Integration of a CAN bus in an Onboard Computer for Space Applications

Stijn Wielandt, Geoffrey Ottoy, Jean-Pierre Goemaere, Nobby Stevens and Lieven De Strycker
DraMCo research group,
Catholic University College Ghent, Association KULeuven
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
stijn.wielandt@kahosl.be, info@dramco.be

***Abstract* — The integration of CAN bus networks in space applications has gained interest over the past years because of its high reliability and cost effectiveness. The bus has already been integrated in some low earth orbit satellites with commercial off-the-shelf components, but the harsh environment in deep space demands a more reliable solution. That is why a radiation hardened design of a CAN transceiver and a CAN controller is being investigated. The transceiver is constructed with a radhard RS-485 transceiver and the CAN controller is implemented in a radhard FPGA by means of VHDL IP cores. For this paper, a CAN IP core from Gaisler Research is selected and implemented in an FPGA. Eventually, this design was successfully tested in simulations of a VHDL test bench as well as in a hardware test bench. These tests proved the fitness of the CAN controller for use in deep space applications.**

***Index Terms* — Data buses, Hardware design languages, Integrated circuit radiation effects, Space vehicle computers, Transceivers**

I. INTRODUCTION

Over the past two decades, the controller area network (CAN) bus [1] has been widely adopted in automotive industry and automation. In the last couple of years there has been a growing interest in this bus for space applications. The low development cost, reliability, priority based bus access and presence of commercial hardware make CAN bus a good alternative to classic spacecraft communication busses, such as the MIL1553b [2] and the European Space Agency onboard data handling (ESA OBDH) bus [3]. The bus has already been used in recent low earth orbit (LEO) missions, but an implementation for deep space applications has never been established. In this context reliability and radiation hardness are major concerns.

This paper describes the implementation of a CAN bus in an advanced data and power management system (ADPMS), i.e. an onboard computer for small satellites. A possible approach involves the use of commercial off-the-shelf (COTS) components, an approach that has been adopted in LEO satellites. This method is without doubt the most cost effective, but doesn't meet all requirements such as fault tolerance and reliability. Another method that is more expensive implies the use of space qualified CAN components. However, these components are still in development and not ready for deep space applications. The third option includes the design of custom built space qualified components. Therefore, a radiation hardened implementation of a CAN transceiver needs to be designed, as well as a radhard CAN controller. Because this last option is the only one that meets the strict demands of deep space missions, it was studied, elaborated and tested in detail. Therefor a CAN controller was designed in the very high speed integrated circuit (VHSIC) hardware description language (VHDL) for implementation in a radiation hardened field programmable gate array (FPGA). This FPGA drives a compact peripheral component interconnect (cPCI) module that can be plugged into an ADPMS. The design was tested in simulations as well as in a hardware test bench.

This paper is organized as follows. First a general description of the CAN protocol is given, together with the space requirements such as fault tolerance and reliability. Afterwards the possible solutions are considered, ranging from COTS components to custom built components. This last option is completely elaborated. A CAN transceiver is put together with a radhard RS-485 transceiver and a CAN controller is designed with the help of intellectual property (IP) cores and an FPGA.

Next (Section III), the design is tested by means of computer simulations of a VHDL test bench. After these tests, the design can be tested in a hardware test bench. The tests are executed as much as possible in accordance with the application-specific integrated circuit (ASIC) and FPGA development guidelines of the European Cooperation for Space Standardization (ECSS) [4]. This enables us to assess the success of this project (Section IV), in order to facilitate future implementations.

II. THEORY

A. CAN bus in space

The CAN bus was developed in the late 80's by Robert Bosch GmbH [5] for use in the automobile industry and was subsequently standardized as ISO 11898. This protocol only defines a part of the physical layer and the data link layer. The application layer can be implemented in various ways, however CANopen [6] has been pointed to be the most appropriate protocol in space applications, according to ECSS [7]. The high reliability of the CAN bus, together with reduced costs and a multimaster structure, make this bus an ideal medium for onboard communication in space applications.

In spite of the proven reliability in terrestrial applications in harsh environments, a CAN bus implementation according to the ISO 11898 does not meet space standards. This is why ECSS published its recommendations for

implementing a CAN bus in space systems [7]. The CAN bus has already been implemented in several LEO satellites, which operate at an altitude of 100 km to 1000 km. At these altitudes, the radiation levels are still relatively low and the environmental characteristics such as temperature and electromagnetic compatibility are comparable to the situation in a car's engine block. This is why COTS components are usable in LEO missions [8][9][10]. Since the use of COTS components for CAN bus implementations in LEO missions, no failures have been encountered. However, a redundant bus should be foreseen [11]. In the case of a hot redundant bus, both CAN busses actively participate in transfers. A cold redundant bus is easier to implement but cannot recover lost messages, since only one bus is active at a time. The implementation of a redundant bus requires a form of redundancy management, an option that is not foreseen in the ISO 11898 or CANopen standard.

In deep space applications, the use of COTS components is not appropriate because of the high temperature range and radiation levels. They don't provide the fault tolerance that is needed because it would raise development and production costs [12]. Besides, these components don't offer the long term survivability that is required for deep space missions. In order to resolve these issues while maintaining low development costs, COTS IPs can be transferred to space qualified ASICs [13].

It's also possible to use space qualified instead of COTS components, but in this domain the choice of CAN components is very limited. There is only one radhard CAN controller available, the AT7908E from Atmel. But this controller doesn't support redundant CAN busses and it's not compatible with the onboard PCI bus of the ADPMS, so the only solution that is left, is the implementation of a CAN controller in a radhard FPGA, the Actel RTAX2000S/SL.

The CAN transceiver can be implemented in various ways. The most logical option involves the use of CANTRAN, a radiation hardened CAN controller from Aurelia Microelettronica. But since this design is currently only available in DIL28 package, it's not usable for deep space missions because of its size and weight. A CAN transceiver built of discrete components also isn't usable because of these reasons. That's why an implementation with a radiation hardened RS-485 transceiver [5] is the best option.

B. Hardware implementation

1) *CAN transceiver:* The CAN transceiver is based on a radhard RS-485 transceiver with the drive enable input connected to the CAN drive output from the CAN controller, as described in [14]. At the CAN bus side of the transceiver, a biasing network is added in order to terminate the network with the characteristic impedance (120 Ω) and to regulate voltage levels on the CAN bus in undriven states.

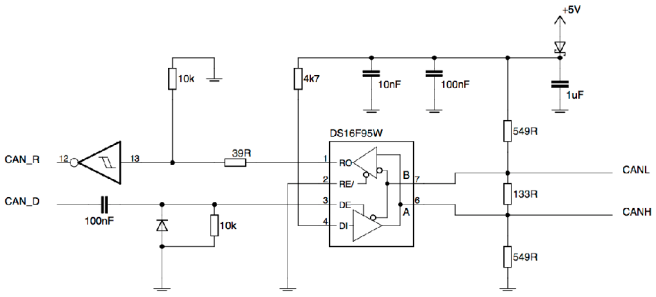

Fig. 1. Radhard CAN transceiver

The values of the resistors in the biasing network are calculated as follows. At first the differential voltage on the CAN bus is calculated in equation 1.

$$V_{CAN\_diff} = \frac{V_{cc}}{\frac{1}{R_1} + \frac{1}{R_{in}} + 2 \cdot \left(\frac{1}{R_{term_1}} + \frac{1}{R_{term_2}} \right)} \tag{1}$$

The RS-485 standard [15] demands a minimum load of 375 Ω on both outputs of the transceiver, which is represented by equation 2.

$$\frac{1}{R_1} + \frac{1}{R_{in}} \geq \frac{1}{375} \tag{2}$$

The input impedance of the transceiver is AC coupled and fixed at 120 Ω, the characteristic impedance of the CAN bus network. This is represented by equation 3.

$$\frac{1}{2 \cdot R_1} + \frac{1}{R_{term_1}} = \frac{1}{120} \tag{3}$$

A substitution of equation 2 and 3 in equation 1, together with a 120 Ω value of R_{term_2}, yields equation 4.

$$R_1 = \left(\frac{V_{cc}}{V_{CAN\_diff}} + 1 \right) \cdot \frac{250}{9} \tag{4}$$

This equation enables the calculation of the bias resistors. The minimum differential input voltage of the RS-485 transceiver is 200 mV. Together with a noise margin of 50 mV and a V_{cc} voltage of 4.7 V, this equation results in bias resistors of 550 Ω, practically realizable with 549 Ω resistors.

The value of the termination resistor can then be calculated with equation 3:

$$R_{term_1} = \frac{1}{\frac{1}{2 \cdot 549} + \frac{1}{120}} = 134.72\,\Omega \tag{5}$$

This value can be realized with a 133 Ω resistor.

2) *CAN controller:* As mentioned in section 2.1, the CAN controller is implemented in a radhard FPGA with the help of VHDL IP cores, interconnected with an on-chip advanced microcontroller bus architecture (AMBA) advanced high-performance bus (AHB). The most important IP cores are the PCI bridge from QinetiQ Space NV for communication with the ADPMS and the CAN core to implement CAN bus functionality. From all available CAN cores, the GRCAN IP core from Gaisler Research seemed the best fit for integration in the ADPMS. Unlike other CAN cores, this core provides an AHB interface, direct memory access (DMA) and a cold redundant CAN bus. The only disadvantage is the use of an advanced peripheral bus (APB), which can be solved with an APB to AHB translate core.

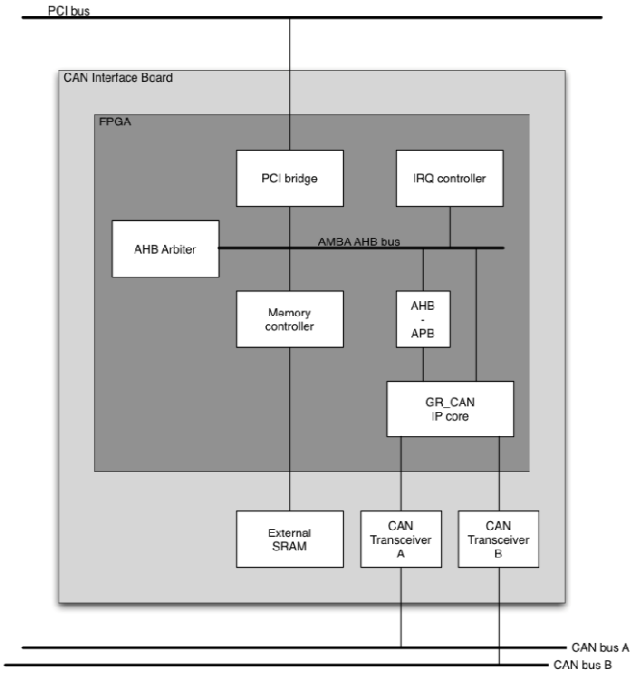

Fig. 2. CAN interface module with radhard CAN controller

III. EXPERIMENTS

One of the most important steps in the development of space applications, is testing. All tests are being executed as much as possible in accordance with [4]. This means that the complete design is first tested in simulations and afterwards in a hardware test bench.

A. Simulations

At first, a complete test bench is created in VHDL code. Simulations of this test bench have to prove the correct operation of the FPGA design. The VHDL test bench is built of several simulation models, such as a simulation central processing unit (SCPU), a static random-access memory (SRAM), some CAN test nodes and a model of the CAN interface board, as can be seen in figure 3. These models are only intended for simulation purposes and contain no synthesizable code. The SCPU from QinetiQ Space NV enables us to execute read and write commands in registers or memory. It is connected with other simulation models through a PCI bus. The executable assembler commands are loaded from a test script at the start of a simulation. The SRAM model from QinetiQ Space NV is meant for the setup of receive and transmit buffers of the CAN controller. The design of the CAN controller ─which is being tested─ is implemented in a simulation model of the CAN interface cPCI module that also contains simulation models of two CAN transceivers. These transceivers connect the CAN interface module with two CAN networks in the test bench, a nominal and a redundant one. The CAN networks are connected to several CAN test nodes, simulation models that can send CAN frames or check received frames on one of the CAN busses. The test nodes were constructed with a high level of autonomy, in order to enable automatic CAN frame construction and control of received frames.

As demanded in [4], the operation of all simulation models is tested in a separate test bench before implementing them in the general VHDL test bench of the CAN controller.

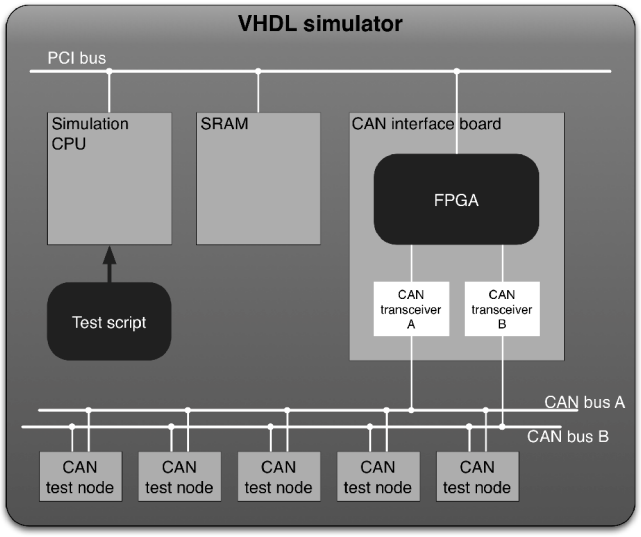

Fig. 3. VHDL test bench

After writing a test bench, the basic functionality of the CAN controller is tested with assembler test scripts for the SCPU. A first test proves the correct read and write capabilities of every register of the CAN IP core. The subsequent data reception test was also successful, which proves that CAN messages can be received correctly and stored in a buffer in external memory. The last test however revealed a problem with the CAN IP core from Gaisler Research. A data transmission could not be initiated by the CAN IP core because of a bug in the AHB part of the core. A bug fix in version 13 of this IP core solved the issue, resulting in a successful data transmission test. A code coverage test showed that 79.9 % of the statements and 44.6 % of the branches were covered during the simulations. These results are far below the requirements of [4], that demand a coverage of respectively 90 % and 85 %. As a consequence, more testing will have to be performed before a final implementation for space applications can be manufactured, but a proof of concept has been delivered.

B. Hardware tests

After successful simulations, the design can be tested thoroughly in a hardware test bench. Therefor the design of the CAN controller is programmed in an FPGA, which is

implemented in a prototype of the CAN interface module. Apart from that, the radiation hardened CAN transceivers are implemented in the same prototype. Thereafter, the prototype is implemented in a test system built on a cPCI backplane. This test system is connected to another test system through a double CAN interface. The second test system contains a commercial CAN test card and runs commercial monitor and test software. This setup is visualized in figure 4.

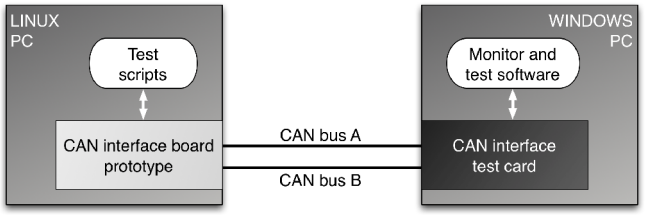

Fig. 4. Hardware test bench

The hardware tests can be divided into three categories. The first tests verify the functional behaviour of the CAN controller. These functional tests include a register access test and reception and transmission tests for different sorts of CAN frames, such as standard and extended remote transfer request (RTR) and data frames. The difference between standard and extended frames lies in the length of the identifier field of the CAN frame. RTR frames are transmitted as a request for data, whereas a data frame contains 0 to 8 bytes of information.

Consequently, several error injection tests are performed in order to verify the behaviour of the CAN controller in abnormal circumstances. First the behaviour of the controller is observed when no acknowledgement is being generated by the receiving node. In the next test a frame reception on the inactive (redundant) bus is tested and finally a buffer overrun test is executed. In this last test, an amount of data larger than the receive buffer is transmitted to the CAN controller.

Finally the design of the CAN controller is put through two performance tests. These tests verify the operation of the CAN controller in case an amount of data equal to the size of the receive or transmit buffer is being received or transmitted at the maximum bitrate of 1 Mbps.

All hardware tests finished successfully. The functional tests proved the correct operation of the CAN controller in normal circumstances, whereas the error injection tests proved a correct error handling. Also the performance tests didn't result in any anomalies and a data transmission rate of 496 kbps could be measured, a number that can deviate slightly, depending on the data content.

IV. CONCLUSION

An implementation of a CAN bus for space applications can be achieved in different ways, but for deep space applications only one option is appropriate. Since COTS components don't offer the required reliability and since space qualified CAN components are still to be finalized, the design of custom components is the only option left.

The CAN transceiver can be constructed with an RS-485 transceiver and a biasing network. The CAN controller can be programmed into a radhard FPGA with the help of VHDL IP cores.

As a test, the GRCAN IP core from Gaisler Research was integrated in a CAN controller design. This design was first tested in simulations and consequently in a hardware test bench. The tests revealed a bug in the GRCAN IP core, but as of version 13 of the core, all test requirements were met. This proves the GRCAN core's feasibility for use in space applications, however further testing needs to be performed, particularly in simulations. These tests only covered 79.9 % of the statements and 44.6 % of the branches instead of the required 90 % and 85 %.

ACKNOWLEDGMENTS

This research was made possible thanks to the structural and intellectual supplies of QinetiQ Space NV.

REFERENCES

[1] ISO 11898-1:2003 - Road vehicles ─Controller area network (CAN)─ Part 1: Data link layer and physical signalling. 2003.
[2] MIL-STD-1553b: Aircraft internal time division command/response multiplex data bus. 1979.
[3] ESA OBDH TTC-B-01: Spacecraft Data Handling Interface Standards. 1978.
[4] ECSS_Q_ST_60_02 Working Group. Space product assurance: ASIC and FPGA development. Technical report, ESA-ESTEC, 2008.
[5] Bosch, R. CAN specification version 2.0. 1991.
[6] BS EN 50325-4: Industrial communication subsystem based on ISO 11898 (CAN) for controller-device interfaces. CANopen. 2002.
[7] Onboard bus LAN working group,. Recommendations for CAN bus in spacecraft onboard applications, Draft 2.1. Technical report, ECSS, 2005.
[8] Woodroffe, A. and Madle, P. Application and Experience of CAN as a low cost OBDH bus system. *Data systems in aerospace, 2004. Proceedings.*, 2004.
[9] Klaus Janschek and Annerose Braune. Application of industrial CAN bus technology for LEO-satellites. *Acta Astronautica*, 46(2-6):313 - 317, 2000. 2nd IAA International Symposium on Small Satellites for Earth Observation.
[10] Khurram, M. and Zaidi, S.M.Y. CAN as a spacecraft communication bus in LEO satellite mission. *Recent Advances in Space Technologies, 2005. RAST 2005. Proceedings of 2nd International Conference on*, pages 432 - 437, 2005.
[11] Jun Yang and Tao Zhang and Jingyan Song and Hanxu Sun and Guozhen Shi and Yao Chen. Redundant design of A CAN BUS Testing and Communication System for space robot arm. *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 1894 -1898, 2008.
[12] Tai, A.T. and Chau, S.N. and Alkalai, L. COTS-based fault tolerance in deep space: Qualitative and quantitative analyses of a bus network architecture. *High-Assurance Systems Engineering, 1999. Proceedings. 4th IEEE International Symposium on*, pages 97 -104, 1999.
[13] Savio Chau. Experience of using COTS components for deep space missions. *High-Assurance Systems Engineering, 1999. Proceedings. 4th IEEE International Symposium on*, pages 116, 1999.
[14] Dr. Emrich, A. CAN application in avionics. Technical report, Omnisys Instruments, ESTEC, 2001.
[15] ANSI/TIA 485-A1998, Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems. 2003.
[16] Plummer, C. and Roos, P. and Stagnaro L. CAN bus as a spacecraft onboard bus. *Data systems in aerospace, 2003. Proceedings.*, pages 51 - 62, 2003.
[17] Tortosa Lopez, F. and Plummer, C. and Roos, P. and Stagnaro L. and Storni, B. The CAN bus in spacecraft on board applications. *Data systems in aerospace, 2004. Proceedings.*, 2004.