Dependability assessment of by-wire control systems using fault injection

S. Blanc, A. Bonastre, and P. J. Gil

Fault Tolerant Systems Group

Department of Computer Systems and Computation

Universidad Politécnica de Valencia (Spain)

e-mail: {sablacla, bonastre, pgil}@disca.upv.es

Corresponding author:

Dr. Sara Blanc;

email: sablacla@disca.upv.es;

Telephone number: 00 34 96 387 70 00 ext. 75722

Fax number: 00 34 96 387 75 79

Correspondence address: Higher Technical School of Applied Computer Science (Escuela técnica superior de informática aplicada)

Universidad Politécnica de Valencia

Camino de Vera s/n

Valencia 46022

Spain

Abstract

This paper is focused on the validation by means of physical fault injection at pin level of a time-triggered communication controller: the TTP/C versions C1 and C2. The controller is a commercial off-the-shelf product used in the design of by-wire systems. Drive-by-wire and fly-by-wire active safety controls aim to prevent accidents. They are considered to be of critical importance because a serious situation may directly affect user safety. Therefore, dependability assessment is vital in their design.

This work was funded by the European project 'Fault Injection for TTA' and it is divided into two parts. In the first part, there is a verification of the dependability specifications of the TTP communication protocol, based on TTA, in the presence of faults directly induced in communication lines. The second part contains a validation and improvement proposal for the architecture in case of data errors. Such errors are due to faults that occurred during writing (or reading) actions on memory or during data storage.

Keywords: Dependability, TTA Architectures, By-wire systems, TTP, Fault injection

Dependability assessment of by-wire control systems using fault injection

S. Blanc, A. Bonastre, and P. J. Gil
Fault Tolerant Systems Group
Department of Computer Systems and Computation
Universidad Politécnica de Valencia (Spain)
e-mail: {sablacla, bonastre, pgil}@disca.upv.es

Abstract

This paper is focused on the validation by means of physical fault injection at pin level of a time-triggered communication controller: the TTP/C versions C1 and C2. The controller is a commercial off-the-shelf product used in the design of by-wire systems. Drive-by-wire and fly-by-wire active safety controls aim to prevent accidents. They are considered to be of critical importance because a serious situation may directly affect user safety. Therefore, dependability assessment is vital in their design.
This work was funded by the European project 'Fault Injection for TTA' and it is divided into two parts. In the first part, there is a verification of the dependability specifications of the TTP communication protocol, based on TTA, in the presence of faults directly induced in communication lines. The second part contains a validation and improvement proposal for the architecture in case of data errors. Such errors are due to faults that occurred during writing (or reading) actions on memory or during data storage.

Keywords: Dependability, TTA Architectures, By-wire systems, TTP, Fault injection

1. Introduction

Embedded systems are becoming increasingly complex. Their technological advance has allowed results that were inconceivable some years ago. This is the case of the 'by-wire' concept in the automotive industry – with a constant increase in the number of electronic control units (ECUs) with each new car model. The number of ECUs is not negligible and they are normally distributed throughout the vehicle and share a required fail-safe communication network.
Dependability assessment is vital in the design of by-wire active safety controls for preventing accidents. Both reliability and safety are critical attributes of dependability in these control systems because a serious situation may directly affect user safety. Reliability concerns the

continuity of a correct service during a time interval, while safety is the probability that a catastrophic failure does not happen.

An architecture considered as dependable will give the system the ability to detect errors and recover in time to continue offering a correct service. An example is the Time-Triggered Architecture (TTA) [1] that places reliability and safety as the most important issues. However, although the TTA provides a worthy structure in the development of by-wire systems, the fulfilment of the expectations concerning reliability and safety mainly depend upon communication protocols. There are several protocols based on TTA such as: FlexRay [2]; TTCAN [3]; FTT-CAN [4]; Time-Triggered Ethernet [5]; or Time-Triggered Protocol (TTP) [6]. TTP was originally focused on backbone communication buses for automotive systems and is now open to the high safety standards required in the aerospace industry.

TTP has been comprehensively verified by formal methods [6] and surpasses the other protocols because it uses fault injection techniques to achieve a precise dependability validation. Two European Commission projects: PDCS (‘Predictably Dependable Computing Systems’ 1992-1993); and FIT (‘Fault Injection for TTA’ 2000-2002)[1] funded research into fault injection. The FIT project is the most recent project and presented tools with improved technology for validating distributed systems. It helped to improve the TTP-C1 and the TTP-C2 communication-controller chips with cost-optimal safe solutions. The TTP/C chips are commercial off-the-shelf products, hence the importance of their validation. Techniques used during the project were: *Software Implemented Fault Injection* by the Vienna University of Technology; *Heavy Ion Radiation* by Chalmers University in Sweden; *C-Sim Simulation* by the Czech Technical University in Prague and the University of West Bohemia; *VHDL-based Fault Injection* and *Physical Fault Injection at Pin-level*, both by the Universidad Politécnica de Valencia in Spain. All the techniques have been demonstrated to be compatible and sufficiently enriching to merit individual study.

This paper is focused on fault injection at pin-level. This was the only technique that enabled the direct injection of faults without overhead. Injections are made on buses or control lines, emulating a wide variety of real faults that can occur inside the chip as, well as in the pins and control lines themselves. The technique is applied to a real prototype of the system, so enabling observation of the reaction of the communications protocol, as well as observation of the goodness of the supporting physical layer.

This paper is organized as follows: Section 2 briefly describes the target of our experiments. Section 3 reviews the most frequent causes of physical faults and how they can be modelled in an experimental context. This section also gives the details of the pin-level fault injection, as well as a description of the tool used in the experiments. Section 4 describes the experiments, Sections 5 and 6 offer an in-depth analysis of the experimental results. Finally, section 7 summarizes the paper.

2. TTP/C: target of the experiments

A by-wire control system includes many components – ranging from basic input-output devices to computational electronic units, plus the communication network and the protocol used to enable sharing: all of which are part of the architecture [7]. Both reliability and safety are critical in most of by-wire applications (steer-by-wire, brake-by-wire, etc.) and consequently, require a very dependable architecture. Architectures for distributed systems can be divided into Event-Triggered Architectures and Time-Triggered Architectures. Generally speaking, an Event-Triggered Architecture is more flexible than a Time-Triggered Architecture (TTA). However, TTA places reliability and safety before flexibility. TTA [1] always guarantees a consistent state, even in the presence of faults. Besides, TTA assures known latency times in the detection of communication errors. However, although the theoretical description of TTA presents such commendable characteristics, the error detection coverage and continuity of service depend on the communication protocol. Therefore, differences exist among the time-triggered protocols with regard to implemented (or non-implemented) services.

2.1. The Time-Triggered Protocol

The Time-Triggered Protocol (TTP) [6] is oriented to systems of n units (TTP nodes) synchronized and interconnected using two replicated broadcast channels. The TTP nodes access the replicated channels according to a Time Division Multiple Access (TDMA) scheme. Every node in the system has a unique time slot for transmitting (transmission window) which is specified in a static data structure of the TTP/C controller chip.

Figure 1 shows an example of transmission. There are four interconnected TTP nodes (A, B, C and D). The space between consecutive transmission windows is the Inter Frame Gap, used for internal updates. The figure shows two rounds of transmission: TDMA $_i$ and TDMA $_{i+1}$.

In TDMA, the sequence of transmission is fixed. After transmitting node D in window $_j$, node C will follow in window $_{j+1}$. Node C then succeeds node D. Node B is the successor of the successor of node D because it transmits in window $_{j+2}$. Finally, node A is the successor of node B because it transmits in window $_{j+3}$.

Frames can be transmitted redundantly by both channels - window $_j$, TDMA $_i$ - or not - window $_j$, TDMA $_{i+1}$ – but fault tolerance is not provided for non-redundant transmissions. However, a node must always transmit a frame using both channels during its window. In case of 'nothing to transmit', the node sends a special frame containing its C-state (or internal state consisting of

[1] 'Fault Injection for TTA': IST-1999-10748 FIT

several parameters). The C-state is always transmitted explicitly or implicitly with data.

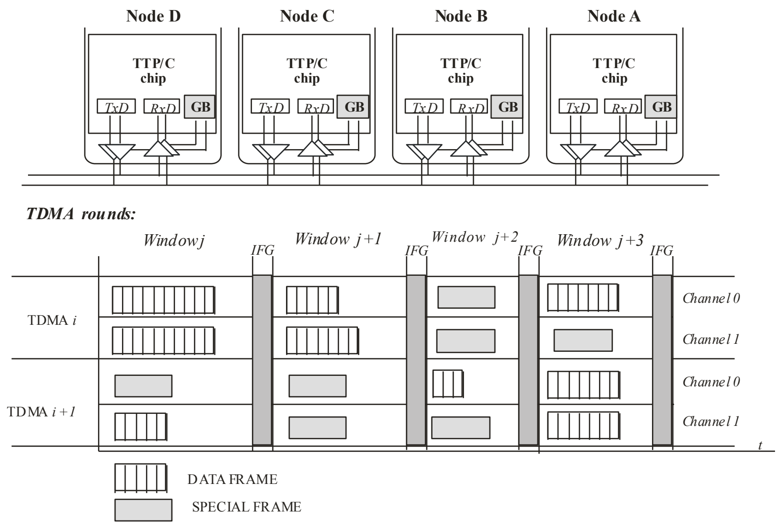

Figure 1. TDMA example

To maintain the TDMA scheme, TTP implements:

- A Global Time Base that allows the nodes to maintain their local clocks as synchronised as possible.
- A TDMA scheme known a-priori by all the TTP nodes. If a node does not transmit during its window because of an error, its partners will realize the problem during that same window. The scheme is stored in nodes as the Message Description List (MEDL).
- There are services implemented at communication layer that guarantee the consistency of the system: after a node sends a frame it needs a mutual confirmation of reception and a global acceptance of this confirmation. In case of disagreement, the majority group with the same knowledge will remain non-faulty and connected, while those that are in minority will assume an error and are disconnected.

Services used as error detection mechanisms are **membership**, **acknowledgement,** and **clique detection**:

- The *membership* service maintains a membership vector with a flag per node. The flag indicates when the node is non-faulty and active or not. The C-state contains the membership vector. When a frame is transmitted, the controller compares its vector with the vector received from the successor. In case of disagreement, it compares both with the successor of the successor. In case of agreement between successors, the node assumes that it has made an error; otherwise, it assumes a successor error.
- When a node transmits a frame, it needs a global *acknowledgement* from at least half of the connected nodes. When a frame is incorrectly received (for example, the CRC does not match), the receiving node sets the node transmitter as non-active in its membership vector. Acknowledgment is then given only if the membership vectors from receiver and

transmitter agree.

- A clique division means that there is not a common knowledge of the system: membership vectors differ among the nodes. *Clique detection* executed individually by each node will avoid this situation if it is assumed that the majority knowledge group always prevails.

2.2. The TTP/C controller

In a TTP system, one TTP node consists of the host processor, the TTP/C controller, and the Communication Network Interface (CNI) [6]. The host processor has three software layers: the time-triggered operating system, the fault-tolerant communication layer (FTcom), and the application layer. The FTcom layer is responsible for reading/writing redundant messages to/from the CNI. It performs the mapping of redundant messages (frames sent redundantly by both channels) on to a single message, thus making the message redundancy transparent to the application. The CNI is a data exchange interface between the host and the communication controller and it is implemented using a dual ported RAM.

The static communication schedule is stored in the MEDL – which stores information about message size, message type, message sending/receiving point in time, and other information related to the communication schedule and configuration of the node. The MEDL is prepared off line and cannot be changed at runtime.

Several versions such as TTP/C-C1 and TTP/C-C2 controllers deploy an additional unit called Bus Guardian (BG). The BG enables and disables the transmission accesses to the bus. The BG prevents a faulty node from sending a message outside its transmission window, avoiding collisions in the bus.

TTP/C-C1 is a 16-bit controller organized around the Protocol Control Unit (PCU). The PCU executes high level protocol mechanisms such as message redundancy, membership, etc. – and controls the interaction of some internal blocks such as the Time Control Unit (TCU) and the CRC Unit. The TCU maintains the Global Time Base algorithms, and the CRC unit supports the calculation of cyclic redundancy check sums. The CRC unit allows the concurrent calculation of two checksums, one for each channel. The frame data is added to the check sum word or byte wise.

The main changes in the TTP/C-C2 version are the extended length of the frames, the MEDL on chip, and the addition of some new blocks.

3. Validation by Fault Injection

To carry out a consistent experimental validation with regard to physical faults, we need previously to analyze which are their sources in current sub-micron technologies.

Firstly, this section summarizes the most frequent causes of physical faults and how they can be experimentally modelled. Secondly, the details for the pin-level fault injection are summed up, as well as the description of the tool used in the experiments.

3.1. Physical faults in current semiconductors

Most electronic devices are designed taking into account the possibility of physical disturbances due to electromagnetic interferences (EMI), power variations, radiation, and other possible causes. Moreover, higher operational frequencies and integration densities combined with the lower power voltages of technologically advanced semiconductor devices make them more susceptible to the effects of neutron and alpha particles [8] [9] [10]. The sources of radiation are cleverly and concisely described in [11]. For example, the U-238 and Th-232 radioactive isotopes, being natural alpha emitting particles in semiconductors, are the cause of soft errors in the form of transient bit-flips in memory. The special case of Pb-210 must be taken into account as a source of alpha particles from the solder bumps in flip chip packages, but only for a limited period of time. Some soft errors, and less probable hard errors [12], are also caused by cosmic radiation. A relevant phenomenon is the induction of secondary particles; the nature, energy, and direction of which could result in complex chain reactions [13] [14], as for example, the case of the B-10 isotope [15]. Due to a neutron incidence, this isotope is capable of fission into a Li-7 recoil nucleus, a gamma photon, and an alpha particle.

The number of transistors belonging to different memory cells which are upset by a single incidence increases with integration density. Besides the possibility of multiple incidences, high energy ions can induce multiple bit upsets if crossing through sensitive adjacent regions – either due to a single ion track or secondary particles caused by an ion collision [16] [17].

Apart from radiation, the environmental working conditions of some critical systems, and the resulting degradation as a result of device wear are also sources of physical faults [18] [19].

A challenge for experimental validation is the modelling of harmful environmental effects (crosstalk, junction defects, bad connectors, coupling, etc.), degradation problems, as well as radiation. In this field, a recent study regarding fault models can be found in the documentation of the European funded project DBench (‘Dependability Benchmarking’ 2001-2004) [20] and [21]. Both permanent and intermittent faults, as well as transient faults, are modelled at the Logic and Register Transfer (RT) levels. The study traces the representation of the electrical stress, hot e-trapping, ionic contamination, electromigration, thin-oxide breakdown, radiation or temperature variation effects in the form of permanent and intermittent faults modelled as shorts, open-lines, stuck-at, stuck-open, indetermination in voltage values, and delays in transmitted signals. Causes of transient faults are radiation, interconnection shrinking at high frequencies, transients in power lines, crosstalk, or temperature variation.

In addition to this study, there are several relevant considerations concerning solder joints and metallizations:

- The risk of shorts due to undesirable whiskers produced by electromigration and thermal coefficient mismatches in junction elements increases with new packages and deep submicron circuits that feature smaller pad distances and internal metallizations.
- Excessive intermetallic compound formation or silicon cratering [22], and mechanical stress are manufacturing defects which produce coarse junctions.

High current density and junction temperatures, as well as moisture and ionic contamination, are factors which speed up solder fatigue and metal migration and result in microcracks and open-lines – regardless of whether manufacturing defects are present. Microcracks increase the resistance of a junction in a signal transmission and would be likely to cause a delayed reception. The problem is significant at high frequencies where wire distance shrinkage worsens the coupling capacities. The variation of resistance and capacity at high frequencies could result in signal attenuation at indeterminate voltage values.
With improved understanding of the causes of faults, and the methods in which these can be modelled in an experimental environment, the next step is to look for techniques and tools which enable us to work with these models.

3.2. Fault injection at pin level

There are three general categories of fault injection (FI): (i) Simulated FI; (ii) Software Implemented FI (SWIFI); and (iii) Hardware Implemented FI (HWIFI).

- *Simulated FI* mainly refers to tools that model the system using high level languages. Hardware description languages such as Verilog or VHDL are currently often used. An example is VHDL-based fault injection [23], a powerful technique with a wide range of fault models.
- *SWIFI* consists in reproducing, at software-accessible RT level, errors caused by physical faults in the hardware or by hidden bugs in the software. It provides a relatively cost effective and simple methodology.
- *HWIFI* tools disturb the electrical properties of hardware components, and indirectly affecting the software components as well. Some examples are heavy ion radiation, laser radiation, electromagnetic interferences, and pin-level FI. An advantage is the application of the technique on a real system prototype. It enables us to observe both the reaction of the software and the goodness of the hardware.

Two HWIFI tools were applied in the FIT project: heavy ion FI, and FI at pin level. Together they provide the effect of faults injected with laser radiation or electromagnetic interferences. FI at pin-level is a technique based on the notion of perturbing integrated circuits: usually by means of shorts on the pins or by sticking them at a predefined value. Pins belong to address or data buses, control lines, clock or oscillator inputs, peripheral inputs or outputs, etc. The technique emulates both external and internal physical faults [24].

Location, duration, and persistence (transient, intermittent, or permanent) of the injections are easily controlled in FI at pin-level. Faults can be injected in one or several points at the same time (reproducing common mode faults) or correlated in a short period of time. There is no overhead but a degree of intrusiveness is caused by an increment in the capacity of the line.

One restriction of FI at pin-level is chip accessibility. However, when this technique is used in the validation of a communication protocol (more specifically in the validation of a communication controller chip) this restriction becomes less effective if fault injections are focused on the communication lines. For example, Figure 2 shows injections in the TTP/C1 chip at pin-level: (i) lines that connect the communication units with the 82C250 CAN driver (in the physical layer for TTP/C1 chip) or the MAX3485 driver (for TTP/C2); and (ii) the CNI host memory busses.

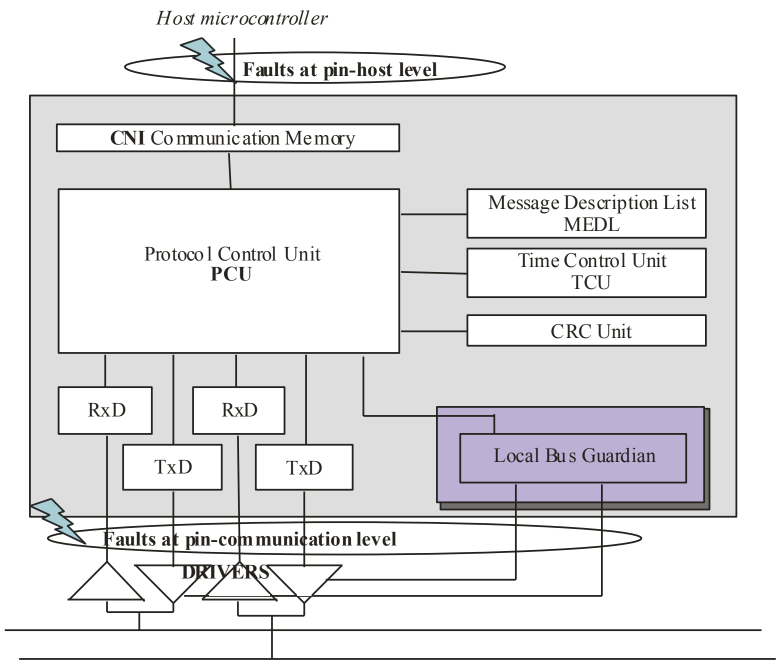

Figure 2. Location of faults at pin-level in the TTP/C1 chip

Location of faults injected on the TTP/C chips:

Faults at communication-pin level on the TTP/C1 chip are located a: (1) the Bus-Guardian line (BG) that enables transmission only during the transmission window assigned to the node, (2) the Output-Enable line (OE) which enables the drivers, (3) the Clear-To-Send line (CTS) which sets the transmission init; and (4) both the Transmission and Reception TTL lines (TxD and RxD). Lines controlled by the Bus Guardian in the TTP/C2 chip are not accessible, and injections are only located on the TxD and RxD and the Reset line.

Faults at host-pin CNI level are located on the Chip Enable line (CE), the Write Enable line (WE), the Output Enable line (OE), as well as the address and data buses.

Faults injected at pin level cause three possible types of alteration in the signal: the generation of non-existing pulses, the removal of the signal, or the variation of a pulse shape.

Depending on the location, persistence of the injected fault and type of alteration observed in the signal fault models reached by FI at pin level are shown in Tables 1 and 2.

Table 1. FI at pin-communication level: Fault Models

Location	***Persistence***	***Type of alteration***	***Fault Model***
BG, OE	Transient, Intermittent	Non-existing pulses	Pulses
TxD	Transient	Non-existing pulses	Bit-flips
Any line	Permanent	Signal removal	Stuck-at
TxD, RxD	Transient	Signal Shape Variation	Delay, Indeterminate voltage at high frequency
Any control line	Transient	Signal Shape Variation	Delay

Table 2. FI at pin-host CNI level: Fault Models

Location	***Persistence***	***Type of alteration***	***Fault Model***
CNI buses	Transient	Non-existing pulses	Bit-flips
CNI WE, CE, OE	Transient	Signal removal	Transient faults at RT level
Any line	Permanent	Signal removal	Stuck-at
Any control line	Transient	Signal Shape Variation	Delay

For example, Figure 3 (A) shows two examples of non-existing pulses generated in a control signal. Figure 3 (B) shows the effect of a stuck-at-0 in a memory bus line.

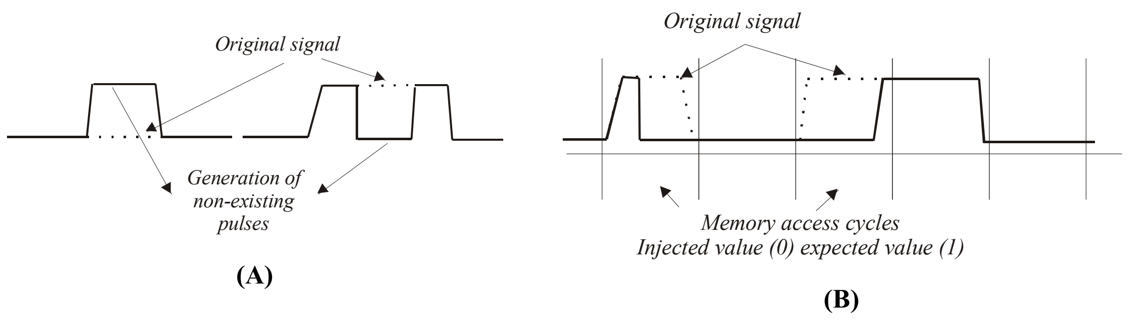
Original signal
Generation of non-existing pulses
(A)
Original signal
Memory access cycles
Injected value (0) expected value (1)
(B)

Figure 3. Generation of non-existing pulses

It depicts a transient fault injected over a short time, but long enough to modify the original signal in consecutive memory accesses, thereby distorting the information (bit-flips) of several words. Likewise, a transient on the TxD line can flip one or several bits that belong to the transmitted frame.

Besides, any pin permanently stuck-at-0 or at-1 will remove the original signal maintaining the line at the injected value. But if it is a transient fault, the pin-level fault injection emulates those transient faults at RT level that may cause an execution error.
Finally, the variation of a pulse shape is focused on yielding violations of set-up and hold times.

3.3. The injection tool: AFIT

Three processes are involved in an experimental validation: fault injection, data acquisition, and the analysis of results [25].

Fault injection: In the FIT project, the injection process at pin level was in charge of AFIT: *Advanced Physical Fault Injection Tool* at pin level. AFIT was developed at the Universidad Politécnica de Valencia [30] with several updates [24]. The tool is completely external and, it is not necessary to halt or delay the target execution. AFIT runs automatically, enabling many experiments to be conducted without supervision. Faults are injected by sticking the lines at-0 or at-1 logic values.

Data acquisition: The last version of AFIT was designed for distributed systems [25]. This version incorporates a monitor as acquisition tool – combined with software tasks running in the host processors of the TTP nodes. The tasks obtain read-outs of the detected errors and failures. Failures can appear in either the injected node or another node connected to the network, which is known as 'error propagation'. Injection and acquisition are synchronized. Each experiment lasts from the injection to the last reaction of the system (i.e. fault effectiveness, error activation, error detection, error treatment including diagnostics, isolation and recovery, error propagation and failure). ***Fault effectiveness*** has a special importance in physical fault injection. It determines whether the fault has caused a real perturbation. The injected fault is not always effective, for example, if we stuck-at-1 a line whose logic value was already 1 the fault will be ineffective and the experiment is useless.
The collection and structured storage of system events in the presence of a fault, together with required measured latencies from the error activation to its detection or failure, are all parts of the acquisition process.

Analysis of results: It would be desirable to have an automatic processing of the 'injection-acquisition' that takes up the smallest possible period of time per experiment (i.e. up to one second in the TTP/C experiments). Post-processing these results, which is usually a time consuming task, would then be accepted if data storage is well conducted during the campaign and generate databases which contain all the required information for tracing back an observed event. An analysis of results may reveal defects in the injection and data acquisition processes that could be rectified in a new injection campaign (feedback).

4. Experimental description

The experimental system was a cluster of four TTP nodes with two configurations:

A) TTP nodes with a TTP/CTM-C1 controller and a Motorola MC68360 used as host.

B) TTP nodes with a TTP/CTM-C2 and a Motorola power PC PPC555.

The workload was a brake-by-wire (BBW) simulation program designed by Volvo Technological Development. It was a distributed control system that required four TTP nodes:

- The pedal brake: angled to calculate the brake force.
- The vehicle: calculates the speed of the vehicle body in m/s, the speed of the four wheels, and the distance covered by the vehicle. It uses two parameters, the *acceleration torque* on one wheel, and the *frictional force* on one side of the wheel brake disc. The vehicle sends the speed of the vehicle body and the speed of the front left wheel.
- The front left wheel: implements the ABS control algorithm – requiring as inputs the speed of the front left wheel in *rad/s* during the last cycle, the speed of the vehicle body in *m/s,* and the pedal force. As outputs, the node sends the pressure applied to one wheel brake pad's piston and the new speed of the front left wheel, and resends the received data from the vehicle node.
- The replica of the left wheel. The injection takes place in the front left wheel node. The replica runs free of faults. Messages transmitted by the replica are compared with those transmitted by the 'front left wheel' node in order to detect communication failures or errors propagated in the message data.

Experiments are divided into two groups: (i) at the communication-pin level and, (ii) at the host-pin CNI level.

<u>Experiments at pin-communication level</u>

In general, failures at the communication level are grouped under a limited number of modes. For example, the loss of connection either due to a crash in a unit or a communication-link-fault, both of which cause the unit to omit messages; babbling-idiot transmissions including commission failures [26] and physical disturbances; the Slightly-Off-Specifications (SOS) problems [27] [28]; Byzantine messages [29], and masquerading failures that occur when a unit assumes the identity of another unit and damages the system. In safety-critical systems, the communications protocol must guarantee a dependable behaviour despite failure or shutdown and reinstatement actions.

The TTP detection mechanisms used by TTP to ensure consistency at the communication layer are membership, acknowledgment, and clique detection. Experiments evaluate their effectiveness in the presence of faults shown in Table 1.

<u>Experiments at pin-host CNI level</u>

The distribution of control algorithms in by-wire applications involves exposing the data values to physical faults during a long period. TTP guarantees the consistency of the system and the continuity of service at the communication layer. However, physical faults can also appear at the application level or during the transference of data between the application and the TTP/C chip.

Because dependability should be achieved at every level in the architecture, the experiments carried out in this section try to identify error patterns caused in the presence of faults in Table 2. It also analyses error detection coverage with Error Detection Codes.

5. Results at communication-pin level

The TTA bus topology admits a network with channel redundancy. Therefore, TTP uses two channels. It sends a copy of the frame can be sent through each of the channels. If one copy does not arrive or is disrupted, the receiver will use the second copy – as both arrive at the same time.

In a sample of 3000 injection experiments on BG, OE, CTS, TxD and RxD (see section 3.2), TTP correctly accomplished the ‘single fault’ hypothesis. In other words, the protocol maintains its consistency in the presence of any single fault at pin level.

But, double faults were also injected: one fault per channel simultaneously.

5.1. Control lines

Table 3 shows the failures that FI at pin level can explicitly induce. Faults are located in BG and OE lines.

Table 3. Emulated failures with faults at BG and OE lines

Description of the emulated failure	**Signal alteration**	**Synchronism**	**Persistence and multiplicity**	**TTP Activated mechanisms**
Message omission	Signal removal	Frame transmission	Double transient faults	Membership
Halts an on-going transmission	Signal shape variation	Frame transmission	Double transient faults	Membership
Babbling-idiot spurious pulses	Non-existing pulses	Frame reception	Single transient faults	Clique avoidance Membership

Failures occurring during the frame transmission were solved with the TTP membership service. Because frames were not received or not correctly received by the receivers (and all nodes are receivers in TTP), they will set the transmitter as faulty and outside the membership vector.

Since the membership vector is always sent as part of the message, the transmitter only needs to compare its own vector with its successors to recognise the failure.
A special case of failures are the babbling-idiot spurious pulses. These pulses emulate a faulty node that tries to transmit outside its transmission window. In the experiments, faults were synchronised with a frame reception. Ten out of every 500 single injections resulted in common-mode faults: frames received by both channels became disrupted. In these cases, failure should be detected by the clique avoidance algorithm. The faulty receiver sees the frames as invalid, but failed to coincide with the majority (it only needed to compare the membership vectors). However, clique avoidance did not detect the failure if the fault was injected during a reception just before its own transmission. The faulty node transmits and does not recognise the failure until the execution of the membership service. Therefore, both mechanisms are needed to achieve 100% detection coverage.

5.2. TxD and RxD lines

FI at pin level can change correct frames into syntactically or semantically incorrect frames. Faults are overlapped with the transmission of real frames, emulating internal errors to the controller, and defects in the communication channel or in the physical layer. Faults frequently cause syntactically incorrect frames. Only rarely (1 in 1000 instances) does the frame remain syntactically correct. Mechanisms used to detect these erroneous frames are CRC and message replication.
FI at pin level can speed up the validation in the case of physical faults which cause errors in both channels. When injecting faults in either TxD signals or RxD signals, Slightly-Off-Specifications failures appear. SOS indicates that the effect of the fault is marginal: it is detected as a fault by some components but accepted as a non-fault by others [28].
An SOS failure is not necessarily produced by a malicious or ‘absent-minded’ node. Its origin is diverse, and ranges from an incoherency between system design and system integration to architectural weaknesses. Coherency will be achieved by replacing or redesigning some components. However, architectural weaknesses require conceptual change. For example, TTP/C developers have appealed for a change in the network topology towards a central star unit.
SOS failures in a bus topology are divided into: SOS in time domain and SOS in value domain.
SOS in time domain:

> In a TTP system, nodes have slight differences in their reception windows, i.e. because the precision of the oscillators drift away from each other.
> There are several parameters in a TTP window. For example, in Figure 4 (in TTP/C specifications), a sender opens its window at $t_{AT\ s}$ but starts the transmission at time

$t_{AT'\,s}$: it waits for any receiver to open its window. $\Delta_{rw\ r}$ defines the limits of valid receptions. In the middle, the frame is expected at $t_{AT'\,r}$, while $t_{reception\ r}$ indicates the real reception init.

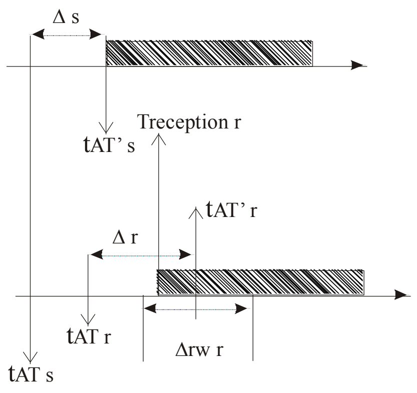

Figure 4. Some parameters in a TTP frame transmission

Spurious short pulses between $t_{AT\ s}$ and $\Delta_{rw\ r}$ left limit, can generate SOS failures in the time domain.

Differences in the window openings divide the cluster into: (1) nodes that receive the fault and the real frame, and (2) nodes that only receive the real frame.

TTP avoids clique formation by comparing the number of valid received frames with the number of invalid frames. Before transmission, valid frames should exceed the number of invalid frames and silences (non-receptions during a window).

In the experiments, it was observed that nodes in case (1) in TTP/C-C1 (version 0.1) increased the invalid frames counter by two (the fault and the frame). Subsequently, in a four TTP node system it was possible to observe the consecutive shutdown of the four nodes due to the fact that none were able to count more valid than invalid frames. Version 1.1 in TTP/C-C2 changes the concept of the valid frames counter into an agreed slots counter. By limiting the number of potential invalid receptions to one per window, shutdowns can be avoided and the protocol can be made more reliable.

<u>SOS in value domain:</u>

SOS failures in value domain are those generated by a marginal incorrect encoding or by an indeterminate value, for example, a signal attenuated below the established threshold.

The experiments emulate SOS failures with double faults. For example, the RxD lines of Nodes 1 and 2 are stuck-at-0 while Node 3 receives the signal free of faults. Faults are aimed to cause discrepancies and the behaviour of the protocol is observed.

In the experiments, it was seen that the TTP always avoids discrepancy by means of a Never Give Up strategy [6] that forces a clean restart and recovery. Such a strategy enables the system to work degradedly, maintaining minimumal functionality which reduces the system recovery latency.

The SOS experiments were carried out with AFIT and reproduced positively with VFIT [33]. It produces more reliable results.

5.3. Reinstatement actions

After a node or nodes recover their functionality, they may perform reinstatement actions. In the context of validation, it is important to verify that reinstatement actions are safe and do not involve any correlated failure.

We observed a new sample of 5000 experiments on the reset line. The reinstatement of one or more nodes did not result in any frame collision. Nevertheless, only 4 out of every 1500 achieved reinstatement in just one TDMA. A faster reinstatement is a difficult challenge for any communication protocol. For this reason, it is recommended to replicate nodes with critical functions in the system.

6. Results at host-pin CNI level

When variables calculated by the application are available (for example, the wheel speed) the host writes these in a dedicated part of the CNI memory: the message memory. Message memory writing (or reading) is susceptible to electromagnetic interferences, radiation, solder joint degradation, etc. Likewise, data stored in the message memory can be exposed to single event upsets (i.e. radiation) that cause bit-flip errors.

In TTP network transmission, frames are protected with a Cyclic Redundant Check (CRC) code. However, before the CRC calculation, data is not protected either during writing (or reading) actions on the message memory, or during data storage in the memory. Moreover, multiple faults are possible, and these would be correlated but not necessarily common-mode faults.

6.1. Errors patterns

There are different references regarding the reality of error multiplicity on data [16], [17], [25] and [31]. Bit-flips are the consequence of such faults, and depending on the source of the fault, they appear following different patterns. In general, it is possible to observe unidirectional or random errors.

Unidirectional errors:

> Unidirectional errors are caused by stuck-at lines. For example, in a serial transmission, a line stuck-at-0 changes adjacent bits in the sequence into '0'. In a parallel transmission, bits stuck at 0 are those transmitted through the same bus line, belonging to different words but adjacent in the transmission sequence. Figure 5 shows an example of unidirectional errors in a 16-bit parallel bus transmission. Each **vertical combination**

represents a bus line and each **horizontal combination** represents a word read or written on memory in consecutive cycles. The left-hand side of Figure 5 is free of faults, while the right-hand side contains 2 bit-flips due to a transient fault in line 6.

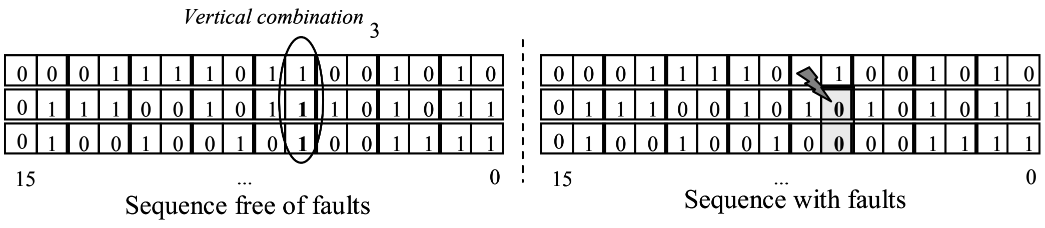

Figure 5. Unidirectional errors

Random errors:

Random means that bits do not necessarily change into the same value '0' or '1':

- Shorts or open lines cause changes in adjacent bits but not to a fixed value.
- Memory or register bit-flips.
- Ineffective attempt: Transient faults during accesses to memory have a negative impact on the logic. Figure 6 shows an example where the host does not manage to update all the selected message memory addresses [25].

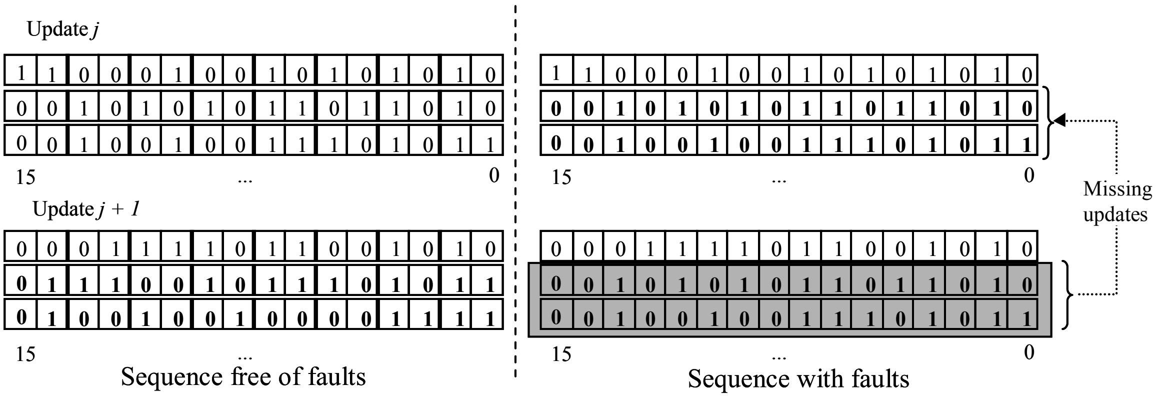

Figure 6. Ineffective writing attempt.

Similar effects can be emulated by changing specific control registers [32].

Furthermore, if the fault alters the destination memory address, it has a double effect. Firstly, data is not updated in the expected memory addresses. Secondly, data overwrites non-expected memory addresses.

6.2. Detection coverage without Error Detection Codes

The detection of bit-flips during writing, reading, or storage in message memory is outside the scope of the TTP, but not outside the scope of TTA architecture – which must guarantee system dependability at every level. Because these errors occur prior to frame formation, the CRC field calculated in TTP to protect the frame is not enough. End-to-end error detection mechanisms at the application level could protect data held in memory and waiting to be transmitted (or after reception).

In the experiments, using a synthetic application, we observed several events caused by FI at pin-level when no end-to-end mechanism is implemented:

- ***Host Exception*** **(HostE)**: A bus error is detected by the host.
- ***Operating System Exception*** **(OSE)**: The task in execution did not finish before the assigned deadline.
- ***Sender Status Flag*** **(SSF):** The sender status flag of a word is received as false. The status flag indicates if the word has been updated in the last TDMA round. It is a very simple end-to-end[2] mechanism that sends an additional word in which one bit is assigned to each word indicating its status [6].
- ***Unidirectional Error*** **(UE)**: At least one word of the faulty node differs in *i* bits from the received word of the golden node (the wheel replica node). The variable *i* depends on the number of faulty lines.
- ***Ineffective Write Attempt*** **(IWA):** At least one word was not effectively updated (see Figure 6).
- **HostE + UE or OSE + UE:** Despite the exception, an incorrect data block is sent containing a unidirectional error.
- **HostE + SSF or OSE + SSF**: A missing update is detected and an exception occurs simultaneously.
- **HostE + IWA or OSE + IWA**: The missing update remains undetected and the incorrect data block is sent. The fault causes an exception as a secondary effect.

Table 4 shows the percentages obtained in a sample of 3000 experiments. 1500 faults stuck-at-0 one line with a 76.39% of effectiveness, and 1500 faults stuck-at-1 one line with a 94.95% of effectiveness. Faults were injected on data bus lines with duration uniformly distributed in [500ns…5μs].

Table 4. Percentage of events caused by faults at CNI pin level

	HostE				OSE			SSF	UE	IWA
Type: Errors/Faults	HostE	HostE+UE	HostE+SSF	HostE+IWA	OSE	OSE+UE	OSE+SSF			
Stuck-at-0: 76.39%	19.28%	**1.54%**	1.54%	**0.0%**	0.0%	**0.0%**	0.0%	0.37%	**37.10%**	**16.57%**
Stuck-at-1: 94.95%	49.25%	**0.0%**	15.54%	**0.53%**	16.37%	**0.53%**	1.73%	0.0%	**11.01%**	**0.0%**

There is a 55.21% rate of undetected errors with stuck-at-0 faults and a 12.07% rate with stuck-at-1 faults. Error Detection Codes (EDC) on data would improve the error detection coverage. The following section analyses some proposals.

6.3. Improving the detection coverage

A fault at pin-level on the TTP/C sticks a line during one or several CNI memory accesses. The reason must be found in the memory access model of the TTP/C controller. Memory accesses are also Time-Triggered and in predefined points of time, the host processor reads or writes a block of words. The size of the block depends upon the application. Because the words of a block are accessed consecutively, the stuck-at line will affect one or several words causing unidirectional or random errors. This relation is shown in Figure 7. For example, a fault in the range of [2µs…<2.7µs] causes up to two unidirectional errors or four ineffective updates, while a fault in the range of [4.3µs…<5µs] causes up to three unidirectional errors or seven ineffective updates.

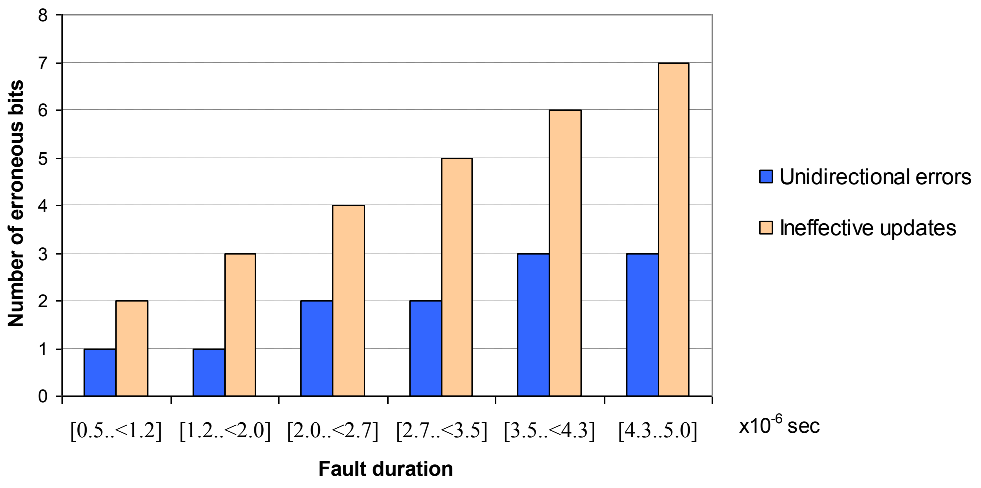

Figure 7. The fault duration determines the number of bit-flips

This section analyses two vertical codes to improve the error detection coverage. They consist in separable codes that add some bits (codeword) to those to be protected (information bits).

6.3.1. CRC per bus line

As well as in serial transmissions, data transferred in parallel can be protected by a CRC assuming a serial alignment of data in the CRC calculus. The number of undetected patterns of errors by a CRC will be 2^{n-k}-1 in 2^n, *n* being the number of transferred bits – and k the generator polynomial grade. Theoretically, CRC coverage is very high. However, from a practical viewpoint, the coverage depends upon where bit-flips in the data are spread. For example, Figure 8 shows several errors undetected by a CRC_{16} (generator polynomial: $x^{16}+x^{12}+x^{11}+x^8+x^5+x^4+1$); bit-flips can be caused by an ineffective update (examples 1, 2, 3, and 4), stuck lines (example 5), upsets due to radiation (example 6), or an upset combined with an ineffective update (example 7).

[2] A mechanism implemented at the sender application level and checked at the receiver application level.

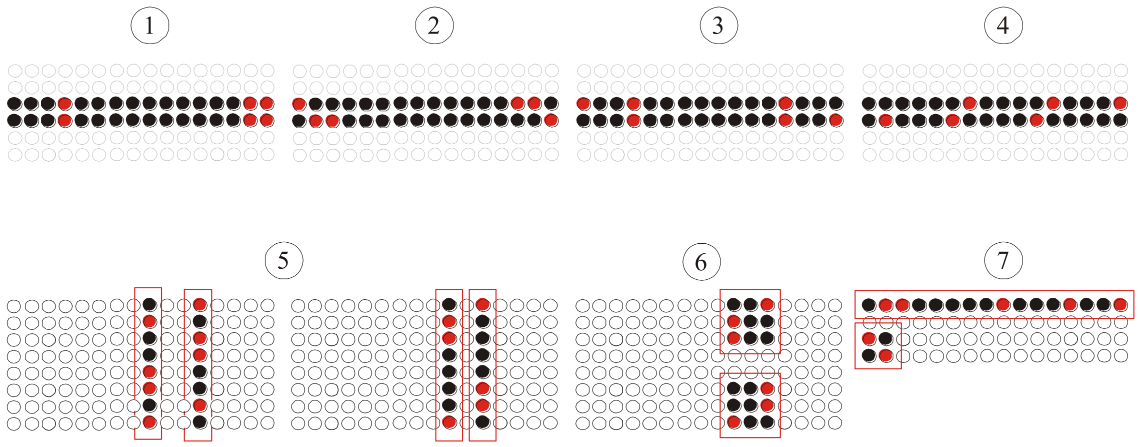

Figure 8. Undetected errors; bits susceptible to flip by the fault effects appear in black; bit-flips in red

Vertical codes will present a good coverage in such cases. With one codeword per bus line (or vertical combination in the data block), the more vertical combinations that contain bit-flips, the better the detection coverage will be. Since each combination is checked independently, the possibility of detecting at least one bit-flip increases. A CRC per bus line combines the high coverage of this EDC with the advantages of vertical codes in a parallel transmission.

Experimentally we used a CRC$_3$ (generator polynomial: x^3+1). The block size is 11 words: 7 words of information plus 4 words for the CRC codewords.

The experimental sample produced 1376 single effective faults and 1411 double effective faults. 1758 messages were transmitted with errors, both UE and IWA.

Figure 9 compares the estimated percentage of uncovered errors with the percentage observed in the sample.

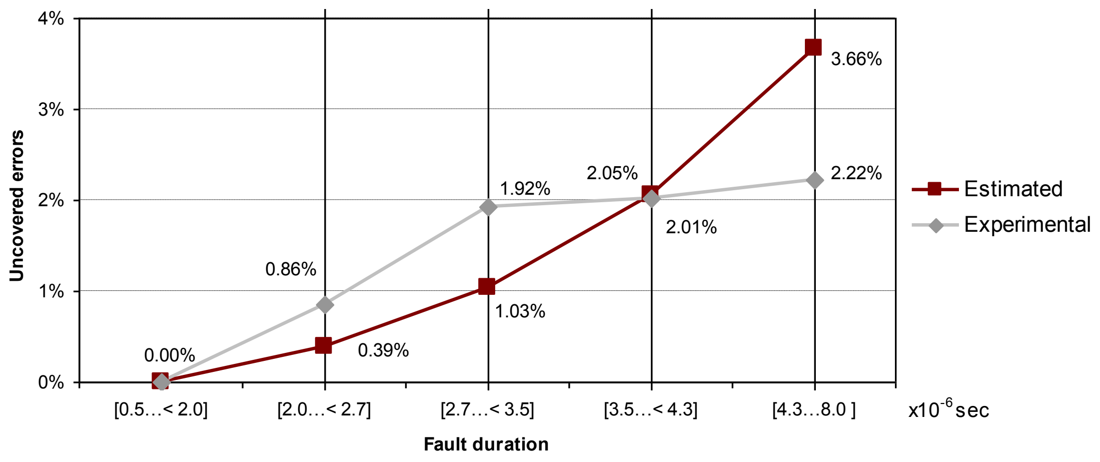

Figure 9. Percentage of uncovered errors: estimated and observed in the sample

The optimistic coverage observed in the sample with faults longer than 4.3µs is due to the low variation in the transferred values: the variables calculated by the control algorithms are normally stabilized at fixed values (although these vary according to how the vehicle is driven).

The estimated coverage includes every undetectable error in 2^n combinations (2^{11} in the example), but many of these undetectable errors never appear in the experiments, and differences are produced between estimated and observed.
However, percentages of uncovered errors, both estimated and experimental, are still high. Detection coverage near of 99% would be advisable.

6.3.2. A vertical parity code

The simplest EDC is a parity code of a bit per memory bus line. However, the effectiveness of the code decreases with the fault duration (Figure 7).
It may be possible to improve our experimental coverage bearing in mind the parity. For example, a Berger code can detect any unidirectional error and many random errors. We presented in [25] a separable error detection vertical code, based on Berger code. The code is focused on covering unidirectional errors but with a good coverage for random errors. It is based on two definitions:

Definition: with *m* being the number of bits to be protected by a codeword, where C is the set of 2^m binary m-tuples. C can be structured as the $\bigcup_{i=0}^{m} C_i$, in such a way that C_i= $\{ c_i^0, c_i^1, \ldots, c_i^{\overline{\frac{m!}{(m-i)!\,i!}}} \}$, denoting *i* the number of 1's in the information bits, where $C_i \cap C_j = \varnothing$, if $i \neq j$.

Definition: with *n* being the number of bits which form the codeword, where U_n denotes the set of 2^n binary n-tuples. Each n-tuple is a codeword. u_k denotes the set of codewords with *k* number of 0's, for all *k* from 0 to *n*. Codewords that belong to U_n can be ordered in a vector Y. For example, Y = $[\, y_0, y_1, \ldots, y_{(2^n-1)}]$. To order the vector, we consider that if y_i belongs to u_k and y_j belongs to u_p where $k \neq p$, then $y_i < y_j$ if it is satisfied that $k<p$. But, if y_i and y_j belong to the same u_k, their relative position within their own group is insignificant. Thus, it is possible to obtain several vectors. From these, the chosen vector must accomplish the following condition in the assignment:

Assignment: *One codeword y_i that belongs to a vector Y is assigned to all combinations included in $C_k \in$C, and k number of 1's in the information bits.*

Condition: *y_i, y_j being two codewords that belong to a Y vector, then $0<i<j<2^n-1$. y_i can be assigned to C_k, with k number of 1's, and y_j can be assigned to C_{k+1} if the minimum Hamming* *distance* $(y_i, y_j) \geq 2$.

<u>Algorithm</u> (*m* information bits, *n* code bits):

1. If $2^n - 1 = m$, go to step 4.

2. If $2^n - 1 > m$, and *m* odd. Then, $2^n-(m+1)$ consecutive codewords have to be removed from the used vector Y taking into account the Condition. Go to step 4.

3. If $2^n - 1 > m$, and *m* even, go to step 7.

4. With Y being the used vector, *m* codewords are needed; the codeword y_0 is assigned to $c_0^0 \in C_0$

5. y_1 is assigned to all combinations included in C_1, y_2 is assigned to all combinations included in C_2, and so on so that y_i is assigned to all combinations included in C_i, $i= 1,...,m-1$.

6. The codeword y_m is assigned to $c_m^0 \in C_m$. Break.

7. Being Y the used vector, $m+1$ codewords are needed, the codeword y_0 is assigned to $c_0^0 \in C_0$

8. There is a C_k for $k=0,...,m$ where the number of 0's in any combination c_k that belong to C_k is equal to the number of 1's. C_k is the biggest group of C and can be divided into two subgroups C_k' and C_k''. Assignments are made following step 5 by taking into account that Condition is a strong requirement. Thus, because of the necessary use of two codewords for C_k' and C_k'', then, with y_i, y_j being two codewords that belong to a vector Y, $0<i<j<2^n-1$, y_i can be assigned to C_k' and y_j can be assigned to C_k'', if the distance $(y_i, y_j)=1$, such that, being y_u the codeword of C_{k-1} and y_v the codeword of C_{k+1}, the minimum distance between either (y_u, y_i), (y_u, y_j), (y_v, y_i), (y_v, y_j), is ≥ 2. $2^n-(m+2)$ codewords of the used vector Y are unused.

9. The codeword y_{m+1} is assigned to $c_m^0 \in C_m$. Break.

Apart from unidirectional bit errors, a codeword covers many random bit-flips. ci $\in$Ci could reach another valid cj $\in$Cj, being $i \neq j$, only if the non-unidirectional inversion of at least three bits occurs. This means that three bits should change to different '0' or '1' values. This is useful because it enables a satisfactory mix of solutions, one of which consists in using the code together with a couple of additional bits to cover any random change in three consecutive bits.

Two additional bits have been added to every y codeword. These are calculated in a similar manner to the CRC_3, but only LSB and (LSB-1) are needed.
The experimental sample produced 1497 single effective faults and 1575 double effective faults. 1969 messages were transmitted with errors, both UE and IWA. Figure 10 shows the improvement in the detection coverage.

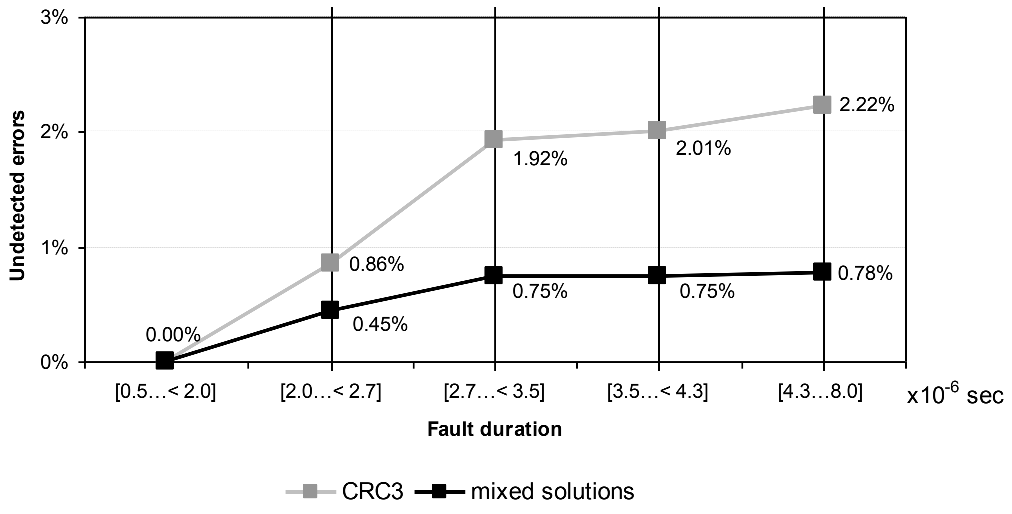

Figure 10. Detection coverage: CRC3 and mixed solutions
Percentages of undetected errors in the graph of mixed solutions are below 1%. This is a good result and means detection coverage is higher than 99%.
However, in general, vertical codes entail a considerable overhead in the message. The final decision must be made by the human integrator. References for making a balanced decision are the criticality of the data and the dedicated field length within the frame.

7. Summary

This paper is part of the work carried out in the European funded project FIT ('Fault Injection for TTA'). The objective was the validation by means of fault injection techniques of a commercial-off-the-shelf product: the Time Triggered Protocol chip, versions TTP/C-C1 and C2, used in the automotive and aerospace industries.
TTP/C chips are used in active safety controls implemented in vehicles. Nowadays, these are implemented under the 'by-wire' concept using specific dependable architectures as the Time-Triggered Architecture (TTA).
Critical distributed systems need to be validated with techniques such as fault injection before

manufacture. This paper is focused on the validation of a TTP/C chip controller by means of fault injection at pin-level. Firstly, the paper traces the representativeness of the injected faults by looking for
their causes in current sub-micron technologies and how they can be emulated in an experimental environment with different fault models. Secondly, experiments are divided into two groups, those oriented towards protocol communication and those oriented towards message data.

Faults at communication-pin level help enable observation of protocol service behaviour in the presence of failures in message omission, on-going transmission halting, babbling-idiot spurious pulses, SOS failures, and reinstatement actions. In the first three cases, an experimental coverage of 100% was obtained. In the case of SOS failures, a defect in the TTP/C-C1 chip controller solved in version 1.1. of TTP/C-C2 chip was observed. Experiments also tested the worthiness of strategies such as Never Give Up or the reinstatement implemented in TTP/C. In general, results highlighted the importance of avoiding the clique formation, as well as the improved reliability provided by a membership service implemented at the data link level to maintain system consistency.

The set of experiments oriented to message data have been concentrated on the CNI message memory. Injections are synchronised with memory accesses causing multiple bit-flips. The detection of these bit-flips caused by physical faults during writing, reading, or storage in message memory is outside the scope of TTP, but not outside the scope of the TTA architecture – which must guarantee system dependability at every level. Because these errors occur prior to frame formation, the CRC field, calculated in TTP to protect the frame, is not enough.

Two EDC's have been considered in this paper and evaluated experimentally: a cyclic redundant check per bus line; and vertical parity code. The second proposal produces detection coverage higher than 99%. However, since data bytes increase with error detection codes, the final decision about data protection rests with human integrators searching for a balanced solution.

References

[1] H. Kopetz, The Time-Triggered Architecture, Proc. First Inter. Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), 1998.

[2] BMW AG, DaimlerChrysler AG, Robert Bosch GmbH and General Motors/Opel AG. FlexRay Requirements Specification. v. 2.0.2.2000, www.flexray.com.

[3] T. Führer, B. Müller, W. Dierterle, F. Hartwich, R. Hugel, M. Walther and Robert Bosch GmbH, Time-Triggered Communication on CAN (Time Triggered CAN – TTCAN), CAN in Automation 2002, www.can-cia.de/can/ttcan/

[4] M.J. Calha and J. Fonseca, Adapting FTT-CAN for the Joint Dispatching of Tasks and Messages, Proc. 4th Inter. Workshop on Factory Commuunication Systems, pp.117-124, 2002.

[5] H. Kopetz, A. Ademaj, P. Grillinger and K. Steinhammer, The Time-Triggered Ethernet (TTE) Design, Procs. of 8th ISORC (2005), pp. 22-33.

[6] www.tttech.com, Time-Triggered Protocol TTP/C High-Level Specification Document Protocol Version 1.1 and other documents and papers (public access).

[7] X-By-Wire Consortium, Safety Related Fault Tolerant Systems in Vehicle, Project no. BE 95/1329, Contract no. BRPR-CT95-0032, Final Report.

[8] P. Hazucha, C. Svensson, Impact of CMOS Technology Scaling on the Atmospheric Neutron Soft Error Rate, IEEE Trans. on Nuclear Science, Vol. 47, No. 6, (2000) 2586-2594.

[9] C. Constantinescu, Impact of Deep Submicron Technology on Dependability of VLSI Circuits, Proc. of Inter. Conf. on Dependable Systems and Networks (DSN) (2002) 205-209.

[10] P. Sivakumar, M. Kistler, S.W. Keckler, D. Burger, L. Alvisi, Modelling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic, Proc. of DSN (2002) 389-398.

[11] R. Baumann, J. Borel, E. Daly, J. Gasiot, J. Gautier, J.L. Leray, J. F. Ziegler, Earth and Space Single-Events: in present and future electronics, RADECS Short Course (2001).

[12] R. Koga, S. Crain, K. Crawford, P. Yu, Heavy Ion Induce Hard Errors in Memory Devices with sub-micron Feature Sizes, Proc. of 6th European Conference on Radiation and Its Effects on Components and Systems, (2001) 423-430.

[13] F. Wrobel, J.M. Palau, C.M. Calvet, O. Bersillon, H. Duarte, Simulation of Nucleon-Induced Nuclear Reactions in a Simplified SRAM Structure: Scaling Effects on SEU and MBU Cross Sections, IEEE Transactions on Nuclear Science, Vol. 48, No. 6, (2001) 1946-1952.

[14] K. Castellani-Coulié, J.M. Palau, G. Hubert, M.C. Calvet, P.E. Dodd, F. Sexton, Various SEU Conditions in SRAM Studied by 3-D Device Simulation, IEEE Transactions on Nuclear Science, Vol. 48, No. 6, (2001) 1931-1936.

[15] R.C. Baumann, E.B. Smith, Neutron-Induced Boron Fission as a Major Source of Soft Errors in Deep Submicron SRAM Devices, Proc. of 38th Annual Inter. Reliability Physics Symposium, (2000) 152-157.

[16] O. Musseau, F. Gardic, P. Roche, T. Corbière, R.A. Reed, S. Buchner, P. McDonald, J. Melinger, L. Tran, A.B. Campbell, Analysis of Multiple –bit Upsets (MBU) in a CMOS SRAM, IEEE Transactions of Nuclear Science, Vol. 43, No.6 (1996).

[17] R.A. Reed, M.A. Carts, P.W. Marshall, C.J. Marshall, O. Musseau, P.J. McNulty, D.R. Roth, S. Buchner, J. Melinger, T. Corbière, Heavy Ion and Proton-Induced Single Event Multiple Upset, IEEE Transactions on Nuclear Science, Vol. 44, No. 6, (1997).

[18] J.A. Abraham, V.K. Agarwal, B. Bose, Y. Levendel, E.J. McCluskey, P.R. Menon, J. Metzner, Y. Tohma, Fault-Tolerant Computing: Theory and Techniques, D.K. Pradhan, (Prentice-Hall, 1986).

[19] E.A. Amerasekera, F.N. Najm, Failure Mechanisms in Semiconductor Devices, (Wiley, second edition, 1997).

[20] Fault Representativeness, ETIE2 of Dependability Benchmarking Project, (2002).

[21] J. Gracia, D. Gil, J. C. Baraza, and P. J. Gil, Using VHDL-Based Fault Injection to exercise Error Detection Mechanisms in the Time-Triggered Architecture, Proc. of Pacific Rim Inter. Symposium on

Dependable Computing, (2002) 316-320.

[22] D.S. Patterson, Understanding Reliability Criteria for Solder Bumped Devices, Meptec Report, (January-February 2002) 25-28.

[23] J.C. Baraza, J. Gracia, D. Gil, P.J. Gil, A Prototype of a VHDL-Based Fault Injection Tool: Description and Application, Journal of Systems Architecture, Vol. 47, No. 10, (2002) 847-867.

[24] P.J. Gil, S. Blanc, J.J. Serrano, Pin-level Hardware Fault Injection Techniques, Handbook of Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation, Vol. 23, 63-80 (A. Benso y P. Prinetto ed. Book series: Frontiers in Electronic Testing, Kluwer Academic Press).

[25] S. Blanc, P.J. Gil, Improving the Multiple Error Detection Coverage in Distributed Embedded Systems, Proc. of 22nd Inter. Symp. on Reliable Distributed Systems (2003) 303-312.

[26] A. Burns, A. Wellings, Real-Time Systems and Programming Languages, Ed. Addison-Wesley, 2001

[27] J. Rushby, Bus Architectures For Safety-Critical Embedded Systems, Proc. First Workshop on Embedded Software (2001) 8-10.

[28] A. Ademaj, Slightly-Off-Specification Failures in the Time-Triggered Architecture, Proc.of 7th Annual IEEE Inter. Workshop on High Level Design Validation and Test (2002) 7-12.

[29] L. Lamport, R. Shostak, M. Pease, The Byzantine Generals Problem, ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, (19982) 382-401.

[30] R. J Martínez, P. J. Gil, G. Martín, C. Pérez, J. J. Serrano, Experimental Validation of High-Speed Fault-Tolerant Systems Using Physical Fault Injection, Proc. of DCCA-7 (1999), pp 233-249.

[31] J. Barak, J.L. Barth, C.M. Seidleck, C.J. Marshall, M.A. Carts, R.A. Reed, Single Event Upset in the Dual-Port-Board SRAMs of the MPTB Experiement, IEEE Transactions on Nuclear Science, Vol. 47, No. 3, (2000) 712-717.

[32] A. Ademaj, H. Sivencrona, G. Bauer, J. Torin, Evaluation of Fault Handling of the Time-Triggered Architecture with Bus and Start Topology, Proc. of DSN (2003) 123-132.

[33] S. Blanc, J. Gracia, P.J. Gil, A Fault Hypothesis Study on the TTP/C using VHDL-based and Pin-Level Fault Injection Techniques, Proc. of IEEE 17th Inter. Symposium on Defect and Fault Tolerance in VLSI Systems (2002) 254-262.