The Design and Realization of CAN Bit Timing Logic

Guo Jinyan
1. Key Laboratory of Advanced Display and System Applications (Shanghai University), Ministry of Education
2. College of Mechanical and Electronic Engineering and Automation, Shanghai Key Laboratory of Power Station Automation Technology, Shanghai University
Shanghai 200072, China
Jennifer_sunny@163.com

Hu Yueli
1. Key Laboratory of Advanced Display and System Applications (Shanghai University), Ministry of Education
2. College of Mechanical and Electronic Engineering and Automation, Shanghai Key Laboratory of Power Station Automation Technology, Shanghai University
Shanghai 200072, China
huyueli@shu.edu.cn

***Abstract*—In the CAN bus, mistakes in bit timing will lead to serious decline in bus performance. How to deal with the bit timing of CAN bus communication decided whether the CAN controller can receive or transmit data correctly. According to bus protocol, bit synchronization will fix bit errors resulting from improper setting in many cases. This paper illustrated the structure of nominal bit time and the principle of bit synchronization, gave an optimized method of nominal bit time of CAN2.0 protocol, that the four non-overlapping segments of traditional nominal bit time is simplified to 3 non-overlapping segments, and on this basis, proposed a design method of bit timing processor of CAN bus based on the synchronized state machine. It also provides programmable time segments to compensate for the propagation delay times and phase shifts, and gave simulation and verification of the design. The results show that the design work which obeys the CAN2.0 protocol, can more easily deal with the CAN bus communication bit timing, realized the control of CAN bus protocol on the bit timing and bit synchronization, and better optimizes the CAN network.**

Keyword: Can bit timing, synchronization, CAN bus

I. INTRODUCTION

Controller Area Network, a serial communication bus defined by the ISO[1], was first put forward by the German Bosch Company. Due to its high reliability, no destruction of the arbitration, multi-master and easy to achieve such advantages, CAN have been widely used in the industrial control and automotive industries, and has become one of the most widely used field bus today[2].

Bit timing processor is used to control bit timing and bit synchronization in the CAN bus protocol. It can program one bit time in CAN bus communication, which is known as the nominal bit time, or as the bit period. On the one hand, CAN times sending-data-bits in accordance with the nominal bit time, and on the other hand, CAN synchronizes (hard synchronization and re-sync) the data on the bus to receive data [3].

In order to more easily deal with the bit timing of the CAN bus communication, this paper proposed a nominal bit time optimization method, which adopted a synchronous state machine design approach. At last, we gave some results of the simulation and verification of this design.

II. BIT TIMING-RELATED CONCEPTS

A. Nominal Bit Time

Nominal Bit Time, or bit cycle, is the transmission time of each bit. Nominal Bit Time in CAN2.0 agreement consists of four non-overlapping segments. Details can be obtained in the CAN2.0 protocol [4] [10]. This paper proposed a nominal bit time optimization method. In this article, Nominal Bit Time consists of three non-overlapping segments: synchronization segment (SYNC), the time buffer in segment 1 (TSEG1) and time buffer segment 2 (TSEG2). That is,

$$T_{BIT} = T_{SYNC} + T_{TSEG1} + T_{TSEG2} \qquad (1)$$

so $T_{TSEG1} = T_{PROP\_SEG} + T_{PHASE\_SEG1}$

$T_{TSEG2} = T_{PHASE\_SEG2}$

This division will not affect reliability and stability of the CAN data communication, makes the bit timing much simpler, and furthermore, will strengthen the CAN controller's versatility.

The bit value is closely tied to the bus level of the sample. The sampling point is a time point to receive the bus level and convert to the corresponding bit value. Bit sampling mode specifies the sampling frequency of the bus, including the one-time sampling and three samples. The relationship between Oscillator cycle, the system clock cycle and the bit cycle are shown as in Figure 1 [5-6].

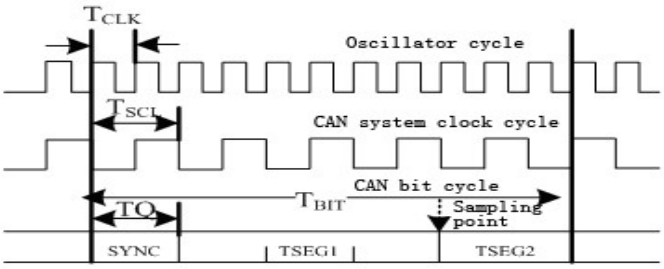

Figure 1 Bit cycle structure

B. Synchronization

In order to ensure that messages can be completely decoded, CAN Bus Specification introduced CAN bus synchronization. Through synchronization, the bus can effectively filter out the noise with less length than the sum of the propagation section and the buffer segment 1. In order to compensate for the phase error generated by the oscillator drift, propagation delay between nodes, or noise etc, CAN bus protocol defines two types of synchronization: hard synchronization and re-synchronization [12].

In one bit time, only one type of synchronization is allowed. When the bus transmission occurs from recessive to dominant transition at the beginning of a message, each bus controller in the CAN network will synchronize the bit stream on the bus. This synchronization is hard sync, that is, whether with phase error at this time, the bit time of all the nodes will start over.

When receiving a recessive to dominant transition outside the synchronization segment in the rest of each message, the controller will implement synchronization, which is called re-synchronization. If the transition edge occurs after SYNC and before the sampling points, the phase error is positive; if the transition edge occurs after the sampling points and before SYNC, the phase error is negative [9-10]. As bit filling is used in CAN coding, a re-synchronization will be implemented with a maximum interval between 10-bit cycle (i.e., five dominant bits followed by 5 recessive bits), which ensures the phase error will not be accumulated [6-8]. Re-synchronization mechanism is based on the jump edge to grow or reduce the bit time, and to adjust the location of sampling points to ensure proper sampling.

III. Design of Bit Timing Processor

A. General Structure Design

According to CAN2.0B protocol, the work process of bit timing processor can be divided into three steps. Firstly, complete the configuration of each time segment referred in the protocol. Secondly, detect the skipping edge and then synchronize according to the sync type (hard synchronization and re-synchronization). Thirdly, sample. Therefore, it's divided into baud rate generating logic, synchronizing logic and sampling logic. Figure 2 shows its structure.

According to the baud rate defined by the bit timing registers, baud rate generating logic can give the system clock cycle derived from the frequency dividing of the oscillator cycle, that is, the length of a time quantum. Then, according to the time segment and the length of each time segment defined by the bit timing register, we can get the counting of time quantum.

Sync Logic consists of synchronous detection circuit, synchronous state machines, phase detection circuit and the synchronous control latch.

Detecting sync type, prior to the implementation of synchronization, is performed by synchronous detection circuit.

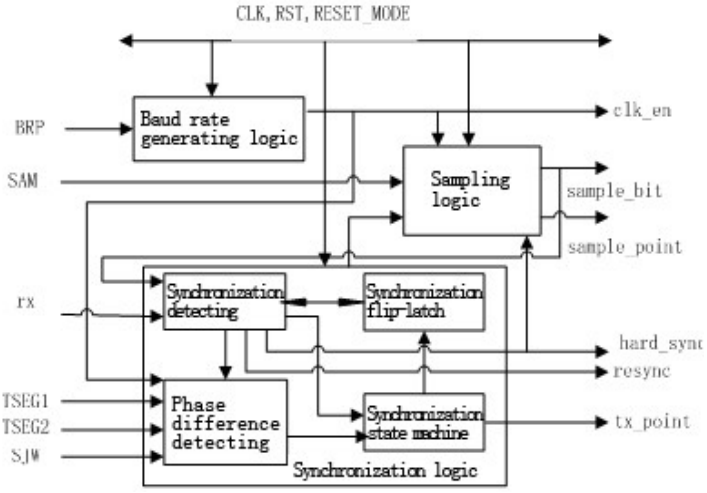

Figure 2 Bit timing processor architecture

Synchronous state machine controls synchronization logic to implement bit synchronization. When the synchronous detecting circuit detects the re-synchronization, the phase difference detecting circuit calculations simultaneously extend or shorten the phase synchronization, the maximum width of which is defined by the bit timing control register bits SJW. When the phase difference is greater than or equal to SJW, the width is equal to the SJW. When the phase difference is less than SJW, the width is equal to the calculation phase. Synchronous control latch can implement synchronization latch to meet the only synchronization in one bit time.

Sampling logic ensures sampling at TSEG1 to TSEG2 conversion. The SAM-bit in the bit timing control register controls sampling mode.

While single sampling, the sampling value (sample_bit) is equal to the end receiving value of TESG1, that is

$$sample\_bit = rx \qquad (2)$$

Three sampling means two to one arbitration. The sampling logic as follows:

$$sample\_bit = (x1 \,\&\, x2) \mid (x2 \,\&\, x3) \mid (x3 \,\&\, x1) \qquad (3)$$

X3 is the sample value of the current system clock cycle; X2 is the sample value of the last one system clock cycle; X1 is the sampled value of the clock cycle before last.

B. Design and Implementation of Synchronous State Machine

Synchronous state machine control synchronization logic to implement bit synchronization, which defines the five kinds of state: SYNC, SGE1, SEG2, WINDOW1 and WINDOW2. SGE1 and WINDOW1 is the front segment of the sampling (TSEG1), SGE2 and WINDOW2 is the back segment of the sampling (TSEG2). WINDOW1 state is the synchronization extension state, and WINDOW2 state is the synchronization shortening state. Figure 3 is a jump diagram of its state.

After receiving the reset signal, the state machine comes into the SEG1 state to initialize the current bit period timing parameters. While in the absence of synchronization, the state machine implements the SYNC, SEG1 and SEG2 in turn between the cycles of the three states. Namely, the three time segment, SYNC, TSEG1 and TSEG2, cyclically implement in turn. At this time, SYNC costs one TQ, and the size of TSEG1 and TSEG2 is appointed by the bit timing control register. TSEG1 segment is equal to the SEG1, and TSEG2 segment is equal to SEG2.

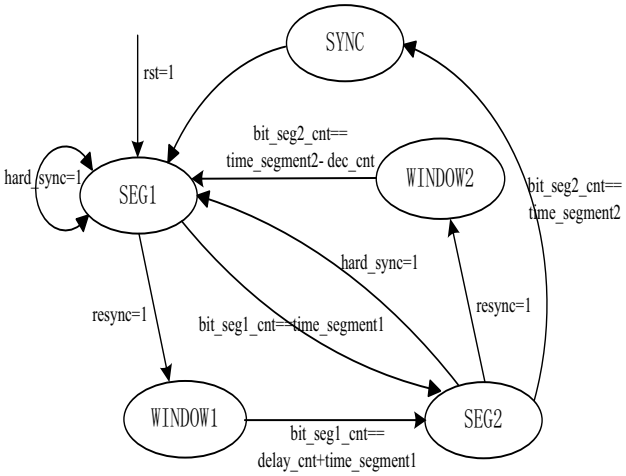

Figure 3 Sync jump state machine state jump diagram

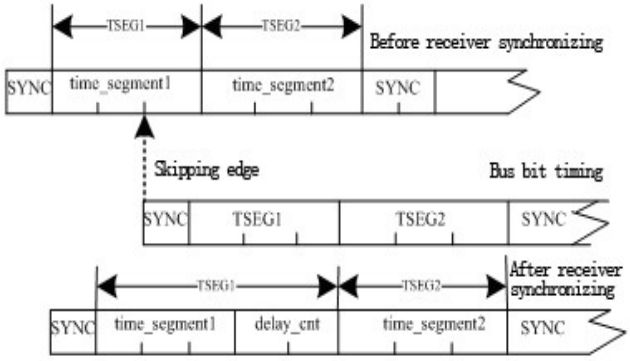

Figure 4 synchronization extending

In the SEG1 state, if there is no sync signal, it enters into the SEG2 state when TSEG1 count full. If the bus transmission occurs from recessive to dominant transition in the idle cycle, the receiver will interpret it as the beginning of a message frame, and implement a hard sync "Hard_sync = 1" to synchronize the bit-stream on the bus, and then the state machine re-enter the SGE1 state. If the skipping edge from recessive value to dominant value is detected in the TSEG1 segment or behind SYNC but in front of the sampling point, the receiver will translate it into a lagging edge sent by a slow transmitter, implement re-synchronization "Resync=1" and extend TSEG1 segment (synchronization extending). Then the state machine comes into WINDOW1 state. In this case, the time TSEG1 segment is equal to TSEG1 plus delay_cnt, and TSEG2 segment is equal to TSEG2, which is shown in figure 4.

If there is no sync signal in the SEG2 state, it enters into the SYNC state when TSEG2 count full. The time of TSEG2 segment is equal to TSEG2. If the bus transmission occurs from recessive to dominant transition at the beginning of the message, the receiver will implement a hard sync "Hard_sync = 1" and the state machine go into SGE1 state. If the skipping edge from recessive value to dominant value is detected in the TSEG2 segment or behind the sampling point, the receiver will translate it into an advance edge sent by a fast transmitter, implement re-synchronization "Resync=1" and shortening TSEG1 segment (synchronization shortening). Then the state machine comes into WINDOW2 state. In this case, TSEG1 segment is equal to TSEG1 and TSEG2 segment is equal to TSEG2 minus dec_cnt, as shown in Figure 5.

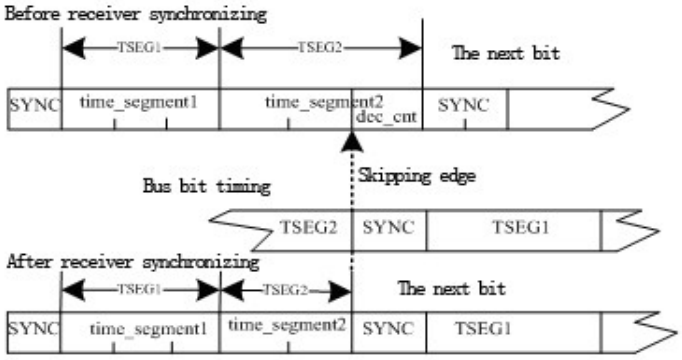

Figure 5 synchronization shortening

IV. IMPLEMENTATION AND VERIFICATION

A. Pre-simulation

The Verilog language of the CAN bit timing design have done functional simulation in the ModelSim SE 6.0. In order to reduce the burden of computer, we assumed that the work clock frequency was 100MHz, CAN bus communication baud rate was 100Kbps, and set the baud rate preset device baud_r_presc = 6'b000100, synchronization jump width SJW = "01" sampling mode SAM = "0", the Baud Rate Prescaler baud_r_presc = 6'b000100, synchronization jump width SJW = 2'b01, sampling mode SAM = 0, TSEG1 = 4'b0100 and TSEG2 = 3'b011, and set the acceptance data signals "rx" based on test need. The parameters of the synchronization state machine are set as follows: seg1 = 3'd0, window1= 3'd1, seg2 = 3'd2, window2= 3'd3, sync_seg = 3'd4. So I can know that bit time is 1000ns, and clk_en is 100ns.

In the Figure6, we can see that the controller will implement data sampling "sample_point = 1" in the end of the TSEG1. At this time, we can also find that the value of sample_bit is just equal to the value of rx. Similarly, each of the following sample values only changes at the end of TSEG1. Again, CAN system clock cycle "clk_en" is 10 times of the oscillator clock cycle "osc_clk", which is just accordance with my former setting. On the other hand, clk_bit clock is the clock of each receiving bit.

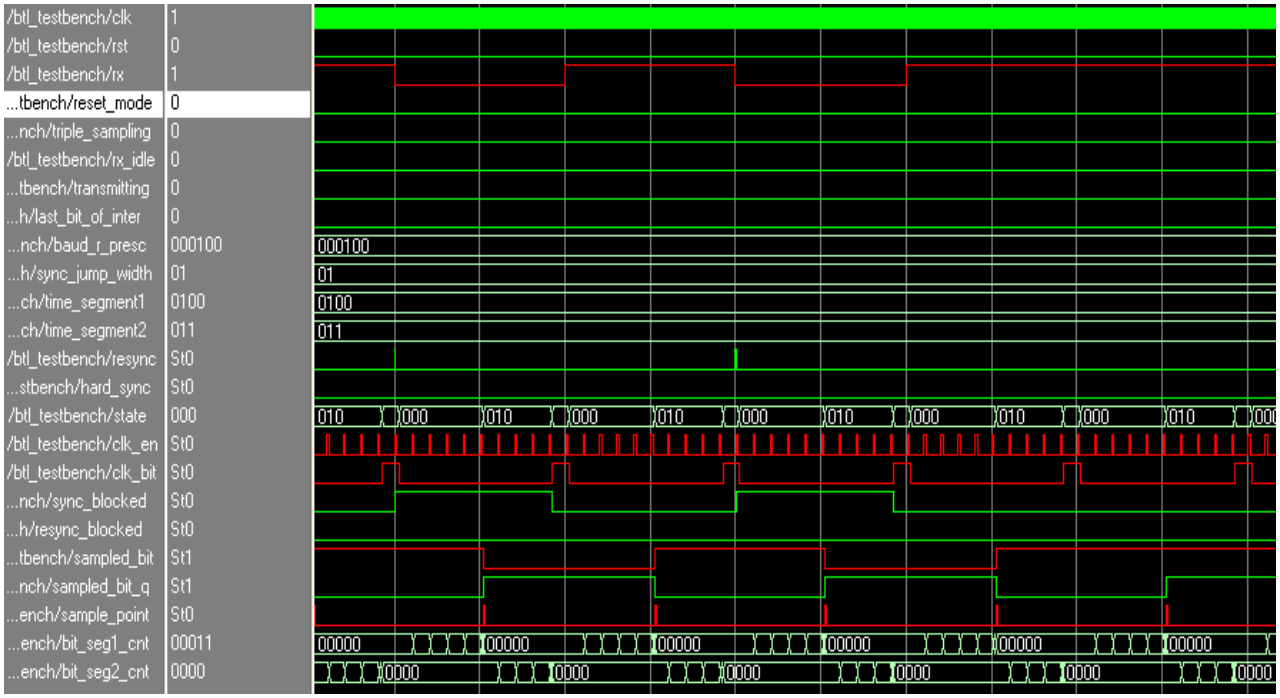

Figure 6 Pre-simulation of sampling and the baud rate generation

B. *Post-simulation*

This paper used Synopsys's Design Compiler to synthesis, Astro to Layout. The following is post-simulation waveforms. We can see that although with a slight delay, the function of the figure is correct.

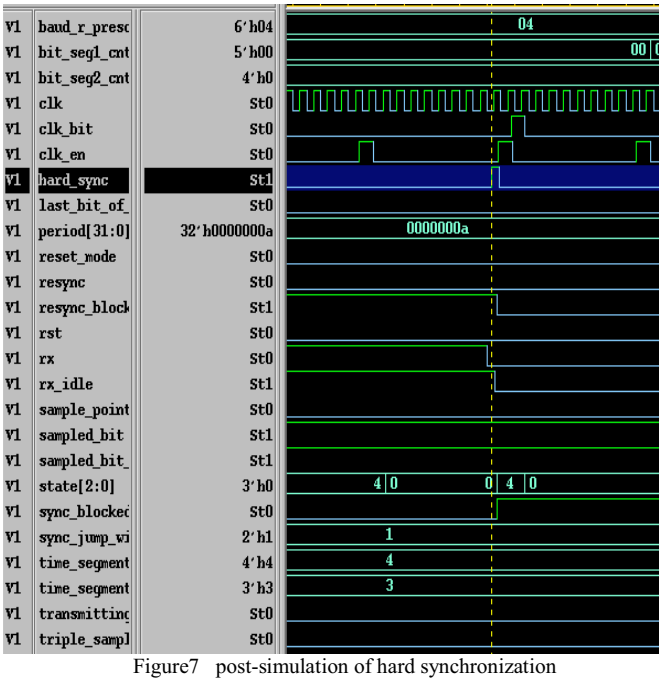

Figure7 post-simulation of hard synchronization

Figure 7 shows the post-simulation of hard synchronization. We can see from the figure, when a hard synchronization comes, the bus transmission immediately re-start to transmit a new bit, which is the start bit of a frame. After the falling edge of the rx, where is the starting point of the bit? In fact, just ensuring the rx jump edge which caused the hard synchronization is located inside the SYNC of the re-starting bit, the starting point of the bit will not change. That is to say, CAN controller doesn't need to ensure one TQ of SYNC at this time, or the accurate start point of the bit is not very important at this time.

When controller finds resynchronization in SEG1 segment, the synchronization state machine enters into WINDOW1 state to implement synchronization extending, shown in Figure 8.

When finding resynchronization in SEG2 segment, the synchronization state machine enters into WINDOW2 state to implement synchronization shortening, shown in Figure 9.

V. CONCLUSIONS

Accurately sampling and processing bus bit timing is important for can bus node to send and receive data correctly [11]. This paper presents an optimization method for a nominal bit time in CAN2.0 protocol, which reduces the traditional nominal bit time from non-overlapping segments to three non-overlapping segments, and proposes a CAN bit timing processor design approach based on synchronous state machine. This paper also gives some pre-simulation and post-simulation waveforms at each state in synchronous state machine. Results show that the designed CAN bit timing processor accords with CAN2.0 protocol specification, can more easily deal with CAN bus communication bit timing, control bit timing, bit synchronization and optimize the CAN network performance more better.

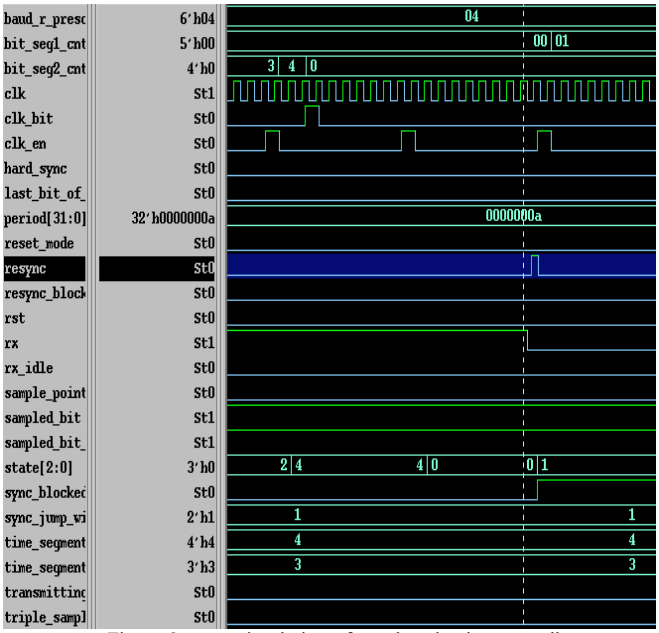

Figure 8 post-simulation of synchronization extending

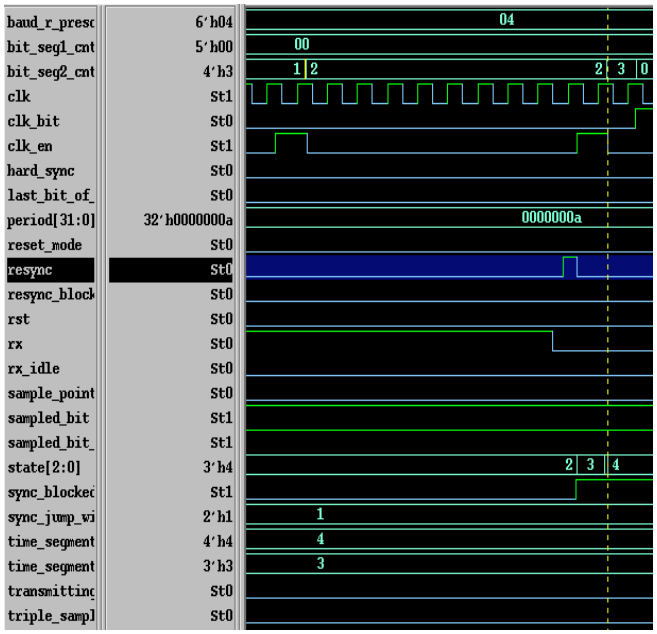

Figure 9 post-simulation of synchronization shortening

ACKNOWLEDGMENT

This work was supported by IC Special Foundation of Shanghai Municipal Commission of Science and Technology (Grant No. 09706201300), Shanghai Municipal Commission of Economy and information (Grant No.090344), Shanghai High-tech Industrialization of New Energy Vehicles (Grant No.09625029) and Graduate Innovation Fund of Shanghai University.

REFERENCES

[1] Steve Corrigan. Introduction to the Controller Area Network (CAN)[Z]. Texas Instruments Application Report. SLOA101A. August 2002–Revised July 2008.

[2] Karl Henrik Johansson, Martin Torngren, Lars Nielsen. Vehicle Applications of Controller Area Network[Z]. 2005.

[3] Karthik Ranganathan. RTL System Design of CAN 2.0A Controller [D]. Master's Thesis. Lubbock, Texas: Texas Tech University, 2005.

[4] Philips Semiconductors. CAN Specification Version 2.0, Parts A and B [S].1992

[5] Florian Hartwich, Armin Bassemir. The Configuration of the CAN Bit Timing [C]. 6th International CAN Conference. Robert Bosch GmbH. Abt. K8/EIS Tübinger Straße 123, 72762 Reutlingen.

[6] Sreeram Krishnamoorthy. Design of an ASIC chip for a controller area network (CAN) protocol controller [D]. Texas: Texas Tech University, 2006.

[7] Blagomir Donchev, Marin Hristov. Implementation of CAN Controller With FPGA Structures [C]. The 7th International Conference CADSM. Lviv-Slasko, Ukraine: 2003.577-580.

[8] Kim Namsub, Cho Kyuhyung, Kim Dawi, Kim Jinsang, Cho Wonkyung. Design and Verification of a CAN Controller for Custom ASIC [C]. Proc. 10th International CAN Conference on Semiconductor solutions (iCC2005).

[9] Peter Dzhelek, Arski, M.SC, Volker Zerbe, DR., Dimiter Alexiev, Assoc. PROFESSOR DR. FPGA Implementation of Bit Timing Logic of CAN Controller[C]. International Spring Seminar on, Volume 2, 13-16 May 2004 Page(s):214 - 220.

[10] Fang Li, Ru yuan Liu, Wei jie Lv. The Analysis of Bit Timing and Synchronization Mechanism of CAN Bus[J]. Electronic Engineering & Product World, 2005, Vol 9:106-107.

[11] Stuart Robb, CAN Bit Timing Requirements, Motorola Semiconductor Application Note, 1999.

[12] Wei Chen. Synchronization Method for CAN Real-Time Control System [J]. Ordnance Industry Automation, 2006, Vol 12:55-58.