EXCELLENT PUBLISHERS

International Journal of Current Research and Academic Review

ISSN: 2347-3215 Volume 4 Number 3 (March-2016) pp. 76-85

Journal home page: http://www.ijcrar.com

doi: http://dx.doi.org/10.20546/ijcrar.2016.403.009

CAN Controller Modelling and Simulation for its Implementation in FPGA (SoC) using Verilog for Integrated CAN Node

Duhita B. Paratane* and A.M.Patil

M.E. Electronics and Telecommunication, J. T. Mahajan College of Engineering, Faizpur, India

Corresponding author

KEYWORDS

CAN (Controller area Network) controller, CAN node, FPGA, SoC, HPS, Verilog, Modules.

A B S T R A C T

A CAN (Controller Area Network) node typically consist of a CAN controller, the microcontroller and CAN transceiver. This system requires the microcontroller containing a CAN controller in it and a separate transceiver. Microcontroller integrated with CAN controller suffers performance penalty as it has to read input gives output to actuators along with CAN message reception processing and communication. All these tasks are done in microcontroller in sequential manner as per standard software application. Also we cannot use a general microprocessor for controlling the CAN network system, since we required a system built around the standalone controllers which is bulky and not cost effective. To resolve this problem, this paper presents Verilog implementation of a CAN controller in Altera Cyclone V SoC (FPGA) with Cortex A9 microprocessor for controlling CAN network. This facilitates implementation of multiple CAN controller in one device with configurable priorities. Also avails highest possible speed of controller due to parallel processing of data in FPGA. In this paper a CAN controller RTL (Register Transfer Level) logic is implemented using Verilog in accordance with CAN protocol (version 2.0A and 2.0B). Each and every functional modules of CAN controller is simulated and verified. This controller connected via Avalon Memory mapped interface, with the Hard Processor (HPS) present in Cyclone V SoC for achieving data transfer control. Complete implementation of CAN node is synthesized into Altera's Cyclone V SoC using Quartus-II software.

Introduction

The Controller Area Network (CAN) is a serial communications protocol that efficiently supports distributed real-time control with a very high level of data integrity [Bosch Controller Area Network].

This protocol was defined by BOSCH in Germany for automotive application, but it is used in wide variety of embedded application like industrial automation due its advantages such as efficiency, high flexibility, low cost and simplicity [Obermaisser and Kammerer]. Day by day the interest in CAN network is increasing rapidly due to CAN related device availability in market. Home automation system, medical devices, industrial application are new control networks which use CAN protocol in it. Some of these applications will require higher levels of integration to reduce the size, the power consumption and the price of the final system [De Lucas and Quintana].

Nowadays, there are FPGA-based integrated solutions where programmable logic is included in addition to general purpose processors, allowing dedicated hardware to be synthesized according to the application needs. Besides configurability features, low turnaround time of rapid prototyping using FPGA devices is an attractive alternative of system validation, specially when fast time-to-market is required. Equally important, FPGA technology is being largely used in final products when total demand is restricted to few units because of the high cost associated to ASIC fabrication [Carvalho *et al*., 2005].

The ability of integrating the protocol handling in a single chip together with a microprocessor core, memory and other peripherals, is called a System-on Chip [De Lucas and Quintana].

This paper aims at achieving this kind of integrity by using latest System on Chip (SoC) concept based FPGA. Altera SoC-FPGA integrate an ARM-based hard processor system (HPS) consisting of processor, peripherals, and memory interfaces with the FPGA fabric using a high-bandwidth interconnect backbone [SoC Product Brochure]. It combines the performance and power savings of hard intellectual property (IP) with the flexibility of programmable logic [SoC Product Brochure]. Paper presents the implementation of integrated CAN node where CAN controller functionality is implemented in FPGA fabric along with block memory utilisation as buffers. Which is connected with the ARM Processor (HPS) present in the same chip for achieving data communication control. This interconnection is done with high speed and high bandwidth interconnect backbone present in FPGA.

Another objective of this implementation of custom CAN controller is to increase the reaction time with the help of parallel processing of data. All the sub-modules designed in the CAN controller block works in parallel with very high frequency clock. This leads to achieve minimum possible data processing time which finally helps taking necessary decisions and starts the respective actuator within nano-seconds for Real time (Time critical) applications.

Since multiple CAN controller logic blocks can be instantiated and controlled via processor it will be easier to achieve configurable priority implementation. Each of this CAN controller module present in FPGA fabric will be connected with the HPS via Avalon memory mapped interface which will allow processor to set the priority message of particular module with runtime configuration.

Materials and Methods

CAN Overview

CAN protocol: - CAN is ISO standardised protocol defined in standard ISO 11898, it covers Data link layer and Physical layer of OSI model. The CAN protocol comprises of the Data Link Layer -composed of the

Logical Link Control (LLC) sub layer and the Media Access Control (MAC) sub layer and the Physical Layer composed of physical signalling mechanism, Medium dependent and medium independent interace. Below given Figure 1 shows layered structure of CAN protocol and its respective implementation [Dzhelekarski and Zerbe, 2004].

The CAN Data Link Layer controls the message communication. The Data Link Layer builds data frames to hold data and control information. It also provides other services such as frame identification, bus arbitration, bit stuffing, error detection, and error signalling, fault confinement and automatic retransmission of error-free frames.

The CAN Physical Layer is responsible for transfer of data between different nodes in a given network; it defines how signals are transmitted and therefore deals with issues like encoding, timing and synchronization of the bit stream to be transferred.

The application layer is specific to the particular application depending upon the sensors actuator system.

System Concept Implementation

Scope of this paper includes Data link layer and Application layer integration into one single chip. In which each layer will perform its own function in parallel so that system will work efficiently without decreasing performance. Application layer includes the HPS present in Altera Cyclone V SoC and the bare metal software program built for it. This program will control the data transfer between different nodes in the network, message analysis, receives data from sensor and controls actuators as per the requirement. If SoC contains more than one CAN controller in it then priority allocation, message controlling of different controllers is also done with the help of this software application.

CAN controller RTL is built in FPGA fabric using Verilog RTL design language. Its functionality includes CAN frame reception, data analysis, error checking, error reporting, message transfer to processor and Data transmission. Below given **Error! Reference source not found.** shows the CAN node implementation diagram.

CAN Controller Design Description

Functionality of CAN controller is mainly divided into five sections CAN Frame Transmitter, CAN Frame Receiver, CRC checking and Error handling, Message processing control, FPGA-HPS interconnect block as shown in **Error! Reference source not found.**.

CAN Frame Transmitter and CAN frame Receiver blocks are mainly responsible for serial message transmission and reception respectively from CAN bus. CRC Checking and Error Handling block calculates and checks the CRC, reports transmitter block to transmit error frame if error occurred etc. Message Processing Control block is responsible for managing the proper functionality all remaining blocks.

CAN Controller’s Internal Modules Description

CAN Frame Transmitter Block

This block contains many smaller module which performs different tasks for CAN message transmission.

The CAN FRAME TRANSMITTER module is responsible for the transmission of messages, in accordance with the CAN protocol. CAN FRAME TRANSMITTER verifies if there is a message to be transmitted and if the bus is idle? When a transmission is started, this block verifies the bus state, comparing the TX bit and RX bit. When TX bit is recessive (logical level 1) and RX bit is dominant (logical level 0), the block assumes that the unit lost the bus arbitration and must withdraw without send any more bits. When a dominant bit is sent and a recessive bit is received, the block understands an error has occurred and stops transmission. When the Message processing block informs the need of a stuff bit, CAN FRAME TRANSMITTER automatically attaches a complementary bit to the last transmitted bit as stuff bit. After receiving the Data Field from HPS, CAN FRAME TRANSMITTER requests that the CRC block to calculate the CRC Sequence and attaches that sequence in the CAN frame for its transmission.

CAN Frame Receiver Block

Sub-modules of CAN Frame Receiver Block are work synchronously to receive the CAN Frame, bit de-stuffing, calculates and validates CRC, error reporting and valid messages transfer to HPS for decision making. It receives identifier and understands the message. Depending upon the identifier field a CAN node filter out the message addressed to it. With the help of CRC and Error Handling block it validates the data which is received and if that message is intended to this particular CAN node then it forwards that data to processor for its utility. When the CRC sequence is received, Frame Receiver block compares it with the CRC sequence calculated. If CRC is matches, CAN Frame receiver acknowledges the message by sending a dominant bit.

CRC Checking and Error Handling Block

This block is responsible for transmit message and Receive message CRC calculation. Received message CRC verification. If match not found then error frame generation and invoking its transmission.

Message Processing Control

Generates control signals to synchronise all other blocks. Also responsible for bit stuff handling. If five consecutive equal bits arrives in message frame then to create additional signal transitions to ensure reliable reception one opposite polarity bit is added this is called the bit stuffing. Arbitration control is also one of its important task.

FPGA-HSP Interconnect Block

This is the Avalon MM interface which Altera’s standard interface for Qsys system interconnect. HPS uses this interface to interact with the custom peripherals modules connected to it. Here HPS acts as the Master and custom modules acts as the memory mapped slaves. HPS master provides data to be transferred to Transmitter block which is stored in transmit buffer through Avalon MM interface. And receive the message data from the receiver block buffers via the same interface.

Hard Processor System (HPS)

It is the Dual-core ARM Cortex-A9 (HPS) Embedded core used to achieve the data control and decision making unit of CAN node. Its functionality includes reading the external data from Sensor or other peripherals. Give this data to CAN controller to transmit to another node. HPS also receives data from the CAN controller to either starts some actuator or store that data as information for processing it. HPS is configured by the bare metal software application written in C language.

Functionality Explanation of Implementation

CAN controller mentioned above is implemented in Verilog language with various different module for each block mentioned in Figure 3. Explanation of the functionality of each module in the particular block is given below.

CAN frame transmitter block requires Data to be transmitted, it receives the data the Processor system. This data is then stored into the Transmission Buffers via processor to FPGA module interface. CAN frame data is then given to the parallel to serial convertor block, which then serial passed to the CRC calculation module. Here we get the CAN frame data along with the CRC attached to it. After that according to the CAN protocol the data needs bit stuffing. If five consecutive bits of equal polarity appears in the data frame then the sixth bit of opposite polarity is added in the frame to be transmitted. This task is done by the Bit stuff module. It completes the CAN data frame/ Remote frame generation which is then transmitted to the network by module sterilised frame transmitter.

Also transmitter section needs to transmit Error or overload frame depending upon the Error or overload message condition. If Error handling module indicates that the frame needs to be transmitted belongs to one of the above two category then the serialised frame transmits either Error Frame or Overload frame. CAN Frame receiver block receives serial data from network it uses bit synchroniser module to detect and collect the incoming bit and generated clock for synchronisation purpose. Next to bit synchroniser module it requires bit dyestuff module to extract the useful data from the incoming CAN frame. After removing redundant data the frame is given to the CRC block to calculate the CRC of the frame same is also given to acceptance checker module to verify that the frame received is intended for receiving CAN controller module or not. If it is intended for that particular CAN controller then it is stored in the Receiver buffers to send it to host processor via Avalon interface.

CRC checking and Error handling module consists of CRC generator module to calculate transmit or receive data CRC. CRC checker to verify the incoming CRC with calculated CRC of the received CAN frame. Form checker module verifies if any of the fixed-form fields in a received CAN message is violated. The fixed form fields include the CRC delimiter, ACK delimiter and the EOF field. A Bit Monitor module checks if a CAN node acting as the transmitter of a message, samples back the bit from the CAN bus after putting out its own bit. If the bit transmitted and the bit sampled by the transmitter are not the same, a bit error is generated. Acknowledgment Checker module functions during the transmission of the acknowledgement slot. Transmitter transmits a recessive bit and expects to receive a dominant bit. If the node receives a recessive bit in the acknowledgement slot an ACK error is signalled.

Message Processing and control block is the central unit which provides all the control and the status signals to the various other blocks in the controller. This unit routes the different signals generated in various blocks to the necessary target blocks. It consists of the module named message processor.

FPGA to HPS interconnect is the standard Avalon MM interconnect provided by

Altera Quartus II IP block. Through which the Hard Processor System (HPS) send the data based on the memory mapped data transactions. Each transaction sends 32 bits of data to the 32 bit slave and which is stored in the 8 bits transmit buffers. Data length of message data to be transmitted is also send to this slave module. If it requires data from some other node then the remote frame command is given to the slave module. After receiving the remote frame command the transmitter block of the CAN controller module send the remote frame requesting the required data to the host processor. Host processor keeps polling the receiver buffer slave after sending remote request command to get the requested data from another node.

Simulation and Synthesis Result

The design presented in this paper is implemented in verilog HDL. Described RTL logic is synthesized in Quartus II 15.1 software and the synthesis results are given in the table 1. It is simulated in Modelsim-ALTERA STARTER EDITION 10.3d – Custom Altera version. And simulation results are given in below given figures.

In Figure 4 transmit frame functionality of CAN message is shown. After reset becomes low the message transmission process starts. Processor sends the parameter and data to be transmitted to the CAN Controller. It makes the param_ld signal to go high. The data to be transmitted is stored in tx_buff_1 to tx_buff_3. This data is then given to CRC calculation module which is upended in the dt_rm_frm message. Dt_rm_frm is our message we need to transmit. After calculating the Frame length and upending the frame length into the Transmit frame the message is transmitted over the can_bus_out signal.

CAN frame receiver functionality is simulated and shown in Figure 5. Here module gets input data on can_bus_in signal. Which is given to the bit de_stuff module by indicating bit_destf_intl bit de-stuffing initialisation. Then module analyses data by asserting signals like rcvd_data_frm, rcvd_crc_flg, send_ack etc. rx_success indicates that the received frame is successfully received. Then the data written into rx_buff0.

When two nodes transmit frame on the network then one node has to stops its message transmission. This is done by the arbitration control. Message with the higher priority wins the arbitration and transmits its data. This functionality is shown in simulation Figure 6. Here CAN node 1 wins the priority and transmits the data and CAN node 2 loses the priority and stops transmitting its data. It will only then acts as the message receiver. After that CAN Node 2 transmits its data.

Synthesis Result

CAN node module is synthesized in Quartus II software with Cyclone V SoC with device part number 5CSEMA5F31C6 and its resource utilisation of FPGA fabric elements is given in table 1.

Table.1 Resource Utilisation of FPGA Fabric

Sr. No.	Resources	Usage
1	Total logic elements	2,146
2	Total combinational functions	2106
3	Total registers	1110
4	I/O pins	26

Figure.1 CAN Protocol and its Implementation

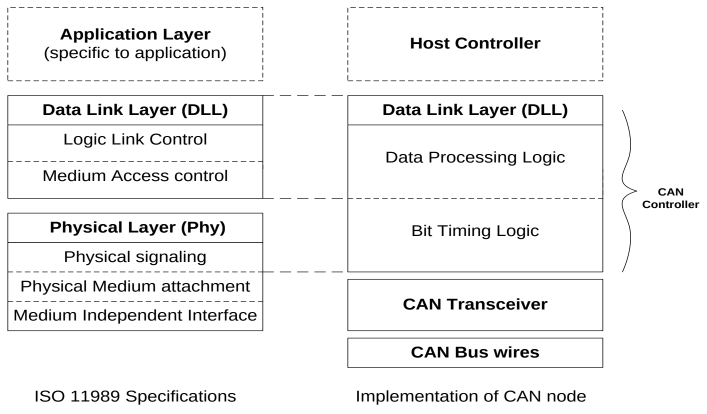


Figure.2 CAN Node Implementation Diagram

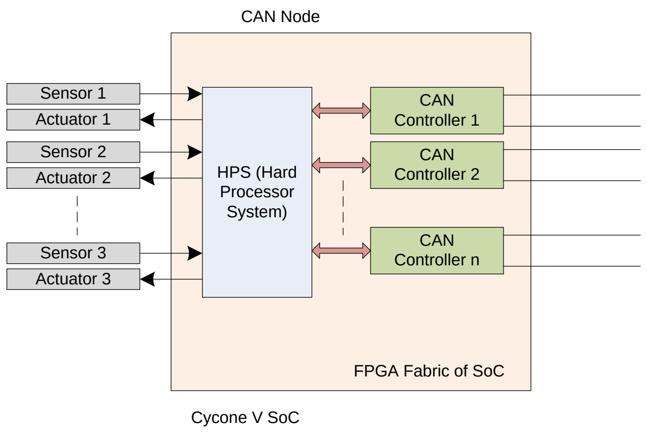


Figure.3 CAN Controller Implementation Block Diagram

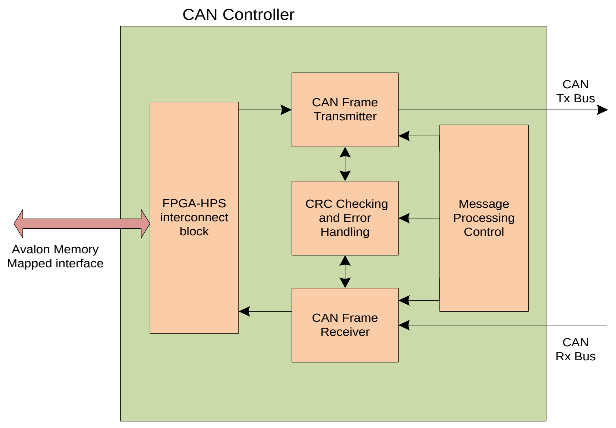

CAN Controller
CAN Frame Transmitter
CAN Tx Bus
FPGA-HPS interconnect block
CRC Checking and Error Handling
Message Processing Control
Avalon Memory Mapped interface
CAN Frame Receiver
CAN Rx Bus

Figure.4 CAN Frame Transmission Simulation Diagram

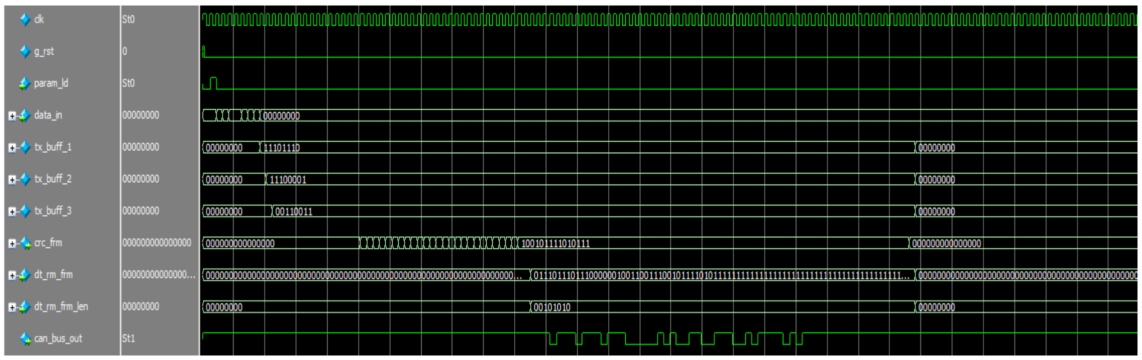


Figure.5 CAN Frame Reception Simulation Diagram

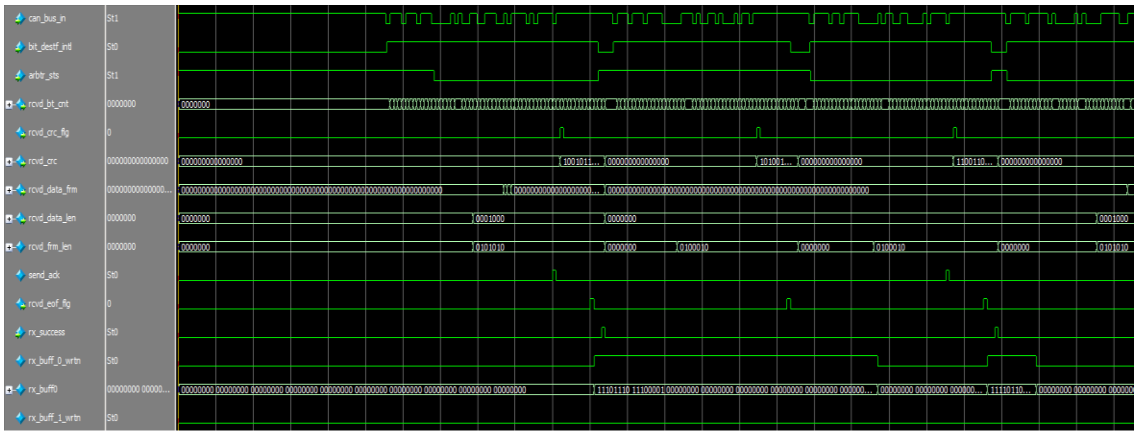
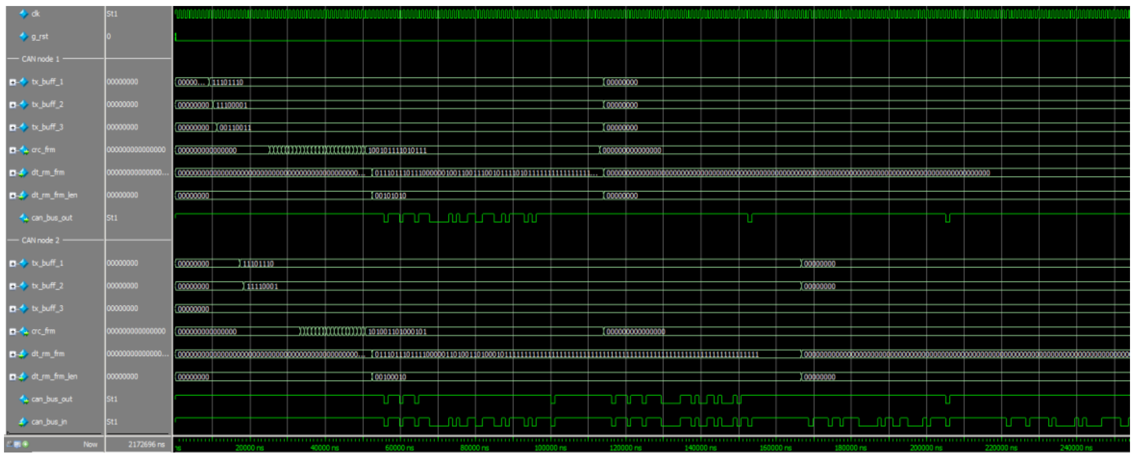


Figure.6 CAN Nodes Arbitration Simulation Diagram



Conclusion

A CAN node is successfully designed and implemented on Cyclone V SoC development kit. Its functionality is simulated on modalism software and its simulation images are given in this paper. Also hardware functionality is verified by transmitting the data from one CAN controller and the same data is given to another CAN controller in the same system via general purpose connector. After every successful communication a user led is made to glow to indicate CAN node data transfer is successful. Also the CAN controller functionality is simulated as whole and the Transmit and receive frame are observed successfully. Individual module functionality of FPGA fabric is simulated and verified and the results are noted.

References

Bosch Controller Area Network (CAN) Version 2.0 PROTOCOL STANDARD.

Obermaisser, R., Kammerer, R. 2010. “A router for improved fault isolation, scalability and diagnosis in CAN” Industrial Informatics (INDIN), 8th IEEE International Conference Pages 123–129 IEEE.

De Lucas, J., Quintana, M. 1999. “Design of CAN interface of custom circuits” Industrial Electronics Society, 1999. *IECON '99 Proceedings*, 2: 662–667, IEEE.

Carvalho, F.C., Jansch-Porto, I., Freitas, E.P. 2005.“The TinyCAN: an optimized CAN controller IP for FPGA based platforms” Emerging Technologies and Factory Automation, Pages 4, pp. 374, IEEE.

SoC Product Brochure- Altera’s user customizable ARM-Based SoC from www.altera.com/products/soc/overview.html.

Reges, J.E.O., Santos, E.J.P. 2008. “A VHDL CAN module for smart sensors” Programmable Logic, 4th Southern Conference, pp. 179–182, IEEE.

Dzhelekarski, P., Zerbe, V. 2004. “FPGA Implementation of Bit Timing Logic of CAN Controller” Electronics Technology: Meeting the Challenges of Electronics Technology Progress, pp. 214–220, Vol.2, IEEE.

Kyung Chang Lee, Hong-Hee Lee. “Network-based Fire-Detection System via Controller Area Network for Smart Home Automation” Consumer Electronics, *Consumer Electronics Soc.*, pp. 1093–1100, IEEE.

Barranco, M., Proenza, J. 2006. “An Active Star Topology for Improving Fault Confinement in CAN Networks”. *Industrial Informatics*, pp. 78–85, IEEE.

Salem HASNAOUI, Oussema KALLEL. 2003. “An Implementation of a Proposed Modification of CAN protocol on CAN Fieldbus Controller Component for Supporting a Dynamic Priority Policy”. Industry Applications Conference, pp. 23–31, Vol.1, IEEE.

http://www.can-cia.de 2004. Homepage of the organization CAN in Automation (CiA).

How to cite this article:

Duhita B. Paratane and A.M. Patil. 2016. CAN Controller Modelling and Simulation for its Implementation in FPGA (SoC) using Verilog for Integrated CAN Node. *Int.J.Curr.Res.Aca.Rev.*4(3): 76-85. doi: http://dx.doi.org/10.20546/ijcrar.2016.403.009