High Availability in Controller Area Networks

Ricardo Pinto*
FC - UL
Lisbon, Portugal
rcp@lasige.di.fc.ul.pt

José Rufino*
FC - UL
Lisbon, Portugal
ruf@di.fc.ul.pt

Carlos Almeida†
IST - UTL
Lisbon, Portugal
carlos.r.almeida@ist.utl.pt

***Abstract*—The Controller Area Network (CAN) fieldbus is a popular technology for distributed embedded system networking. Its usage spans several domains, ranging from home automation to factory control. There are, however, restrictions to its application in a specific domain: highly dependable applications.**

A crucial step towards CAN-based dependable systems was taken by the CAN Enhanced Layer (CANELy) architecture. This architecture provided the analytic models of CAN operation. Based on these models, it defines both the hardware and software mechanisms for dependable and timely CAN-based network service provision.

This paper provides the materialization of CANELy network availability mechanisms. The mapping of the error monitoring and fault-confinement mechanisms defined by CANELy into hardware has proven to be resource-effective, allowing their integration in an inexpensive Field Programmable Gate Array (FPGA) device. This opens room for cost-effective high dependability CAN-based solutions in several domains, e.g. the aerospace industry.

***Index Terms*—Embedded and Real-time Systems, Controller Area Network (CAN), Dependability, High Availability**

I. Introduction

Current embedded systems possess significant computing resources, allowing their usage in complex control systems. Moreover, these systems are usually distributed in nature. An instance of such a system is a satellite, where the OnBoard Computer (OBC) gathers information from physically dispersed sensors, in order to adjust the spacecraft's attitude via actuators. Such distribution raises several issues, from component physical interconnection to logical entity interaction. A class of computer networks, named *fieldbuses*, was devised to address the issue of control system interconnection, in harsh industrial environments.

The Controller Area Network (CAN) is an instance of a fieldbus technology, traditionally used to interconnect embedded applications in the automotive industry. Its current popularity among other embedded settings, such as aeronautics [1] and space [2] comes mainly from two factors: *low-cost*, being available in many cheap microcontrollers, having also modest requirements regarding cabling and connectors; *robust operation*, having (to a certain extent) fault-tolerant behaviour in the presence of network errors. However, is standard CAN usable as a building block of highly dependable systems?

The CAN Enhanced Layer (CANELy) architecture [3] was designed to provide a definite and affirmative answer to the previous question. CANELy consists of both hardware and software components, with well defined interfaces. The architecture is modular, specifying solutions for a set of different problems, such as: high network availability; enhanced network timeliness; reliable communication.

This paper addresses the design and materialization of the CANELy components providing a highly available CAN-based network infrastructure. This paper is organized in the following manner: Section II introduces the CANELy architecture and provides a context for its design, discussing the impairments of the standard CAN layer regarding dependable network operation; Section III discusses the CANELy mechanisms for enforcing network availability; Section IV details how these mechanisms can be materialized, and mapped into reconfigurable hardware. Finally, Section V concludes this paper.

II. CAN Enhanced Layer (CANELy)

The CANELy architecture [3] enables CAN as a solution for dependable system interconnection. Such interconnection is needed in domains having stringent requirements, e.g. distributed hard-real time control.

A. CAN Protocol and its Dependability

The CAN fieldbus [4] is a multi-master bus, where the nodes communicate via a shared medium. Medium access is achieved through a non-destructive arbitration process where the node transmitting the message having the lowest identifier gets through. The remaining competing nodes listen to the winning node, and reschedule the transmission of their messages to the next bus idle period. Such arbitration method comes from two factors: the bus signalling having logical AND behaviour, with *dominant* (d) bits overwriting *recessive* (r) bits, corresponding also to *idle* bus state; the *quasi*-stationary network operation mode, where the nodes sample the bus almost simultaneously.

There are four types of frames: *data*, *remote*, *error* and *overload*. The data and remote frames are used during normal operation: data frames have a payload field, carrying a piece of user-level information; the remote frames do not possess such field. Conversely, error frames are issued upon bus error

*Faculdade de Ciências da Universidade de Lisboa, Campo Grande - Edifício C8, 1749-016 Lisboa, Portugal. Tel: +351-217500254. Navigators Home Page: http://www.navigators.di.fc.ul.pt. This work was partially supported by Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology), through the Multiannual Funding and CMU-Portugal Programs.

†Instituto Superior Técnico, Universidade Técnica de Lisboa, Av. Rovisco Pais n. 1, 1049-001 Lisboa, Portugal. Tel: +351-218418397.

detection and overload frames are intended to delay the start of a data or remote frame transmission.

The CAN bus possesses extensive error detection mechanisms, both at the physical and data-link layers, such as: bus bit monitoring; Cyclic Redundancy Check (CRC) for assessing message integrity. During normal operation, all nodes participate in error detection procedures. Each node, however, has internal counters that increment upon the issuing of an error frame. Upon exceeding a certain threshold the node does not participate actively in the error detection procedures, and if the error persists it might be disconnected from the network.

Although CAN is traditionally regarded as a robust fieldbus, it has several impairments against dependable operation. Whilst some of these impairments are architectural, such as lacking a standard mechanism to protect against physical network partitioning [5], other have been uncovered in the course of modelling its operation [6]. Therefore, highly dependable applications using CAN as the communication fieldbus must have these issues dully addressed.

B. CANELy Architecture

The CANELy architecture [3] is a fundamental building block for the construction of dependable distributed systems. It provides a fault-resilient network infrastructure based on the CAN bus, together with a rich set of services for distributed operation. The CANELy architecture is depicted in Fig. 1.

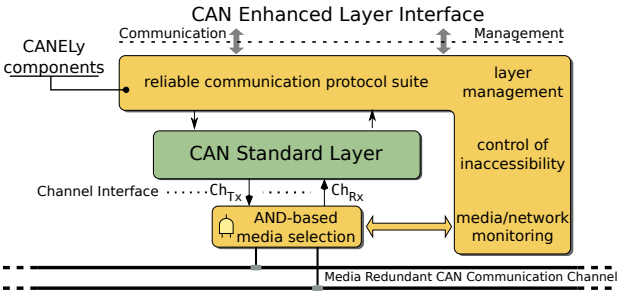

Fig. 1. CANELy Architecture Block Diagram

This architecture was designed around the standard CAN layer, contemplating the provision of both software and hardware mechanisms. These mechanisms deal with (dependable) distributed system services, such as reliable group communication, node failure detection, site membership and clock synchronization. Low-level enhancements to reliability and timeliness include: replicated communication media; network inaccessibility detection and control [7].

III. High Availability Provision

The standard CAN specification [4] does not contemplate native mechanisms for high network availability enforcement. These are in need, since "real-world" networks are not fault-free. The CANELy architecture addresses the issue of network availability, by:

- introducing replicated media, which convey the CAN channel, thus providing resilience to medium partitioning;
- defining a set of mechanisms and functions, ensuring correct management of the replicated media and providing fault confinement.

These mechanisms are mapped into several components, as illustrated in the diagram of Fig. 2. The fundamental components provide: *AND-based Media Redundancy Management*; *Channel and Media Monitoring* functions assessing media status, error detection and fault confinement; *Omission Control*, accounting for omission errors, disabling any media exhibiting an excessive error rate; *Media Quarantine* keeping a faulty medium disabled until recovery.

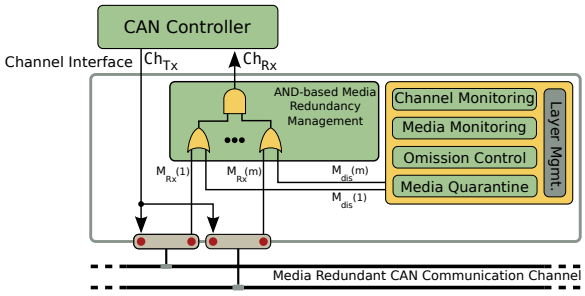

Fig. 2. CANELy Media Selection Unit diagram

A. AND-based Media Redundancy Management

The crux of network media redundancy is its management. In CANELy, redundancy management is ingeniously accomplished by exploiting CAN bus operating properties: the logical AND and *quasi*-stationary network operation.

The AND-based Media Redundancy Management component shown in Fig. 2 receives the several replicated media signals, $M_{Rx}(m)$, extracting the incoming bit stream representing the channel, Ch_{Rx}. This is achieved through a simple and effective solution, by extending the logical AND function over all the incoming media $M_{Rx}(m)$. The mapping of this mechanism into VHDL yields the description shown in Fig. 3.

```
-- ChRx  : Incoming bit stream of the Channel
-- M_Rx  : Vector aggregating the several incoming media
-- In CAN: recessive (r) <=> logical '1'
--         dominant  (d) <=> logical '0'

ChRx <= '1' when MediumRX = (MediumRX'range => '1')
        else '0';
```

Fig. 3. AND-Based Media Redundancy description in VHDL

The description of Fig. 3 implements the logical AND function over a vector aggregating the incoming media bit streams. If all the elements of that vector are logical value '1', then the output is also '1'.

A fragment of simulated Media Selection Unit operation is shown in Fig. 4, which serves as the working basis for detailing and exemplifying the network availability enhancements provided by CANELy.

The recovered channel Ch_{Rx} (ChRx in Fig. 4) can now be delivered to the CAN controller and used by CANELy components for channel monitoring functions.

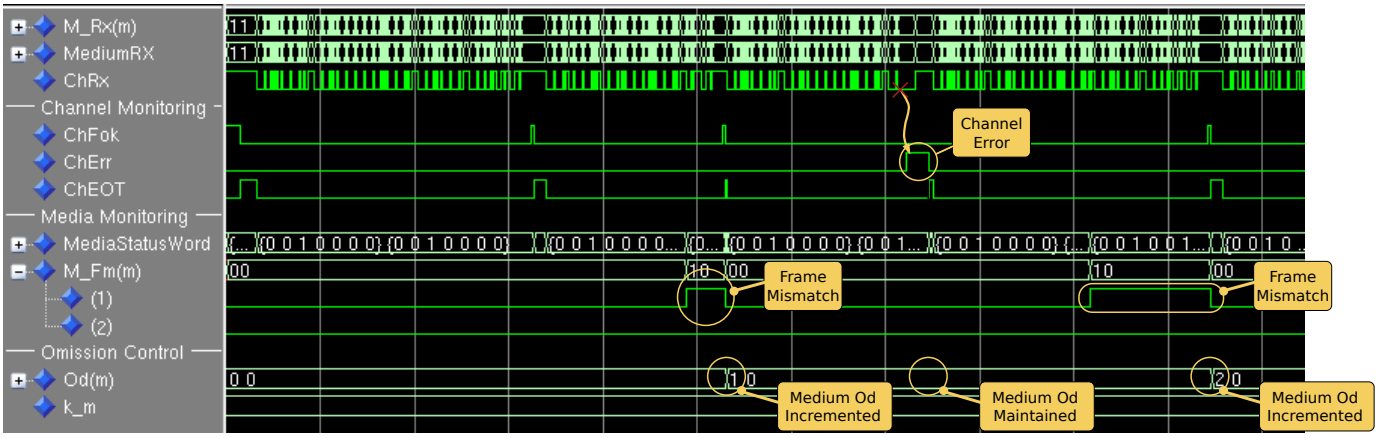

Fig. 4. CANELy Media Selection Unit simulation

B. Channel Monitoring

The CANELy *Channel Monitoring* facilities provide indications of the channel status, through the assertion of the following signals:

- Ch_{Fok} - Correct Frame Reception. This signal is asserted at the end of a successful data or remote frame termination sequence. This results from the detection of the following bit sequence: *rdrrrrrrr*.
- Ch_{Err} - Frame Error. This signal is asserted upon an error/overload frame. This is equivalent to detect a sequence of at least six dominant (d) bits.
- Ch_{EOT} - End Of Transmission. This signal is asserted upon the end of a successful frame transfer followed by a minimum two bit bus idle period.

In general, Channel Monitoring functions are based on the detection of certain and unique channel incoming bit sequences. The operation of these mechanisms is illustrated in the simulation fragment of Fig. 4.

C. Media Monitoring

The *Media Monitoring* mechanisms provide error detection for each medium, monitoring its interface and comparing it against the channel receive signal. Each medium has an associated *Medium Status Word*, whose elements are updated by the medium monitoring functions, assessing if the medium:

- is in a correct *idle state*;
- has conveyed at least one *dominant bit*, such as the start-of-frame delimiter, during the current frame transfer;
- is experiencing an *abnormal recessive period*;
- is *stuck-at-dominant*;
- is *disabled*, either by quarantine actions or by higher layers' management request;
- has experienced a *frame mismatch* with respect to the channel receive signal, during the current frame transfer, as illustrated in Fig. 4;
- has exceed the omission degree bound.

The *stuck-at-dominant* monitoring signal is used to automatically disable the contribution of a faulty medium, through the assertion of a medium disabling signal. A given medium can also be disabled upon an order from media quarantine mechanisms, or if its omission degree[1] exceeds the allowed *omission degree bound*, k_m.

D. Omission Control

The last set of monitoring functions detects and evaluates *omission errors*. An omission error occurs whenever a message fails to be delivered, due to its corruption by the network. The error may affect only a subset of the media, or all the media in the case of common-mode errors. The omission degree of each medium, $O_d(m)$ is evaluated upon the assertion of Ch_{EOT}, being:

- *incremented*, if the frame mismatch signal is asserted, although the frame has been received correctly; or if a channel error is signaled and the corresponding frame mismatch signal remains negated;
- *maintained*, if it was not possible to assign the error to a given medium, despite the fact the frame was not correctly received;
- *cleared*, if the frame was correctly received and the medium frame mismatch signal was not asserted.

The operation of the Omission Control mechanisms is illustrated in Fig. 4. Medium 1 exhibits a frame mismatch, `M_Fm(m)` with $m = 1$, during a correct frame transfer. That means the medium error has not affected the channel, but should be accounted for as an error for that Medium. Upon the frame End-of-Transmission, the omission degree evaluation mechanisms increment the omission degree of Medium 1.

The next message transfer is aborted by an error frame.The Ch_{Err} flag is asserted, and the omission degree of all media is maintained, since there is no frame mismatch in this situation. That means the source of the error cannot be assigned to a given medium.

Upon the retransmission of a frame affected by errors, the Medium 1 has another frame mismatch, which upon evaluation implies the increment of its omission degree.

[1]A medium omission degree is the number of consecutive omission errors in a given reference interval.

IV. Component Engineering and Prototyping

The final issue to be discussed is the engineering of the mechanisms, and their integration both in a Field Programmable Gate Array (FPGA) device and in a computing platform prototype, materializing a CANELy node.

A. FPGA Resource utilization

The world of embedded systems' design is ruled chiefly by production cost. Therefore, mapping the CANELy mechanisms into hardware must be made cost-effective. The target implementation technology is any low-cost FPGA family, e.g. Xilinx's Spartan-3E or Altera's Cyclone. Resource usage of the Media Selection Unit (MSU) core unit is shown in Table I.

TABLE I
FPGA RESOURCE OCCUPATION

Device	CANELy MSU Mechanisms			
	Flip-Flops	LUTs*	Slices	
			Absolute	Relative (%)
XC3S500E	121	228	148	3.2
XC3S100E	121	228	148	15.4
XC3S50	121	228	148	19.3

Look-Up-Tables

The figures of Table I were obtained through synthesis and place & route of the design, having as target several devices from Xilinx. The synthesizer was Xilinx's XST 11.4, with the optimization effort put into area reduction. The integration of these mechanisms with existing CAN controller cores [8], [9] represents only a small overhead in resource usage. Fig. 5 shows usage w.r.t. to the total resource utilization, i.e. the sum of CANELy MSU and CAN core resource utilization.

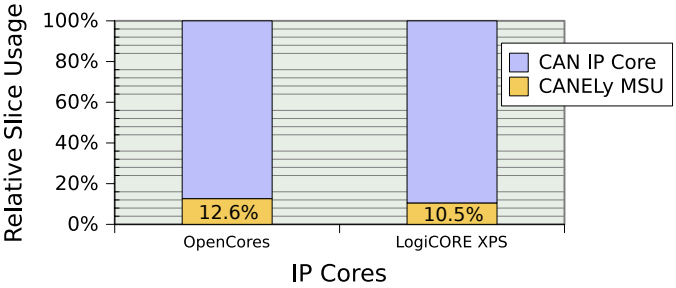

Fig. 5. CANELy MSU Resource Usage Comparison

B. Proof-of-Concept Prototype

The CANELy computing platform must include a processing element to support the execution of software components, such as the reliable group communication protocol suite and specific layer management entities. Together with the hardware components materialized in an FPGA device they form the CANELy node. The current CANELy node prototype is shown in Fig. 6.

The enhanced 8051 core used in the prototype of Fig. 6 includes a dual standard CAN interface, which enables the implementation of a quad-media/dual-channel CAN infrastructure - possible due to the CANELy's reduced resource usage.

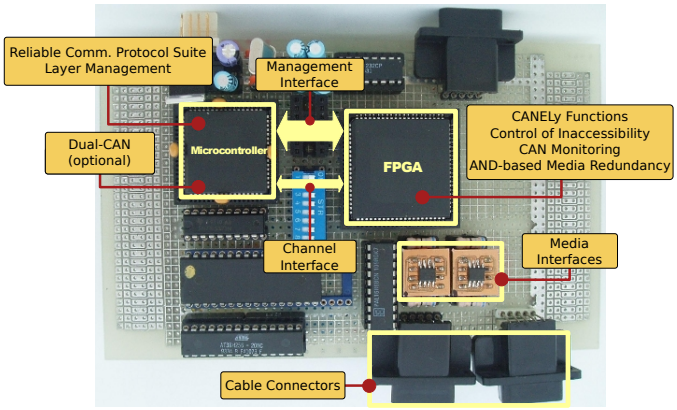

Fig. 6. CANELy Prototype Board

V. Conclusions and Future Work

The CAN Enhanced Layer (CANELy) architecture defines a CAN-based infrastructure capable of extremely reliable communications. A fundamental block concerns the implementation of a highly available network infrastructure.

This paper addresses CAN network availability issues, and how the CANELy mechanisms can be mapped into a platform including a state-of-the-art microcontroller and a companion FPGA device.

These mechanisms can be further integrated with the standard CAN controller, given the current availability of Intelectual Property (IP) cores [8], [9], [10]. Future work will focus on the integration of the CANELy components with the LEON3 spaceborne processor [11].

The long term objective is the integration of the CANELy architecture into emergent time- and space-partitioned architectures (TSP), acting as the network building block for dependable mixed-criticality applications, e.g. aerospace and automotive.

References

[1] AEEC, "General standardization of CAN (Controller Area Network) for airborne use," ARINC Specification 825-1, May 2010.

[2] ECSS, "ECSS draft standard ECSS-E-ST-50-15C - recommendations for CAN bus in spacecraft onboard applications," ECSS Draft, European Cooperation for Space Standardization (ECSS), May 2005.

[3] J. Rufino, "Computational system for real-time distributed control," Ph.D. dissertation, Technical University of Lisbon - Instituto Superior Técnico, Lisboa, Portugal, Jul. 2002.

[4] *International Standard 11898-1 - Road vehicles - Controller Area Network (CAN)*, ISO Std., Dec. 2003.

[5] H. Kopetz, "A comparison of CAN and TTP," in *Proc. 15th IFAC Workshop on Dist. Computer Control Systems*, Como, Italy, Sep. 1998.

[6] J. Rufino, P. Verissimo, G. Arroz, C. Almeida, and L. Rodrigues, "Fault-tolerant broadcasts in CAN," in *Digest of Papers 28th Annual Int. Symposium on Fault-Tolerant Computing*, Jun. 1998, pp. 150–159.

[7] J. Rufino, P. Verissimo, G. Arroz, and C. Almeida, "Control of inaccessibility in CANELy," in *Proc. of the 6th. Int. Workshop on Factory Communication Systems*. Torino, Italy: IEEE, Jun. 2006, pp. 35–44.

[8] I. Mohor, *CAN Protocol Controller IP core*, OpenCores, Nov. 2004.

[9] *LogiCORE IP XPS Controller Area Network (CAN)*, Xilinx Inc., Jul. 2010.

[10] *HurriCANe - Controller Are Network IP core User's Manual*, European Space Agency, Sep. 2007.

[11] *GRLIB IP Library User's Manual*, Aeroflex Gaisler AB. [Online]. Available: http://www.gaisler.com/