injection [3]. Moreover, fault tolerance of the TTP/C protocol and the failures have been studied using heavy-ion fault injection and simulation-based fault injection [1].

An evaluation of the FlexRay protocol using simulation-based fault injection has been reported in [10] and [9]. However, this evaluation has two main limitations: 1) faults were injected only into 10% of the FlexRay registers, and 2) only three error types, i.e., Boundary violation, Content, and Syntax were considered. In an experiment reported in [17], faults were injected into all 408 registers of the FlexRay to study the fault effects. In [18], the activated faults presented in [17], have been classified into six error types.

This paper extends the work presented in [18]. The extension includes three parts: 1) A new error type called Invalid frame error is added in the study; 2) Fault effects are classified individually in the target node and in the neighbor node; and 3) Error propagation rates from the target node to the neighbor node are investigated. The work is evaluated by simulation-based fault injection using the ModelSim5.5. In the simulation, a FlexRay bus network, composed of four nodes, is modeled by the Verilog HDL and the effects of a total of 135,600 injected faults in the target node are studied. In this study, the behavior of the target node and one of its neighbor nodes is studied.

The remainder of the paper is organized as follows: Section 2 introduces the FlexRay protocol briefly. Error types and error handling mechanisms in this protocol are presented in Section 3. In Section 4, the experimental environment is introduced. Section 5 includes the experimental results and finally, conclusions are given in Section 6.

## 2 The FlexRay Protocol

The FlexRay protocol provides key features of synchronization that include scalable data transmission in both synchronous and asynchronous modes. It can support the 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. In addition, its physical layer provides support for bus, star, and multiple star topologies [5].

From the dependability point of view, the FlexRay documents [5] specify solely bus guardian mechanism and clock synchronization algorithms. Other features, such as a membership service or mode management facilities, 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 [11].

The main purpose of this paper is to instruct the developers of the FlexRay communication controller to know that: 1) which errors are widespread, 2) which error type is the most destructive, and 3) which error types are propagated through the network. Based on fault injection results and error analyses in this paper, it seems that in addition to existing techniques in FlexRay communication controller, other useful techniques should be applied.

### 2.1 Protocol Structure

The FlexRay communication controller consists of six modules [5]: 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 1 illustrates the relation between these modules.

- The CHI module, manages data and control flow between the host processor and the FlexRay protocol engine within each node.
- Operational modes of the FlexRay modules are adjusted by POC module. The purpose of the POC is to react to host commands and protocol conditions by triggering coherent changes to core modules in a synchronous manner, and to provide the host with the appropriate status regarding these changes.
- 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 a recurring communication cycle. Within one communication cycle, the FlexRay offers the choice of two media access schemes, i.e., Time Division Multiple Access (TDMA) [22] scheme for a time-triggered (or static) window, and Flexible TDMA (FTDMA) [4] scheme for an event-triggered (or dynamic) window. The communication cycle is the fundamental element of the media access scheme within the FlexRay. Figure 2

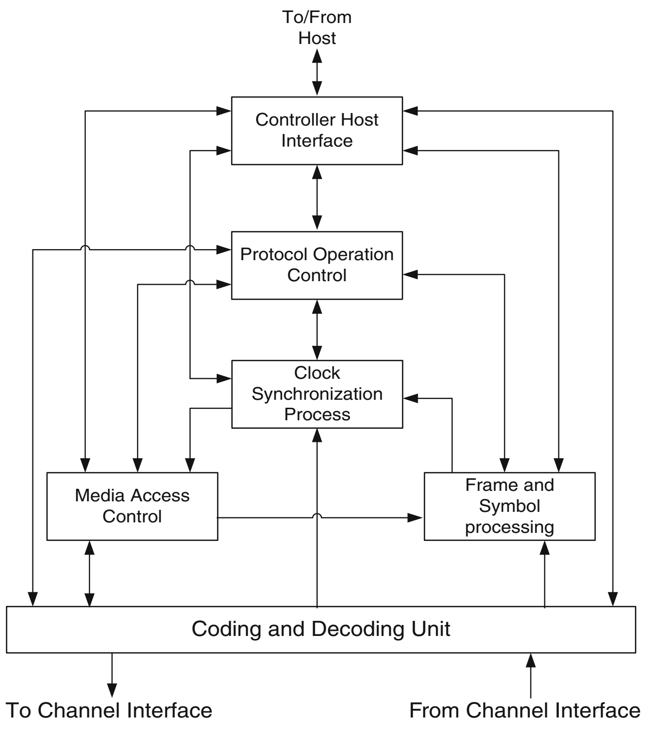

To/From Host
Controller Host Interface
Protocol Operation Control
Clock Synchronization Process
Media Access Control
Frame and Symbol processing
Coding and Decoding Unit
To Channel Interface
From Channel Interface

**Fig. 1** The FlexRay structure [5]

shows an example of a communication cycle in the FlexRay protocol.

- The FSP module is responsible for checking the correct timing of received frames and symbols with respect to the TDMA scheme, applying further syntactical tests to received frames, and checking the semantic correctness of received frames.
- The CSP module is responsible for generation of 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 sync frames from other nodes.

In the FlexRay protocol, frames are sent in static slots or dynamic slots of each communication cycle. Figure 3 shows the frame format in the FlexRay protocol. For more details about the frame format in the FlexRay protocol, readers are referred to FlexRay specifications [5], in Chapter 4.

## 3 Error Types and Error Handling Approaches

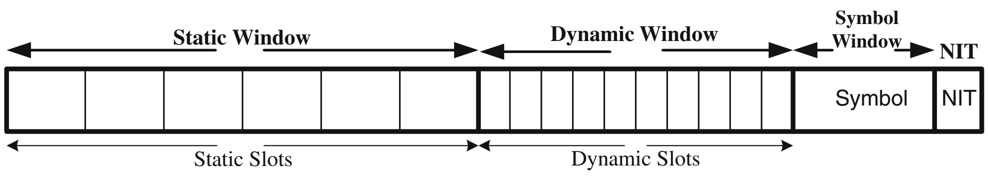
Safety-critical applications have to function correctly even in presence of faults. Faults can be permanent (e.g., damaged microcontrollers or communication links), transient (e.g., caused by single event upsets or electromagnetic interferences), or intermittent (appear and disappear repeatedly). The transient faults are the most common, and their number is dramatically increasing due to the continuously raising level of integration in semiconductors [8]. Transient single bit-flip errors, which are considered in this paper, are more common consequences of the transient faults [2].

### 3.1 Error Types in the FlexRay Protocol

More detailed studies of the FlexRay specification documents [5] showed that in the FlexRay protocol, due to faults, in addition to six introduced error types [18], another error type, which is named Invalid frame, is also possible. Then these seven error types are:

1) **Boundary violation error** denotes whether a boundary violation has occurred at the boundary of the corresponding slot. A boundary violation occurs if the node does not consider the channel to be idle at the boundary of a slot.
2) **Conflict error** denotes whether reception was ongoing at the time the node started a transmission.
3) **Content error** denotes the presence of an error in the content of a received frame.
4) **Synchronization error:** In the FlexRay protocol, a clock rate correction mechanism is used. In synchronization process, if a fault results in failing the clock rate correction mechanism, a Synchronization error is occurred.
5) **Syntax error** denotes the presence of a syntactic error in a time slot; e.g. when a decoding error occurs.
6) **Freeze error:** In addition to mentioned error types, in the FlexRay communication protocol, there are three general conditions that trigger functional state of communication controller to halt state, immediately. In this state, POC stops activities of all other modules and freezes the communication controller. After resolving freeze conditions, communication controller should be started up through the host. Three freeze conditions are: 1) Product-specific error conditions such as Built-In Self-Test (BIST) and sanity check errors, 2) Error

**Fig. 2** Communication cycle in the FlexRay protocol



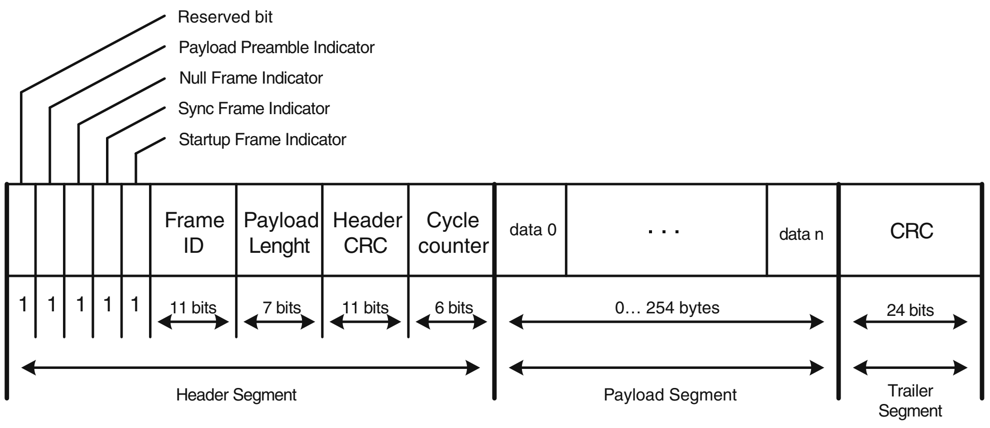


**Fig. 3** Frame format in the FlexRay protocol [5]

conditions detected by the host that result in a FREEZE command being sent to the POC via the CHI, and 3) Fatal conditions detected by the POC or one of communication controller mechanisms.

Product-specific and host detected errors, which are accommodated by the POC and are beyond the scope of the FlexRay specification, were not considered in this paper. Since freezing the communication controller could be result in failing this controller, occurrence of a freeze condition was assumed as an individual error type which was named Freeze error.

7) **Invalid frame error** denotes whether an invalid frame has been received in a slot of static or dynamic segments.

More detailed reinvestigations of the fault injection results showed that due to an injected fault, a received frame which has none of mentioned error types in [18] may be dropped by the communication controller. Hence, the [5] were reviewed again, carefully. Detailed study of the FlexRay specifications showed that in the FSP module, when a syntactically correct frame is received, this frame is checked to meet frame acceptance conditions. If the frame passes this check, it is marked as an accepted frame; otherwise, an Invalid frame error occurs and the received frame is dropped [5]. The frame acceptance conditions are [5]:

1- The frame ID included in the header of the frame equals the slot number of static or dynamic slot which the frame has been received in it.
2- The cycle count included in the header of frame matches the communication cycle which the frame has been received in it.
3- The payload length included in the header of a received static frame matches the globally configured value of the payload length of static frames (this value is determined in the design time of the network).
4- The indicator bits of the received frame (please see Fig. 3) matches with the expected type of received frame.

It should be noticed that, none of the above conditions are checked for detecting the previous error types. Furthermore, the occurrence rate of this error type has a direct effect on the estimation of network bandwidth utilization in the presence of faults. Then, in addition to the previous error types [18], the Invalid frame error type was also considered.

It should be mentioned that, investigations showed that in the FlexRay communication controller, an injected transient single-bit flip fault may be overwritten, or may result in one or more discussed error types, simultaneously. For example, if a fault causes a Freeze error, other error types such as the Syntax error, Content error, Boundary violation error, and even the Invalid frame error can be occurred.

## 3.2 Error Handling Approaches in the FlexRay Communication Controller

To handle the mentioned error types in the FlexRay protocol, only two main approaches are used. 1) Occurrence of the Boundary violation, Conflict, Content, Invalid frame, and Syntax errors is reported to the host via the CHI module and the host responses to these error types, based on its error handling mechanisms. 2) Occurrence of the Synchronization and Freeze errors is reported to the POC module and this module, based on a graceful degradation mechanism [5], reacts to it.

In the FlexRay protocol, to check the operational conditions of the communication controller some registers are employed in the FlexRay modules. In the normal operation of the controller, the behavior of these registers has been determined by the FlexRay specifications [5]. If one of these registers behaves unexpectedly during the operation of the controller, this indicates that an error condition has fulfilled. Then, the related module to that register determines the type of the occurred error and reports it to the CHI or POC modules. In this paper, these registers are referred as “error indicator registers”. Table 1 shows these registers and their related modules of the FlexRay communication controller. This table also shows error types which are determined by these error indicator registers. For example, if a frame is received by the controller and there is an error in it such that the CRC check for this frame is failed, the CODEC module sets the "Frame_crc_error" error indicator register and reports a Syntax error to the CHI module.

## 4 Experimental Environment

The FlexRay communication controller, according to its specifications [5], was implemented by hardware description language, Verilog HDL, and specifications of this controller, e.g. timing and configuration, were tested according to the FlexRay protocol conformance test specification [6].

As for the experiment setup, a cluster was formed consisting of four nodes with single bus topology. In this topology, a node is composed of a host and a communication controller. The host typically is a hardware unit that generates data to exchange with other neighbor 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. Within the cluster, all nodes were scheduled to send/receive frames to/from the communication channel.

The simulation time included five communication cycles. Each communication cycle composes of ten static slots and 20 mini-slots. Two, three, two, and three static slots assigned to Node1, Node2 (the target node), Node3, and Node4 (the observed neighbor node), respectively. Moreover, 20 mini-slots were assigned to four nodes, equally. In the dynamic segment of the each communication cycle, data generators generates dynamic frame randomly for their nodes. For example, it is possible that in a communication cycle only Node 1 transmits a dynamic frame in its mini-slots and the other nodes do not send any thing.

Consequently, to decrease the effects of the used bench mark on the results, faults were injected into the communication controller registers of the target node during two communication cycles, at a random time. Also, to guarantee that presented results have lower dependency on the used bench mark, 50 faults were injected to each bit of all communication registers in these two communication cycles, at a random time.

To investigate the fault tolerance of the FlexRay communication controller, transient single bit-flip faults were injected into all registers of communication controller modules of a node which is named target node and their effects on the error indicator registers were observed in the target node and in a neighbor node which is called an observed neighbor node. It should be noticed that in this investigation, the faults were not injected to the data registers, i.e. input and output data registers, of the communication controller; because these registers have no effects on the operation of the communication controller and if a fault affects these registers, only transmitted or received data is affected.

It should be mentioned that, if effects of an injected fault in the target node is observed on a neighbor node, because of broadcasting data in the FlexRay bus-based network, the same effects are also observable in all other neighbor nodes. Consequently, to investigate the effects of injected faults on the neighbor nodes, it is enough to observe the effects on one of the neighbor nodes.

### 4.1 Error Classification Methodology

The error classification methodology includes two phases. In the first phase, a fault should be injected into a flip-flop (one bit of a register) of communication controller; and if

**Table 1** Error indicator registers in the FlexRay communication controller

| Registers | FlexRay module | Error types | Registers | FlexRay module | Error types |
|---|---|---|---|---|---|
| decoding_error_on_A | CODEC | Syntax | zSyncCalcResult | CSP | Sync* |
| TSS_ok<br>TSS_too_long | | | vPOC_Freeze<br>vPOC_CHIHaltRequest | POC | Freeze |
| FSS_ok | | | vPOC_ErrorMode | | |
| payload_ok | | | Content_error_on_A | FSP | Content |
| trailer_ok | | | Fatal_protocol_error | | Freeze |
| BSS_ok | | | T_StatusSlot_SyntaxError | | Syntax |
| FES_ok | | | T_StatusSlot_ContentError | | Content |
| zBssError | | | T_StatusSlot_TxConflict | | Conflict |
| Header_crc_error | | | T_StatusSlot_BViolation | | Boundary violation |
| Frame_crc_error | | | T_StatusSlot_ValidFrame | | Invalid frame |

* Synchronization

injected fault results in at least one error, in the second phase, type of occurred error(s) should be determined.

For the first phase and injecting the transient single bit-flip faults at the behavioral level in the target node, the SINJECT fault injection tool [23] was used. There are three steps to use of SINJECT tool:

Step1: When the given workload is applied, behaviors of the error indicator registers of the target node and the observed neighbor node in a fault-free network are simulated and stored.

Step2: In the second step, to consider fault effects, the given workload is applied again to the network, a single transient bit-flip fault is injected to one bit of a communication controller register of the target node at a random time, and the behavior of the error indicator registers of the target node and the observed neighbor node are observed.

Step3: In the last step, the faulty network behavior is compared with the behavior of the fault-free network, which is gathered at first step, and if there is a mismatch, that injected fault which causes at least one error, is considered as an activated fault and otherwise, that injected fault is considered as an overwritten fault.

After these steps, if the injected fault was activated, the second phase of error classification methodology is started. In the second phase, based on unexpected changing of error indicator registers, types of occurred errors are determined. To obtain a careful investigation of the fault tolerance of one bit, several faults should be injected to this bit and then these two phases must be repeated severally.

## 5 Experimental Results

To assess effects of activated faults due to injected faults in the FlexRay communication controller, presented in [17] and [18], and to determine widespread and destructive error types in the FlexRay protocol, the nodes were connected through a passive bus network. The main reason of selecting bus topology was preventing some error propagations by a bus guardian unit [21] which sited in star coupler of star topology. This prevention can result in hiding effects of activated faults in communication controller registers of one node on its neighbor nodes.

For simulating the experiments, the ModelSim 5.5 simulation environment was used. In this paper in comparison with [17], 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.

As mentioned in the description of the used methodology for error classification, the behaviors of the error indicator registers of the target and the observed neighbor nodes are observed during the simulations. In the first communication cycle, some of the error indicator registers have an unknown value, i.e., 'x' value, until to the second communication cycle. If a fault is injected when at least one of the error indicator registers has an unknown value, the fault injection results will not be trustworthy. Consequently, in this paper, fault injections were allowed after the first communication cycle.

To obtain a careful investigation of the fault tolerance, 50 transient bit-flip faults were injected to each bit of all FlexRay controller registers. Hence, a total of 135,600 transient single bit-flip faults were injected to all 408 single-bit and multiple-bit registers of the communication controller in the target node. After that in [18], all occurred error types in the network nodes (i.e., the target node and an observed neighbor node), due to injected faults in the target node, were determined. Also In this paper, error types which occurred in the target node and in the observed neighbor node were analyzed, separately. Finally, rates of fault effects which have propagated from the target node to its neighbor nodes (e.g., the observed neighbor node) were calculated.

$$\begin{aligned}&\textit{Occurence rate of one error type for a unit}\\&=\frac{\textit{Number of the error type in the unit}}{\textit{Number of all activated faults in the unit}}\times 100\%\end{aligned}$$

### 5.1 Error Types Statistics in the Overall Network [18]

In this section, various types of all occurred errors in the network which are due to injected faults in the target node have been classified. To reach this goal, occurrence rate of each error type for every bit of communication controller registers of the target node, using Equation 1, was calculated; and then occurrence rate of these error types was achieved for each communication controller modules.

Table 2 shows the total of activated faults and their rates, occurrence number of each error type, and the occurrence rate of them for all FlexRay communication controller modules. It should be mentioned that, in this sub-section, in addition to reported occurrence rates which were reported in [18], the occurrence rate of the Invalid frame error have been reported.

The presented results in the Table 2 show that the Syntax error in the CODEC module and the Invalid frame error in the CHI module are the more frequent error, whereas in the other modules the Synchronization error is the more frequent error. Furthermore, the Conflict error is not occurred due to single bit-flip fault injections. In the POC module which is responsible for controlling the protocol operation, the occurrence rate of the Freeze error is noticeable; based on its destructive property, we convince FlexRay developers to pay more attention to design the POC module.

Figure 4 shows the occurrence rate of the error types for each FlexRay communication controller modules. According to this figure, the Synchronization error in the CSP module is the most probable and the occurrence probability of the Conflict error due to injected faults in the all FlexRay modules is zero.

The overall results in Fig. 5 show that the Synchronization error, the Invalid frame error, and the Boundary violation error are widespread occurred errors; but the Synchronization error is the more occurred than both the Invalid frame and the Boundary violation errors. In the FlexRay protocol, to tolerate the Synchronization errors, for deferring to enter a halt state, at least temporarily, a graceful degradation mechanism is used; Moreover in FlexRay based communication networks, to prevent the Boundary violation error from propagation, bus guardian mechanism which is responsible for watching communication boundaries, is applied; however, it should be mentioned that no technique is applied in order to prevent the Invalid frame, the Syntax, and the Content errors from propagation.

In [18], all occurred error types in the FlexRay network due to injected faults in the target node have been classified, but these results cannot be used for analyzing effects of injected faults on the network nodes, individually. In distributed embedded systems, a fault not only affects the target node (fault site node), but also it affects the other neighbor nodes. Individual investigations of fault effects on the target and observed neighbor nodes are important for system designers; because these results help them to understand which error types may occur in the neighbor nodes and which error types may occur only in the target node. For example, investigations show that the Freeze error type only occurs in the target node. Hence, a Freeze error in a network node does not occur due to a fault in another node. Contrary to the Freeze error, the Synchronization error type may occur in the observed neighbor node due to a fault in the target node. Then, the designers should provide a useful mechanism to protect network nodes against this error type. Consequently, in this paper the work presented in [18] has been extended by investigating the fault effects on the target node and on a neighbor node, individually.

## 5.2 Error Types Statistics in the Target Node

Table 3 shows statistics on all activated faults and all occurred error types in the target node due to fault injections in this node. As seen in this table, about 6.0% of all injected faults to communication controller registers of the target node were activated in this node. As described

**Table 2** Statistics on the all error types in the overall FlexRay network

| Modules | Injected Faults in the target node | All activated faults in the FlexRay network | | Boundary violation errors | | Conflict errors | | Content errors | | Freeze errors | | Synchronization errors | | Syntax errors | | Invalid frame errors | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | # | % | # | % | # | % | # | % | # | % | # | % | # | % | # | % |
| CHI | 32350 | 5628 | 17.4 | 2054 | 36.5 | 0 | 0 | 1290 | 22.9 | 406 | 7.2 | 3606 | 64.1 | 1899 | 33.7 | 4165 | 74.0 |
| CODEC | 32300 | 291 | 0.9 | 40 | 13.7 | 0 | 0 | 45 | 15.5 | 40 | 13.7 | 98 | 33.7 | 135 | 46.4 | 56 | 19.2 |
| CSP | 47850 | 1314 | 2.7 | 570 | 43.4 | 0 | 0 | 333 | 25.3 | 0 | 0 | 1272 | 96.8 | 50 | 3.8 | 222 | 16.9 |
| FSP | 6850 | 164 | 2.3 | 29 | 17.7 | 0 | 0 | 51 | 31.1 | 51 | 31.1 | 76 | 46.3 | 5 | 3.0 | 51 | 31.1 |
| MAC | 11050 | 655 | 5.9 | 240 | 36.6 | 0 | 0 | 109 | 16.6 | 144 | 22.0 | 509 | 77.7 | 121 | 18.5 | 83 | 12.7 |
| POC | 5200 | 1290 | 24.8 | 90 | 7.0 | 0 | 0 | 93 | 7.2 | 979 | 75.9 | 987 | 76.5 | 87 | 6.7 | 235 | 18.2 |
| TOTAL | 135600 | 9342 | 6.9 | 3023 | 32.4 | 0 | 0 | 1921 | 20.6 | 1620 | 17.3 | 6548 | 70.1 | 2297 | 24.6 | 4812 | 51.5 |

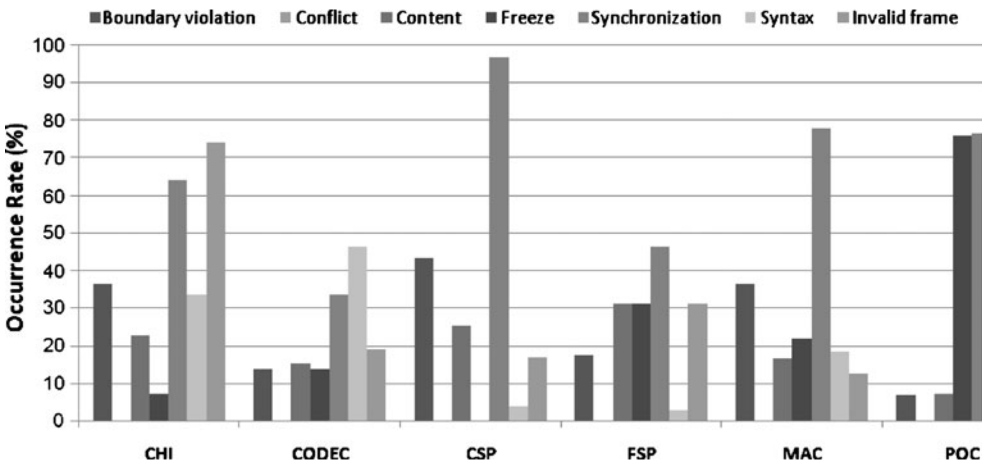


**Fig. 4** All error types in the overall FlexRay network

before, each activated fault resulted in one or more error types.

Figure 6 shows the occurrence rate of the error types due to fault injections in each FlexRay communication controller modules of the target node. In this figure, the Synchronization error in the CSP module, the Invalid frame error in the CHI module, and the Synchronization error in the MAC module are more probable than the other error types; also occurrence rates of Conflict error in all module, the Syntax error in the FSP module, and the Freeze error in the CSP module are zero.

Whereas the occurrence rate of the Syntax error due to fault injections in the FSP module in Table 2 is not zero, this fact shows that all reported Syntax errors for the FSP in Table 2, are related to the occurred Syntax error in the observed neighbor node. It is important to mention that, Table 2, as compared with Table 3, reports a union of occurred error types in both the target node and the observed neighbor node.

Figure 7 shows the overall occurrence rate of all the error types which occurred in the target node due to fault injections in the communication controller of this node. As illustrated in this figure, the Synchronization, the Invalid frame, and the Boundary violation errors are higher occurrence rate than the other error types. This figure also shows that, the Syntax error as compared with Fig. 5 has a noticeable lower rate. This means that injected faults result in more Syntax errors in the neighbor nodes than the target node.

## 5.3 Error Type Statistics in the Observed Neighbor Node

As mentioned before, in the distributed embedded systems, because of using communication networks, an error in a node not only occurs due to a fault in that node, but also an error can occur due to propagated fault effects from an occurred fault in a neighbor node. Also, it should be mentioned that handling the faults as close as possible to the fault origin, i.e. in the target node, does not provide a sufficient protection. The main reasons are: 1) Protecting all fault sensitive registers of the communication controller in the target node, incurs noticeable hardware overhead to the controller. Moreover, the imposed delays due to the hardware overhead cause to threat timing constraints of the communication protocol. 2) Also, it is possible that, before an occurred fault is detected in the target node (because of fault latency), its effects affect the neighbor nodes, simultaneously. Furthermore, investigations showed that some injected faults in the target node had no effect in this node, nevertheless these faults caused to error(s) in the observed neighbor node. Consequently, the designers should be known about the error types which can be occur

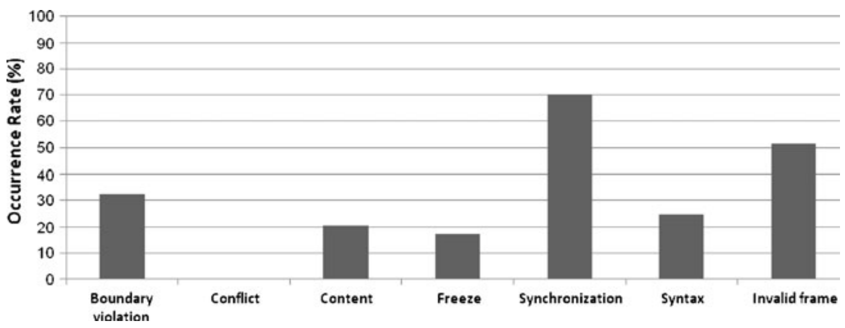


**Fig. 5** Overall error types in the FlexRay network

**Table 3** Statistics on the error types in the target node

| Modules | Injected Faults in the target node | Activated faults in target node | | Boundary violation errors | | Conflict errors | | Content errors | | Freeze errors | | Synchronization errors | | Syntax errors | | Invalid frame errors | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | # | % | # | % | # | % | # | % | # | % | # | % | # | % | # | % |
| CHI | 32350 | 4460 | 13.8 | 1613 | 36.2 | 0 | 0.0 | 1146 | 25.7 | 406 | 9.1 | 3467 | 77.7 | 274 | 6.1 | 3701 | 83.0 |
| CODEC | 32300 | 280 | 0.9 | 38 | 13.6 | 0 | 0.0 | 45 | 16.1 | 40 | 14.3 | 93 | 33.2 | 121 | 43.2 | 56 | 20.0 |
| CSP | 47850 | 1314 | 2.7 | 541 | 41.2 | 0 | 0.0 | 333 | 25.3 | 0 | 0.0 | 1272 | 96.8 | 33 | 2.5 | 23 | 1.8 |
| FSP | 6850 | 164 | 2.4 | 29 | 17.7 | 0 | 0.0 | 51 | 31.1 | 51 | 31.1 | 76 | 46.3 | 0 | 0.0 | 51 | 31.1 |
| MAC | 11050 | 620 | 5.6 | 180 | 29.0 | 0 | 0.0 | 83 | 13.4 | 144 | 23.2 | 508 | 81.9 | 19 | 3.1 | 83 | 13.4 |
| POC | 5200 | 1240 | 23.8 | 88 | 7.1 | 0 | 0.0 | 88 | 7.1 | 979 | 79.0 | 987 | 79.6 | 66 | 5.3 | 235 | 19.0 |
| TOTAL | 135600 | 8078 | 6.0 | 2489 | 30.8 | 0 | 0.0 | 1746 | 21.6 | 1620 | 20.1 | 6403 | 79.3 | 513 | 6.4 | 4149 | 51.4 |

due to propagation of fault effects. In this section, the effects of injected faults in the target node on one of its neighbor nodes, called the observed neighbor node, have been presented.

Statistics on all activated faults and all occurred error types in a neighbor node due to fault injections in the target node have been presented in Table 4. The presented results show that from 135,600 injected faults in the communication controller modules of the target node, effects of 8,301 faults (about 6.1%) caused one or more error types in the observed neighbor node. It should be noticed that an injected fault in the target node may result in error(s) in the target node and also its neighbor nodes, simultaneously.

As mentioned before, Table 2 reports a union of occurred error types in both the target node and the observed neighbor node; whereas, Table 4 reports only error types which have been occurred in the observed neighbor node. For example, among all of reported Syntax errors due to injected faults in the CSP module (50 errors), all of these errors were observed in the observed neighbor node; whereas only 33 Syntax errors were observed in the target node. Also, among all reported Invalid frame errors due to injected faults in the FSP module (51 errors), all of these errors were observed in the target node; whereas only 15 Invalid frame errors were observed in the observed neighbor node.

As seen in this table, the Freeze error has not occurred in the observed neighbor node; in the other hand, transient single-bit faults in one node cannot freeze the other network nodes. Figure 8 shows the occurrence rate of the error types in the observed neighbor node due to injected faults in each module of the FlexRay controller in the target node. As seen in this figure, the Invalid frame error and the Synchronization error due to fault injections in the CHI and the POC modules of the target node are the widespread occurred error types in the observed neighbor node.

Also Fig. 9 shows the overall occurrence rate of all error types which occurred in the observed neighbor node due to injected faults in the communication controller of the target node. As illustrated in this figure, the Synchronization, the Invalid frame, and the Syntax errors are higher occurrence rate than the other error types. Moreover, in the observed neighbor node, occurrence rate of the Conflict and the Freeze errors are zero.

## 6 Discussion

As seen in Figs. 5, 7, and 9, the Synchronization error type is the more frequent error type. Investigation on Tables 2, 3, and 4, shows that injecting faults in the CSP module of the

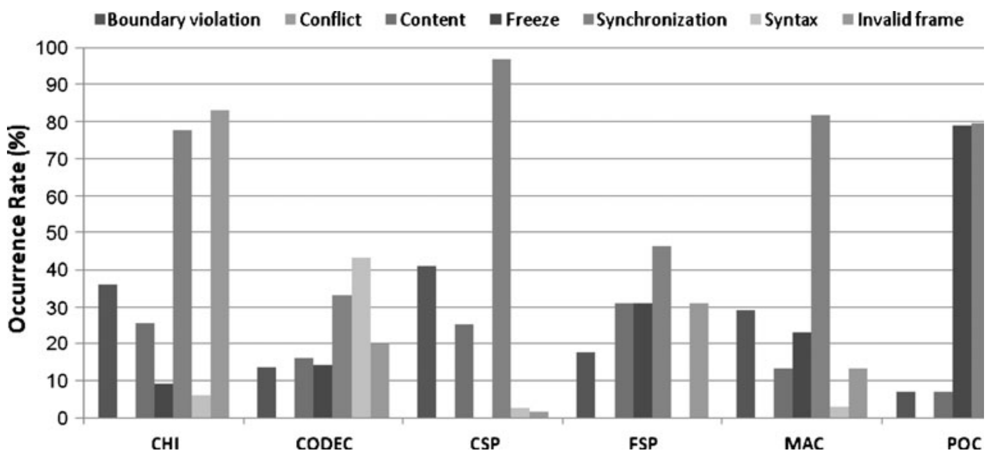


**Fig. 6** All Error types in the target node

target node results in noticeable the Synchronization errors in the target node. This is because the CSP module is responsible for generating timing units in the FlexRay communication controller. In addition, in this module a distributed clock synchronization mechanism is used to synchronize the node with other neighbor nodes. Also, fault injections in the MAC module (which controls node access to the communication channel), and in the POC module (which reacts to the Synchronization errors using a graceful degradation mechanism), of the target node results in noticeable the Synchronization errors in both nodes.

Another widespread error type is the Invalid frame error. Invalid frame errors cause to drop received frames by the communication controller and to reduce the bandwidth utilization. Investigations on the presented results show that injecting faults to the CHI modules of the target node results in noticeable the Invalid frame errors in both the target and the observed neighbor nodes. This is because that the CHI module contains all configuration information and registers, e.g., the frame ID, the payload length of static frames, and other configuration registers.

The Freeze error type is the most critical error type as it mostly results in communication controller failures. The fault injection results show that this error type does not propagate to neighbor nodes. The presented results show that injecting faults to the POC module of the target node result in noticeable freeze errors in the target node. The main reason of this vulnerability is that the POC module is responsible for adjusting the operational modes of the FlexRay modules.

The Syntax error type occurs when a syntactic error in a timing slot is observed. Because the CODEC module encodes the communication elements into a bit stream, receives communication elements, makes bit streams, and investigates correctness of bit streams, injecting faults to this module cause to occur noticeable syntax errors in the both nodes.

Also, the Boundary violation error type occurs when a node does not consider the channel to be idle at the boundary of a slot. Occurrence of this error type is more probable when the faults are injected to the CSP, MAC, and CHI modules. This is because that the CSP module generates timing units and determines boundaries of each slot, the MAC module controls the access to the bus (includes the access time), and the CHI module contains all configuration information, specially timing information.

Finally, occurrence of the Content error type is more probable in the target node when the faults are injected to

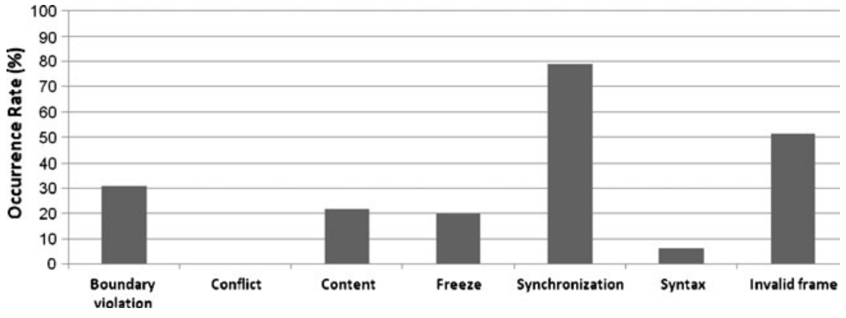


**Fig. 7** Overall error types in the target node

**Table 4** Statistics on error types in the observed neighbor node

| Modules | Injected Faults in the target node | Activated faults in the observed neighbor node | | Boundary violation errors | | Conflict errors | | Content errors | | Freeze errors | | Synchronization errors | | Syntax errors | | Invalid frame errors | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | # | % | # | % | # | % | # | % | # | % | # | % | # | % | # | % |
| CHI | 32350 | 5113 | 15.8 | 972 | 19.0 | 0 | 0 | 180 | 3.5 | 0 | 0 | 2750 | 53.8 | 1696 | 33.2 | 4165 | 81.5 |
| CODEC | 32300 | 129 | 0.4 | 6 | 4.7 | 0 | 0 | 0 | 0.0 | 0 | 0 | 58 | 45.0 | 55 | 42.6 | 20 | 15.5 |
| CSP | 47850 | 1191 | 2.5 | 554 | 46.5 | 0 | 0 | 308 | 25.9 | 0 | 0 | 548 | 46.0 | 50 | 4.2 | 222 | 18.6 |
| FSP | 6850 | 55 | 0.8 | 10 | 18.2 | 0 | 0 | 10 | 18.2 | 0 | 0 | 19 | 34.5 | 5 | 9.1 | 15 | 27.3 |
| MAC | 11050 | 580 | 5.2 | 91 | 15.7 | 0 | 0 | 40 | 6.9 | 0 | 0 | 350 | 60.3 | 119 | 20.5 | 40 | 6.9 |
| POC | 5200 | 1233 | 23.7 | 85 | 6.9 | 0 | 0 | 87 | 7.1 | 0 | 0 | 980 | 79.5 | 80 | 6.5 | 26 | 2.1 |
| TOTAL | 135600 | 8301 | 6.1 | 1718 | 20.7 | 0 | 0 | 625 | 7.5 | 0 | 0 | 4705 | 56.7 | 2005 | 24.2 | 4488 | 54.1 |

the FSP, CHI, and CSP modules; whereas, occurrence of this error type is more probable in the observed neighbor node when the faults are injected to the CSP and FSP modules of the target node. This is because that the CHI module contains all configuration information and registers, the FSP modules is responsible for checking the semantic correctness of received frames, and the CSP module generates timing units in the FlexRay controller.

As mentioned before, in the FlexRay protocol, to tolerate the Synchronization errors, a graceful degradation mechanism is used. Also, to prevent the Boundary violation error from propagation, bus guardian mechanism is applied; however, it should be mentioned that no technique is applied in order to tolerate the other error types. The above analyses in this discussion instruct the designer to reduce occurrence rate of a specific error type by specifying FlexRay modules which should be protected. Moreover, these analyses show that the designer should pay more attention to fault-tolerant design of the CHI and the CSP modules.

## 6.1 Error Propagation Rates

The presented results in Tables 2, 3, and 4 showed that about 6.1% of all injected faults in the modules of the target node caused at least one error type in the observed neighbor node, whereas only 6.0% of all injected faults caused at least one error type in the target node. This means that fault injections in a node of the bus-based FlexRay network, affect all its neighbor nodes more than the target node.

Investigations showed that among all 9,342 activated faults, 7,037 activated faults (about 75.3%) caused at least one error type in both the target node and the observed

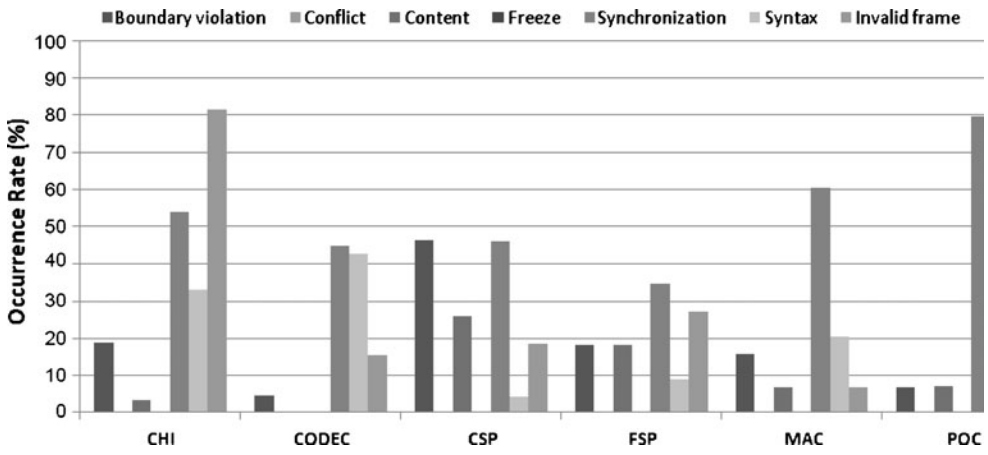


**Fig. 8** Error types in the observed neighbor node

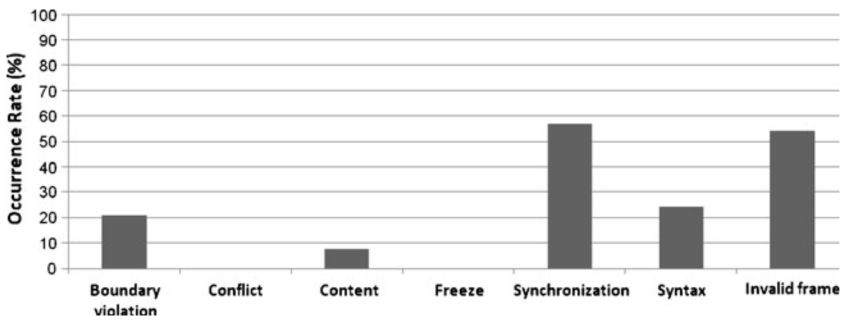


**Fig. 9** Overall error types in the observed neighbor node

neighbor node, simultaneously. Moreover, 1041 of all activated faults (11.1%) were observed only in the target node and 1264 of all activated faults (13.5%) were observed only in the observed neighbor node.

Table 5 shows fault activation rates in the target and in the observed neighbor node as compared with the rates of all activated faults in the network. As seen in this table, in the modules of the target node, only injected faults in the CHI module have more effects on the observed neighbor node and injected faults in the other modules more affect the target node.

Moreover, Table 5 also shows that more than 90% of activated faults due to injected faults in the POC, CHI, and the CSP modules of the target node, propagated their effects to the observed neighbor node. Investigations showed that these ultra high rates are because of more important roles of these modules in the FlexRay protocol. The POC module adjusts operational modes of the communication controller of a node, the CHI module includes all registers which configure all network parameters, and the CSP synchronizes its node to other network nodes.

## 7 Conclusion

To design a fault-tolerant system, in addition to fault models, probable error types must be known. In this paper, widespread and destructive error types in the FlexRay communication protocol, based on 135,600 injected transient bit-flip faults, have been introduced and their occurrence rates in two network nodes, i.e., the target node and a neighbor node, have been determined. Investigations of the activated faults in the FlexRay network showed that the Synchronization error has the occurrence rate of 70.1%. After that, the Invalid frame, Boundary violation, Syntax, Content, Freeze, and Conflict errors have the occurrence rates of 51.1%, 32.4%, 24.6%, 20.6%, 17.3%, and 0.0%, respectively. Among the error types, the Freeze errors are the most critical errors as they mostly result in system failures. The results also showed that most of the activated faults, about 75.3%, were observed simultaneously in the target node and a neighbor node. About 11.1% of the activated faults were observed only in the target node and about 13.5% only in the neighbor node.

**Table 5** Fault activation rates in the target and in the observed neighbor node as compared with all activated faults

| Modules | All activated faults | Activated faults in target node | | Activated faults in the observed neighbor node | |
|---|---|---|---|---|---|
| | | # | % | # | % |
| CHI | 5628 | 4460 | 79.2 | 5113 | 90.8 |
| CODEC | 291 | 280 | 96.2 | 129 | 44.3 |
| CSP | 1314 | 1314 | 100 | 1191 | 90.6 |
| FSP | 164 | 164 | 100 | 55 | 33.5 |
| MAC | 655 | 620 | 94.6 | 580 | 88.5 |
| POC | 1290 | 1240 | 96.1 | 1233 | 95.6 |
| Total | 9342 | 8078 | 86.5 | 8301 | 88.9 |

## References

1. Ademaj A, Sivencrona H, Bauer G, Torin J (2003) 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, June 22–25, pp 123–132
2. Armengaud E, Rothensteiner F, Steininger A, Horauer M (2005) 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, April 13–16, pp 69–74
3. Blanc S, Gil PJ (2003) Improving the multiple errors detection coverage in distributed embedded systems. Proc of 22nd International Symposium on Reliable Distributed Systems (SRDS'03), Florence, Italy, October 6–18, pp 303–312
4. Cena G, Valenzano A (2004) Performance analysis of byteflight networks. Proc of 5th IEEE international Workshop Factory Communication Systems (WFCS'04), Vienna, Austria, September 22–24, pp 157–166
5. FlexRay Communications System - Protocol Specification V2.1 Revision A, www.flexray.com
6. FlexRay Communications System - Protocol Conformance Test Specification V2.1, www.flexray.com
7. Hagiescu A, Bordoloi UD, Chakraborty S (2007) Performance analysis of FlexRay-based ECU networks. Proc of 44th ACM/ IEEE Design Automation Conference (DAC '07), San Diego, CA, USA, June 4–8, pp 284–289
8. Izosimov V, Pop P, Eles P, Peng Z (2005) 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, March 7–11, pp 864–869
9. Lari V, Dehbashi M, Miremadi SG, Amiri M (2007) 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, November 7–9, pp 8
10. Lari V, Dehbashi M, Miremadi SG, Farazmand N (2007) Assessment of message missing failures in FlexRay-based networks. Proc of 13th Pacific Rim International Symposium on Dependable Computing (PRDC'07), Melbourne, Australia, December 17–19, pp 191–194
11. Navet N, Song Y, Simonot-Lion F, Wilwert C (2005) Trends in automotive communication systems. Proc IEEE 93(6):1204–1223
12. Perez J, Reorda MS, Violante M (2003) 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, November 3–5, pp 537–544
13. Pop T, Pop P, Eles P, Peng Z (2007) Bus access optimization for FlexRay-based distributed embedded systems. Proc of Design, Automation & Test in Europe Conference & Exhibition 2007 (DATE '07), Nice, France, April 16–20, pp 1–6
14. Pop T, Pop P, Eles P, Peng Z, Andrei A (2006) Timing analysis of the FlexRay communication protocol. Proc of 18th Euromicro Conference Real-Time Systems (ECRTS'06), Dresden, Germany, July 5–7, pp 203–216
15. Salmani H, Miremadi SG (2005) Contribution of controller area networks controllers to masquerade failures. Proc. of 11th Pacific Rim International Symposium on Dependable Computing (PRDC'05), Changsha, Hunan, China, December 12–14, pp 310–316
16. Salmani H, Miremadi SG (2005) Assessment of message missing failures in CAN-based systems. Proc of IASTED International Conference on Parallel and Distributed Computing and Networks, Austria, February 15–17, pp 387–392
17. Sedaghat Y, Miremadi SG (2008) 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, September 22–25, pp 153–166
18. Sedaghat Y, Miremadi SG (2009) Categorizing and analysis of activated faults in the FlexRay communication controller registers. Proc of 14th European Test Symposium (ETS'09), Seville, Spain, May 25–29, pp 121–126
19. Sethna F, Stipidis E, Ali FH (2006) What lessons can controller area networks learn from FlexRay. Proc of 2nd Vehicle Power and Propulsion Conference (VPPC '06), Windsor, UK, September 6–8, pp 1–4
20. Sivencrona H, Johannessen P, Persson M, Torin J (2003) Heavy-ion fault injections in the time-triggered communication protocol. Proc. of 1st Latin American Symposium on Dependable Computing (LADC '03), Sao Paulo, Brazil, October 21–24, pp 69–80
21. Temple C (1998) Avoiding the babbling-idiot failure in a time-triggered communication system. Proc of 28th Annual International Symposium on Fault-Tolerant Computing (FTCS'98), Munich, Germany, June 23–25, pp 218–227
22. Tindell K, Clark J (1994) Holistic schedulability analysis for distributed hard real-time systems. Trans on Microproc Microprog 40(2–3):117–134
23. Zarandi HR, Miremadi SG, Ejlali A (2003) Dependability analysis using a fault injection tool based on synthesizability of HDL models. Proc of 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems(DFT'03), Boston, MA, USA, November 3–5, pp 485–492

**Yasser Sedaghat** received his B.S. in Computer Engineering from Ferdowsi University of Mashhad, and his M.Sc. in computer engineering from Sharif University of Technology in 2004 and 2006, respectively. He is currently a PhD. student at department of Computer Engineering, Sharif University of Technology. His research interests include Dependable Embedded Systems, Distributed Embedded Systems, Embedded Communication Protocols, and FlexRay Protocol.

**Seyed Ghassem Miremadi** is a Professor of Computer Engineering at Sharif University of Technology. As fault-tolerant computing is his specialty, he initiated the "Dependable Systems Laboratory" at Sharif University in 1996 and has chaired the Laboratory since then. The research laboratory has participated in several research projects which have led to several scientific articles, conference papers and technical reports. Dr. Miremadi and his group have done research in Physical, Simulation-Based and Software-Implemented Fault Injection, Dependability Evaluation Using HDL Models, Fault-Tolerant Embedded Systems, Fault-Tolerant NoCs, and Fault Tree Analysis. He was the Education Director (1997-1998), the Head (1998-2002), and the Research Director (2002-2006) of Computer Engineering Department at Sharif University. Dr. Miremadi is currently the Director of the Hardware Group at the Computer Engineering Department. He is the Editor of the Scientia Journal on Computer Science and Engineering. He served as the general chair of the 13th Int'l CSI Computer Conference (CSICC 2008). Dr. Miremadi got his MSc in Applied Physics and Electrical Engineering from Linköping Institute of Technology and his Ph.D. in Computer Engineering from Chalmers University of Technology, Sweden, in 1984 and 1995, respectively. He is a senior member of the IEEE Computer Society, IEEE Reliability Society.