Design and Implementation of CAN Frame Bit Disturbance based on CAN IP Core

Anyu Cheng [a], Xiaofeng Meng [b], Bingyang Chen [c], Dawei Sun [d]

School of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

[a]chengay@cqupt.edu.cn, [b]mengxf187@163.com, [c]chenby@sumarte.com, [d]sundw@sumarte.com

Abstract

This paper has designed a CAN controller soft core using the VHDL language and with the reference of CAN2.0 bus protocol. This core architecture adopts hierarchical design thought and consists of three parts, which are registers, bit timing logic and bit stream processor. Referencing to the fault types about link layer and physical layer in the CAN protocol, combining FPGA with SOPC design method, this paper design and implement CAN network interference test system. A new CAN bus system to test the CAN controller soft core is built based on the SOPC technology on different series of FPGA. After many times of experiments, the results show that this core can realize different baud rate communication and disturbance with good performance, and be transplanted conveniently.

Keywords

CAN2.0 Bus Protocol, Soft Core, CAN Bus System, SOPC.

1. Introduction

SOPC is a kind of special embedded system chip with programmable logic technology to integrate the electronic system including processor, memory, I/O port, DSP, bus and so on. Assembling the various IP core into a large system-on-chip is the essence of designing a system with SOPC. With the emergence of high-density FPGA and the development of SOPC technology, the programmable system implementation based on FPGA has also been widely used.

CAN bus is widely used in industrial field for the reliability, high transmission rate, long transmission distance and other advantages. The key of designing the CAN bus system is to design and use the CAN controller. The normal and easy design is to use a MCU and CAN controller, or to select the MCU embedded CAN controller with CAN interface [1,2]. In the concept of the modern electronic system design, the system described above is low integration, low reliability, not versatility. The problem above can be solved by designing the CAN controller into soft core on FPGA with the SOPC technology. Simultaneously, other digital circuit can be integrated by using the FPGA's resources remained to effectively reduce the number of peripheral chips.

Based on the advantages of soft core designed above, this article designs a CAN controller soft core, referring to the standard CAN2.0 protocol. The soft core that specific circuit function programed with hardware description language, with the advantages of being reused, clipped and transplanted, can enhance the efficiency and flexibility of system design.

2. Design of CAN Controller Soft Core

CAN controller is the kernel of CAN system, that mainly completes the communication of CAN network and network protocol. For the external microprocessor (CPU), the CAN controller is a I/O devices mapped memory, including all hardware and features to control communication of CAN network [5]. The documents for SJA1000 that independent CAN controller designed by Philips are comprehensive, structure clear. So this paper takes SJA1000 as a blueprint to build the CAN controller structure. The structure of SJA1000 is consisted of seven function modules that are

interface management, bit timing logic, bit stream processor, error management, send buffer, receive buffer and acceptance filter. The structure of CAN controller is shown in Fig. 1.

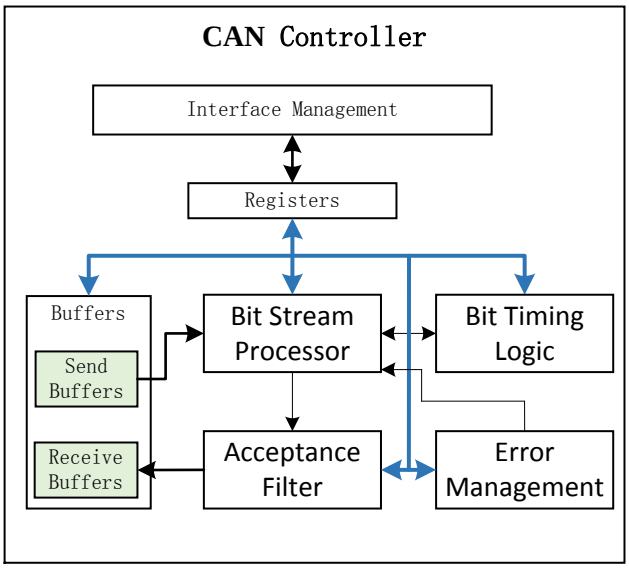

Fig. 1 CAN Controller Diagram

2.1 Register Block

CAN controller registers consist of two parts: the control register and data register. The control register includes mode register, status register, command register, interrupt register and the others registers. It's total about seventeen kinds of registers. Each register has only one address uniformly distributed in order. So, the microprocessor could use the interface of management to read and write registers [3].

There are two mode selections: reset mode and work mode. All the registers could be changed by the microprocessor to initializes the CAN controller in reset mode. After initialization, the microprocessor could start or stop the transmission by sending the request command. When sending, the microprocessor writes the pending data to the transmission buffer, and then sets the transmission request bit in the command register to initiate the transmission. Upon receiving, the bit stream processor saves data in the receive buffer and requests the microprocessor to read. The receive buffer is a 64-byte FIFO that can continue to receive other messages while the CPU is processing a message. During the processing, the microprocessor stores the controller status and interrupt information in the status register and interrupt register respectively [3-6].

2.2 Bit Timing

The bit timing logic mainly task is to achieve the bus timing and synchronization. It would configures the bus timing parameters to adjust the transmission rate of the bus by dividing the system clock to the bus clock. At the same time, the CAN controller can monitor and sample the bus and transmit the pending message on the bus accordance to the setting time stream.

In accordance with the CAN protocol: the bus has been in a recessive position when no messages transmission on the bus. A bit of recessive to dominant, it will trigger a hard synchronization, that is a sign of starting to transmit a message. The hard synchronization is cleared off after the CAN controller sample. A resynchronization is triggered each time when a transition from recessive to dominant bit is detected and the transition falls outside the synchronization segment during the reception of the message. The bit timing could be incremented or shorten based on the edge of transition to ensure the proper sampling. Only one synchronization can be performed in a bit time [3-6].

2.3 Bit Stream Processor.

Being the CAN controller's core, bit stream processor's task is to frame, arbitration, response and bit encoding/decoding for message. The bit stream processor performs functions such as bus arbitration, bit stuffing, and CRC calculation in accordance with the CAN protocol, and performs functions such as removing bit stuffing, CRC check, data acknowledgment, reception filtering, error detection, and error calibration when receiving data [3-5].

Fig. 2 shows the packet transmission process. When sending a message, the microprocessor first reads the status register to check whether the bus is idle. If it is busy, it enters the receiving mode to receive message. If it is idle, it writes the message to the sending buffer and writes a request to the command register. The bit stream processor reads the messages in the transmit buffer, assembling the data and performing CRC calculation. Then it sends the serial data to the shift register for serial-to-parallel conversion, encoding the serial data (bit stuffing) to the bus. Simultaneity, the bit stream processor compares the transmitted data with the data sampled by the bit timing logic from the bus to determine whether having bus access. If the arbitration is lost, the packet is received. If bus access is granted, the sender continues to send data and compares the transmitted data with the sampled data to determine if there are any bit errors. If any error detected, then it reports errors to the error management logic and updates the arbitration lost capture registers and errors code capture register. After the data is sent, the sender waits for the receiver's response. If the response is received correctly, the transmission process ends and the bus enters the idle state. If the response is not received in time, the sender will retransmit the data [6,7].The packet received process,see Fig 3.

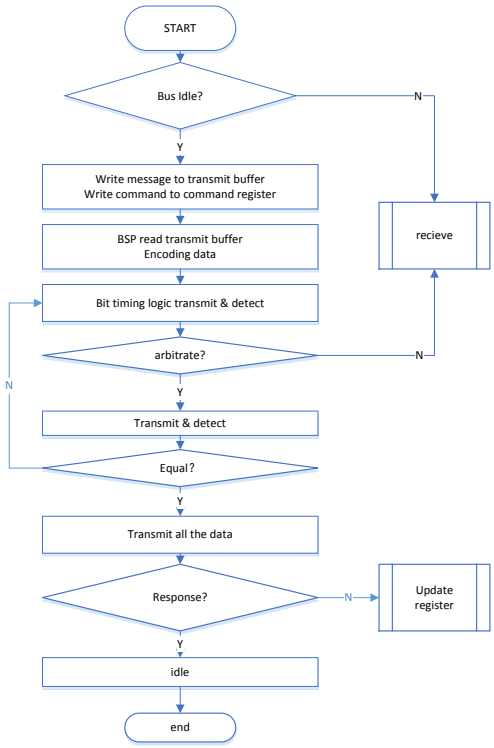

Fig. 2 Transmit flow chart

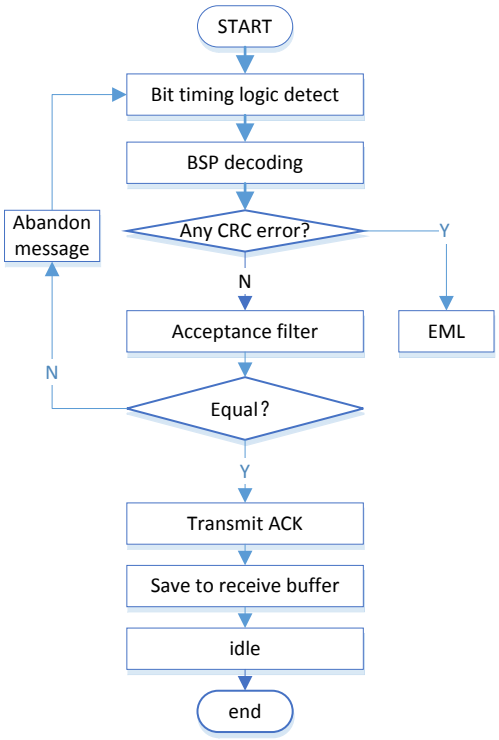

Fig. 3 Receive flow chart

2.4 Error Management state.

The CAN protocol defines five types of errors: bit errors, CRC errors, form errors, stuffing errors, and acknowledgment errors. When the bit stream processor detects any error on the bus, it will report the error to the error management logic. The error management logic adds, subtracts, or clears the receive error counter and the transmit error counter according to the twelve errors monitoring rules described in the protocol. The controller defines three faults: Error Activation, Error Acknowledge or Bus Off according to the error limit register. Once the receive error counter or send error counter reaches the limit, the error management logic will to make the appropriate fault handling based on the protocol error handling policy.The state of error management see Fig. 4.

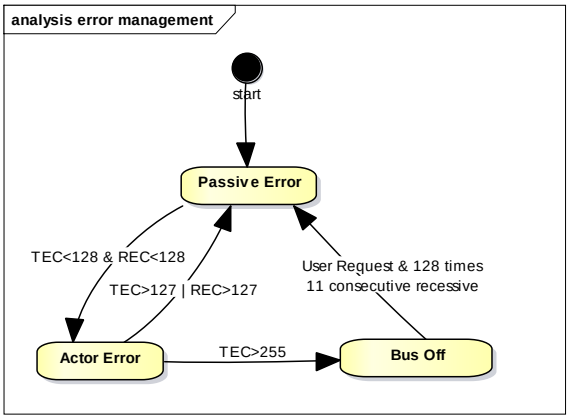

Fig. 4 error management state

2.5 Bit Sequence Trigger Detection and Distortion.

Message transmission have a different type of frames that are data frames, remote frame, error frame, overload frame.For a designated bus system, first to calculate the system transmission baud rate, to determine the message frame width of each bit.So, from the start bit of the frame, the controller translate a bit every other a few wide, high level as the dominant logic 0, low level of implicit logic 1. After the decoding of the data, the controller needs to parsing the meanings of the binary data representation frame message.

According to the CAN communication protocol, it has the fixed frame format characteristics for a certain type of frame. According to these characteristics, the controller could identify the category of frames transmitted over the bus and frame structure of potential field. Then,the controller will make the signal distortion corresponding to sequence preconfigured, forcing the bit stream transmiitting on the bus to distort.The execution flow chart shown in Fig 5.

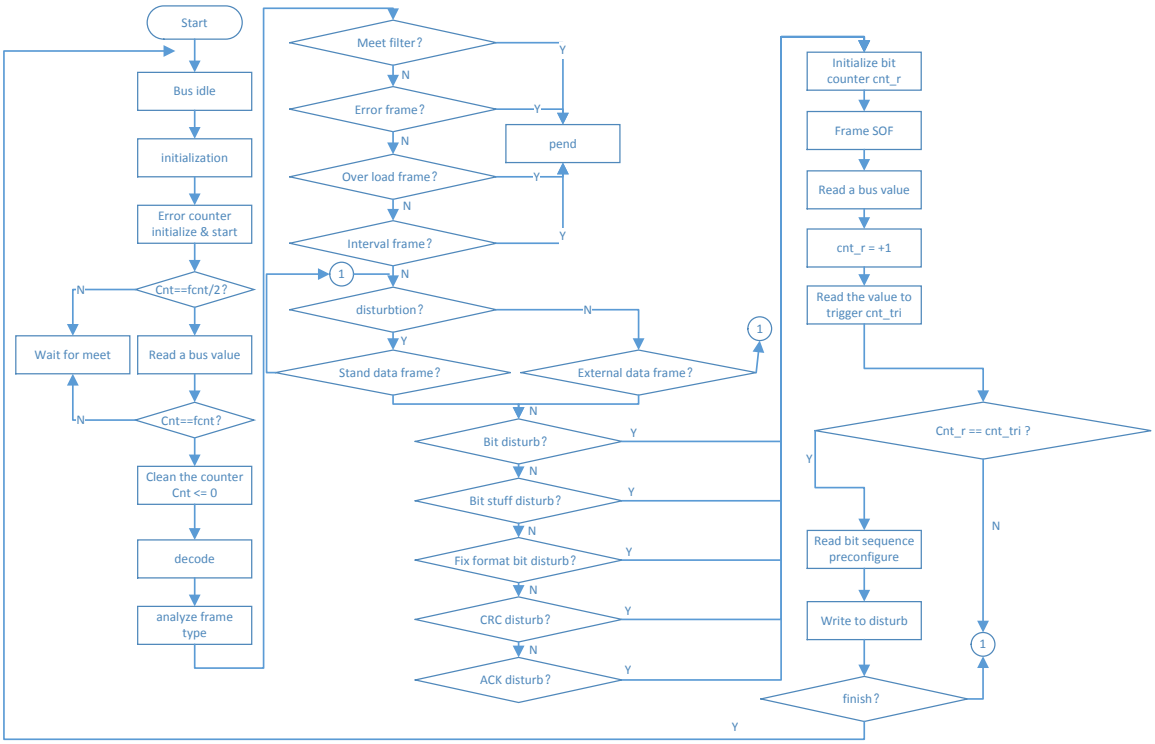

Fig. 5 distortion execution flow chart

3. Functional verification of the controller

When two nodes are set to send packets of different priorities on the bus at the same time, the PC receives the higher priority packets. This proves that CAN controller designed in this paper can deal

with bus collision according to non-destructive arbitration mechanism. In order to verify the error management capability of CAN controller, this paper makes the short circuit and open circuit fault of the bus system. The CAN controller designed in this paper can correct the errors according to CAN2.0 protocol. Using USB-CAN tool to monitor the network transceiver, the test process as follows:

1) Transmission test: Set the baud rate to 1Mbit/s, 500kbit/s and 250kbit/s respectively, and continuously send the message for 5 hours according to the 5ms interval and count the received packets on the PC.

2) Receiving test: When the baud rate is 1Mbit/s, 500kbit/s and 250kbit/s, the PC will continue to send messages for 5 hours according to the interval of 10ms. The CAN controller will reply each message.

3) Distortion test: This paper disturb the SOF, CRC and ACK field to validate the precept. And the system continue to test for 5 hours according to the interval of 10ms. The result shows that the system is stable and reliable.ACK disturb see Fig.6.

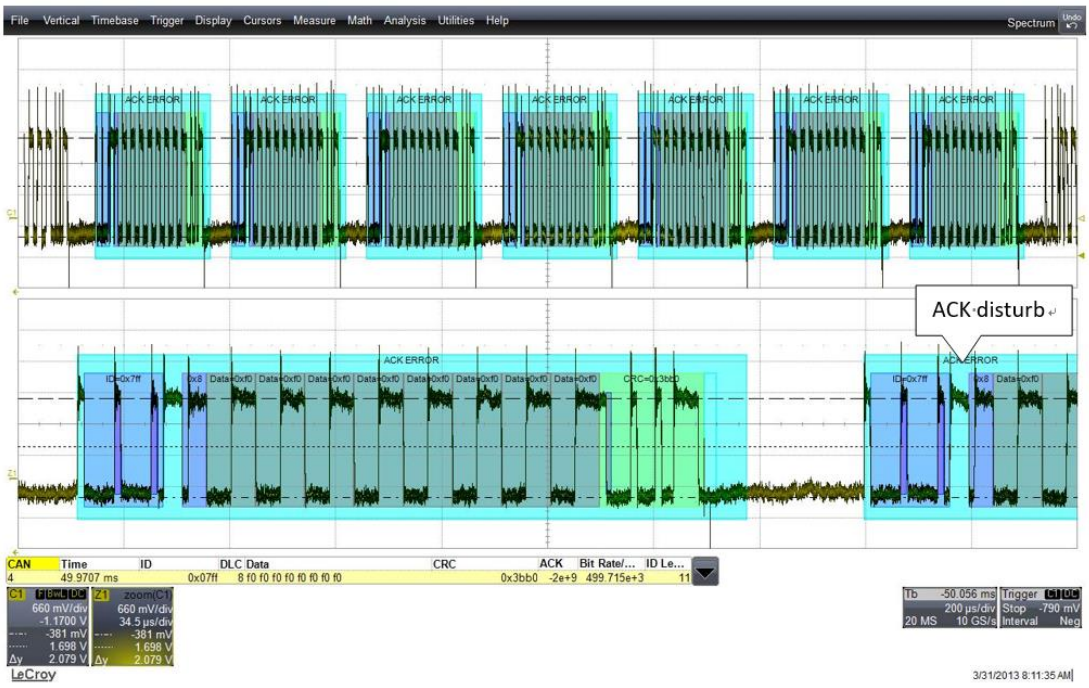

Fig. 6 ACK distortion chart

4. Platform Verification of the Controller

This article tests the soft core on the platform of Altera's FPGA for the utilization of resources and performance reports. The test results show that the soft-core resources designed in this paper occupy little logical. The soft core performance in different series of FPGA, the table one gives the maximum performance achieved. The user could adjust the frequency to determine the bus transfer rate according to requirements. In addition, the controller soft core is simulated with Modelsim 6.0a. The simulation results showed that it can conform to CAN2.0 protocol specification.The platform being tested,see Table 1.

Table 1. Controller on Altera FPGA

FPGA	Level	Logic Unit	Fmax(MHz)
Cyclone	-6	1955	100
Cyclone Ⅱ	-6	1925	100
Cyclone Ⅳ	-6	1970	100

5. Conclusion

In this paper, it designs a CAN controller soft core referencing to the structure and function of SJA1000. It is proved that the soft core implements CAN2.0 protocol standard, with less resource occupation and high performance. The CAN controller soft core designed in this paper can be used as a single chip external controller or integrated with the processor soft core to realize the design on FPGA. This paper verify that the distraction system is stability and reliable by bit interference test. When it is integrated into other systems, only the soft core interface should be modified. CAN bus control system integrated into the FPGA as a module, can effectively improve the system integration, to ensure system reliability.

References

[1] Anyu Cheng, Yan Meng, Xiaopin Wang, Jia Li. The design of the truck CAN network test system based on HIL[J]. International Core Journal of Engineering, 2015, 1(5): 75-81.
[2] Jun Cai, Jia Li, Yan Meng, et al, Vehicle CAN gateway static route design[J]. International Journal of Science, 2016, 3(2): 63-39.
[3] Hu Jian, Li Guangyan, Yu Xiangpeng, et al. Design and application of SAE J1939 communication database in city-bus information integrated control system development [C]// Mechatronics and Automation. 2007 International Conference on. Harbin: IEEE, 2007: 3429 -3434.
[4] Meng Liu, Behnam M, Nolte T. A stochastic response time analysis for communications in on-chip networks [C]//Embedded and Real-Time Computing Systems and Applications (RTCSA), 2015 IEEE 21st International Conference on. Hong Kong: IEEE, 2015: 237-246.
[5] Tindell K, Burns A, Wellings A J. Calculating controller area network (CAN) message response times[J]. Control Engineering Practice, 1995, 3(8): 1163-1169.
[6] Davis R I, Burns A, Bril R J, et al. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised[J]. Real-Time Systems, 2007, 35(3): 239-272.
[7] Navet N, Song Yeqiong, Simonot F. Worst-case deadline failure probability in real-time applications distributed over controller area network[J]. Journal of Systems Architecture, 2000, 46(7): 607-617.
[8] Tindell K W, Hansson H, Wellings A J. Analysing real-time communications: controller area network (CAN)[C]//Real-Time Systems Symposium (RTSS), 1994 IEEE. San Juan: IEEE, 1994: 259-263.
[9] Zeng Haibo, Natale M D, Giusto P, et al. Using statistical methods to compute the probability distribution of message response time in controller area network[J]. IEEE Transactions on Industrial Informatics, 2010, 6(4): 678-691.
[10] Broster I, Burns A, Rodriguez-Navas G. Timing analysis of real-time communication under electromagnetic interference[J]. Real-Time Systems, 2005, 30(1-2): 55-81.