Microelectronics Journal 42 (2011) 863–873

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

An FSM-based monitoring technique to differentiate between follow-up and original errors in safety-critical distributed embedded systems

Yasser Sedaghat, Seyed Ghassem Miremadi*

Dependable Systems Laboratory, Sharif University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 19 September 2010
Received in revised form
15 March 2011
Accepted 5 April 2011
Available online 6 May 2011

Keywords:
Distributed embedded systems
Error propagation
Follow-up errors
FlexRay protocol
Transient faults
FSM-based monitoring

ABSTRACT

Nowadays, distributed embedded systems are employed in many safety-critical applications such as X-by-Wire. These systems are composed of several nodes interconnected by a network. Studies show that a transient fault in the communication controller of a network node can lead to errors in the fault site node (called original errors) and/or in the neighbor nodes (called follow-up errors). The communication controller of a network node can be halted due to an error, which may be a follow-up error. In this situation, a follow-up error leads to halt the correct operation of a fault-free controller while the fault site node, i.e. the faulty controller, still continues its operation. In this paper, an analysis shows that the occurrence probability of follow-up errors in communication protocols is noticeable. Consequently, it is important to provide a technique to recognize the error's nature, i.e. original or follow-up in each node. This paper proposes a novel low-cost monitoring technique to differentiate follow-up errors from original errors. The proposed technique is based on monitoring the operational states of a communication controller. In this paper, this technique has been applied to the FlexRay protocol. However, it is applicable for all communication protocols having an FSM-based description such as FlexRay, TTP/C, and TT-Ethernet. To evaluate the monitoring technique, a FlexRay-based network including 4 nodes was designed and implemented. The low-cost monitoring technique was as well implemented inside each node of the network. A total of 135,600 transient bit-flip faults were injected in the communication controller of one node. The results showed that about 6.0% of injected faults lead to original errors. This figure for follow-up errors was about 6.1%. The results as well showed that the accuracy of the proposed technique to differentiate between the follow-up and original errors is about 97% at merely 1.4% hardware overhead. This level of accuracy and cost makes the proposed technique a feasible solution to enhance the reliability of communication controllers.

1. Introduction

Since the last two decades, there has been an increased use of electronic components in automotive applications. The main motivations behind this include lower cost, reduced weight, new and innovative functionalities and the need for faster design cycles [1]. Modern automotive applications are nowadays implemented by complex distributed embedded systems. These systems 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 to transfer data between the ECUs. Examples of distributed automotive applications are: chassis, air bag, powertrain, body and comfort electronics, diagnostics, X-by-Wire, multimedia and infotainment, and wireless and telematics [2]. Among these applications, chassis, air bag, X-by-Wire, and powertrain are safety-critical [2], demanding a failure rate of lower than one failure per 10^9 h of operation (1 FIT) [3]. To reach such a low failure rate, the system should be distributed into reliable subsystems and reliable communication protocols as well should be employed [4].

For safety-critical automotive distributed systems, several communication protocols such as Byteflight, TTP/C, TT-CAN, and FlexRay have been introduced [2]. Among these protocols, the FlexRay protocol is advancing as a predominant protocol and is expected to become a de-facto industry standard for X-by-Wire and safety-critical automotive applications [1,5–8]. The BMW X5 is the first vehicle, which employed the FlexRay protocol to enable a new and fast adaptive damping system at the end of 2006. A full use of the FlexRay was introduced in 2008 in the new BMW 7 series, the world's first production vehicle. Moreover, in

* Corresponding author.
E-mail addresses: y_sedaghat@ce.sharif.edu (Y. Sedaghat), miremadi@sharif.edu (S.G. Miremadi).

0026-2692/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2011.04.003

the BMW X6 Sports Activity Coupe, the FlexRay protocol has been employed in steering, braking and suspension systems [9]. Driver assistance systems in Audi A8 car as also well been networked utilizing the FlexRay protocol.

There are a lot of studies addressing the reliability assessment and enhancement of communication protocols. Fault tolerance of the TTP/C protocol has been investigated by heavy-ion fault injection [10] and physical pin-level fault-injection [11]. Fault tolerance and the occurred failures in the TTP/C protocol have been studied using a heavy-ion fault injection and a simulation-based fault injection [12]. Fault-containment and error detection mechanisms of the TTP/C and the FlexRay protocols have been investigated [4]; in addition, this paper has introduced and analyzed several critical failures in these two protocols. The removal of babbling idiot failures by designing a bus guardian technique in the TTP/C and the FlexRay-based communication networks have been studied in Refs. [13] and [14]. An evaluation of the FlexRay protocol utilizing a simulation-based fault injection has been reported in Refs. [15] and [16]. In these studies, faults were injected merely into about 10% of the FlexRay controller's registers. In Ref. [17], an exhaustive evaluation of the FlexRay protocol utilizing a simulation-based fault injection has been reported, and the occurred errors have been classified in Refs. [18] and [19]. Single points of failure in the FlexRay protocol have been identified and resolved in Ref. [20]. The startup algorithm of the FlexRay protocol has been analyzed and the vulnerability of this protocol in the presence of failures during the startup has been investigated [21]. Milbredt et al. in Ref. [22] have investigated the clique problem in the FlexRay protocol. An approach to the implementation of an application-level acknowledgment and retransmission scheme for the FlexRay protocol has been proposed in Ref. [23]. A membership service with a low runtime overhead has been proposed for FlexRay-based communication networks [24].

Two main points in the safety-critical distributed embedded systems are: (1) each network node is commonly designed as a fail-silent node, and (2) a transient fault in a communication controller of a network node can cause errors in the fault site node (called original errors) and/or in the neighbor nodes (called follow-up errors) [25,26]. Consequently, follow-up errors can lead a neighbor node to enter an unwanted fail-safe mode while in the fault site node no original errors may occur and that node can continue its operation. One effective way to prevent the neighbor node from entering an unwanted fail-safe operation is to provide this node with a technique to differentiate between the error's natures, i.e. original, and follow-up, and to inform the controller of the nature. It should be noted that this issue has not been addressed in the literatures, while this differentiation is the most important step for the containment of error propagations.

This paper proposes a novel low-cost monitoring technique to differentiate follow-up errors from original errors. The proposed technique is based on monitoring the operational states of a communication controller. In this paper, this technique has been applied to the FlexRay protocol. However, it is applicable for all communication protocols having an FSM-based description such as FlexRay [27], TTP/C [28], and TT-Ethernet [29]. To evaluate the technique, a FlexRay-based network including 4 nodes has been designed and the technique (a differentiating monitor) has been implemented inside each node. The simulation-based fault injection is employed to generate original and follow-up errors in the network and accuracy of the differentiating monitor is assessed. Furthermore, an analysis has been performed, which shows the noticeable occurrence of follow-up errors. In this analysis, TDMA-based communication protocols have been analyzed and the occurrence probability of the follow-up errors has been determined.

The remaining of the paper is organized as follows. The FlexRay protocol is briefly introduced and its FSM-based behaviors are described in Section 2. In Section 3, the proposed FSM-based monitoring technique is presented. Furthermore, the importance of follow-up errors in TDMA-based communication protocols is analyzed in Section 4. Section 5 includes the experimental setup and evaluation results of the proposed technique, and finally the conclusions are given in Section 6.

2. Review of the FlexRay protocol

In 2000, the FlexRay protocol had been developed by an industry consortium with four founding members (BMW, Daimler-Chrysler, Philips, and Freescale) [8]. This consortium now consists of several automotive industry key players including Bosch, General Motors, and Volkswagen [30]. Today, the FlexRay protocol is expected to become the de-facto industry standard in future automotive systems [2]. The FlexRay protocol provides flexibility, robustness, scalability, and high data rate up to 10 Mbps to be employed in automotive applications. This protocol supports different network topologies, i.e. passive bus, passive or active star, cascaded star, and hybrid topologies. The FlexRay as well supports dual channel communications, which can be employed in fault-tolerant systems [27]. From the dependability point of view, in the FlexRay protocol specifications, a bus guardian mechanism, a fault-tolerant clock synchronization algorithm, and Cyclic Redundancy Codes (CRC) are described; however, other complicated mechanisms, such as a membership service or a clique avoidance mechanism should be implemented in software or hardware layers on top of the FlexRay protocol. This will allow the designers to conceive and implement exactly the services, which are required for the drawback that can be corrected and efficient implementations may be more difficult to achieve in a layer above the communication protocol [27].

2.1. Media access control in the FlexRay protocol

Communications in the FlexRay protocol are based on sending messages in recurring communication cycles. 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 employs a Time Division Multiple Access (TDMA) [31] mechanism. In the event-triggered window of the communication cycle, the arbitration mechanism is Flexible TDMA (FTDMA) [32]. 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. It should be noted that in the FlexRay protocol, data frames are sent in the static slots or in the dynamic slots of each communication cycle.

2.2. Architecture of the FlexRay communication controller

The FlexRay communication controller consists of six functional modules [27]: (1) a controller host interface (CHI) module that manages all data and a control flow between the host controller and the FlexRay communication controller within each node, (2) a protocol operation control (POC) module adjusts operational modes of the FlexRay modules, (3) a coding and decoding (CODEC) module that is responsible to encode the communication elements into a bit stream and to receive communication elements, making bit streams and investigating correctness of bit streams, (4) a media access control (MAC) module controls the access to the bus, in the FlexRay protocol, the media access control is based on recurring communication cycles,

(5) a frame and symbol processing (FSP) module that is responsible to check the correct timing of received frames and a received symbol, applying further syntactical tests to the received frames, and checking the semantic correctness of received frames, and (6) clock synchronization process (CSP) module that is responsible to generate timing units in the FlexRay communication controller, e.g., communication cycles. Moreover, this module employs a distributed clock synchronization mechanism in which each node individually synchronizes itself to its cluster by observing the timing of transmitted frames from the other nodes. Fig. 1 shows the relationship between the FlexRay controller modules.

2.3. FlexRay protocol: an FSM-based protocol

The FlexRay consortium has released a version (Version 2.1, revision A) of the FlexRay communication protocol specification [27], which is available to the general public. In this specification, all of the FlexRay mechanisms have been described utilizing Finite State Machines (FSMs) and have been presented utilizing a graphical method loosely based on the Specification and Description Language (SDL) [33] (SDL diagrams). For example, Fig. 2 shows power operation states of the FlexRay communication controller utilizing an SDL-based description.

As shown in this figure, the communication controller has three power operation states. When the power supply of the controller is turned on and its voltage level reaches a required level, the power operation state of the controller changes from the "power off" state to the "reset" state. The power state changes from the "reset" state to the "POC operational" state when the voltage level is sustained for predetermined time duration and a hardware reset is not asserted. Moreover, the POC operational state itself consists of eight functional states. Fig. 3 shows an overview of the relationship between the eight sub-states of the "POC operational" state.

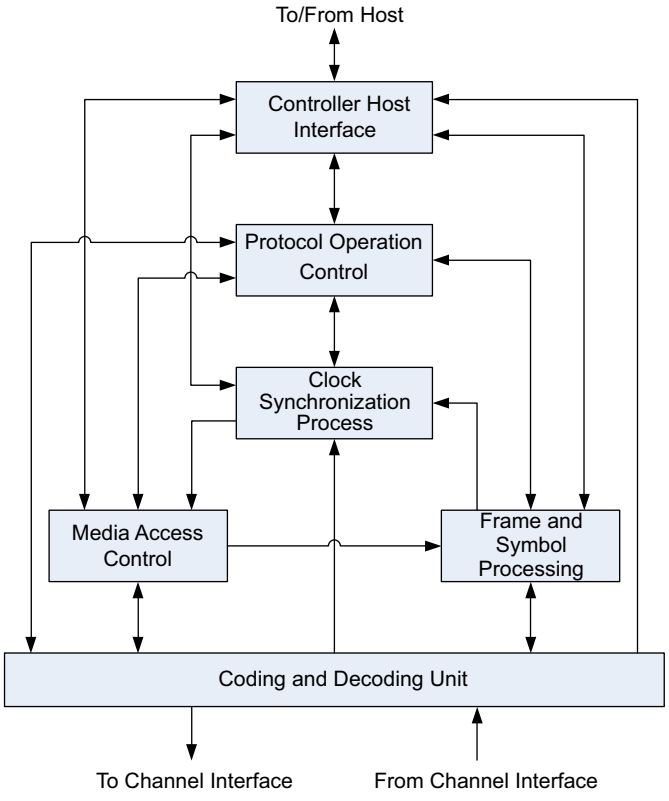

Fig. 1. Structure of the FlexRay communication controller [27].

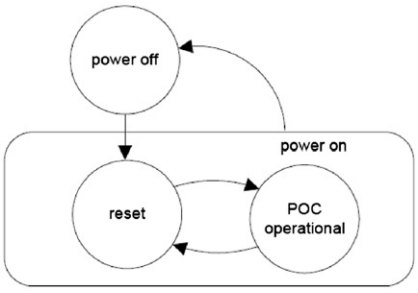

Fig. 2. Power operation states of the FlexRay communication controller [27].

As shown in Fig. 3, the POC module enters the "default config" state when the communication controller enters the "POC operational" power state or the controller is recovered from its "halt" state by a DEFAULT CONFIG command issued from the host controller. In the "default config" state, the POC awaits the explicit command from the host (CONFIG command) to enable the configuration of the controller and afterwards the POC enters the "config" state. It should be noted that an entrance to the "config" state is merely occurred when the CONFIG command is issued by the host and the POC's state is "default config" or "ready". In the "config" state, the host configures the communication controller through the CHI module and afterwards verifies this configuration. When a proper configuration is verified, a CONFIG COMPLETE command is issued by the host to announce the successful completion of the controller's configuration. Next, the POC transits to the "ready" state. On this transition, the POC creates all of the core mechanism processes, incorporating the configuration values that were set in the "config" state.

In the "ready" state, the communication controller is ready to perform the necessary tasks to start or join an actively communicating cluster. In this state, three commands, i.e. CONFIG, WAKEUP, and RUN commands, which are issued by the host controller leads the communication controller to perform several related tasks to the issued command and then the POC operational state is changed to a proper state. Fig. 4 shows all tasks, which and how should be performed in the "ready" state before transiting to a next state, if one of these commands is received from the host.

The CONFIG command causes the host to re-enter the "config" state to allow the host to alter the current controller configuration. Since the core mechanism processes are created on the transition back to "ready" following the configuration process, the processes shall be terminated on the transition to "config". This operation has been shown in Fig. 4 by the TERMINATE_ALL_PROCESSES macro invocation.

The WAKEUP command causes the POC to commence the wakeup procedure in accordance with the configuration loaded into the controller when it was previously configured. This procedure is shown in Fig. 4 by the WAKEUP macro invocation. On completion of the wakeup procedure, the POC activates all the core mechanisms appropriately for "ready" state and returns to this state.

The RUN command causes the POC to commence a sequence of tasks that bring the POC to a normal operation, i.e. the "normal active" state. First, all internal status variables are reset to their starting values. Next, the startup procedure is executed. In Fig. 4 this is represented by the STARTUP macro invocation. This procedure activates the core mechanisms appropriately to perform the sequence of tasks necessary for the communication controller (and its node) to start or enter an actively communicating

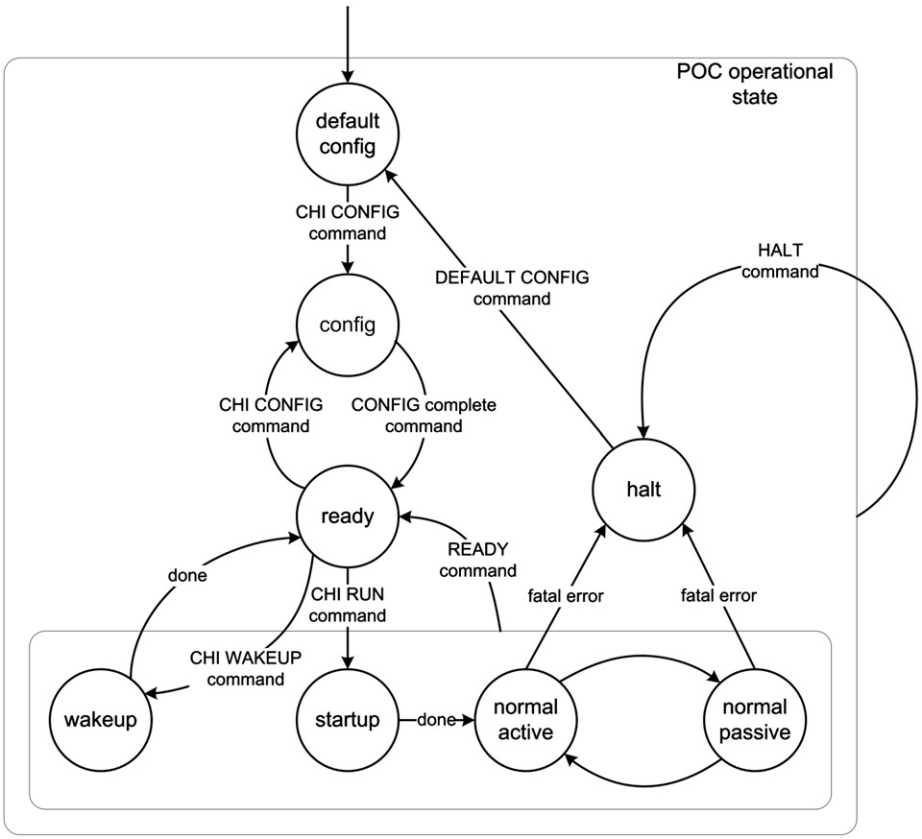

Fig. 3. An overview of the POC operational functional states [27].

cluster. The startup procedure results in the node being synchronized to the timing of the cluster. Following a successful startup, the POC will reside in the "normal active" state. In this state, the communication controller is able to communicate with the other communication controllers, which have been placed in the other network nodes.

Similar to the presented figures, all of the other FlexRay mechanisms have been described using FSMs and have been presented using SDL technique in the FlexRay protocol specifications. For more details, readers are referred to Ref. [27]. In addition to the FlexRay protocol, other communication protocols for safety-critical distributed embedded systems, e.g., TTP/C [28], and TT-Ethernet [29], have been as well described and presented using FSMs and the SDL technique.

As described in this section, FSMs and SDL diagrams present the behavior of a communication protocol and its communication controller in all possible situations. Consequently, if the behavior of a communication controller is monitored during its operation in a communication network, all behavioral errors will be detectable. In this paper, an FSM-based monitoring technique, which is based on monitoring the behavior of the FlexRay communication controller, has been presented. This technique discovers the nature of an occurred error, i.e. original or follow-up in the communication controller and informs the host controller.

3. The proposed FSM-based monitoring technique

In safety-critical distributed embedded systems, fail-silent communication protocols are most frequently employed. In a fail-silent protocol, communication controller of this protocol is a self-checking controller that either functions correctly or stops functioning when a critical error is detected. The FlexRay protocol has been as well designed as a fail-silent protocol. In the FlexRay controller, if a severe error occurs or if various errors persist, the communication controller will be halted [27]. More detailed studies of the FlexRay documents [27] and beforehand experimental results [18,19,34], show that the propagation of an erroneous message can result in follow-up error(s) in the communication controller of the receiving nodes. In this situation, the communication controller of a receiving node cannot discover the nature of errors and thus cannot differentiate between follow-up errors originated from a faulty sending node, and original errors originated from a fault in the communication controller. Consequently, it is possible that a communication controller is halted by the host due to frequently occurred errors, which can be follow-up errors. On the other hand, a perfect communication controller can be halted due to follow-up errors, whereas a faulty communication controller (the origin of follow-up errors) continues to send its messages.

In the FlexRay-based networks similar to the other networks, e.g., TTP/C-based networks, an error in a received message due to a fault in the sending communication controller can appear in two models [4,35]: (1) an error in the timing of the message or (2) an error in the value of the message. A timing error in a received message implies that the "instant of sending the message" or the "instant of receiving the message" are not in agreement with the system specifications, which are determined in the design time. A value error in a received message implies that there is an error in the structure of the message or in the content of the data, which has been packed in the message. Propagation of the timing errors has been contained utilizing a technique, which is called the bus guardian. A bus guardian is an autonomous unit which has a prior knowledge of the temporal access pattern of a node to a communication channel. Hence, this unit, based on its knowledge knows all of the intended message send and receive instants

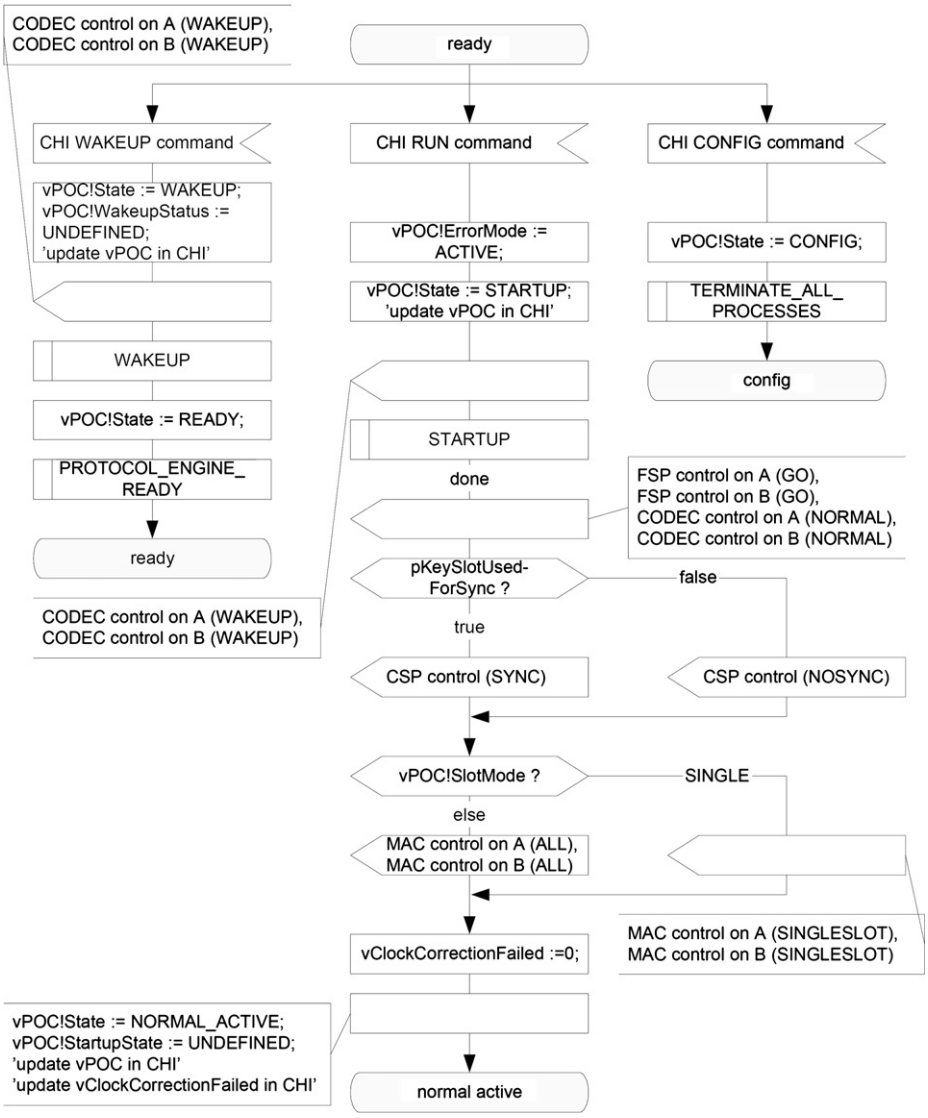

Fig. 4. POC behavior in preparation for normal operation [27].

[36]. Furthermore, it allows transmissions merely in slots assigned to a respective node. The bus guardian has a separate oscillator to prevent temporal coupling with the communication controller and to prevent common mode failures [12]. In the literature, detection of the value errors has been relegated to the host controller of a receiving node [4,37] and thus, propagation of this error model has not been contained.

3.1. Monitoring policy for the FlexRay protocol

As mentioned before, when a fault occurs in a communication controller, the controller can violate its normal behavior. This results in an original error(s). However, when an erroneous message is received by the controller, a follow-up error(s) occurs in the controller. In this situation, there is no violation in the normal behavior of the communication controller. Consequently, to differentiate between follow-up and original errors, an on-line monitoring technique is employed. This technique should monitor the behavior of the communication controller of a network node, concurrently with the operation of the controller.

To propose an on-line monitoring technique to differentiate follow-up errors from original errors, the FlexRay protocol was chosen as a case study. The two main motivations behind this are as follows: (1) the FlexRay protocol has gained an industry-wide acceptance as a next-generation automotive networking standard, and (2) the FlexRay documents [27] present a behavioral specification of this protocol utilizing SDL-based descriptions.

In the FlexRay protocol, the following seven error types are possible [18,19,27]: (1) Boundary violation error, (2) Conflict error, (3) Content error, (4) Synchronization error, (5) Syntax error, (6) Freeze error, and (7) Invalid frame error. The Boundary violation error denotes whether a boundary violation has occurred at the boundary of a corresponding slot. Hence, the boundary violation occurs if a node does not consider the channel to be idle at the boundary of a slot. The Conflict error denotes whether reception was ongoing at the time when the node started a transmission. The Content error denotes the presence of an error in a received frame. Moreover, in the FlexRay protocol, a clock rate correction mechanism is employed. However, in a synchronization process, if a fault results in failing the clock rate correction mechanism, a Synchronization error occurred. The Syntax error denotes the presence of a syntactic error in a time slot, e.g., when a decoding error occurs. Moreover, in the FlexRay communication protocol, there are three general conditions,

which transit functional state of communication controller to the halt state immediately. In this state, the POC module stops the activities of all the other modules and freezes the communication controller. After resolving the freeze conditions, communication controller should be started up through the host controller. Three freeze conditions are: (1) Product-specific error conditions such as Built-In Self-Test (BIST) and sanity check errors, (2) Error 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 the communication controller mechanisms. Finally, in addition to the mentioned error types, when a syntactically correct frame is received in the FlexRay protocol, this frame is checked to meet the 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 [27].

The occurrence of the Boundary violation, the Conflict, the Content, the Invalid frame, and the Syntax errors is reported to the host controller via the CHI module and the host responses to these errors, appropriately. In addition, the occurrence of the Synchronization error is reported in the POC module and this module based on a graceful degradation mechanism reacts to it. Moreover, in the FlexRay protocol, if one of the freeze conditions is satisfied then the CHI module enforces the POC module to stop the activities of all other modules and to freeze the communication controller [27].

Previous investigations showed that an injected transient single bit-flip fault in the FlexRay communication controller can be overwritten or can result in more than one error type in the network nodes, i.e. fault site node and the other neighbor nodes, simultaneously [18]. For example, if a fault causes a Freeze error in the fault site node, other error types such as the Syntax, the Content error, and the Invalid frame errors can occur in the other network nodes (as follow-up errors) [18]. It should be noted that due to an injected fault in one node, merely the Boundary violation, the Conflict, the Content, the Invalid frame, the Synchronization, and the Syntax errors can occur in the other network nodes, on the contrary, the Freeze error occurs only in the fault site node [18,19,27]. Moreover, if the bus guardian technique is applied, the propagation of the Boundary violation error will be contained. Consequently, the monitoring technique should be able to check the occurrence conditions of all error types except the Freeze error.

It should be noted that the main cause, the main consequence, and the possible nature of each error type have been as well presented in Table 1.

3.2. Architecture of the proposed FSM-based monitoring technique

As presented in Table 1, due to a fault in network node, the Boundary violation, the Conflict, the Content, the Invalid frame, the Synchronization, and the Syntax error types can occur as follow-up errors in the communication controller of the other nodes. Hence, if the monitor can check the occurrence conditions of these error types in the communication controller of a receiving node concurrently with the operation of the controller, it will be able, capable to determine whether the nature of an occurred error is an original or a follow-up error. The reason is that when an error occurs its occurrence conditions had not been satisfied, it implies that this error originates from a fault in the error site communication controller and thus the occurred error is an original error. However, if occurrence conditions of the occurred error had been satisfied, it implies that the error is due to an erroneous received message and thus the occurred error is a follow-up error.

To assess the satisfaction of the mentioned error types' occurrence conditions, behavioral FSMs of these error types, i.e. the Boundary violation, the Conflict, the Content, the Invalid frame, the Synchronization, and the Syntax error types, should be monitored by the monitor, concurrently with the operation of the communication controller. These FSMs are easily extractable from behavioral FSMs of the FlexRay mechanisms presented in Ref. [27]. For example, an extracted FSM of the Syntax error type has been presented using SDL as shown in Fig. 5. As shown in Fig. 5, in the standard operation of the FlexRay protocol, when the operational state of the FSP module is "decoding in progress" or "wait for CHIRP" and one of the transition conditions illustrated in this figure is satisfied, the Syntax error will occur. Each illustrated transition condition itself is satisfied in a special operational state of the FlexRay modules. For example, "decoding halted on A" transition occurs merely when the operational state of the CODEC module is "Normal" or "Wakeup". Consequently, if the monitor checks the occurrence conditions of an occurred Syntax error based on monitoring the FSM of the Syntax error type, it will be able, capable to determine whether the nature of the occurred error is an original or a follow-up error.

Fig. 6 shows the behavioral description of the differentiating monitor for the Syntax error type. As illustrated in this figure, once a Syntax error occurs, the monitor checks the operation state of the CODEC module and when the state of the CODEC is "Decoding in progress" or "Wait for CHIRP", it will check illustrated transition conditions and finally will determine the nature of the occurred error. For example, if a fault sets the "decoding halted on A" register, this fault will lead to a Syntax error. In this situation, the differentiating monitor checks occurrence conditions of the Syntax error and discovers that the occurrence conditions of setting "decoding halted on A" have not been satisfied. Hence, the monitor determines the nature of the occurred Syntax error as an original error.

In addition to the Syntax error type, behavioral FSMs of the other error types, i.e. the Boundary violation, the Conflict, the

Table 1
Error types in the FlexRay communication protocol.

Error types	Boundary violation	Conflict	Content	Freeze	Invalid frame	Synchronization	Syntax
Cause	Disregarding the channel to be idle at the boundary of a slot	Starting a transmission while reception is ongoing	Presence of an error in a received frame	Satisfying at least one of the freezing conditions	Failing the frame acceptance check	Failing the clock rate correction mechanism	Presence of a syntactic error in a received frame
Consequence	Depends on the host's response	Depends on the host's response	Depends on the host's response	Freezing the communication controller	Depends on the host's response	Activating a graceful degradation mechanism	Depends on the host's response
Informed unit	Host controller	Host controller	Host controller	The CHI module	Host controller	The POC module	Host controller
Can appear as an original error?	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Can appear as a follow-up error?	Yes	Yes	Yes	No	Yes	Yes	Yes

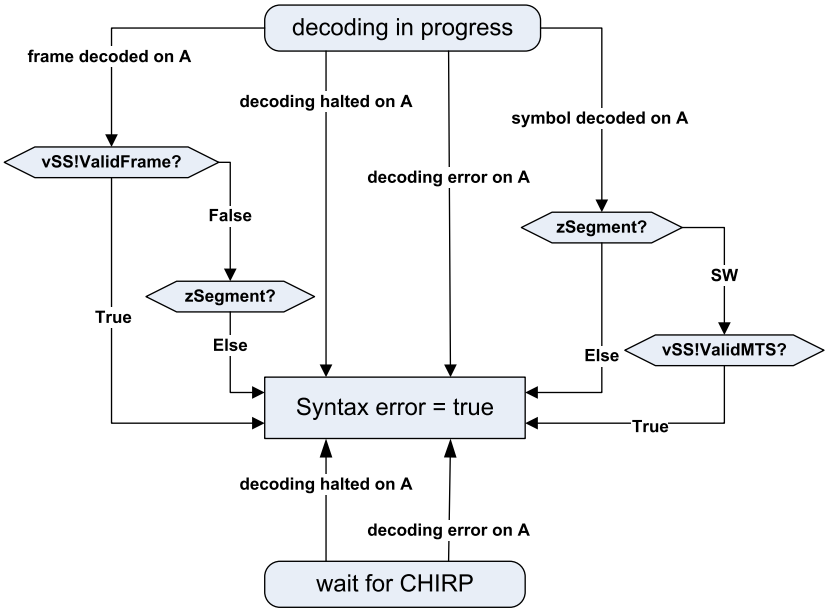

Fig. 5. SDL presentation of Syntax error type.

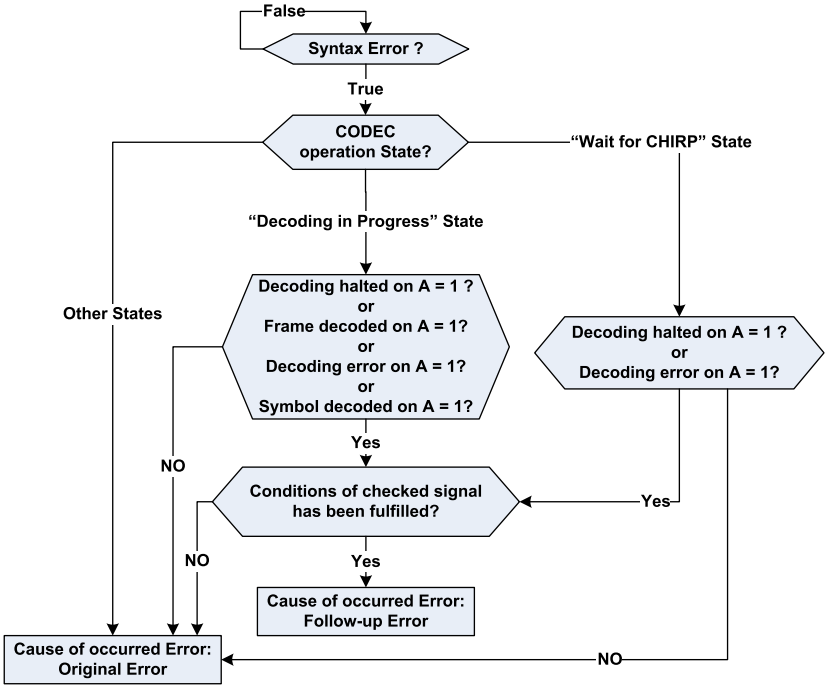

Fig. 6. SDL diagram of the differentiating monitor for the Syntax error.

Content, the Invalid frame, and the Synchronization error types, were extracted from the behavioral FSMs of the FlexRay mechanisms presented in Ref. [27] and based on them, behavioral descriptions for the differentiating monitor were provided. It should be noted that the differentiating monitor observes the behavior of the communication controller and traces the operational state of the controller, concurrently with the operation of the controller. Hence, when one of the error types occurs, the monitor will be able to determine the nature of the occurred error immediately. Moreover, the monitor creates no disturbance in the operation of the communication controller; consequently, it has no effect on the performance of the controller.

Fig. 7 shows the architecture of a FlexRay node in the presence of the differentiating monitor. This figure as well shows the relationship between FlexRay communication controller and proposed differentiating monitor. As illustrated in this figure, the communication controller exchanges configuration data, status information, and communication data with the host controller. Moreover, it informs the type of occurred errors to the host. In addition, the proposed monitor observes the operational status of the communication controller and provides an output for the host controller that determines the nature of occurred errors, i.e. original or follow-up errors.

To apply the technique to the other FSM-based communication protocol, e.g., TTP/C [28], and TT-Ethernet [29], five steps should be performed: (1) as described in this section, all probable error types should be identified, (2) the behavioral FSM of the detection mechanism for each error type is extracted, (3) based on

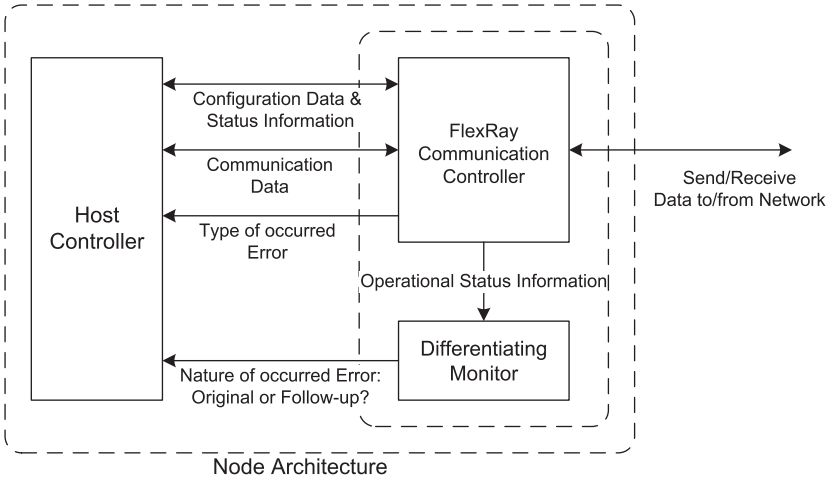

Fig. 7. FlexRay node architecture with differentiating monitor.

the extracted FSM for each error type, an FSM should be provided to assess the satisfaction of the error type's occurrence conditions, (4) all error types' assessing FSMs should be integrated in a differentiating monitor, and Finally (5) the differentiating monitor is placed beside the communication controller in a network node. Hence, the consequent monitor will be able to determine the nature of the occurred errors in each communication controller in a network's node.

4. An analysis on the follow-up errors

In this section, a probability analysis has been presented. In this analysis, TDMA-based communication protocols have been analyzed and the occurrence probability of the follow-up errors has been determined.

Communications in the time-triggered protocols are based on sending messages in predetermined interval times, which are named as communication cycles (bus cycles). These communication cycles are executed periodically. In these protocols, based on Time Division Multiple Access (TDMA) [31] mechanism, a communication cycle is composed of several communication time slots called static slots. Each static slot is assigned merely to one node of the network. Moreover, it is possible that more than one slot is assigned to one node. Hence, each node sends its data frame merely in static slots, which have been assigned to it.

To analyze the occurrence probability of follow-up errors, several assumptions and notations should be established before proceeding with the main development of the section. The system is composed of N_n nodes. Each node is composed of a processing unit and a communication controller. The physical and architectural structures of all system nodes are assumed to be identical. Each communication cycle contains N_s equal length time slots and N_s^i is number of time slots, which have been assigned to the ith node and B^i denotes the bandwidth that has been assigned to the ith node:

$$B^i = \frac{N_s^i}{N_s}.$$

To simplify the analysis, it is assumed that just one fault could have occurred in each communication cycle. The rate of fault occurrence for all nodes is the same and depends on environmental conditions. It is denoted by R_{flt_Occ}. Not all faults in the communication controller lead to an error. The probability of a fault in the communication controller leading to an error depends on where and when that fault occurs and is denoted by P_{flt_act}. It is obvious that the occurrence probability of the original error, P_o for all nodes due to a fault in the communication controller of the fault site node is the same and is equal to:

$$P_o = R_{flt\_Occ} \times P_{flt\_act}.$$

Among all original errors in each node, merely a part of them can be propagated to the other nodes of the network. Let R_{propag} denote the ratio of the propagated original errors. This ratio is estimated by analytical or experimental approaches. Therefore, the probability of propagating an original error in jth node to the other nodes, P_{propag} in a time slot, which has been assigned to the jth node, is equal to:

$$P_{propag} = R_{propag} \times P_o.$$

Now it is ready to calculate the occurrence probability of an error in the ith node, P_e^iin a communication cycle. In each time slot, which has been assigned to the ith node, the occurred error is just an original error due to a fault in that node as a fault site node, therefore the probability of error in this time slot is:

$$P_e^i = P_o.$$

In the other time slots, which have been assigned to the jth, ($j \neq i$) node, the occurred error in the ith node is not merely due to an original error in the ith node, however, it may be due to a propagated error from the jth node to it and therefore the probability of an error in this time slot is:

$$P_e^i = P_o + \mathrm{P}_{propag}.$$

The above equality holds because it is assumed that just one fault could have occurred in each communication cycle. By applying previous equations, the occurrence probability of an error in the ith node in each communication cycle is:

$$P_e^i = P_o + \sum_{\substack{j=1 \\ i \neq j}}^{N_n} \left(P_{propag} \times \frac{N_s^j}{N_s} \right) = P_o + \sum_{\substack{j=1 \\ i \neq j}}^{N_n} \left((P_o \times R_{propag}) \times \frac{N_s^j}{N_s} \right)$$

$$= P_o + \left(P_o \times R_{propag} \times \frac{N_s - N_s^i}{N_s} \right). \quad (1)$$

The first term in Eq. (1) denotes the occurrence probability of an original error in the ith node and the second term denotes the occurrence probability of an error in the ith node due to a fault in the other nodes, called a follow-up error, which is denoted by P_f^i:

$$P_f^i = R_{propag} \times P_o \times \frac{N_s - N_s^i}{N_s}. \quad (2)$$

By applying Eq. (2), the ratio of the occurrence probability of a follow-up error to the occurrence probability of an original error in the ith node, R_f^i , is equal to:

$$R_f^i = \frac{P_f^i}{P_o} = R_{propag} \times \frac{N_s - N_s^i}{N_s}. \quad (3)$$

In Eq. (3), as mentioned before, R_{propag} denotes the ratio of the propagated original errors from a fault site node to the ith node. It should be noted that this ratio is the same for all nodes, which employ a same communication controller. Moreover, this equation shows that the follow-up error ratio for the ith node is increased when the number of assigned time slots to that node is decreased.

This simplified analysis shows that the occurrence probability of follow-up errors is not negligible and is even noticeable. For example, based on the presented results in Ref. [19], ratio of the propagated original errors from a fault site node to a neighbor node in the FlexRay protocol is about 88.8% ($R_{propag} = 0.888$) and if among all 10 time slots ($N_s = 10$), two slot is assigned to the ith node ($N_s^i = 2$), then the ratio of the occurrence probability of a follow-up error to the occurrence probability of an original error in the ith node is approximately 0.71 ($R_f^i = 0.7104$). Hence, designers should take care of the follow-up errors.

5. Experimental evaluation

Safety-critical applications have to function perfectly even in the presence of faults. Faults can be permanent, transient, or intermittent. The transient faults in flip-flops, latches and combinational logic circuits, pose a key challenge in the design of fault-tolerant applications and their rates are significantly increasing due to the ever decreasing technology size of semiconductor devices [38–40]. Transient single bit-flip errors, which are the major consequences of transient faults [41] have been considered in this paper.

5.1. Experimental setup

The FlexRay communication controller was implemented by a hardware description language, Verilog HDL, based on State Diagram Language (SDL) descriptions of this controller in the FlexRay specifications [27]. Later, specifications of this controller, e.g., timing and configuration were tested according to the FlexRay protocol conformance test specification [42].

To evaluate the proposed monitoring technique, a cluster consisting of 4 nodes with single bus topology (Fig. 8) was formed. In this topology, each node was composed of a host controller, a communication controller and a proposed differentiating monitor. The host controller typically was a hardware unit that generates data to exchange with the other nodes through a communication channel. In the experiments, instead of a real host, a data generator was employed 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.

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

5.2. Results

The modeled FlexRay communication controller is not still synthesizable to investigate the error propagation rate in the FlexRay-based networks, transient single bit-flip faults were injected into all accessible registers of the communication controller of node 2 (as a fault site node) and their effects on another communication controller (node 4 as an observed neighbor node) were investigated. To reach a sufficiently accurate investigation, 50 transient bit-flip faults were injected to each bit of all FlexRay controller registers in the fault site node and gathered results were investigated. The fault injection process utilized in this experiment is the same as the one utilized in Refs. [18] and [19].

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 in the fault site node and behaviors of the communication controller in the fault site node as well as the observed neighbor node were observed. As mentioned before in Refs. [18] and [19], investigations showed that in the FlexRay communication controller, an injected transient single bit-flip fault can be overwritten or can result in one or more discussed error types in the fault site and the observed neighbor nodes, simultaneously. For example, if a fault leads to satisfy one of the freeze conditions in a communication controller, the POC module stops the activities of all the communication controller's modules and freezes the controller. In this situation, if the fault site communication controller is sending a message, its transmission is discontinued and the transmitted message will be imperfect. Moreover, if the fault site communication controller is receiving a message, its reception is discontinued and the received message will be imperfect. Hence, several error types such as the Syntax, the Content, the Boundary violation, and even the Invalid frame errors can occur due to a Freeze error in the observed neighbor node and even in the fault site node.

In addition to a communication controller, the proposed differentiating monitor was placed in each node. This monitor reports the nature of the occurred errors, i.e. original or follow-up errors, in each node. Table 2, presents number and occurrence rate of each error types in the fault site and observed neighbor nodes, which have been presented in Ref. [19]. Detailed analyses of the occurrence rate of the error types in the fault site and observed neighbor nodes are beyond the scope of this paper and have been presented in Refs. [18] and [19]. Moreover, this table

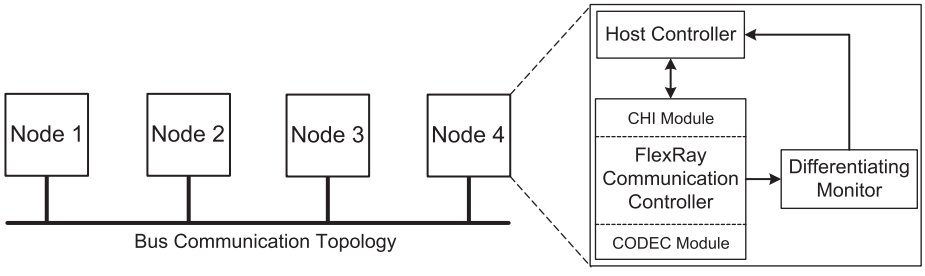

Fig. 8. Experimental setup.

Table 2
Statistics on the error types in the observed neighbor node.

		Boundary violation errors	Conflict errors	Content errors	Synchronization errors	Syntax errors	Invalid frame errors	Total errors
Fault site node	Occurred errors (#) Monitor	2489	0	1746	6403	513	4149	15300
	Original errors (#)	2410	0	1715	6180	498	4073	14876
	Follow-up errors (#)	79	0	31	223	15	76	424
	Accuracy (%)	96.8	–	98.2	96.5	97.1	98.2	97.2
Neighbor node	Occurred errors (#) Monitor	1718	0	625	4705	2005	4488	13541
	Original errors (#)	0	0	0	0	0	0	0
	Follow-up errors (#)	1718	0	625	4705	2005	4488	13541
	Accuracy (%)	100	–	100	100	100	100	100

Table 3
Hardware overhead of the proposed differentiating monitor.

	Standard FlexRay communication controller	Differentiating Monitor							Hardware overhead
		Boundary violation error	Conflict error	Content error	Synchronization error	Syntax error	Invalid frame Error	Total	
Flip-flops	2712	6	4	5	10	7	6	38	1.4%

shows the number of occurred errors, whether their natures have been correctly determined by the monitor and the accuracy rate for the differentiating monitor in each node.

It should be noted that, in this experiment, transient single bit-flip faults were injected in the communication controller of the fault site node; hence, nature of all occurred errors in that node is the original error, whereas the nature of all occurred errors in the communication controller of the observed node is the follow-up error.

The experimental results showed that the differentiating monitor has correctly determined the nature of all occurred errors in the observed neighbor node as the follow-up error. However, in the fault site node, the monitor has determined the nature of 97.2% of all occurred errors as the original error, and has considered other errors (merely 2.8%) as the follow-up errors, incorrectly. Investigations on the experimental results showed that when an error occurs in a communication controller's register, which indirectly affects occurrence conditions of the error types, the nature of this error may be determined as the follow-up error instead of the original error. This issue will be resolved if more error occurrence conditions are checked by the monitor utilizing more imposed hardware overhead. Table 3 shows an estimation of the imposed hardware overhead of the proposed differentiating monitor for each error type. It should be noted that since the modeled FlexRay communication controller is not yet synthesizable, the imposed hardware overhead is estimated based on the number of added flip-flops in comparison with the number of communication controller's flip-flops. The main purpose of this estimation is to show that the hardware overhead of the proposed differentiating monitor is rather negligible.

As shown in Table 3, the differentiating monitor imposes only 38 flip-flops (1.4%) as hardware overheads to the FlexRay communication controller. These flip-flops are employed as flags, which are utilized to trace the behavior of the FlexRay communication controller to check error occurrence conditions. Among all error types, checking occurrence conditions of the Synchronization is the most complicated and needs to apply 10 flip-flops, whereas to check the occurrence conditions of the Conflict errors, only 4 flip-flops is required.

It should be noted that, the differentiating monitor is consulted merely when an error is observed by the controller. Consequently, a single bit-flip transient fault in the differentiating monitor cannot result in any malfunction in the communication controller.

6. Conclusion

It has been shown that a transient fault occurring in the communication controller of a network node can cause errors in the fault site node (called original errors) and/or in the neighbor nodes (called follow-up errors). This paper proposes a novel low-cost monitoring technique to differentiate follow-up errors from original errors. The proposed technique is based on monitoring the states of a communication controller, which is applicable for all communication protocols having an FSM-based description such as FlexRay, TTP/C, and TT-Ethernet. To evaluate this technique, a FlexRay-based network including 4 nodes was implemented. The monitoring technique was as well implemented inside each node of the network. A total of 135,600 transient bit-flip faults were injected into the communication controller of one node. The results showed that about 6.0% of injected faults lead to original errors. This figure for follow-up errors was about 6.1%. The results as well showed that the accuracy of the proposed technique to differentiate between the follow-up and the original errors is about 97% at only 1.4% hardware overhead. This level of accuracy and cost makes the proposed technique a viable solution to enhance the reliability of communication controllers. Furthermore, an analysis has been performed to show that it is important to pay special attention to the follow-up errors. In this analysis, TDMA-based communication protocols have been analyzed and the occurrence probability of the follow-up errors has been determined.

Acknowledgment

The authors would like to thank Mrs. Monireh Houshmand from Electrical Engineering Department of Ferdowsi University of Mashhad for her helpful comments and suggestions which have improved the paper.

References

[1] A. Hagiescu, U.D. Bordoloi, S. Chakraborty, Performance analysis of FlexRay-based ECU networks, in: Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC '07), San Diego, USA, 4–8 June 2007, pp. 284–289.

[2] T. Nolte, H. Hansson, L. Bello, Automotive communications—past, current and future, in: Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'05), vol. 1, Catania, Italy, 19–22 September 2005, pp. 985–992.

[3] N. Suri, C.J. Walter, M.M. Hugue (Eds.), IEEE Press, Los Alamitos, 1995.

[4] H. Kopetz, Fault containment and error detection in the time-triggered architecture, in: Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS'03), Pisa, Italy, 9–11 April 2003, pp. 139–146.

[5] E. Armengaud, A. Steininger, M. Horauer, Towards a systematic test for embedded automotive communication systems, IEEE Transactions on Industrial Informatics 4 (3) (2008) 146–155.

[6] T. Pop, P. Pop, P. Eles, Z. Peng, A. Andrei, Timing analysis of the FlexRay communication protocol, in: Proceedings of the 18th Euromicro Conference Real-Time Systems (ECRTS'06), Dresden, Germany, 5–7 July 2006, pp. 203–216.

[7] T. Pop, P. Pop, P. Eles, Z. Peng, Bus access optimization for FlexRay-based distributed embedded systems, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition 2007 (DATE '07), Nice, France, 16–20 April 2007, pp. 1–6.

[8] R. Makowitz, C. Temple, FlexRay—a communication network for automotive control systems, in: Proceedings of the 6th IEEE International Workshop on Factory Communication Systems (WFCS 2006), Torino, Italy, 28–30 June 2006, pp. 207–212.

[9] J. Voelcker, Top 10 tech cars, IEEE Spectrum Magazine 45 (4) (2008) 27.

[10] H. Sivencrona, P. Johannessen, M. Persson, J. Torin, Heavy-ion fault injections in the time-triggered communication protocol, in: Proceedings of the 1st Latin American Symposium on Dependable Computing (LADC '03), Sao Paulo, Brazil, 21–24 October 2003, pp. 69–80.

[11] S. Blanc, P.J. Gil, Improving the multiple errors detection coverage in distributed embedded systems, in: Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS'03), Florence, Italy, 6–18 October 2003, pp. 303–312.

[12] A. Ademaj, H. Sivencrona, G. Bauer, J. Torin, Evaluation of fault handling of the time-triggered architecture with bus and star topology, in: Proceedings of the International Conference on Dependable Systems and Networks (DSN'03), San Francisco, CA, USA, 22–25 June 2003, pp. 123–132.

[13] G. Bauer, H. Kopetz, W. Steiner, The central guardian approach to enforce fault isolation in the time-triggered architecture, in: Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS'03), Pisa, Italy, 9–11 April 2003, pp. 37–44.

[14] G.N. Sung, C.Y. Juan, C.C. Wang, Bus guardian design for automobile networking ECU nodes compliant with FlexRay standard, in: Proceedings of the IEEE International Symposium on Consumer Electronics (ISCE'08), Algarve, Portugal, 14–16 April 2008, pp. 1–4.

[15] V. Lari, M. Dehbashi, S.G. Miremadi, N. Farazmand, Assessment of message missing failures in FlexRay-based networks, in: Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing (PRDC'07), Melbourne, Australia,17–19 December 2007, pp. 191–194.

[16] V. Lari, M. Dehbashi, S.G. Miremadi, M. Amiri, Evaluation of babbling idiot failures in FlexRay-based networks, in: Proceedings of the 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems (FET'07), Toulouse, France, 7–9 November 2007, 8 pp.

[17] Y. Sedaghat, S.G. Miremadi, Investigation and reduction of fault sensitivity in the FlexRay communication controller registers, in: Proceedings of the 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP'08), Newcastle upon Tyne, UK, 22–25 September 2008, pp. 153–166.

[18] Y. Sedaghat, S.G. Miremadi, Categorizing and analysis of activated faults in the FlexRay communication controller registers, in: Proceedings of the 14th European Test Symposium (ETS'09), Seville, Spain, 25–29 May 2009.

[19] Y. Sedaghat, S.G. Miremadi, Classification of activated faults in the FlexRay-based networks, Journal of Electronic Testing: Theory and Applications (JETTA) 26 (5) (2010) 535–547.

[20] Y. Sedaghat, S.G. Miremadi, A low-cost on-line monitoring mechanism for the FlexRay communication protocol, in: Proceedings of the 4th Latin–American Symposium on Dependable Computing (LADC'09), Joao Pessoa, Brazil, 1–4 September 2009, pp. 111–118.

[21] W. Steiner, H. Kopetz, The startup problem in fault-tolerant time-triggered communication, in: Proceedings of the Conference on Dependable Systems and Networks (DSN'06), Philadelphia, PA, USA, 25–28 June 2006, pp. 35–44.

[22] P. Milbredt, M. Horauer, A. Steininger, An investigation of the clique problem in FlexRay, in: Proceedings of the 3rd International Symposium on Industrial Embedded Systems (SIES'08), Montpellier, France,11–13 June 2008, pp. 200–207.

[23] W. Li, M.D. Natale, W. Zheng, P. Giusto, A.L. Sangiovanni-Vincentelli, S.A. Seshia, Optimizations of an application-level protocol for enhanced dependability in FlexRay, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'09), Nice, France, 20–24 April 2009, pp. 1076–1081.

[24] M. Mitzlaff, M. Lang, R. Kapitza, W. Schröder-Preikschat, A membership service for a distributed, embedded system based on a time-triggered FlexRay network, in: Proceedings of the 8th European Dependable Computing Conference (EDCC-8), Valencia, Spain, 28–30 April 2010, pp.155–162.

[25] P. Marwedel, Embedded System Design, Springer, 2006.

[26] H. Kopetz, Architecture of safety-critical distributed real-time systems, invited talk, in: Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'03), Munich, Germany, 3–7 March 2003.

[27] FlexRay Communications System—Protocol Specification V2.1 Revision A, 〈www.flexray.com〉.

[28] Time-Triggered Protocol TTP/C High Level Specification Document, TTTech, Vienna, Austria 2002.

[29] H. Kopetz, A. Ademaj, P. Grillinger, K. Steinhammer, The time-triggered ethernet (TTE) design, in: Proceedings of the 8th International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05), Seattle, WA, USA, 18–20 May 2005, pp. 22–33.

[30] F. Sethna, E. Stipidis, F.H. Ali, What lessons can controller area networks learn from FlexRay, in: Proceedings of the IEEE Vehicle Power and Propulsion Conference (VPPC'06), Windsor, UK, 6–8 September 2006, pp. 1–4.

[31] K. Tindell, J. Clark, Holistic schedulability analysis for distributed hard real-time systems, Transactions on Microprocessing and Microprogramming 40 (2–3) (1994) 117–134.

[32] G. Cena, A..Valenzano, Performance analysis of byteflight networks, in: Proceedings of the 5th IEEE international Workshop Factory Communication Systems (WFCS'04), Vienna, Austria, 22–24 September 2004, pp. 157–166.

[33] ITU-T Recommendation Z.100 (03/93), Programming Languages—CCITT Specification and Description Language (SDL), International Telecommunication Union, Geneva, 1993.

[34] M. Dehbashi, V. Lari, S.G. Miremadi, M. Shokrolah-Shirazi, Fault effects in FlexRay-based networks with hybrid topology, in: Proceedings of the 3rd International Conference on Availability, Reliability and Security (ARES'08), Barcelona, Spain, 4–8 March 2008, pp. 491–496.

[35] F. Cristian , H. Aghili , R. Strong, D. Dolev, Atomic broadcast: from simple message diffusion to byzantine agreement, in: Proceedings of the 25th International Symposium on Fault-Tolerant Computing (FTCS-25), Pasadena, California, USA, 27–30 June 1995, pp. 431.

[36] C. Temple, Avoiding the babbling-idiot failure in a time-triggered communication system, in: Proceedings of the 28th Annual International Symposium on Fault-Tolerant Computing (FTCS-28), Munich, Germany, 23–25 June 1998, pp. 218–227.

[37] G. Bauer, H. Kopetz, Transparent redundancy in the time-triggered architecture, in: Proceedings of the International Conference on Dependable Systems and Networks (DSN'00), New York, USA, 25–28 June 2000, pp. 5–13.

[38] V. Izosimov, P. Pop, P. Eles, Z. Peng, Design optimization of time- and cost-constrained fault-tolerant distributed embedded systems, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition 2005 (DATE'05), vol. 2, Munich, Germany, 7–11 March 2005, pp. 864–869.

[39] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, T. Toba, Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule, IEEE Transactions on Electronic Devices 57 (7) (2010) 1527–1538.

[40] S. Mitra, M. Zhang, T.M. Mak, N. Seifert, V. Zia, K.S. Kim, Logic soft errors: a major barrier to robust platform design, in: Proceedings of the 36th annual IEEE International Test Conference (ITC'05), Austin, Texas, USA, 8–10 November 2005, pp. 687–696.

[41] E. Armengaud, F. Rothensteiner, A. Steininger, M. Horauer, A method for bit level test and diagnosis of communication services, in: Proceedings of the 8th International IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS'05), Sopron, Hungary, 13–16 April 2005, pp. 69–74.

[42] FlexRay Communications System—Protocol Conformance Test Specification V2.1, 〈www.flexray.com〉.