Indonesian Journal of Electrical Engineering and Computer Science
Vol. 6, No. 3, June 2017, pp. 602 ~ 614
DOI: 10.11591/ijeecs.v6.i3.pp602-614

An Optimized FPGA Implementation of CAN 2.0 Protocol Error Detection Circuitry

Md Farukh Hashmi*[1], Avinash G. Keskar[2]
[1]Anurag Group of Institutions, Ghatkesar RR District, Hyderbad 500088
[2]Visvesavraya National Institute of Technology, South Ambazari Road, Nagpur
*Corresponding author, e-mail: farooq786engg@gmail.com

Abstract

Controller Area Network is an ideal serial bus design suitable for modern embedded system based networks. It finds its use in most of critical applications, where error detection and subsequent treatment on error is a critical issue. CRC (Cyclic Redundancy Check) block was developed on FPGA in order to meet the needs for simple, low power and low cost wireless communication. This paper gives a short overview of CRC block in the Digital transmitter based on the CAN 2.0 protocols. CRC is the most preferred method of encoding because it provides very efficient protection against commonly occurring burst errors, and is easily implemented. This technique is also sometimes applied to data storage devices, such as a disk drive. In this paper a technique to model the error detection circuitry of CAN 2.0 protocols on reconfigurable platform have been discussed? The software simulation results are presented in the form of timing diagram.FPGA implementation results shows that the circuitry requires very small amount of digital hardware. The Purpose of the research is to diversify the design methods by using VHDL code entry through Modelsim 5.5e simulator and Xilinx ISE8.3i.The VHDL code is used to characterize the CRC block behavior which is then simulated, synthesized and successfully implemented on Sparten3 FPGA .Here, Simulation and Synthesized results are also presented to verify the functionality of the CRC -16 Block. The data rate of CRC block is 250 kbps .Estimated power consumption and maximum operating frequency of the circuitry is also provided.

Keywords*: cyclic redundancy check (CRC), controller area network (CAN), field programmable gate array (FPGA), error detection, linear feedback shift register (LFSR)*

1. Introduction

Network must be able to transfer data from one device to another with complete accuracy. A system that cannot guarantee that the data received by one device is identical to the data transmitted by another device is essentially useless. Yet anytime data are transmitted from source to destination, they can become corrupted in passage. Infect it is more likely that some part of the message will be altered in transit than entire content will arrive intact. Many factors including line noise can alter or wipe out one or more bit of the given data. Reliable system must have a mechanism for detecting and correcting such errors. The acceptance and introduction of serial communication to more and more applications has led to requirements that the assignment of message identifiers to communication functions be standardized for certain applications. These applications can be realized with CAN more comfortably, if the dress range that originally has been defined by 11 identifier bits is enlarged. Therefore a second message format ('extended format') is introduced that provides a larger address range defined by 29 bits. This will relieve the system designer from compromises with respect to defining well-structured naming schemes [1]. Users of CAN, who do not need the identifier range offered by the extended format, can rely on the conventional 11 bit identifier range ('standard format') further on. In this case they can make use of the CAN implementations that are already available on the market, or of new controllers that implement both for-mats [2].

Embedded networks are used widely in most of critical embedded systems driven by any transmission medium. In fact many embedded systems are distributed, consisting of multiple microprocessors communicating over one or more networks to accomplish shared tasks. Amongst various buses used in embedded communication, the most widely used and important communication bus is Controller Area Network as CAN [3] in which worldwide

Received January 22, 2017; Revised April 6, 2017; Accepted April 21, 2017

researching is going on. With high data flow through such channels, there is a chance of potentially high noise. Reliable detection of bit errors is very important to prevent data corruption in such systems. In CAN, Cyclic Redundancy Check (CRC) is used to detect multi bit errors.

There are some works related to can protocol available in literature.Milind Khanapurkar et al. in [4] proposed an automotive Black Box design intends for storing and retrieving the data of various Electronics Controller Units (ECUS) on standards CAN protocol frame. The data thus stored and retrieved can be used for introspection of cause of failure or unfortunate miss happening with the vehile. Mazran Esro et al. [1] focused on method of application of CAN bus system in place, the methods of controlling each station in the security system has changed significantly. The system permits each station to send and receive data according to the message priority.Reinder J Bril et al. [5] revisits the basic message response time analysis of Controller Area Network (CAN) [6]. It was shown that existing response time analysis, as presented as optimistic.

In this paper, we presented a technique to model the error detection circuitry of CAN protocol using hardware description language (HDL).The HDL model is implemented on FPGA platform in the laboratory environment. Logic circuit requirement and FPGA resource utilization is presented. The estimated power consumption of the implementation is found to be 38mW which is very suitable for battery power application. The resource utilization report shows that very small amount of FPGA resources are used keeping ample scope to implement the rest of the circuitry in the same FPGA.

The VHDL source code has been edited and synthesized using Xilinx ISE 13.1, and then simulated and tested using ISim (VHDL/Verilog). Spartan 3A FPGA starter kit from Xilinx has been used for downloading the design into Xilinx Spartan 3A FPGA chip. The design has been tested in a hardware environment for different data inputs

The materials in this article are organized as follows: in Section II, a brief description of the backgrouond of CAN protocol; problem formulation for CRC detection circuitary algorithm is discussed in Section III; the methodoldogy for hardware modeling for error correction will be described in Section IV; as well as the top-level design, RTL view; the simulation results and discussion is given in Section V;In Section VI, FPGA implementation and synthesis results will be concluded, at the end, a conclusion will be given in Section VII.

2. Background of CAN 2.0 Protocol

Controller Area Network (CAN) is a serial communications bus designed to provide simple, efficient and robust communications for in-vehicle networks. CAN was developed by Robert Bosch GmbH, beginning in 1983, and presented to a wider audience at the Society of Automotive Engineers (SAE) Congress in 1986-effectively the birth of CAN. In 1987, the first CAN controller chips were released by Intel (82526) and Philips (82C200). In the early 1990s, Bosch submitted the CAN specification (Bosch, 1991) for standardization, leading to publication of the first ISO standard for CAN (11898) in 1993 (ISO, 1993). Mercedes was the first automotive manufacturer to deploy CAN in a production car, the 1991 S-class. By the mid 1990s, the complexity of automotive electronics was increasing rapidly [1]. The number of networked Electronic Control Units (ECUs) in Mercedes, BMW, Audi and VW cars went from 5 or less at the beginning of the 1990s to around 40 at the turn of the millennium. With this explosion in complexity traditional point-to-point wiring became increasingly expensive to manufacture, install, and maintain due to the hundreds of separate connections and tens of kilograms of copper wire required [7-8]. As a result CAN was rapidly adopted by the cost-conscious automotive industry, providing an effective solution to the problems posed by increasing vehicle electronics content. Following on from Mercedes, other manufacturers including Volvo, Saab, BMW, Volkswagen, Ford, Renault, PSA, Fiat and others all adopted CAN technology [9].

In order to distinguish standard and extended format the first reserved bit of the CAN message format, as it is defined in CAN Specification 1.2, is used. This is done in such a way that the message format in CAN Specification 1.2 is equivalent to the standard format and therefore is still valid. Furthermore, the extended format has been defined so that messages in standard format and extended format can coexist within the same network [10-11].

This CAN Specification 2.0 consists of two parts, with:

1. Part A describing the CAN message format as it is defined in CAN Specification
2. Part B describing both standard and extended message formats.

2.1. Message Format

In CAN data transmission is done using formatted message frames. There are two protocols versions in which a CAN network may be configured namely 2.0A and 2.0B. The former su2pports 11-bit message identifiers while the later supports 2.0B which supports both 11-bit and 29-bit identifiers. A data frame is shown in Figure 1. A data frame is composed of seven different fields: start of frame, arbitration field, control field, data field, CRC field, Ack field and end of frame. In CAN 2.0 A arbitration field consist of 11 bit identifier and RTR bit where as in CAN 2.0B it consist of 29 bit identifier, IDF bit and RTR bit [12].

a. Data length code

The number of bytes in the data field is included by the data length code. This data length code is 4bits wide and is transmitted within the control field .Here the admissible number of data bytes (0,1…7,8). And other values may not be used. The DLC format is shown in the Table 1.

2.2. Error Detection CAN

Data flow though channels may subject to unpredictable changes due to interference, which may lead to change in the shape of the signal. For reliable communication error must be detected. Error detection communication error must be detected. Error detection mechanism uses redundancy means addition of extra bit information to the information. in CAN cyclic redundancy check is used to detect multi bit errors [13-14].

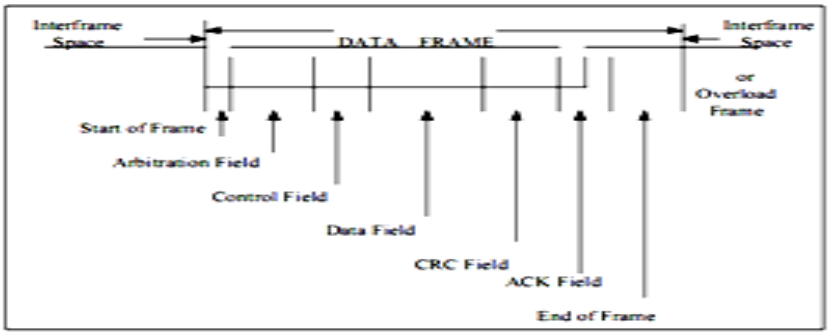

Figure 1. Format of Data Frame

Explaining research chronological, including research design, research procedure (in the form of algorithms, Pseudocode or other), how to test and data acquisition [1-3]. The description of the course of research should be supported references, so the explanation can be accepted scientifically [4-5].

3. Problem Formulation as a CRC-16 Error Detection Circuitry

3.1. Cyclic Redundancy Check (CRC-16)

In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probality whether or not an error occurred in the transmission [15]. If an error occurred the receiver sends a ‘negative acknowledgement (NAK) back to the sender, requesting that the message be transmitted. The CRC is based on polynomial arithmetic. The redundancy bits used by CRC are derived by dividing the data unit by a pre-determined divisor and the remainder is CRC, Addition and Subtraction are done in modulo 2 that is, they are both the same as theexculisive or operator. In this technique at Sender ‘side a sequence of redundant bits the CRC or the CRC remainder, is

appended to the end of a data unit so that the resulting data unit becomes exactly divisible by the same predetermined binary number. At its destination, the incoming data unit a divided by the same number. If at this step there is no remainder, the data unit is assumed to be intact and is therefore accepted. A reminder indicates that the data unit has been damaged in transit and therefore must be rejected [16-17].

a. At Sender's Side

First n Numbers of 0s bits are appended to the original data where n is a less than the predetermined divisor, which is n+1 bit.Secondaly, drawn out data unit is divided by divisor; using binary divison.The remainder resulting from this division is the CRC. Third the CRC is replaced with the 0s that has been appended in the first step. A CRC may consist of all 0s also.

b. At Receiver side

The receiver divides the whole data with the predetermined divisor that was used to find the CRC remainder. If there is a change in data, the division yields a non zero remainder and the data unit does not pass. The sender reminder and the data unit do not pass the sender side and receiver side technique has been demonstrated in the Figure 2.

3.2. Algorithm for CRC Computation

The hardware implementation of cyclic Redundancy check computation is done by using linear shift register (LFSR) the shift register is driven by a clock. At every clock pulse, the input data is shifted in to the register in addition to transmitting the data. When all the input bits have been processed, the shift register contains the CRC bits, which are then shifted out on the data line. The algorithm is as follows.

Table 1. Representation for Data Length Code

Number of Bytes	Data Length Code			
	DLC03	DLC02	DLC01	DLC00
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

Abbreviations:-d 'dominant'' r 'receive'

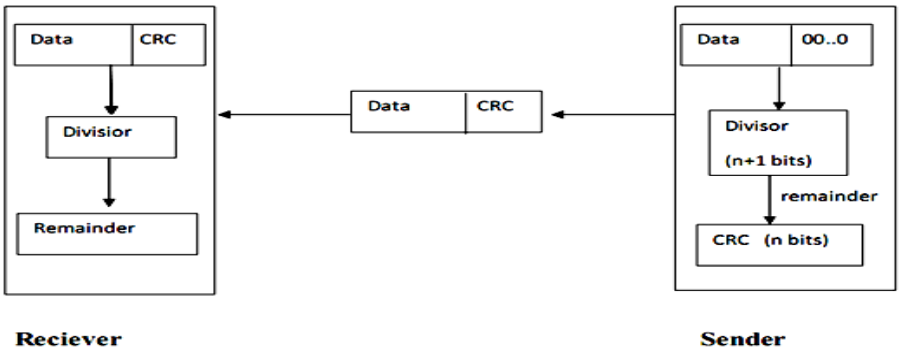

Figure 2. Cyclic Redundancy Check (CRC) Block Diagram

(i) Initialize the CRC register to all o-bits.
(ii) Get first/next message bit m.
(iii) Append (G-1) o bits to the original message
(iv) If the high –order bit of CRC is 1
(v) Shift CRC and m together 1 position and Xor the results with the low order r bits of G.
(vi) Otherwise, just shift CRC and m1 position.

Where G is generating polynomial and r is the degree of polynomial.

Polynomials: Instead of representing the data string in 1's and 0s they are represented in algebraic polynomials. The algebraic polynomials are very short to represent and also they can used to prove the concept mathematically. Two important conditions while selecting polynomials are it should not be divisible by x and it should be divisible but x+1.

4. Methodology for Hardware Modeling

The block diagram of the error detection mechanism of CAN is presented in the Figure 3. The Entire circuitry has been divide into three main blocks; LFSR.CRC Generator and a Counter. The input data bytes are applied to LFSR whose output is supplied to CRC generator. The counter progresses through its count sequences and generates a pulse to latch the output CRC generator based on the DLC input [18]. The output available from the CRC generator is appended with the data field and transmitted over the network. In the receiver side the CRC checking circuit is implanted using the same LFSR of Figure 4.

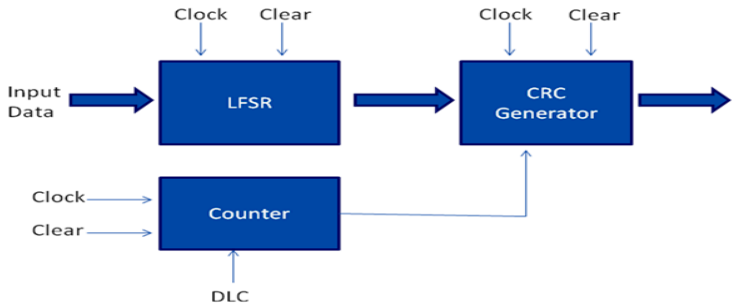

Figure 3. Block Diagram for CAN Bus Error Detector Implementation

Transmitter: CRC process can easily be implemented as a dividing circuit consisting of exclusive OR gates and one bit shift registers. Implementation is as follows.

Process:

1. The input register contains N bit, equal to the length of the FCS.
2. There can be up to n exclusive OR gates.
3. The presence or absence of a gate corresponds to the presence of a term in the divisor polynomial other than to the major significant to the polynomial.
4. Always there is a feedback from the highest corresponding shift register to the input and the same feedback is fed to the other XOR gate, which is given below Figure 4 and Contents of Shift Register at Transmitter is given in Table 1.

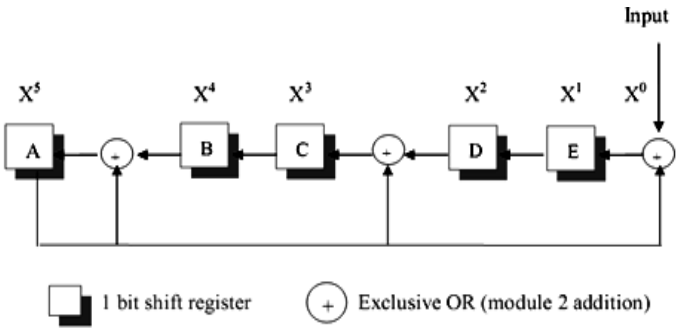

Figure 4. Implementation of CRC polynomial at Transmitter

Table 1. Contents of Shift Register at Transmitter

STEP	A_n xor B_n A_{n+1}	C_n B_{n+1}	A_n xor D_n C_{n+1}	E_n D_{n+1}	I_n xor A_n E_{n+1}	I/P
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	1
3	0	0	0	1	1	0
4	0	0	1	1	0	0
5	0	1	1	0	0	1
6	1	1	0	0	1	0
7	0	0	1	1	1	1
8	0	1	1	1	1	1
9	1	1	1	1	1	1
10	0	1	0	1	0	0
11	1	0	1	0	0	1
12	1	1	1	0	0	0
13	0	1	1	0	1	0
14	1	1	0	1	0	0
15	0	0	0	0	1	0
16	0	0	0	1	0	0
17	0	0	1	0	0	

INF

ZEROS

Frame Check Sequence

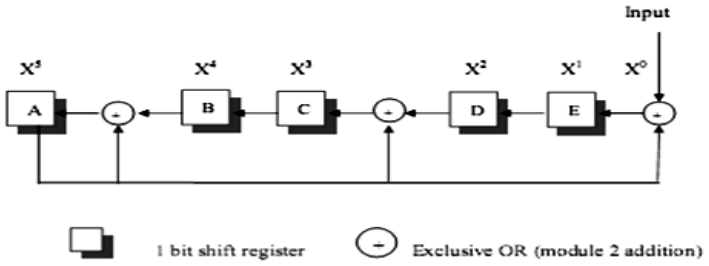

Figure 5. Implementation of CRC polynomial at Receiver

Receiver: At the receiver the same circuit deployed in the transmitter to formulate the CRC will be deployed. In this case the information will be the received message which has been transmitted from the transmitter M+R.

Process:

1. At the zero clock pulse or step 0, all shift registers reset to zero.
2. From Step 1 the received message will be input, one bit at a time with the most significant bit. [11]

Contents of Shift Register at Reciever is given in Table 2.

Table 2. Contents of Shift Register at Reciever

STEP	A_n xor B_n A_{n+1}	C_n B_{n+1}	A_n xor D_n C_{n+1}	E_n D_{n+1}	I_n xor A_n E_{n+1}	I/P
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	1
3	0	0	0	1	1	0
4	0	0	1	1	0	0
5	0	1	1	0	0	1
6	1	1	0	0	1	0
7	0	0	1	1	1	1
8	0	1	1	1	1	1
9	1	1	1	1	1	1
10	0	1	0	1	0	0
11	1	0	1	0	0	1
12	1	1	1	0	0	0
13	0	1	1	0	0	0
14	1	1	0	1	0	1
15	0	0	0	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	0	

INF

CRC bits

4.1 Hardware Modules

4.1.1. Linear Feedback Shift Register (LFSR)

An n-bit LFSR is an n-bit length shift register with feedback to its input. The feedback is formed by XORing or XNORing the outputs of selected stages of the shift register - referred to as 'taps' - and then inputting this to the least significant bit (stage 0). Each stage has a common clock. The 'linear' part of the term 'LFSR' derives from the fact that XOR and XNOR are linear functions [18]. An example of a 5-bit LFSR is shown below Figure 5.

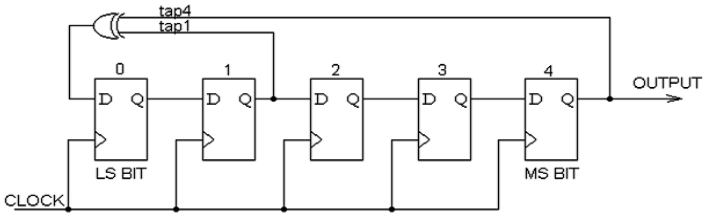

Figure 6. Implementation of Linear Feedback Shifft Register (LFSR)

This has taps at stages 1 and 4 with XOR feedback. Note also that the LS bit of the shift register is, by convention, shown at the left hand side of the shift register, with the output being taken from the MS bit at the right hand side. So what is it about a LFSR that makes it interesting? It will produce a pseudorandom sequence of length 2n-1 states (where n is the number of stages) if the LFSR is of maximal length. The sequence will then repeat from the initial state for as long as the LFSR is clocked. Assume that the example LFSR above is set to $1F after initialization. The output of the feedback XOR gate will be 0 (since 1 XOR 1 = 0) and the first clock edge will load 0 into stage 0. [13] An LFSR is of 'maximal' length when the sequence it generates passes through all possible 2n-1 values. The LFSR sequence depends on the seed value, the tap positions and the feedback type. So far we have seen how to implement LFSRs in VHDL such that any device can be targeted. Xilinx devices however, will allow optimal implementation of LFSRs with their internal distributed RAM. This type of RAM is available in the XC4000, Spartan/XL, Spartan-II and all Virtex families. Each CLB can be configured as a RAM and this allows very compact shift registers to be built. [5]

4.1.2. Up-Down Counter

In digital logic and computing, a counter is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.Counter that can change state in either direction, under the control of an up/down selector input, is known as an up/down counter. When the selector is in the upstate the counter increment its value. When the selector is in the down state, the counters decrement its count [18].

Proceed through a well-defined sequence of states in response to count signal
3 Bit Up-counter: 000, 001, 010, 011, 100, 101, 110, 111, 000,...
3 Bit Down-counter: 111, 110, 101, 100, 011, 010, 001, 000, 111,...

A counter is a degenerate finite state machine/sequential circuit where the state is the only output. A counter can be easily made by using T (Toggle) flip-flop [5]. An Example of Up/Down Counter is shwn in Figure 7.

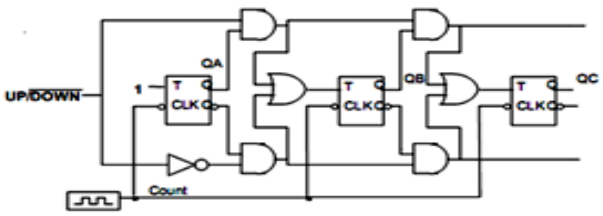

Figure 7. Implementation of Up/Down Counter

5. Simulation Results and Discussion

This design is implemented in VHDL platform using Xilinx ISE 13.1. The Register transfer logic(RTL) and Internal RTl View view are shown in the Figure 8 and Figure 9. LFSR is represented by 1st two blocks & the third block is CRC generator. The input data & DLC is applied to the circuit and generates the CRC and Input Data at Tansmitter is presented in Figure 10.The simulation results of the CRC generator in the form of waveforms are presented in Figure 11. Here input data is 10101010 is applied at the input of CRC & the output obtained is 100001110010001. Now the output CRC is appended taking the MSB first to the main data & transmitted over the network. Figure 12 is showing the simulation waveform for CRC checker. Here the transmitted data is checked by CRC checker. The input is data and the CRC. The waveform shows that after 23rd clock the output is 0 which indicates no error. In Figure 13, an error has been purposefully introduced in 16th clock period. The result in Figure 14 shows that the output of CRC checker is non-zero meaning that an error has occurred during transmission.

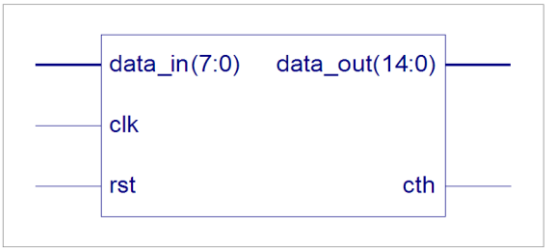

Figure 8. RTL View of CRC -16 Can 2.0 Error Detector

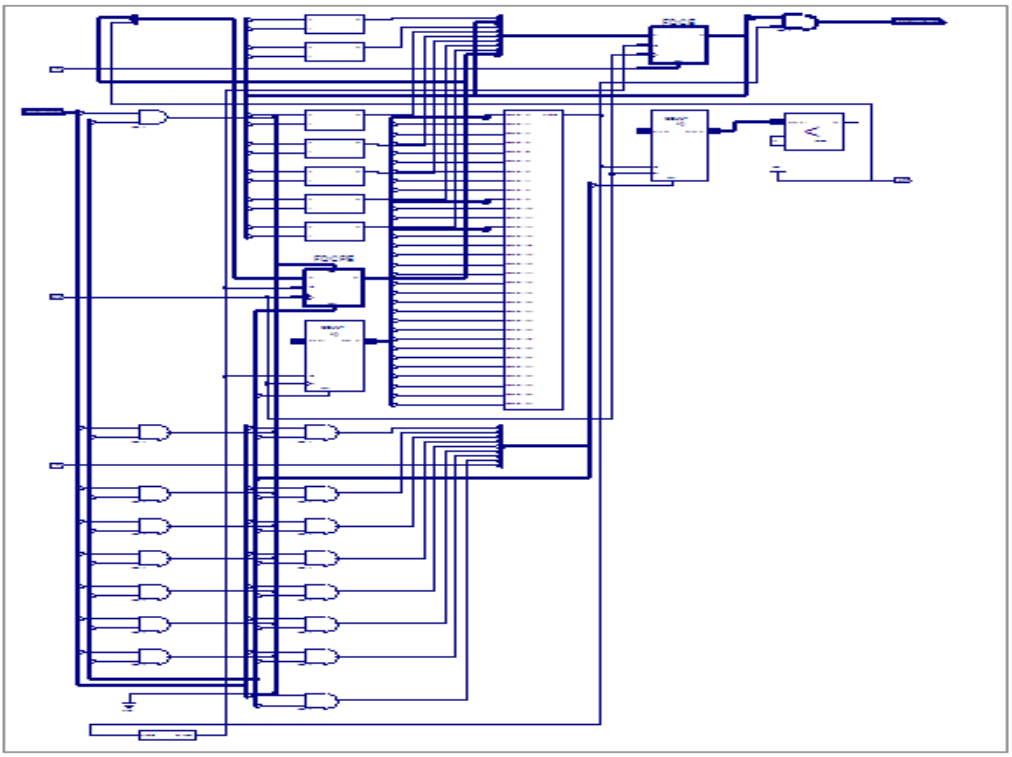

Figure 9. Internal RTL View of CRC -16 Can 2.0 Error Detector

For Input data: It shows the input data given at transmitter side and also shows the initial values that are assigned to the different registers and counters.

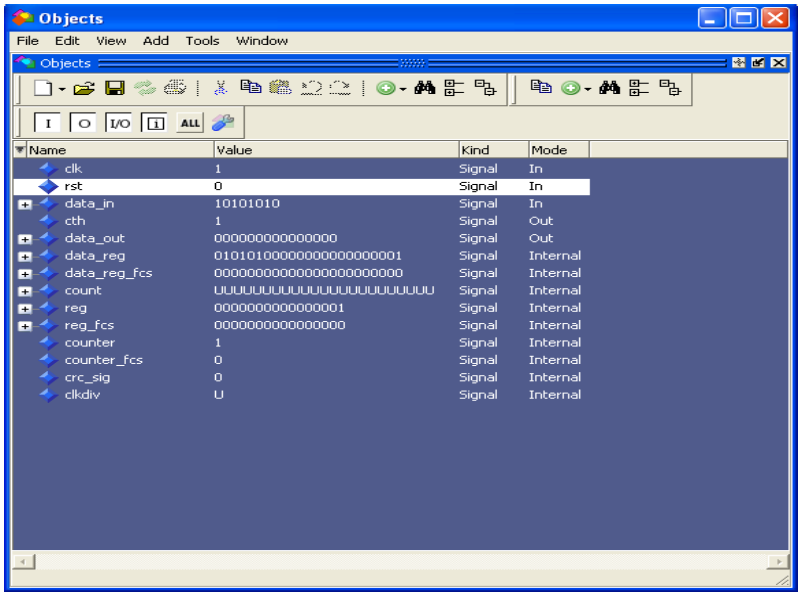

Figure 10. Input data at Transmitter

For Transmitter: This figure shows the CRC bits that are generated in CAN bus. Here the CRC bits are appended with given input data and then transmitted.

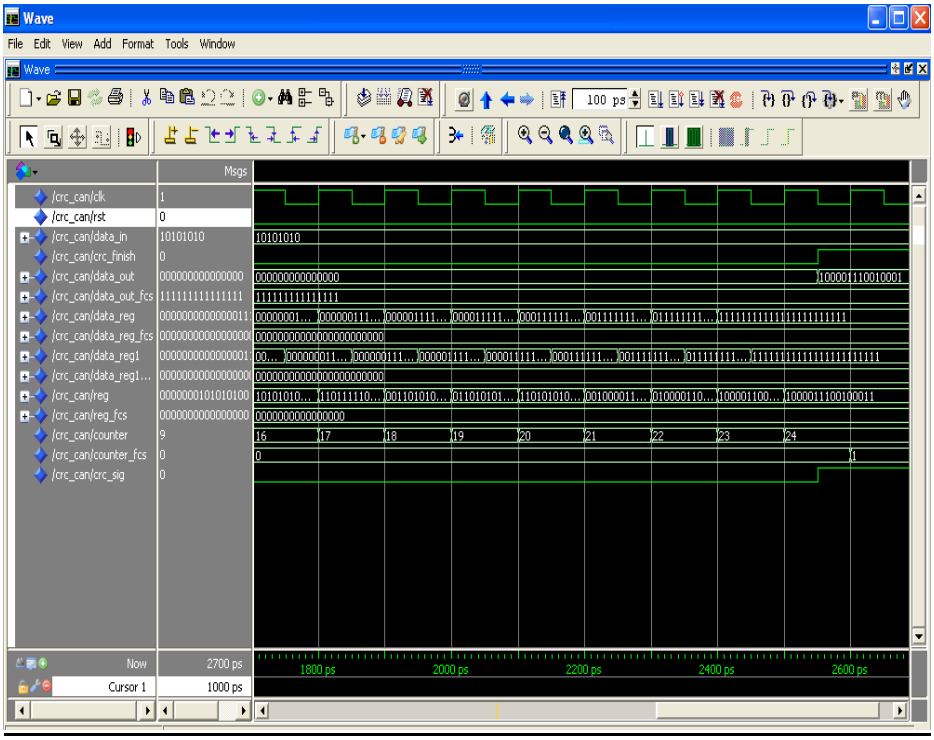

Figure 11. Output waveforms for Transmitter

This figure shows the output at receiver side. The waveform shows that after 26rd clock the output is 0 which indicates no error.

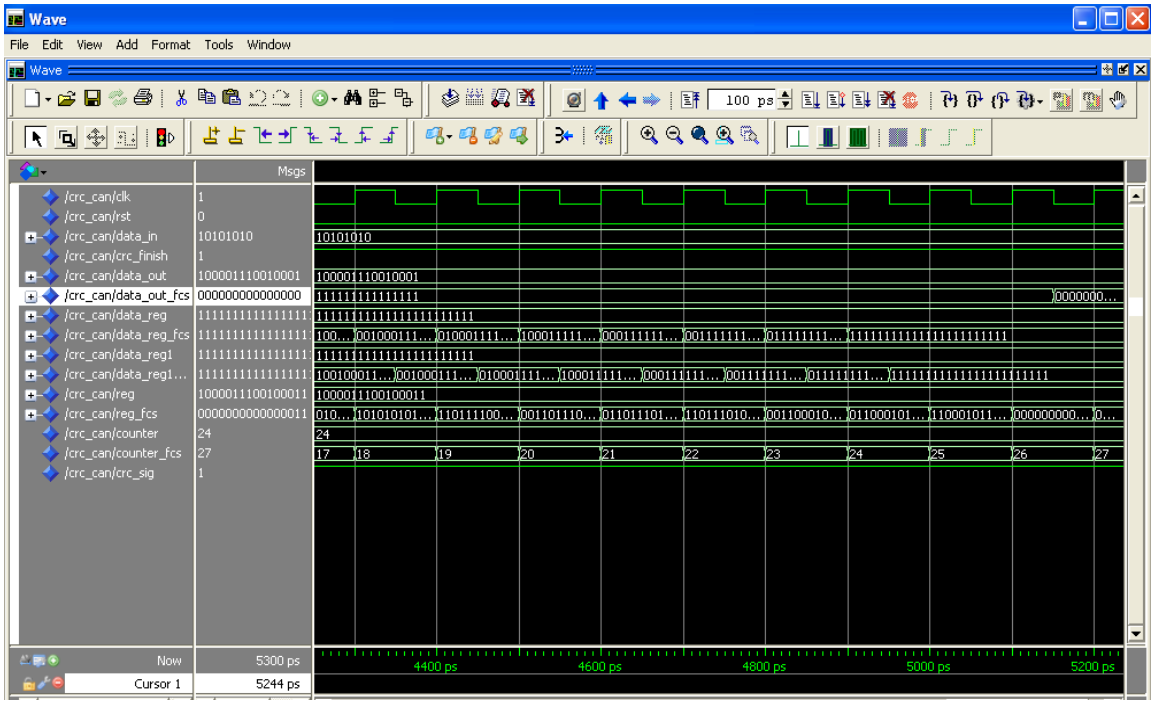

Figure 12. Output waveforms for Receiver

Output waveforms in Error Case:-

This figure shows output waveform at transmitter part in case of error. It generates the CRC bits for transmitter when any error is occurred in given data.

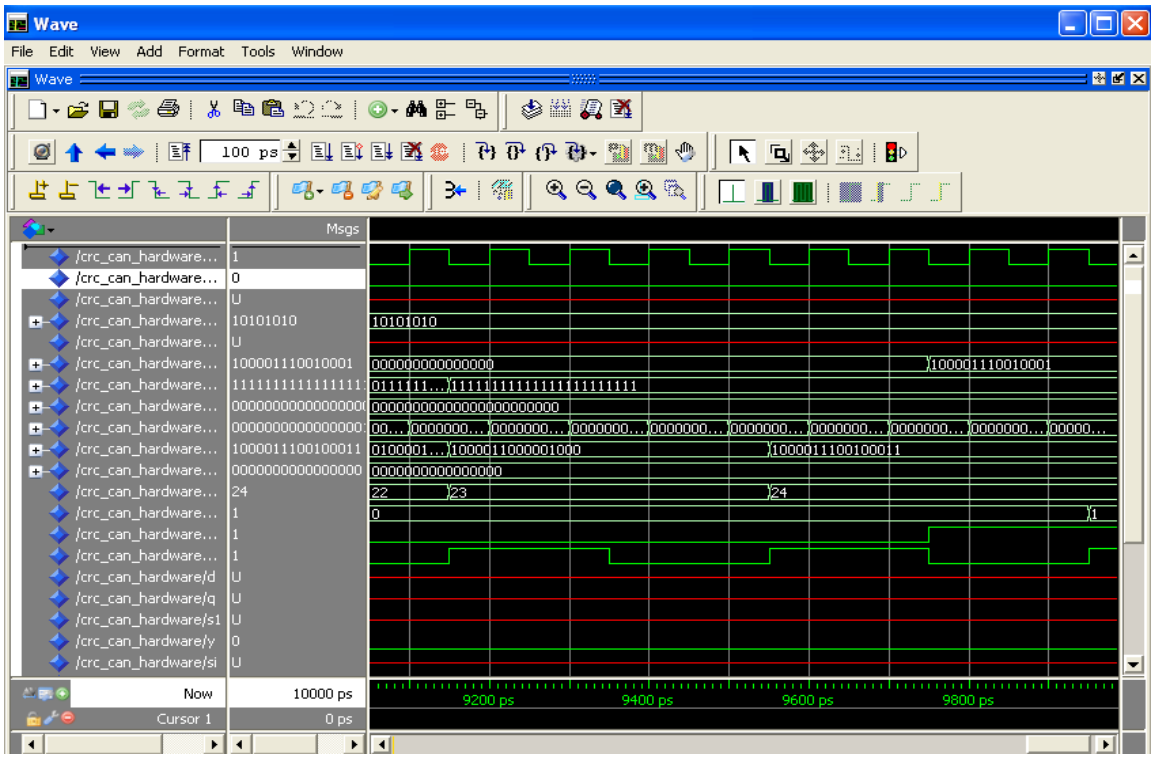

Figure 13. Output Waveforms at Transmitter Side in case of Error

Output in case of Error

This figure shows the output at receiver side. The waveform shows that after 26rd clock the output is non zero which indicates error.

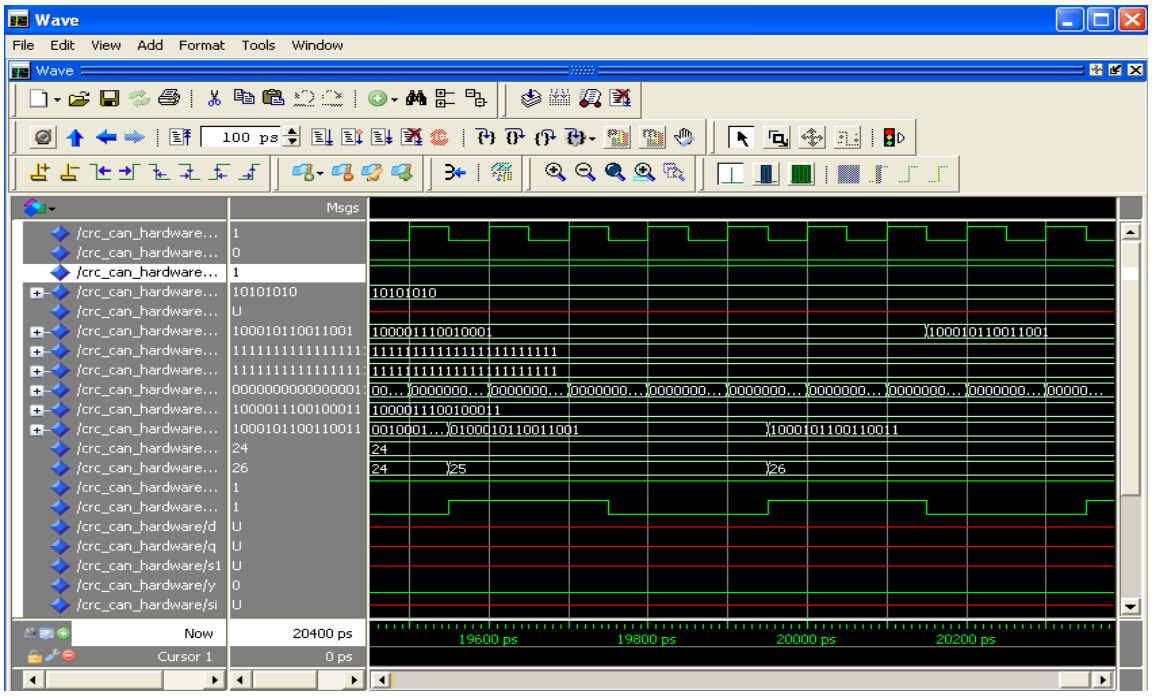

Figure 14. Output Waveforms at Reciever Side in case of Error

6. FPGA Implementation and Synthesis Result

The design is implemented in Xilinx Spartan 3A (device XC3s400) FPGA platform.HDL Synthesis report (Macro Statistics)and Advanced Synthesis report(Macro Statistics) of the implementation is presented in Table 3 and Table 4.The devices has been used are adder, multiplexers, registers and xors gates. The 7 bit counter uses an adder to increment its value. The multiplexers are used in the CRC generator, Based on the value in DLC, the CRC generator with the help of multiplexer selects one of the output instances from LFSR.Registers have been used to store the data at different level of the CRC generation process.Seven, 1 –bit XOR gates are used in LFSR to produce the new bit in the string for each clock pulse. Table 5 shows the Device Utilization Summary that the circuit utilizes a very small amount of the available resources. The percentage of device utilization for number of slice Flip Flops, number of 4 input LUTS, number of IOBS and numbers of GCLKS are shown. It has been found that the estimated power consumption using Xilinx Xpower is very low and it is 38mW. The maximum operating frequency found is 117.08 MHz's

Table 3. HDL Synthesis Report Macro Statistics

#Counters	2
32-bit up cunter	2
#Register	26
1-bit register	23
16-bit register	2
23-bit register	1
#Comparators	1
32-bit comparator or less	1
# Xors	14
1-bit xor2	14

Table 4. Advanced HDL Synthesis Report (Macro Statistics)

#Counter	2
32- bit up counter	2
#Registers	78
Flip-Flops	78
#Comparators	1
32-bit Comparator Less	1
#Xors	14
1-bit Xor2	14

Table 4. Device Utilization Summary

Number of Slices	56 out of 3584	1%
Number of Slice Flip Flops	71 out of 7168	0%
Number of 4 Inputs LUTs	77 out of 7168	1%
Number of Bonded IOBs	26 out of 141	18%
Number of GCLKs	1 out of8	12%

TIMING REPORT: Timinng Analysis is presented in following Table 5.

Table 5. Timinng Analysis

Timing Parameter	Calculated Results
Minimum period: 8.097ns (Maximum Frequency:	117.08 MHz.
Minimum input arrival time before clock:	No path found
Maximum output required time after clock	12.426ns
Maximum combinational path delay: No path found	No path found

7. Conclusion

A Technique to model the error detection circuitry of CAN protocol in VHDL is described. The VHDL Model of CRC generator and checker is implemented on FPGA Xilinx ISE 13.1.The FPGA implementation result shows that the design is attractive from resource utilization, power consumption and operating frequency Here we have utilized 114696 kilobyte memory. The referred paper describes error circuitry in case of 'no error' our contribution – after reviewing the paper we designed error controlling circuitry for CAN bus in case of error.

Usually components, like LFSR, XOR and Counters are used, so the entire circuit can be easily designed. This approach is efficient both in terms of hardware, speed and power cocsumption. The additional hardware required is very simple. This technique works efficiently in case of ASIC design also. We have shown that hardware implementation on FPGA can be effectively used to improve the performance of CRC implementation for CAN protocol.

References

[1] Shukla, Sunil, and Neil W. Bergmann. Single bit error correction implementation in CRC-16 on FPGA. *Poceedings of the IEEE International Conference on Field-Programmable Technology*. Brisbane, Australia. 2004; 319-322.
[2] Al-Ali Y, Al-Habian G, Saifi S, Al-Rousan M. Automobile Black Box for Accident Simulation. CSIDC. American University of Sharjah. 2005.
[3] Specification C.A. N. Version 2.0. Robert Bosch GmbH. 1991 Sep.
[4] M Khanapurkar, P Bajaj, SDH Wandhare. Design approach for VHDl and FPGA Implementation of Automotive Black Box using Can Protocol. *International journal of Computer Science and Network Security.* 2008; 8(9): 214-218.
[5] M Esro, AA Basari, S Kumar, A Sahiqin MI, Z Syariff. Controller Area Network (Can) Application in security System. *World Academy of Science, Engineering and Technology*. 2009; 59: 299-302.
[6] Eisenreich, Dan, and Brian De Muth. Designing Embedded Internet Devices. Newnes. USA: Elsevier Science. 2003; 467-482.
[7] BA Forouzen, Chapter 10. Data Communication and Networking. 3rd ed. Tata Mcgraw-Hill publishing Company Limited. 2004; 243-252.
[8] Xing W, Chen H and Ding H. *The application of controller area network on vehicle*. Proceedings of the IEEE International on Vehicle Electronics Conference. 1999; 455-458.

[9] Carvalho FC, Jansch-Pôrto I, Freitas EP, Pereira CE. *The TinyCAN: an optimized CAN controller IP for FPGA-based platforms*. Proceedings of the 10th IEEE Conference on Emerging Technologies and Factory Automation.The TinyCAN: An Optimized CAN Controller IP for FPGA-Based Platform. Catania, Italy. 2005; 374-377.
[10] Donchev B, Hristov M. *Implementation of CAN controller with FPGA structures*. Proceedings of the IEEE 7th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics. Lviv–Slavsko, Ukraine. 2003; 577-580.
[11] Khanapurkar M, Bajaj P, Gharode D. *A design approach for intelligent vehicle black box system with intra-vehicular communication using lin/flex-ray protocols*. Proceedings of the IEEE International Conference on Industrial Technology. Chengdu, China. 2008; 1-6.
[12] Koopman, Philip, and Tridib Chakravarty. *Cyclic redundancy code (CRC) polynomial selection for embedded networks*. Poceedings of the IEEE International Conference on Dependable Systems and Networks. Florence, Italy. 2004; 145-154.
[13] Peterson, William Wesley, and Daniel T Brown. *Cyclic codes for error detection*. Proceedings of the IRE. 1961; 49(1): 228-235.
[14] Rajib Sarkar & Bijoy Kumar. *Design of VHDL Based Error Detection Circuitry in CAN*. Poceedings of the international Conference on Emerging Trends in Engineering & Technology. 2010; 106-110.
[15] Williams, Ross. *A painless guide to CRC error detection algorithms*. Internet publication, August 1993.
[16] Jagannathan Sarangapani, Krishna Chaitany Emani, Keong Kam, Maciej Zawodniok Yahong Rosa Zheng. *Improvement of can bus performance by using error-correction codes*. Department of Electrical and Computer Engineering, University of Missouri-Rolla.
[17] Emani, Krishna Chaitanya, Keong Kam, and Maciej Zawodniok. *Improvement of CAN bus performance by using error-correction codes*. Proceedings of the IEEE Region 5 Technical Conference, Fayetteville, AR. 2007; 205-210
[18] Bhasker, Jayaram. *A VHDL Primer*. PHI Pvt. Ltd 3, 2008.