FPGA-based Engineering of Bus Media Redundancy in CAN

José Rufino – FCUL*
Ricardo Pinto – IST/UTL
Carlos Almeida – IST/UTL

Abstract: Distributed control systems are ubiquitous nowadays, with applications ranging from vehicle to shop-floor control. With the increased penetration of control systems in all application, the need for reliable communication among the members of the system is more crucial than ever. Standard CAN bus networks already have embedded a mechanism to guarantee the provision of service in face of a one-wire fault in a two-wire differential cabling infrastructure. However there is no mechanism to guarantee continuity of service in the event of a two-wire fault. To achieve this, one needs spatial redundancy for the cabling infrastructure. However, the design and implementation of such kind of solutions places problems with a non-trivial solution, despite the end result is quite simple. This paper addresses how a FPGA-based solution may be used to enhance CAN network availability through the management of the several physical media which support a single, logical medium.

1 - Introduction

With the advances in miniaturization of processing units, microcontrollers found their way into nearly all fields of application, including control applications.

Control is nowadays a reality present in almost all systems we interact with in a daily basis, like cars, lifts, automatic doors. These systems are composed by sensors, which gather information about the surrounding environment, controllers who process this information and actuators who act accordingly to the output of the controllers. Sometimes these elements are not physically located in the same place, and need some sort of interconnect to convey the information between them. One of the solutions to solve the problem of component interconnection in a distributed system is CAN, the Controller Area Network fieldbus [1, 2].

In addition, almost all distributed control applications have high dependability requirements, meaning the overall system must exhibit a set of strict availability, reliability and timeliness requirements which cannot be fulfilled by the native CAN protocol [19, 9, 12].

However, given the amount of applications based on standard CAN controllers, the most effective solution to enhance the dependability of a CAN fieldbus infrastructure without compromising the cost would involve the use of "components off the shelf".

In this paper the implementation of the mechanisms described in [8, 9, 11], is thoroughly analysed in respect to the engineering problems they pose and how to solve them, achieving a cost effective solution to enhance the dependability of CAN-based systems and applications. The provision of a highly available network infrastructure that uses only bus media redundancy is a simple solution of utmost importance for building highly dependable CAN-based systems.

This paper is organized in the following manner: Section 2 introduces the framework to enhance CAN dependability and timeliness characteristics, the CAN Enhanced Layer, CANELy; Section 3

* This work was supported by EU and FCT, through Project POSC/EIA/56041/2004 (DARIO) and through the FCT Multiannual Funding Program. Project DARIO Web site – http://pandora.ist.utl.pt/projects/dario.

describes the CANELy hardware architecture, and how it provides support for the implementation of CAN protocol enhancement mechanisms, such as bus media redundancy; Section 4 analyses the design and implementation of the specific mechanisms to handle the bus media redundancy; Section 5 concludes the paper. The paper assumes the reader to be fairly familiar with CAN operation [1, 2].

2 - The CANELy Framework

The CAN fieldbus is a very important design component. However, would CAN technology be used by system designers as a core building block of dependable, hard real-time systems and applications?

A large set of devices, intended for the implementation of the CAN standard layer are available in different configurations: stand alone CAN controllers [18]; devices integrating in a single chip, single/dual CAN controllers and an intelligent processing infrastructure [3]; or, more recently, CAN controller cores [15, 16]. This represents a very strong argument in favor of using commercially available CAN technology.

The shortcoming of that approach is well known: the native CAN protocol exhibits a set of severe limitations in respect to the provision of dependability properties, such as strict availability, reliability and timeliness attributes [19]. This problem was thoroughly investigated in [9] and it was discovered that to attain the levels of dependability provided by other similar technologies (e.g. TTP [20]) a set of fault tolerance and timeliness-related services was missing.

However, those services can be provided off-the-shelf (i.e. without modifications to the CAN standard or to existing CAN controllers), meaning CAN operation can (and should) be complemented/enhanced with some simple machinery and low-level protocols, able to secure the strict reliability, availability and timeliness guarantees needed by highly fault-tolerant real-time systems and applications [9,12].

In [9], this concept is called CANELy, the CAN Enhanced Layer, a CAN-based infrastructure able to extremely reliable communication. The CANELy architecture represented in Figure 1 identifies the main functional modules. The CAN standard layer is the central component of CANELy. Other CANELy specific components are required to secure:

- highly available network infrastructure, through redundancy;
- hard real-time operation, even in the presence of glitches in network operation;
- reliable CAN communications.

The CANELy architecture follows a modular approach, allowing system designers to include only the components needed for a given application. For example, in the architecture of Figure 1 it is assumed the use of simple media redundancy, although we do not preclude the utilization of full space-redundant network architectures, whenever glitch-free operation and tight timeliness guarantees are required.

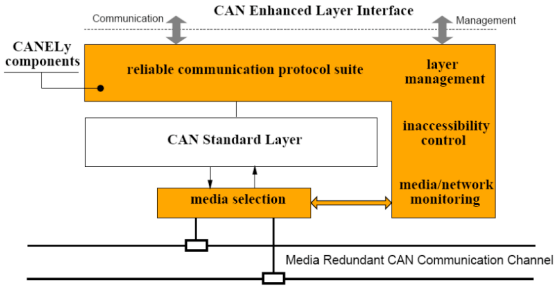


Figure 1 CANELy Architecture

The functional modules in the CANELy reliable communications protocol suite are built on top of the exposed CAN standard layer interface as a software layer composed of simple low-level protocols and aim to render a programming interface enhanced with a set of semantically rich services, such as reliable group communication [7], failure detection and membership [10], clock synchronization [6]

and support to group management. These services are extremely useful to the implementation of distributed fault-tolerance techniques [9].

Other modules in the CANELy architecture are intended to support: the control of CAN inaccessibility [11]; configuration and layer management [9]. The additional information provided by these modules can be made available to upper layer entities, through a management interface and used by other components in the system able to secure hard real-time operation [21, 22, 23].

3 - CANELy Hardware Architecture

The basic mechanisms needed to enforce timeliness and dependability in the CANELy system architecture must be implemented by special-purpose logic machinery. Such additional mechanisms are not complex to implement, being able to fit in small sized devices, such as Programmable Logic Devices (PLDs). However the low-cost and high density of today programmable logic devices such as current Field Programmable Gate Arrays (FPGAs) opens room for the integration of both basic and advanced dependability and timeliness enforcement functionalities.

The CANELy hardware architecture is therefore composed by three essential blocks:

- a microcontroller, containing the native CAN controllers and providing support for the execution of the CANELy protocol suite [7, 10, 6, 11, 9];
- a FPGA, holding the implementation of the required special-purpose low-level machinery functions;
- the CAN transceivers, dealing with the CAN physical (PHY) layer.

The FPGA machinery performs the continuous monitoring of the standard CAN channel interfaces, detecting CAN bus errors such as frame omissions and disabling medium interfaces affected by failures. In addition, the microcontroller interfaces the FPGA for the overall management of the machinery being introduced to enhance CAN dependability and timeliness attributes. The CANELy hardware architecture is depicted in the diagram of Figure 2.

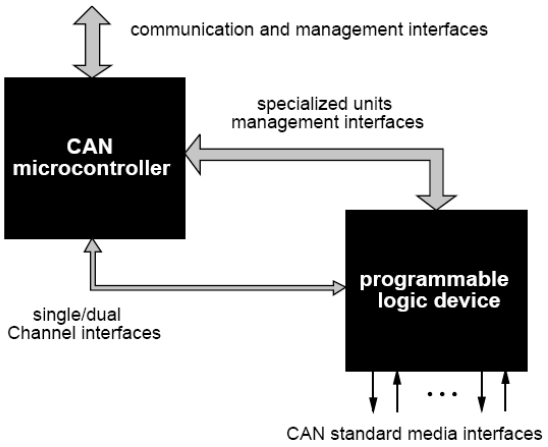


Figure 2 CANELy Hardware Architecture

Apart from replication, a completely standard physical layer interface is used, allowing multiple design solutions for the cabling infrastructure. In particular, it allows the use of inexpensive two-wire differential cabling plus commercial non fault-tolerant transceivers.

In the current prototype implementation (Figure 3) it is used: a MaximDallas DS80C390 microcontroller [3]; a FPGA Xilinx Spartan 3E device [4]; MAX13054 CAN transceivers [5].

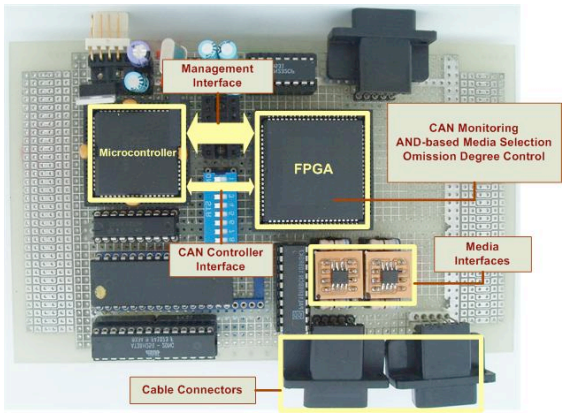


Figure 3 CANEly Prototype

4 – CANELy Low-Level Engineering

The FPGA infrastructure integrates a set of basic dependability enforcement mechanisms. Roughly, they correspond to the low-level mechanisms of the CANELy architecture.

4.1 Handling Physical Network Partitioning

A fundamental problem regarding the dependability of the fieldbus infrastructure concerns network availability. For example, in harsh environments cabling may be damaged, which in turn affects the behaviour of the transmission medium and therefore network operation.

In [8] a Columbus' egg idea was devised to solve the problem of CAN network availability, managing the interface of the different media in the PHY layer with only one MAC* sub-layer. The Columbus' egg strategy is based on the fundamental nature of CAN and on its quasi-stationary operation, extending the wired-AND properties of physical bus operation to the CAN Channel interface. This means only a single conventional AND gate, inserted between the medium interfaces and the standard CAN controller (cf. Figure 4) is needed to provide resilience to medium partitioning faults.

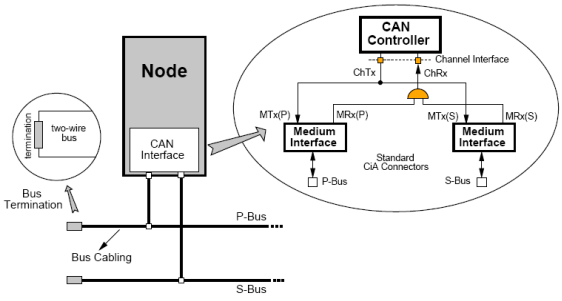


Figure 4 CAN Columbus' Egg Strategy

That means a medium exhibiting a dominant bit value will always prevail. Additional mechanisms are required to handle medium stuck-at-dominant faults, but these are not hard to implement. The standard CAN specification [1] does not allow a correct medium to be at a dominant state for more than a given number of bits. Such kind of errors can be easily detected and the contribution of a stuck-at-dominant medium to the Channel receiver signal can be disabled.

The Columbus' egg strategy, together with the management of stuck-at-dominant medium faults is implemented in VHDL† by the mechanisms specified in Figure 5, resorting to implicit loop unrolling techniques in the VHDL synthesizer and following the guidelines present in [17].

```
-- Bare Columbus' egg strategy
Ch_Rx  <= '1'
when  M_Rx_in = (M_Rx_in'range=>'1')
else '0';

-- Medium disable logic
M_Rx_disable: process (M_Rx, M_dis)
begin
  for m in 1 to NumberMedia loop
    M_Rx_i(m) <= M_dis(m) or M_Rx(m);
  end loop;
end process M_Rx_disable;
```

Figure 5 CAN Media Selection in VHDL

4.2 Handling Stuck-at Faults

Though resilience to medium stuck-at-dominant faults is handled at the media selection level, network operation can be affected by other stuck-at failures:

- stuck-at-recessive medium, which does not disturb network operation. Nevertheless, such kind of failures are detected by the CANELy low-level machinery and signaled to high-level management entities, e.g. for diagnose and repair purposes;
- stuck-at-dominant Channel, which may severely disturb the operation of the CAN protocol. This kind of failure is handled by the CANELy machinery through low-level mechanisms allowing early failure detection and disable of the corresponding Channel interface.

* Medium Access Control.

† Very High-Speed Integrated Circuits (VHSIC) Hardware Description Language.

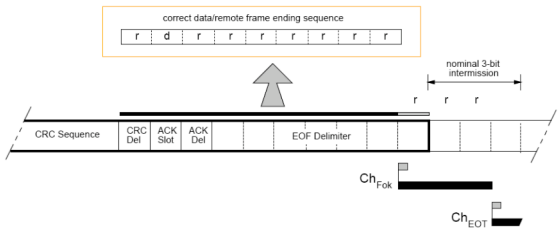
Note that commercial state-of-the-art CAN transceivers such as the MAX13054 [5] also integrate additional mechanisms for transceiver disabling upon the detection of a stuck-at-dominant transmitter. In any case, the CANELy mechanisms are highly effective allowing in general an earlier detection of such failures;

- stuck-at-recessive Channel, which must be (and usually is) handled directly by the CAN controller.

Naturally, other CAN controller failures, such as deaf receiver, stuck-at-dominant receiver and general receiver/transmitter failures are handled directly by the native error confinement mechanisms of the standard CAN controller [9, 13, 14].

4.3 Frame Monitoring

The assertion of frame monitoring signals, as specified in [8, 9,11] can be engineered through a simple general pattern detection methodology. There is always a pattern of recessive (r) and dominant (d) bits to be detected, upon monitoring the sequence of incoming bits at the Channel and Medium interfaces.



Frame Monitoring Signal	Reference	Detection Pattern
Frame correct	Ch_{Fok}	rdrrrrrrr
Transmission correct	Ch_{Tok}	rdrrrrrrrrr
End of Transmission	Ch_{EOT}	rrrrrrrrr

Figure 6 CANELy Frame Monitoring

A clear example of this methodology is the detection of a correct frame reception, with the assertion of signal Ch_{Fok}, upon the detection of the pattern specified in the diagrams of Figure 6.

The engineering of a generic implementation for the pattern detection machinery is presented in Figure 7. The idea of the corresponding VHDL description is the continuous comparison of the incoming bitstream with the desired pattern, stored in an internal Read Only Memory (ROM).

The correct detection assessment is done through the comparison of the current ROM address with the programmed pattern length. Otherwise, the process returns to the initial position and starts comparing all over again.

```
begin
  -- purpose: To detect a pattern
  -- type   : sequential
  -- inputs : Clk, Reset_N, D_in
  -- outputs: count
  PDetection: process (Clk)
  begin
    if Clk'event and Clk='1' then
      if Reset_N = '0' then
        count <= 0;
      else
        count <= cnta;
      end if;
    end if;
  end process PDetection;

  cnta <= (count + 1)
  when D_in = pattern_rom(count) and
       count/=pattern'length
  else 0;
  P_Ok <= '1'
  when count=pattern'length
  else '0';
end
```

Figure 7 Detection of patterns in VHDL

Each frame monitoring signal is implemented through a particular instantiation of the general description specified in Figure 7, using advanced features of the VHDL language such as dynamic sizing and parameterization of the component internal structures. This has allowed an extremely simple design solution, as shown in Figure 7.

The pattern detection scheme can be further extended to enhance the resource utilization efficiency, through a sort of resource sharing. For example, the assertion of Ch_{Tok} can be seen as an

extension of Ch_{Fok}, the assertion of the former implies the assertion of the later.

This way we can pack the machinery in a denser form at the cost of losing a generic pattern detection machinery description.

4.4 Additional Dependability Mechanisms

The engineering of the CANELy low-level mechanisms also includes advanced machinery functions such as: accounting and management of each medium frame *omission degree*, i.e. the number of consecutive frame omissions on that medium; media quarantine mechanisms, which temporarily disables a given medium when it exhibits an excessive number of errors, until (if) it behaves well again; the provision of a system level management interface.

5 – Conclusions and Future Work

This paper has addressed the engineering of the low-level machinery mechanisms of CANELy, an architecture able of extremely reliable communication using the standard CAN components enhanced with a set of additional hardware/software mechanisms.

The low-level machinery mechanisms are deployed in a Field Programmable Gate Array (FPGA) and specified using VHDL descriptions. Advanced features of VHDL such as component dynamic sizing and parameterization were used in order to reduce the complexity of the design flow.

A prototype board of the CANELy architecture that combines a Xilinx Spartan 3E FPGA with a MaximDallas DS80C390 microcontroller and integrates this design is currently being developed.

José Rufino

FCUL – Faculdade de Ciências da Universidade de Lisboa.

Campo Grande - Bloco C8,
1749-016 Lisboa, Portugal.

Phone +351-217500254

Fax +351-217500084

E-mail ruf@di.fc.ul.pt

Website http://www.navigators.di.fc.ul.pt

Website http://pandora.ist.utl.pt/CAN

Ricardo Pinto

IST/UTL – Instituto Superior Técnico, Universidade Técnica de Lisboa.
Avenida Rovisco Pais,
1049-001 Lisboa, Portugal.
Phone +351-218418397

Fax +351-218417499

E-mail ricardo.pinto@ist.utl.pt

Website http://pandora.ist.utl.pt

Carlos Almeida

IST/UTL – Instituto Superior Técnico, Universidade Técnica de Lisboa.
Avenida Rovisco Pais,
1049-001 Lisboa, Portugal.
Phone +351-218418397

Fax +351-218417499

E-mail cra@comp.ist.utl.pt

Website http://pandora.ist.utl.pt

References

[1] ISO. International Standard 11898 - Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication, November 1993.

[2] CiA - CAN in Automation. CAN Physical Layer for Industrial Applications - CiA Draft Standard 102 Version 2.0, April 1994.

[3] Maxim/Dallas Semiconductors. DS80C390 Dual-CAN High-Speed Microprocessor, February 2005.

[4] Xilinx Inc. Spartan-3E FPGA Family: Complete Data Sheet, May 2007.

[5] Maxim. MAX13054 Industry-Standard High-Speed CAN Transceivers with ±80V Fault Protection. February 2005.

[6] L. Rodrigues, M. Guimarães, and J. Rufino. Fault-tolerant clock synchronization in CAN. In Proceedings of the 19th Real-Time Systems Symposium, pages 420-429, Madrid, Spain, December 1998. IEEE.

[7] J. Rufino, P. Veríssimo, G. Arroz, C. Almeida, and L. Rodrigues. Fault-tolerant broadcasts in CAN. In Digest of Papers, The 28th Int. Symposium on Fault-Tolerant Computing Systems, Munich, Germany, June 1998. IEEE.

[8] J. Rufino, P. Veríssimo, and G. Arroz. A Columbus' egg idea for CAN media redundancy. In Digest of Papers, The 29th International Symposium on Fault-Tolerant Computing Systems, Madison, Wisconsin - USA, June 1999. IEEE.

[9] J. Rufino. Computational System for Real-Time Distributed Control. PhD thesis, Technical University of Lisbon - Instituto Superior Técnico, Lisboa, Portugal, July 2002.

[10] J. Rufino, P. Veríssimo, and G. Arroz. Node failure detection and membership in CANELy. In Proceedings of the International Conference on Dependable Systems and Networks, San Francisco, California, USA, June 2003. IEEE.

[11] J. Rufino, P. Veríssimo, G. Arroz, and C. Almeida. Control of inaccessibility in CANELy. In Proceedings of the 6th. Int. Workshop on Factory Communication Systems, Torino, Italy, June 2006. IEEE.

[12] J. Rufino, C. Almeida, P. Veríssimo, and G. Arroz. Enforcing dependability and timeliness in controller area networks. In Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON06). Paris, France, Nov. 2006, IEEE.

[13] J. Rufino and P. Veríssimo. A study on the inaccessibility characteristics of the controller area network. In Proceedings of the 2nd International CAN conference, London, England, October 1995, CiA.

[14] P. Veríssimo, J. Rufino, and L. Ming. How hard is hard real-time communication on field buses? In Digest of Papers, The 27th International Symposium on Fault-Tolerant Computing Systems, Washington - USA, June 1997. IEEE.

[15] L. Stagnaro. HurriCANe - Free VHDL CAN Controller core. Eupean Space Agency (ESA), Noordwijk, The Netherlands, March 2000.

[16] I. Mohor, CAN Core. www.opencores.org

[17] Peter Sinander.*VHDL Modelling Guidelines*. European Space Agency (ESA), ASIC/001 Issue 1 edition, September 1996.

[18] Philips Semiconductors. SJA1000 Stand-alone CAN controller. January 2000.

[19] H. Kopetz. A comparison of CAN and TTP. In Proceedings of the 15th IFAC Workshop on Distributed Computer Control Systems, Como, Italy. September 1998. IFAC.

[20] H. Kopetz and G. Grunsteidl. TTP – a protocol for fault-tolerant real-time systems. IEEE Computer 27(1), January 1994.

[21] M.A. Livani, J. Kaiser, and W.J. Jia. Scheduling hard and soft real-time communication in CAN. In Proc. of the 23rd Workshop on Real-Time Programming, Shantou, China, June 1998. IFAC/IFIP.

[22] L. Pinho, F. Vasques, and E. Tovar. Integrating inaccessibility in response time analysis of CAN networks. In Proceedings of the 3rd Int. Workshop on Factory Communication Systems, Porto, Portugal, September 2000. IEEE.

[23] S. Punnekkat, H. Hansson, and C. Norstrom. Response time analysis under errors for CAN. In Proceedings of the Real-Time Technology and Applications Symposium, pages 258-265, Washington, USA, May 2000. IEEE.