Categorizing and Analysis of Activated Faults in the FlexRay Communication Controller Registers

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

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

Abstract

FlexRay communication protocol is expected becoming the de-facto standard for distributed safety-critical systems. In this paper, transient single bit-flip faults were injected into the FlexRay communication controller to categorize and analyze the activated faults. In this protocol, an activated fault results in one or more error types which are Boundary violation, Conflict, Content, Freeze, Synchronization, and Syntax. 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 were injected in only one node, where 9,342 (6.9%) of the faults were activated. The results show that the Synchronization error is the widespread error with the occurrence ratio of about 70.1%. The Boundary violation and the Syntax errors have the occurrence ratios of 32.4% and 24.6%, respectively. The results also show that the Freeze error which more frequent resulted system failures has the occurrence ratio of about 17.3%.

Keywords: Safety-critical applications, Distributed embedded systems, FlexRay protocol, Fault injection.

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 [1], [2]. This means that if a communication network fails, the overall operation cannot be succeeded; however, if a node fails, its task can be assigned to another node. Thus, the communication network is a single point of failure. A network failure occurs 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 [3]. Examples of time-triggered protocols are SAFEbus, SPIDER, and TTP/C. The main drawback of time-triggered protocols is their lack of flexibility [3]. To resolve the drawbacks of both event-triggered and time-triggered protocols, other protocols such as TTCAN, FTT-CAN, and FlexRay [4] 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 [5], [2], [3], [6]. As an example, the next edition of the BMW X5 will use the FlexRay protocol in its electronically controlled dampers [5].

Several work have investigated the reliability and fault tolerance of communication protocols. Effects of masquerade failures [7] and message missing failures [8] have been studied for the CAN protocol by the simulation-based fault injection. Also, in [9], 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 [10] and physical pin-level fault-injection [11]. Moreover, fault tolerance of this protocol and the failures have been studied using heavy-ion fault injection and simulation-based fault injection [12].

An evaluation of the FlexRay protocol using simulation-based fault injection has been reported in [13] and [14]. However, these studies have two main limitations: 1) faults were injected only into about 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 [15], faults were injected into all 408 registers of the FlexRay to study the fault sensitivity.

This paper extends the work in [15] by categorizing the activated faults into six error types: Boundary violation, Conflict, Content, Freeze, Synchronization, and Syntax. In this study, a FlexRay bus network, composed of four nodes, was modeled by Verilog HDL and a total of 135,600 faults were injected in one node. The results show that about 9,342 of the faults were activated.

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 presented. 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 10Mbit/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 [4].

From the dependability point of view, the FlexRay documents [4] 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 [1].

The main purpose of this paper is to instruct the developers of the FlexRay communication controller, which errors are widespread and which error type is the most destructive. 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 [4]: 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.

Figure 1. The FlexRay structure [4]

- 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) [16] scheme for a time-triggered (or static) window, and Flexible TDMA (FTDMA) [17] 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 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.

Figure 2. Communication cycle in the FlexRay protocol

3. Error types and error handling mechanisms in the FlexRay protocol

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 [18]. Transient single bit-flip errors, which are considered in this paper, are more common consequences of the transient faults [19].

3.1. Error types in the FlexRay protocol

In the FlexRay protocol, due to occurring faults, the following six categories of errors are possible [4]:

- 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 a received frame.
- 4) **Synchronization error**: In the FlexRay protocol, a clock ratio correction mechanism is used; but in synchronization process, if a fault results in failing the clock ratio 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.

Occurrences of the Boundary violation, the Conflict, the Content, and the Syntax errors are reported to the host via the CHI module and host responses to these errors properly. Also occurrence of the Synchronization error is reported to the POC

module and this module, based on a graceful degradation mechanism, reacts to it.

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 host. 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 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, the freeze conditions were assumed as an individual error type. Moreover, our investigations show that in the FlexRay communication controller, an injected transient single-bit flip fault may be overwritten, or may result in one or more discussed errors simultaneously. For example, if a fault causes a freeze error, other error types such as the Syntax error, the Content error, and even the Boundary violation error can be occurred.

3.2. Error indicator registers of the FlexRay communication controller

In the FlexRay protocol, there are some registers that are behaved unexpectedly in the mentioned error conditions. In this paper, these registers are referred as "error indicator registers". Table 1 shows these registers and their locations in the FlexRay communication controller. This table also shows the error type which is related to each error indicator register.

4. Experimental environment

The FlexRay communication controller, according to its specifications [4], 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 [20].

As for the experiment setup, a cluster was formed consisting of 4 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 nodes through a communication channel. In the experiments, instead of a real host, a data generator

Table 1. Error indicator registers in the FlexRay protocol

Registers	FlexRay Module	Error Type
decoding_error_on_A	CODEC	Syntax
TSS ok		
TSS too long		
FSS ok		
payload ok		
trailer ok		
BSS ok		
FES ok		
zBssError		
Header Crc error		
Frame Crc error		
zSyncCalcResult	CSP	Sync*
vPOC Freeze	POC	Freeze
vPOC_CHIHaltRequest		
vPOC_ErrorMode		
Content error on A	FSP	Content
Fatal protocol error		Freeze
T_StatusSlot_SyntaxError		Syntax
T_StatusSlot_ContentError		Content
T_StatusSlot_TxConflict		Conflict
T_StatusSlot_BViolation		Boundary Violation

*Synchronization

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, any node was allowed to send/receive frames to/from the communication channel.

To investigate the fault tolerance of the FlexRay communication controller, transient single bit-flip faults were injected in all accessible registers of communication controller modules of the node 2 and their effects on the error indicator registers were observed in node 4 and node 2; because the occurrence of the Syntax, the Boundary violation, the Content, and the Conflict errors in one node, is also observable on the other nodes (e.g. node 4); whereas, the occurrence of the Freeze error is only observable where they are occurred (i.e. node 2).

4.1. Error types determination process

An error type determination process includes two phases. In the first phase, a fault should be injected into an accessible flip-flop (one bit of a register) of communication controller; and if 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 node 2, the SINJECT fault injection tool [21] 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 node 2 and

node 4 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 node 2 at a random time, and the behavior of the error indicator registers of node 2 and node 4 are observed.

Step3: In the final 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 an 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 type determination process 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

In this paper, to assess the fault tolerance of the FlexRay communication controller and to determine widespread and destructive errors in the FlexRay protocol, the nodes were connected through a passive bus network. The main reason of selecting bus topology is to prevent some error propagations in star coupler of star topology. This prevention results in hiding the fault sensitivity of some communication controller registers.

To simulate the experiments, the ModelSim 5.5 simulation environment was used. In this paper in comparison with [15], 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; because of unstable behaviors of the network in the first communication cycle, the fault injection process was started at the end of this cycle.

To obtain a careful investigation of the fault tolerance, 50 transient bit-flip faults were injected to each bit of all accessible FlexRay controller registers. Hence, a total of 135,600 transient single bit-flip faults were injected to all 408 accessible single-bit and multiple-bit registers of the communication controller in node 2.

$$\text{Occurrence ratio of one error type for one unit} = \frac{\text{Number of the error type in the unit}}{\text{Number of all activated faults in the unit}} \times 100\% \quad (1)$$

Table 2. Statistics on the error types in different modules of the FlexRay

Module	Injected Faults	Activated Faults		Boundary Violation Errors		Conflict Errors		Content Errors		Freeze Errors		Synchronization Errors		Syntax Errors	
		#	%	#	%	#	%	#	%	#	%	#	%	#	%
CHI	32350	5628	17.4	2054	36.5	0	0	1290	22.9	406	7.2	3606	64.1	1899	33.7
CODEC	32300	291	0.9	40	13.7	0	0	45	15.5	40	13.7	98	33.7	135	46.4
CSP	47850	1314	2.7	570	43.4	0	0	333	25.3	0	0	1272	96.8	50	3.8
FSP	6850	164	2.3	29	17.7	0	0	51	31.1	51	31.1	76	46.3	5	3.0
MAC	11050	655	5.9	240	36.6	0	0	109	16.6	144	22.0	509	77.7	121	18.5
POC	5200	1290	24.8	90	7.0	0	0	93	7.2	979	75.9	987	76.5	87	6.7
TOTAL	135600	9342	6.9	3023	32.4	0	0	1921	20.6	1620	17.3	6548	70.1	2297	24.6

In this paper, various types of occurred errors which are all due to injected faults have been categorized. To reach this goal, occurrence ratio of each error type for every bit of communication controller registers, using Equation 1, was calculated; and then occurrence ratio of these error types was achieved for each communication controller modules. Table 2 shows the total of activated faults and their ratios, number of occurrence of each error type, and the occurrence ratio of them for all FlexRay communication controller modules.

The presented results in the Table 2 show that in the CODEC module which is responsible for coding and decoding communication elements, the Syntax error is the most common occurred error, Whereas in the other modules the Synchronization error is the most common occurred 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 ratio of the Freeze error is noticeable; based on its destructive property, we convince FlexRay developers to pay more attention to the freeze conditions carefully.

Figure 4 shows the occurrence ratio 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 Figure 5 show that the Synchronization error, the Boundary violation error, and the Syntax error are widespread occurred errors; but the Synchronization error is the more occurred than both the Syntax error and the Boundary violation error. 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 Syntax and the Content errors from propagation.

■ Boundary violation ■ Conflict ■ Content ■ Freeze ■ Synchronization ■ Syntax

Figure 4. Error types in different modules of the FlexRay **Figure 5. Overall error types in the FlexRay protocol**

6. Conclusions

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, are introduced. The results show that the Synchronization error has the occurrence ratio of 70.1%. After that, the Boundary violation and the Syntax errors have the

occurrence ratios of 32.4% and 24.6%, respectively. Furthermore, although the Freeze error has low occurrence ratio (about 17.3%), this error is more destructive than the other errors. Moreover, the authors of this paper recommend applying efficient techniques to prevent the Syntax and the Content errors from propagation and convince the FlexRay developers to reduce the occurrence ratio of the Freeze errors by applying fault-tolerant mechanisms to check the freeze conditions, carefully.

7. References

- [1] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in Automotive Communication Systems", *Proc. of IEEE*, vol. 93, Issue 6, June 2005, pp. 1204-1223.
- [2] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing Analysis of the FlexRay Communication Protocol", *Proc. of 18th Euromicro Conference Real-Time Systems (ECRTS'06)*, Dresden, Germany, 5-7 July 2006, pp. 203-216.
- [3] T. Pop, P. Pop, P. Eles, and Z. Peng, "Bus Access Optimization for FlexRay-based Distributed Embedded Systems", *Proc. of Design, Automation & Test in Europe Conference & Exhibition 2007 (DATE '07)*, Nice, France, 16-20 April 2007, pp. 1-6.
- [4] FlexRay Communications System - Protocol Specification V2.1 Revision A, www.flexray.com
- [5] F. Sethna, E. Stipidis, and F.H. Ali, "What Lessons Can Controller Area Networks Learn From FlexRay", *Proc. of 2nd Vehicle Power and Propulsion Conference (VPPC '06)*, Windsor, UK, 6-8 September 2006, pp.1-4.
- [6] A. Hagiescu, U.D. Bordoloi, and S. Chakraborty, "Performance Analysis of FlexRay-based ECU Networks", *Proc. of 44th ACM/IEEE Design Automation Conference (DAC '07)*, San Diego, CA, USA, 4-8 June 2007, pp. 284-289.
- [7] H. Salmani and S.G. Miremadi, "Contribution of Controller Area Networks Controllers to Masquerade Failures", *Proc. of 11th Pacific Rim International Symposium on Dependable Computing (PRDC'05)*, Changsha, Hunan, China, 12-14 December 2005, pp. 310-316.
- [8] H. Salmani and S.G. Miremadi "Assessment of Message Missing Failures in CAN-based Systems", *Proc. of IASTED International Conference on Parallel and Distributed Computing and Networks*, Austria, 15-17 February 2005, pp. 387-392.
- [9] J. Perez, M.S. Reorda, and M. Violante, "Dependability Analysis of CAN Networks: an Emulation-based Approach", *Proc. of 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03)*, Boston, MA, USA, 3-5 November 2003, pp. 537-544.
- [10] H. Sivencrona, P. Johannessen, M. Persson, and J. Torin, "Heavy-ion Fault Injections in the Time-triggered Communication Protocol", *Proc. of 1st Latin American Symposium on Dependable Computing (LADC '03)*, Sao Paulo, Brazil, 21-24 October 2003, pp. 69-80.
- [11] S. Blanc and P.J. Gil, "Improving the Multiple Errors Detection Coverage in Distributed Embedded Systems", *Proc. of 22nd International Symposium on Reliable Distributed Systems (SRDS'03)*, Florence, Italy, 6-18 October 2003, pp. 303-312.
- [12] A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin, "Evaluation of Fault Handling of the Time-Triggered Architecture with Bus and Star Topology", *Proc. of International Conference on Dependable Systems and Networks (DSN'03)*, San Francisco, CA, USA, 22-25 June 2003, pp. 123-132.
- [13] V. Lari, M. Dehbashi, S.G. Miremadi, and N. Farazmand, "Assessment of Message Missing Failures in FlexRay-Based Networks", *Proc. of 13th Pacific Rim International Symposium on Dependable Computing (PRDC'07)*, Melbourne, Australia, 17-19 December 2007, pp. 191-194.
- [14] V. Lari, M. Dehbashi, S. G. Miremadi, and M. Amiri, "Evaluation of Babbling Idiot Failures in FlexRay-based Networks", *Proc. of 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems (FET'07)*, Toulouse, France, 7-9 November 2007, pp. 8.
- [15] Y. Sedaghat, and S.G. Miremadi, "Investigation and Reduction of Fault Sensitivity in the FlexRay Communication Controller Registers", *Proc. of 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP'08)*, Newcastle upon Tyne, UK, 22-25 September 2008, pp. 153-166.
- [16] K. Tindell, and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems", *Trans. on Microprocessing & Microprogramming*, vol. 40, Issue 2-3, April 1994, pp. 117-134.
- [17] G. Cena, and A. Valenzano, "Performance Analysis of Byteflight Networks", *Proc. of 5th IEEE international Workshop Factory Communication Systems (WFCS'04)*, Vienna, Austria, 22-24 September 2004, pp. 157-166.
- [18] V. Izosimov, P. Pop, P. Eles, and Z. Peng, "Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems", *Proc. of Design, Automation and Test in Europe Conference and Exhibition 2005 (DATE'05)*, vol. 2, Munich, Germany, 7-11 March 2005, pp. 864-869.
- [19] E. Armengaud, F. Rothensteiner, A. Steininger, and M. Horauer, "A Method for Bit Level Test and Diagnosis of Communication Services", *Proc. of 8th International IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS'05)*, Sopron, Hungary, 13-16 April 2005, pp. 69-74.
- [20] FlexRay Communications System - Protocol Conformance Test Specification V2.1, www.flexray.com
- [21] H.R. Zarandi, S.G. Miremadi, and A. Ejlali, "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, 3-5 November 2003, pp. 485-492.