Design and Implementation of CAN-bus Controller using Redundancy Management




Mohammed Ismail Muqtadir
M.Tech, VLSI Design Engineering, Electronics and Communication Engineering, VIF College of Engineering and Technology.

B.Kotesh
Head of the Department, Electronics and Communication Engineering, VIF College of Engineering and Technology.

Korani Ravinder
Assistant Prof, Electronics and Communication Engineering, VIF College of Engineering and Technology.

Ms. Imthiazunnisa Begum
Head Of the Department, Electronics and Communication Engineering, VIF College of Engineering and Technology.

Abstract:

At present, the technique of dual redundancy CAN-bus is mainly implemented by software, so that it has the disadvantages of low reliability and bad real-time performance. Built on the error handling rule in CAN specification version 2.0, a hardware redundancy management unit is creatively put forward in this paper. Based on FPGA, a kind of customized Dual Redundancy CAN-bus Controller (DRCC) is designed. By implementing the design into a XILINX's SPARTAN-3 chip to test, it has been verified that the design could completely meet the requirement for high real-time performance and reliability, with a bright prospect for the future.

Keywords:

Dual Redundancy CAN-bus; Verilog; FPGA; IP Core.

I.INTRODUCTION:

With the development of EDA (Electronic Design Automation), digital system designed by FPGA is widely used in all kinds of fields [1] such as communication, aerospace, medical treatments and industrial control system [7]. CAN (Controller Area Network) has become one of the most popular data bus [2] with characteristics such as anti-interference capability, much lower cost and easy maintenance.

There are a great number of CAN chips in market for example PHILIPS' SJA1000 [3]. No matter how perfect the single-channel CAN bus network is, while something happens to the single-channel bus network such as short circuit or open circuit, the whole network won't work. To solve this problem, some concepts of redundancy were put forward in the past. To sum up, there are three kinds of means of redundancy data bus [4, 6, 8].

The first is redundancy of bus driver, which employs one CPU, one CAN controller and two bus drivers. The second is redundancy of bus controller, which employs one CPU, two CAN controllers and two bus drivers. The last is redundancy of software system, which employs two CPUs, two CAN controllers and two bus drivers. But those redundancy means is done by software running in the CPU which has the disadvantages of low reliability and bad real-time performance [14, 17, 18]. So the best redundancy means is that redundancy management is done by hardware logic circuit. But a CAN controller chip is usually a whole component whose function cannot be modified. Thus, a Dual Redundancy CAN-bus Controller (DRCC) based on FPGA chip, a programmable logic component, is put forward in this paper.

II.DUAL REDUNDANCY CAN-BUS (DRC) NETWORK ARCHITECTURE:

The DRC Network architecture is shown in Fig.1. Compared with physical layer of a single-bus CAN network, physical layer of the DRC Network is added an additional channel. In single-bus CAN network, if its only channel is severely interfered or open, the Network will be corrupted. But the DRC Network's physical layer has two completely independent channels, which are Channel 1 and Channel 2 respectively. If the redundancy management fails to transmit message from one channel, it will transmit the message automatically from the other channel.

III. DUAL REDUNDANCY CAN-BUS CONTROLLER DESIGN:

A. DRCC Structure:

The block diagram of DRCC is shown in Fig.2. DRCC is composed of two Bit Stream Processor Blocks (BSPB), one Redundancy Management Block (RMB) and two RAM Blocks.

The BSPB includes one state-machine and one Bit Timing Logic Block (BTLB). The function of several blocks of DRCC can be described as follows:

BTLB [12] monitors the serial CAN-bus line, manages the bus line-related bit timing, does hard synchronization and resynchronization, compensates for the propagation delay times and controls the sample point and the number of samples to be taken within a bit time.

BSPB takes charge of Date Link Layer protocol and manages CAN Message such as recognizing and handling standard frame and extended frame, managing FIFO and filtering Message etc.

RMB manages transmission of CAN Messages while DRCC runs in redundancy mode, and it doesn't work while DRCC runs in normal mode. The block consists of include some "glue" logic and three state-machines which a main state machine and two auxiliary state-machines.

The main state machine manages channels switch, latches bits of the time counter when finishing sending message or switching channels and the two auxiliary state-machines monitor whether a channel is valid and report its state to the main state-machine.

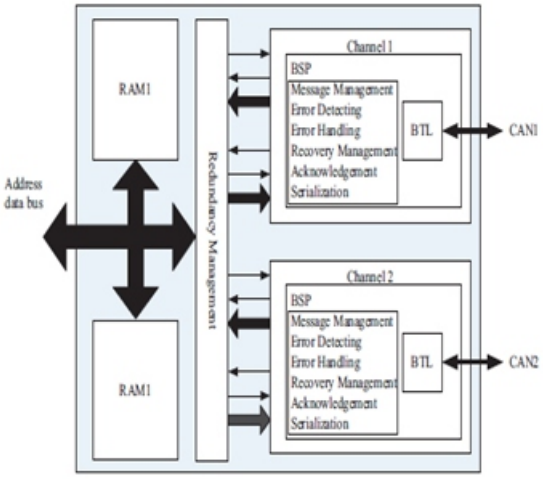

Fig 1. Dual Redundancy CAN-bus Controller Block Diagram

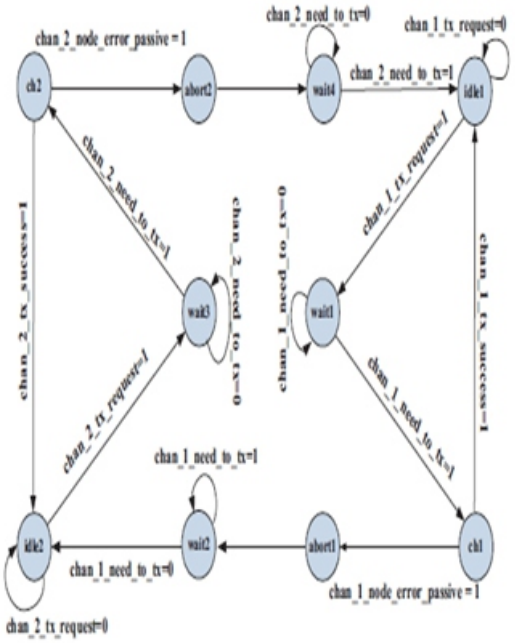

Fig 2. State transition diagram

Two RAMs are used to buffer messages waiting for being transmitted, to buffer received messages and to register all kinds of states which DRCC runs.

B. Redundancy management state-machine:

The RMB includes three state-machines, a main state machine and two auxiliary state-machines. The state transition diagram of main state-machine is shown in Fig.3 and its each state is described in Table 1.

IV.CAN BUS FRAME :

The data link layer defines the CAN bus frame structure, error handling mechanism, and medium access, Bosch specification describes two standards.

- CAN 2.0A (standard) – with 11 bit identifier.
- CAN 2.0B (extended) – with 11 and 29 bit identifier.

Due to the nature of the information transmitted the different types of frames are used.

- data frame.
- remote transmission request frame,
- error frame.
- overload frame.

The data frame structure with 11 bit ID is shown below:

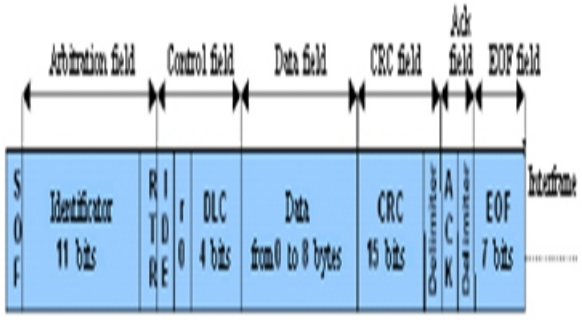

Fig 3: Data Frame Structure

The Respective Frame bits are:

- SOF (Start of Frame) bit - dominant bit, which marks beginning of the frame.

- ID (Identifier) - 11-bit identifier field.

- RTR bit - recessive state on this bit tells that we are dealing with Remote Transmission Request frame, otherwise data frame.

- IDE (Identifier Extension) - recessive state on this bit marks 29-bit identifier frame, otherwise 11-bit ID frame.

- r0 - reserved bit - can be dominant or recessive.

- DLC (Data Length Code) - 3-bit field describes how many data bytes, frame contains, or how many data bytes are requested in case of RTR frame.

- Data field - can contain from 0 do 8 data bytes.

- CRC (Cyclic Redundancy Check) field - 15-bit checksum field. The calculation of the checksum is based on bits starting from the beginning of the frame. This field ends with checksum delimiter (CRC-D), which is always recessive. It is worth to remember, that this field is calculated without stuff bits.

- ACK (Acknowledge) - 2-bit acknowledge field (ACK slot, ACK-D - delimiter). It is used to acknowledge correct frame transmission by a reception node node.

- EOF (End of Frame) - 7-bit end of frame field, which is stream of recessive bits.

- INT (Interval) - 3-bit frame interval - always recessive.

After each transmitted frame a 3-bit inter-frame must occur.

In case of the 29-bit identifier frame the IDE bit (in recessive state) indicates, that successive 18-bits are a part of ID as well. A Substitute Remote Request (SRR) bit is also included in the Arbitration Field. This bit is always transmitted as a recessive bit and ensures that, in the case of arbitration between a Standard Frame and an Extended Frame, the Standard Frame will always have higher priority if both messages have the same value in the 11-bit part.

STATE	FUNCTION	DESCRIPTION
idle1	root state idle1 state	If system has some messages to send, first write the messages to buffer, and then set the chan 1 tx_request. When the main state-machine monitors this change, starts transmitting process and changes to wait1 state.
wait1	wait1 state	If Channel1 isn't ready for transmitting message, the state-machine will still wait. Otherwise, change to ch1 state.
ch1	channel 1 send state	If Channel 1 transmits a message successfully, the state-machine returns to idle1 state and will be ready for transmitting next message from the Channel1. Otherwise, the state-machine changes to abort1 state in order to abort the message which wasn't transmitted successfully from Channel 1.
abort1	channel 1 abort state	The state-machine sets chan_1_abort_send signal of Channel 1, and then changes to wait2 state.
wait2	wait2 state	If abort the corrupted message from Channel 1 successfully, the state-machine changes to idle2 state in order to transmit the same message from Channel 2. Otherwise, will still wait.
idle2	idle2 state	If The past state is wait2 state, the state-machine is directly into wait3 state. Or else, the state-machine needs to wait for chan_2_tx_request signal which is the request signal of Channel 2.
wait3	wait3 state	If Channel 2 isn't ready for sending message, the state-machine will still wait. Otherwise, changes to ch2 state.
ch2	channel 2 send state	If Channel 2 transmits successfully, the state-machine returns to idle2 state and will be ready for next message from the Channel2. Otherwise, the state-machine changes to abort2 state in order to abort the corrupted message from Channel 2.
abort2	abort state	The state-machine sets chan_2_abort_send signal of Channel 2, and then changes to wait4 state.
wait4	wait state	If abort the message from Channel 2 successfully and if Channel 1 recovers from faults, the state-machine changes to idle1 state. Otherwise, changes to idle2 state and latches failure states.

Table 1. state description

It is easy to calculate, that using 11 bit ID it is possible to address 2048 nodes, however specification says, that 7 MSB bits of ID cannot be recessive simultaneously, thus the effective number of addresses is 2032 (2048- 24). For the 29 bit identifier this number grows to about 536 millions. It is physically not possible, because every node increases bus capacity distorting signals on it. Usually the number of nodes shall not be greater than 32 (at maximum speed).

It was mentioned before, that medium access is provided by the "bit arbitration". Each node listen what is going on the bus at the moment, and knows when the medium is available. When it starts transmit data, all other nodes switch into reception mode. Often two or more nodes are trying to transmit data at the same time. In this case "wired and logic solves the problem.

V.RESULTS

Simulation Result:

Fig 4: Simulation Result of CAN bus Controller

RTL Schematic:

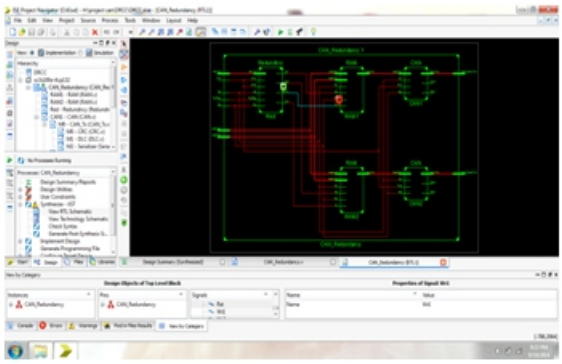

Fig 5: RTL Schematic of CAN bus Controller

VI.CONCLUSIONS:

The DRCC, which is written by synthesizable, behavioral Verilog language, can be used as a component in a project and it must have had a bright prospect for the future.

By implementing the design into a XILINX's SPARTAN-3 chip to test, the design of Dual Redundancy CAN-bus Controller Based on FPGA is successful. It guarantees reliability and real-time performance and compensates for the disadvantage of software redundancy.

REFERENCES:

[1] Ma Xiaojun, Tong Jiarong, "Design and Implementation of A New FPGA Architecture," ASIC, 2003. Proceedings. 5th International Conference, Vol.2, pp.816-819, October 2003.

[2] Yu Zhu, Can and FPGA Communication Engineering: Implementation of a Can Bus Based Measurement System on an Fpga Development Kit, Diplomica Verlag, 2010.

[3] Philips Semiconductors. SJA1000 Standalone CAN controller. January 2000.

[4] Qing Jia, DeviceNet media redundancy iCC 2005.

[5] Robert Bosch GmbH, CAN Specification Version 2.0, September 1991.

[6] Jos´e Rufino, Dual-Media Redundancy Mechanisms for CAN, Technical Report, January 1997.

[7] CiA - CAN in Automation. CAN Physical Layer for Industrial Applications - CiA/DS102-1, April 1994.

[8] C. Mateus, Design and implementation of a non-stop Ethernet with a redundant media interface. Graduation Project Final Report, Instituto Superior T´ecnico, Lisboa, Portugal, September 1993. (in portuguese).