International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.5., 2015
(Sept.-Oct.)

RESEARCH ARTICLE

ISSN: 2321-7758

IMPLEMENTATION OF CAN PROTOCOL CONTROLLER IN VHDL

P. HEMA MOUNIKA[1],Dr. N. BALA SUBRAHMANYAM[2],Ms. V. KIRANMAYI[3]

[1] PG Student, [2] Professor, [3]Sr. IT Engineer, CMC Limited, Hyderabad.

Department of ECE, Gayatri Vidya Parishad College of Engineering (A), Visakhapatnam, Andhra Pradesh, India

ABSTRACT

CAN, Controller Area Network was first developed by Robert Bosch in 1986 to provide communication between ECUs in automobiles. It is a serial communication bus protocol which obeys highly sophisticated set of rules.

Within the network of sensors, a communication module is needed for interfacing these sensors. In this thesis, a VHDL implementation of CAN controller for interfacing those sensors is presented. It deals with the arbitration, transfer protocol, control of frames, arbitration, error detection and error signaling. According to the protocol, the interface can be divided into modules such as FIFO, CRC, Acceptance Filtering, Error Checking, Error signaling and Bit timing Logic to perform various communication tasks. Each module is implemented as a VHDL entity and the performance of each entity is analyzed and then all entities are combined in structural style. The final description is simulated using Modelsim Software. The main objective of this paper is to achieve successful transmission and reception of messages between sensor nodes with synchronization.

I.INTRODUCTION

The CAN was developed by BOSCH in early 1980's, as a multi-master message broadcast system that specifies a maximum signaling rate upto 1Mbps. It is a serial communication protocol which was designed especially for communication between microcontrollers and later its usage is extended for industrial and automotive applications.

It doesn't send large blocks of data from one node to another node under the supervision of a central bus. This protocol was developed for verifying that data from one node gets transferred to another node securely and safely with no information defilement and without missing any of the information. This protocol was basically developed for short length data transfer such as in automobiles.

As it is a multi-master communication protocol, each node is equivalent to every other node in the system. If any node wishes to begin transmission of messages within the network then it has to wait till the bus becomes idle. Each message has a unique identifier and each frame is accessible to each other node in the network. The intended node chooses relevant message and the rest are ignored.

II. CAN PROTOCOL

CAN protocol can be implemented in two formats:

- Standard format, which has 11-bit identifier
- Extended format, which has 29-bit identifier

This protocol contains 5 types of frames:

- Data frame, which usually carries data to the requested node

- Remote frame, which requests data from a specific transmitter node
- Error frame, generated on detecting an error
- Overload frame, is used as delay by a receiver whenever it is not yet ready to receive frames.
- Interframe space, used as spacing between successive data and remote frames.

The structure of a standard and an extended frame are shown in figures 1 and 2 respectively.

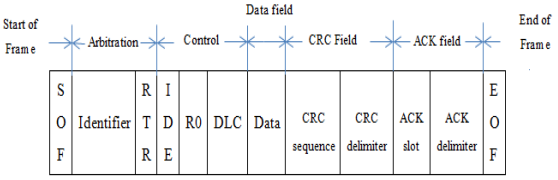

Figure 1: Structure of a Standard frame

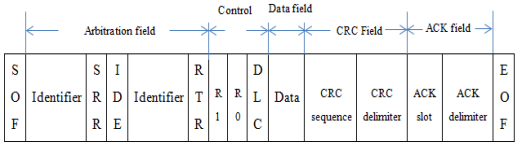

Figure 2: Structure of an extended frame

Where SOF is Start of frame. Identifier indicates the priority of the message that is being transmitted. RTR is Remote transmission request and it is'0' for data frame and '1' for remote frame.IDE is an Identifier Extension which is '0' for standard format and '1' for extended format.R1, R0bits are reserved for future use. SRR stands for Substitute remote request which substitutes RTR bit in extended format. DLC means Data Length Code. It represents the number of bytes of data exists in data field. The data field contains data of length varying from 0 to 8 bytes. The next field is 15 bit CRC sequence followed by a delimiter and this CRC delimiter value is always recessive.

After CRC, the next field is Acknowledgement slot which consists of two fields ACK slot and ACK delimiter. For a transmitter, the ACK slot is always a recessive bit. The receiver after successful reception of the message overwrites this recessive bit into dominant bit. ACK delimiter is always a recessive whether it may be a transmitter or a receiver. EOF indicates the End of frame which contains 7 recessive bits.

A data frame contains SOF, Arbitration field, Control field, Data field, CRC field, ACK field and EOF whereas a remote frame contains same fields as that of a data frame except the Data field.

An error frame is used to indicate an error has arisen in the course of transmission.

An overload frame is used in order to delay the reception of next data or remote frame and is generally used by the receive unit to inform that it is not yet ready to receive the frames. Interframe space, the name itself indicates that this frame is used for spacing between frames i.e., data and remote frames.

III. FUNCTIONAL DIAGRAM

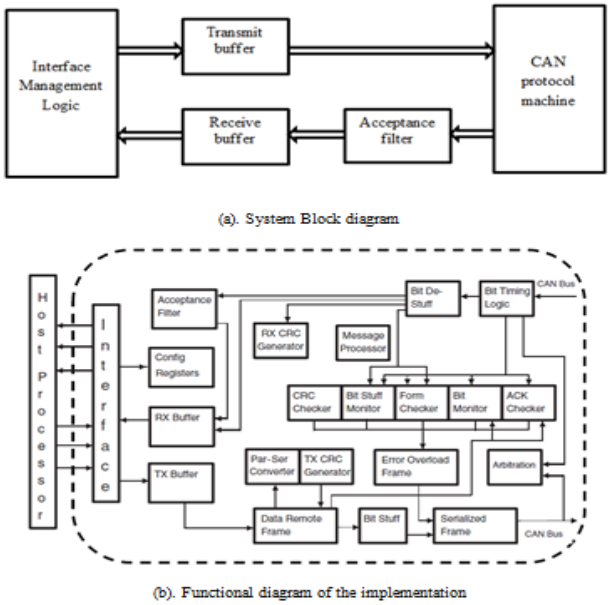

Figure 3: (a) Block diagram (b) Functional diagram of the CAN controller

IV.IMPLEMENTATION

The main objective of this paper is to achieve successful transmission of information between CAN nodes.

Transmitter:

If the host processor wants to send data on to the CAN bus then whatever the data that is to be transmitted is kept in transmit buffers via 8-bit data bus. At every clock cycle, the data is kept in transmit buffers and this data is converted into serial data for ease of calculation of CRC sequence. After calculating the CRC sequence, the bit stuffing is done for the message starting from SOF until the CRC sequence. Once bit stuffing is done, the remaining fields are appended and the entire message is transmitted onto the bus.

Receiver: The host processor can receive the message from the CAN bus only through the interface.

The received message is kept in receive buffers and only two receive buffers are used.

The message that is stored in receive buffer undergoes filtering in order to make sure that the correct message is received. After filtering mechanism, the message is checked for errors and is called error confinement. If there is any error, the same is indicated by signaling an error frame. The error free message is then sent to the host processor via 8-bit data bus.

Whatever the data that may be transmitted to a node or received from a node need to be synchronized according to the

system clock of individual nodes. This synchronization is achieved with the help of bit timing.

V.RESULTS

After calculating the CRC sequence of the data, bit stuffing has to be done and after bit stuffing the entire frame is transmitted and it is shown in fig.4.

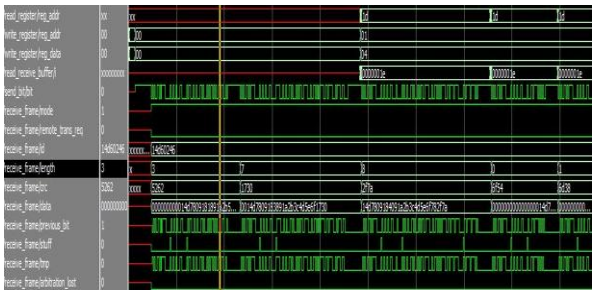

Figure 4: The transmitted data frame

At the receiver, the received data is filtered using acceptance filtering. In this mechanism, the filter checks the identifier field and if it is same as that of the intended identifer, an id_ok signal becomes active and it is shown in fig.5.

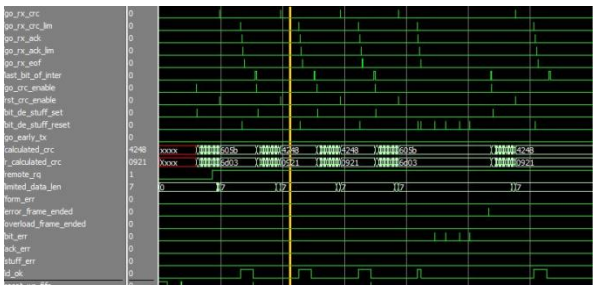

Figure 5: Acceptance filtering mechanism

on receiving every field, enable signals are generated and it is shown in fig.6 and in order to receive the data from the fields in synchronization, bit timing logic is used and it is shown in fig.7.

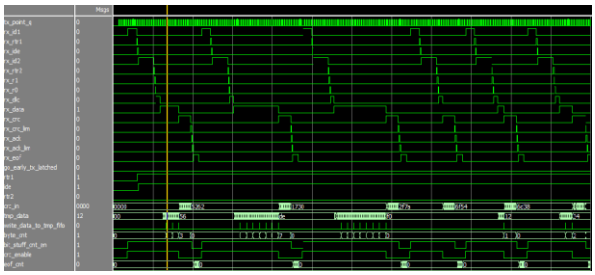

Figure 6: Enable signals that are generated at the receiver

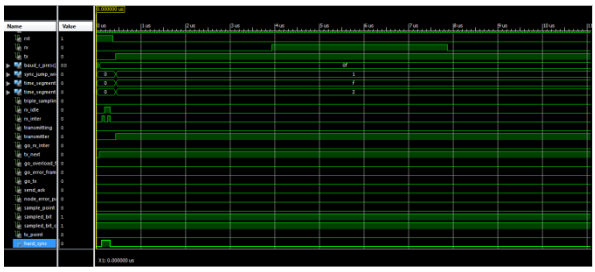

Figure 7: Hard Synchronization is generated at the starting of the frame in order to synchronize all nodes.

If the there is any error in the received frame, the same is notified by signaling an error and the status of the transmission and reception is shown in fig.8.

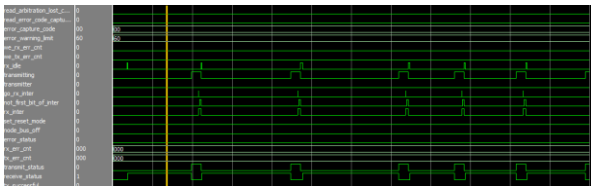

Figure 8: Error signaling and the transmitter and the receiver status

VI.CONCLUSION

In this paper, the CAN frame format is divided into modules such as FIFO, CRC, Acceptance Filtering, Error Checking, Error signaling and Bit timing Logic to perform various communication tasks. Each module is implemented as a VHDL entity and the performance of each entity is analyzed and then all entities are combined in structural style. The described entities were simulated using Modelsim. This CAN protocol controller acts as a communication module between two CAN nodes and it was observed that there is a successful transmission and reception of messages between these nodes and in addition to

this, it was also observed that Synchronization between the nodes is also achieved.

VII. FUTURE SCOPE

This CAN protocol controller can be extended to LIN protocol. The safety critical applications are implemented in CAN and non-critical applications are implemented in LIN network there by reducing the traffic between CAN nodes and increasing the signaling rate.

REFERENCES

[1]. Jose E.O.Reges, Edval J P.Santos, A VHDL CAN MODULE FOR SMART SENSORS.
[2]. MICROCHIP AN713, Controller Area Network(CAN) basics.
[3]. E.S.D. Electronics, Controller Area Network CAN– A serial bus system- Not just for vehicles.
[4]. Robert Bosch GmbH, CAN SPECIFCATION.
[5]. TEXAS INSTRUMENTS, Introduction to the Controller Area Network(CAN).