ISSN (ONLINE): 2454-9762
ISSN (PRINT): 2454-9762
International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE)
Vol. 2, Issue 10, October 2016

Enhancement And Performance Comparison Of I2C, UART and CAN Bus Protocols

Shri. Kiran B Kadakuntla
Lecturer, *E&C Dept.*
Govt. Polytechnic, Hubli, India
Email: kiran.bk800@gmail.com

***Abstract*—The work presented in this paper addresses the concepts of Inter Integrated Circuits (I2C), Universal Asynchronous Transmitter and Receiver (UART) and Controller Area Network (CAN). It presents the simulation of I2C, UART and CAN protocol in ISIM simulator and implementation in FPGA board.**

It gives the architecture details of different protocols with its logical diagrams and provides the information regarding the dataflow. It explains how the data transfer initiates and terminates. Frame format of different protocols are explained thoroughly and shown its simulation results. Here, it is also presented the Resistor Transistor Logic (RTL) diagram of all the protocols. In this, implementation of the different protocols for the application of fuel measurement is explained with the help of practical implementation using the FPGA kit.

This paper provides various performance comparisons such as Timing, Data Rate, Architecture complexity etc. It also covers some of the advancement in different protocols like parity bit addition in I2C Protocol, CRC addition in I2C protocol, CRC addition in UART protocol and Parity addition in CAN protocol. The advancement in each protocol shows how to increase the data rate and decrease the architecture complexity. It also gives the implementation details of these advancements along with the data rate and timing information.

Keywords*—*i2c, can, uart, bus protocol

I. INTRODUCTION

Bus is a channel which connects multiple devices and allows flow of information between devices or units. Bus usually carries signal like address signals, data signals, clock signals etc. As Bus is shared between many devices, rules are necessary to establish a proper communication. These rules are called Protocols. Buses can be classified as synchronous and asynchronous based on whether a communication is controlled by clock or not. Buses can be classified as serial and parallel, based on whether data bits are sent through Parallel wires or multiplexed into single wire.

Serial communication has advantage of having less hardware requirement compared to parallel communication but at the cost of low data rate. Where as in parallel communication, data rate is high with the more hardware requirement.

The examples of serial buses are I2C, UART, CAN etc. and parallel buses are ISA, PCI, EISA and VESA etc.

This paper deals about introduction of I2C bus, CAN bus and UART bus and its implementation in the FPGA board. It also deals about the improvements can be introduced in the different protocols like I2C, CAN and UART.

II. LITERATURE SURVEY

1. *Inter Integrated Circuits (I2C):*

I2C is a serial communication protocol. There are many other serial protocols for serial data communication but they require more number of pins. But as per Moores law statement the transistors on the chip getting double approximately every 18 months and less number of pin connections are available for serial communication hence I2C protocol preferred over other protocols [1][2]. It is a popular among all other serial communication protocols due to its collision detection and arbitration feature [1]. It is a two wire bi-directional bus that provides more efficient method of data transmission [3].

I2C has two bidirectional wires; those are SDA (Serial Data line) and SCL (Serial Clock line). The devices connected by I2C are identified by a unique address.

In this, the device which initiates the action by producing clock is called as Master and all other devices connected in the network are called as Slaves [1]. It supports multi-master operation but only one device is allowed to initiate the data transfer. The device which sends data onto the I2C bus is called transmitter and device which receives the data from bus is called as receiver [1].

The Communication over I2C bus consists of: Generation of Start signal, Transfer of Slave address, and Transfer of Data and Generation of Stop signal [2]. The data transfer initiation is indicated by sending Start signal followed by the address of slave device. Once the slave is addressed, data transfer takes place. The end of data transfer is signaled by the Stop signal [2].

2. *Controller Area Network (CAN):*

Controller Area Network in short called as CAN protocol. CAN is a serial data communication protocol developed by Bosch in the early 1980s [14]. It is mainly developed to solve the problem exists in the automobiles.

The automobile vehicles have complex electronic devices like engine ignition, air bag controller, anti braking system etc.

CAN is a bi-directional bus which is very popular for its error detection and correction mechanism. CAN follows the Open System Interconnect (OSI) model and uses only Physical layer and Data link layer [8].

CAN format has data frame, remote frame, error frame and overload frames. Data frame starts with the start of frame (SOF) bit followed by eleven bits of identifier and remote transmission request (RTR) bit. Then it is followed by six bits of control field and then data field. Data field is followed by sixteen bits of CRC bits which enables error detection and correction. CRC field is followed by one acknowledgment bit (ACK) and seven end of frame (EOF) bits. And every CAN format end with Inter frame space (IFS) [5]. [4] proposed a secure hash message authentication code. A secure hash message authentication code to avoid certificate revocation list checking is proposed for vehicular ad hoc networks (VANETs). The group signature scheme is widely used in VANETs for secure communication, the existing systems based on group signature scheme provides verification delay in certificate revocation list checking. In order to overcome this delay this paper uses a Hash message authentication code (HMAC). It is used to avoid time consuming CRL checking and it also ensures the integrity of messages. The Hash message authentication code and digital signature algorithm are used to make it more secure . In this scheme the group private keys are distributed by the roadside units (RSUs) and it also manages the vehicles in a localized manner. Finally, cooperative message authentication is used among entities, in which each vehicle only needs to verify a small number of messages, thus greatly alleviating the authentication burden. [6] discussed about Reconstruction of Objects with VSN. By this object reconstruction with feature distribution scheme, efficient processing has to be done on the images received from nodes to reconstruct the image and respond to user query. Object matching methods form the foundation of many state- of-the-art algorithms. Therefore, this feature distribution scheme can be directly applied to several state-of- the-art matching methods with little or no adaptation. The future challenge lies in mapping state-of-the-art matching and reconstruction methods to such a distributed framework. The reconstructed scenes can be converted into a video file format to be displayed as a video, when the user submits the query. This work can be brought into real time by implementing the code on the server side/mobile phone and communicate with several nodes to collect images/objects. This work can be tested in real time with user query results.

3. ***Universal Asynchronous Receiver Transmitter Controller (UART):***

UART is a one of the serial communication bus. It is used to connect two devices. As this bus does not use the clock for synchronization it is named as Asynchronous bus. A UART contains both transmitter and receiver. It does the data conversion of parallel to serial during transmission and converts serial to parallel at the receiver [12].

Since UART does not pass the clock along with the data, receiver generates its local clock and in advance both transmitter and receiver agree on timing parameter. As UART is used to communicate between only two devices there is no necessity of destination addressing. The UART is mainly used for low speed, low cost and short distance communication [13].

The UART mainly including three main components such as transmitter, receiver and baud rate generator. Baud rate generator is used as frequency divider [12].

In transmitter of UART, a start bit is added to beginning of the data bits. A start bit is added to indicate the receiver that data bits arriving. For every word of data one parity bit is generated and sent after the data bits. Parity bit helps the receiver to check whether it is received correct data. After parity bit a stop bit is sent to indicate the end of data transmission [11]. In this, time taken to transmit each bit is depends on the baud rate. The baud rate may be 4800, 9600 or 19200 bauds [11].

III. INTER INTEGRATED CIRCUITS (I2C)

1. Introduction of I2C

Inter Integrated Circuit (I2C) was introduced by Philips which allows communication between faster devices and slower devices through a serial data bus without having data loss.

The I2C bus is a two wire, duplex serial bus and it offers a simple and efficient way of short distance data transmission between different devices. It is low-bandwidth, simple, short distance protocol. Standard I2C device operate at speed up to 100kbps and fast mode operate at speed up to 400kbps.

It has in-built address hence multiple devices can be connected together easily. The I2C has two signals i.e., serial data (SDA) signal and serial clock (SCL) signal as shown in figure. Every device connected by the I2C bus has unique address and it is addressable by using software. During data transfer devices are considered as master or slaves. A master is the main device which takes initiation in data transmission over the bus and also takes initiation in generation of clock signals. This protocol supports multiple masters but only one device can be a master at a time. Any other device addressed at that time will be known as slave. Both master and slave can transfer and receive data. The device which sends data onto the I2C bus is called transmitter and device which receives the data from bus is called as receiver.

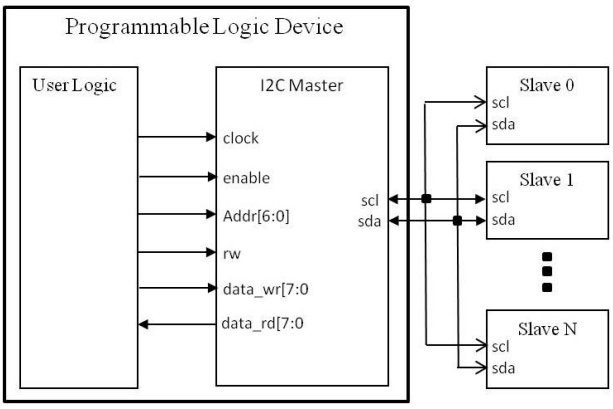

Fig 1.1: Logical diagram of I2C Master-Slave

2. I2C Protocol

According to the specification of I2C protocol the device which starts the data transfer onto bus is called as master.

The four parts of a standard I2C communication is

- Generation of START signal.
- Transfer of Slave address.
- Transfer of Data.
- Generation of STOP signal.

3. Simulation Results

The I2C master and slave has been designed using HDL and simulated using ISim.

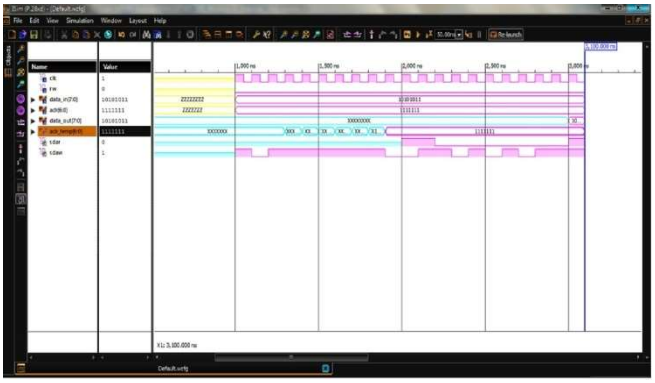

Figure.1.2. ISim simulation for writing data to slave

Figure 1.2 shows the simulation result of I2C developed in verilog, which shows the data transfer from master to slave serially through the SDA.

4. FPGA Synthesis Results

Xilincs version 14.2 is used for synthesizing I2C master and slave design on FPGA sparton3.

HDL (Hardware Description Language) Synthesis Report: Figure 1.3 shows the HDL Synthesis Report obtained from XILINX 14.2

```
=========================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 3
 32-bit adder                                         : 1
 4-bit adder                                          : 2
# Counters                                             : 2
 32-bit up counter                                    : 1
 4-bit up counter                                     : 1
# Registers                                            : 23
 1-bit register                                       : 19
 4-bit register                                       : 2
 8-bit register                                       : 2
# Comparators                                          : 8
 33-bit comparator greater                            : 1
 33-bit comparator lessequal                          : 1
 4-bit comparator greatequal                          : 2
 4-bit comparator greater                             : 1
 4-bit comparator less                                : 2
 4-bit comparator lessequal                           : 1
# Multiplexers                                         : 2
 1-bit 7-to-1 multiplexer                             : 1
 1-bit 8-to-1 multiplexer                             : 1

=========================================================================
```

Figure1.3. HDL Synthesis Report

Device Utilization Summary on FPGA: Table No. 1.1 shows the Device utilization Summary of the I2C Master-Slave. The summary shows that minimum resources are utilized by the I2C Master-Slave as only 4% slices, 2% Flip Flops and 4% of LUTs are utilized.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	93	1920	4%
Number of Slice Flip Flops	83	3840	2%
Number of 4 input LUTs	162	3840	4%
Number of bonded IOBs	32	141	22%
Number of GCLKs	1	8	12%

Table No.1.1 Device Utilization Summary

Timing Summary of I2C: Figure 1.4 gives the timing summary of the I2C Master-Slave design.

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 9.784ns (Maximum Frequency: 102.204MHz)
   Minimum input arrival time before clock: 5.910ns
   Maximum output required time after clock: 6.216ns
   Maximum combinational path delay: No path found
```

Figure 1.4 Timing Summary of I2C Master-Slave

5. Implementation of I2C protocol in FPGA

I2C bus protocol is implemented for the application of fuel measurement in automobile using the FPGA

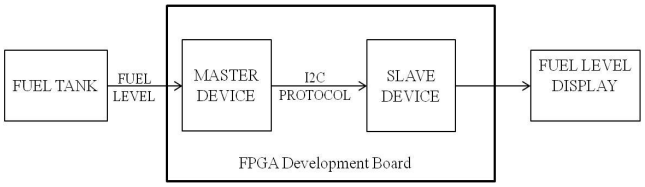

Figure 1.5 Implementation of Fuel level measurement using I2C protocol

Figure 1.5 shows the block diagram of I2C implementation for fuel level measurement using FPGA Development board.

In this fuel level is measured using fuel level sensor and fuel level is sent to Master device. Master device sends the fuel level data to slave device in the I2C format. Slave device separates the fuel level data from received data which is in the I2C data format and sent to display device.

Data transfer using i2c protocol: Fuel level is in the form of analog and ADC0804 is used to convert analog to digital. The digital data is sent to master node designed in FPGA. Master node converts fuel level data into I2C format and sent to slave node designed in the FPGA. Slave node receives data in the form of I2C format and recovers the fuel level data. This data is sent to Display System.

6. Conclusion

The I2C bus design is simplified using verilog VHDL. It shows that designer can prepare their design independent of any particular fabrication technology. Any advancement in technology or new technology can be evaluated by using advanced fabrication technology and feed the design results to the logic synthesis and create a new gate level netlist. The logic synthesis result shows the circuit area and timing information for the new technology.

IV. CONTROLLER AREA NETWORK (CAN)

1. Introduction

Controller Are Network (CAN) was first introduced by Bosch in the early 1980s. CAN protocol is very popular in the automobile applications and also it is used in trains, boats and many other systems. Need of CAN increased as more and more vehicles contain network of many electronic devices. CAN has popularity due to its high reliability and it become an international standard, i.e., ISO 11898 and ISO 11519.

Before CAN protocol was invented, in the automobile all electronic devices used to connect using wires (point to point wiring) as shown in figure

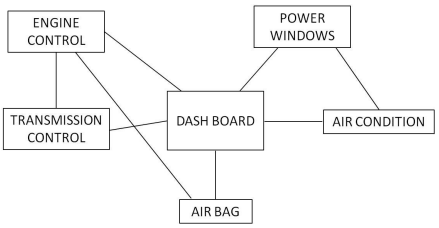

Figure 1.6 Point to point connection

Point to point wiring works fine for limited number of devices but as number of devices increases communication become difficult using point to point wiring. Controllers Area Network protocol was designed to address the above said problem. It defined many rules using which many devices can share the data through a common serial bus and it reduced the complexity and bulkiness of the system.
CAN is a multi-master protocol, means it supports multiple devices to operate as a master. It supports addition of new devices without changing the original hardware. It has error checking feature to prevent the fault in the operation and it has many advantage like low cost, high reliability and high flexibility etc.

2. CAN Protocol

Various nodes connected by CAN network and each node has a host controller which is responsible for proper functioning of node in the network. Along with host controller, each node has CAN Controller and CAN Transceiver. Function of the CAN controller is to convert the messages of node in according to the CAN protocol. As it supports multi masters, every node is able to read and write on to the CAN bus. Whenever any node has data to send on the bus it verifies the availability of the bus then writes onto the bus.

Generally Protocols are classified into two types:
1) Address based and 2) message based.

In address based protocol each data packet contains the address of destination device. In contrast the message based on protocol, a predefined unique ID is used to identify the message.

CAN protocol operate on message based communication. A unique identification number is assigned to each message in CAN.

In CAN messages are sent in frames.

SOF	11 bit Identifier	RTR	IDE	r0	DLC	0...8 Bytes of Data	CRC	ACK	EOF	IFS

Figure 1.7 CAN Frame format

Figure 1.7 shows the CAN frame format. The various fields in CAN are as follows.
SOF – Start of frame bit. Synchronization of the nodes on the bus is done by this and it indicates the initiation of the message. Logic "0" on this bit indicates the starting of the frame.
Identifier – It shows the node which has access to the bus. It is of 11bit length.
RTR – Remote Transmission Request. It differentiates between data frame and remote frame. Logic "0" on this bit indicates it as a data frame and Logic "1" as remote frame.
IDE – Identifier Extension. It specifies the frame format. Logic "0" on this bit indicates as standard frame and Logic "1" as extended frame.
R0 – Reserved bit. It is reserved for future use.
DLC – Data Length Code. It is four bit data length code and it has number of bytes to be transmitted.
DATA – This field stores the application data ranging from 0 to 64 bits to be transmitted.
CRC – Cyclic Redundancy Check. It is of 16bit and it including the checksum of the preceding data for error detection.
ACK – Acknowledge bit. It indicates acknowledgement is positive or negative. When correct data received, logic "1" on this field is overwritten by logic "0".
EOF – End of Frame. It is a 7 bit field indicates end of frame.
IFS – Inter Frame Space. It separates the consecutive messages. It shows the space between two frames. This IFS time allows nodes for internal processing before commencement of the next frame.

3. Simulation Results

The CAN Source and Destination has been developed in HDL and simulated using ISim and is used to verify the functioning of the design.

Figure 1.8. ISim simulation for writing data source to destination
Figure 1.8 shows the simulation result of CAN protocol developed in Verilog, which shows the data transfer from source to destination.

3. FPGA Synthesis Results

Xilincs 14.2 has been used for the synthesis of CAN Source and Destination on FPGA.

HDL (Hardware Description Language) Synthesis Report: Figure 1.9 shows the HDL Synthesis Report obtained from XILINX 14.2

```
=========================================================================
HDL Synthesis Report

Macro Statistics
# ROMs                                          : 1
 4x1-bit ROM                                    : 1
# Adders/Subtractors                            : 4
 32-bit adder                                   : 4
# Counters                                      : 1
 32-bit up counter                              : 1
# Registers                                     : 6
 1-bit register                                 : 1
 32-bit register                                : 4
 8-bit register                                 : 1
# Comparators                                   : 5
 33-bit comparator less                         : 5
# Multiplexers                                  : 1
 1-bit 8-to-1 multiplexer                       : 1

=========================================================================
```

Figure 1.9. HDL Synthesis Report

Device Utilization Summary on FPGA:
Table No. 1.2 shows the Device utilization Summary of the CAN Source and Destination.

The summary shows the resources utilized by the CAN Module is only 10% slices, 4% Flip Flops and 9% of LUTs are utilized.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	201	1920	10%
Number of Slice Flip Flops	176	3840	4%
Number of 4 input LUTs	380	3840	9%
Number of bonded IOBs	10	141	7%
Number of GCLKs	1	8	12%

Table No.1.2: Device Utilization Summary

Timing Summary: Figure 1.11 gives the timing summary of the CAN Source and Destination design.

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 7.874ns (Maximum Frequency: 126.995MHz)
   Minimum input arrival time before clock: 2.346ns
   Maximum output required time after clock: 6.306ns
   Maximum combinational path delay: No path found
```

Figure 1.11 Timing Summary of I2C Master-Slave

Total memory usage is 203212 kilobytes

4. Implementation of CAN protocol in FPGA

CAN protocol is implemented for the application of fuel measurement in automobile using the Xilinx sparton3 development board.

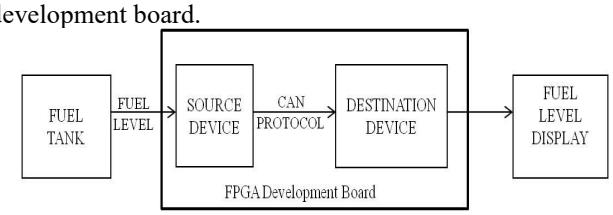

Figure 1.12 Implementation of Fuel level measurement using CAN protocol

Figure 1.12 shows the block diagram of CAN implementation for fuel level measurement using FPGA Development board.

In this fuel level is measured using fuel level sensor and fuel level is sent to source device. Source device sends the fuel level data to destination device in the CAN format. Destination device separates the fuel level data from received data which is in the CAN format and sent to display device.

Data transfer using CAN protocol: Fuel level is in the form of analog and ADC0804 is used to convert analog to digital. The digital data is sent to source node designed in FPGA. Source node converts fuel level data into CAN format and sent to destination node designed in the FPGA. Destination node receives data in the form of CAN format and recovers the fuel level data. This data is sent to Display System.

5. Conclusion

The design of CAN bus is simplified using verilog VHDL.

V. UNIVERSAL ASYNCHRONOUS RECEIVE-TRANSMIT (UART)

1. Introduction

Universal asynchronous receive/ transmit (UART) is a simple way to share the data between two different systems. It is used for long distance communication. It converts parallel data to serial and vice versa. It includes both transmitter and receiver.

During transmission, UART fetches the data word in parallel and converts it into serial and transmits it. Similarly during reception, UART receives data serially and converts it into parallel and send it to a receiver system. Since the UART is asynchronous, the receiver does not know when the data will come, hence Start bit is used to indicate the receiver that the transmitter is about to send the data. The receiver generates the local clock to synchronize with the transmitter. Both transmitter and receiver agreed to common clock well before the communication is established.

The UART adds the extra bit to create error free communication.

2. UART Protocol

UART mainly has three components such as transmitter, receiver and baud rate generator.

2.1 UART Transmitter: UART Transmitter function is to convert the 8 bit data parallel data to serial. Transmitter adds the extra bits to data bits such as Start bit, Parity bit, and Stop bit.

Start Bit: It is to indicate the receiver that transmitter about to send data bits.

Parity bit: It is sent to receiver to detect the error in the received data.

Stop Bit: It is to indicate the receiver about the completion of the data transmission by the transmitter.

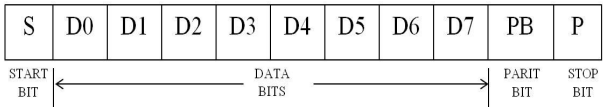

Figure 1.13 UART Frame Format

Figure 1.13 shows the UART frame format. It shows that data transfer begins with the start bit, followed by the 8 bit of data bits. After entire data bits are sent the transmitter sends the one bit of parity bit which is generated using data bits.

2.2 UART Receiver: UART Receiver function is to receive the data serially sent by transmitter. It waits for the start bit and after start bit, it receives the 8 bit data and converts it to parallel. Receiver calculates the parity bit based on the received data bits and compares the calculated parity bit with the received parity bit. If both are same then it considers the received data is error free.

2.3 Baud Rate Generator: It is used to generate the baud rates for the both transmitter and receiver.

$$\text{Baud Rate} = \frac{\text{Clock Frequency}}{(\text{Sampling Rate}) * (\text{Divisor})}$$

Standard baud rates are multiple of 9600bps like 19200 bps, 38400 bps, 57600 bps, 115200 bps, 230400 bps, 460800 and bps, 921600 bps

3. Simulation Results

The UART protocol has been developed in verilog and simulated using ISim and it is used to check the functioning of the design.

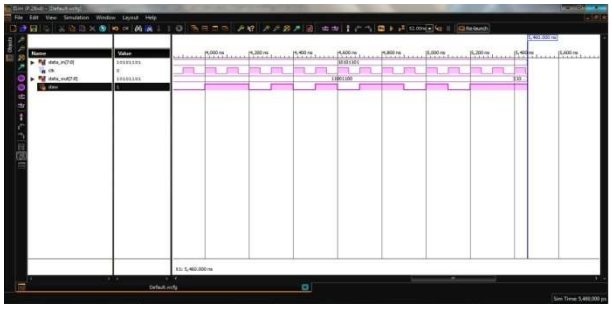
Figure 1.14 ISim simulation for sending data from source to destination using UART protocol

Figure 1.14 shows the simulation result of data transfer from one device to other using UART protocol.

4. FPGA SYNTHESIS RESULTS

Xilincs 14.2 has been used for the synthesis of UART protocol on FPGA.

HDL (Hardware Description Language) Synthesis Report: Figure 1.15 shows the HDL Synthesis Report obtained from XILINX 14.2

```
=========================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 1
 32-bit adder                                          : 1
# Counters                                             : 1
 4-bit up counter                                      : 1
# Registers                                            : 14
 1-bit register                                        : 11
 32-bit register                                       : 1
 8-bit register                                        : 2
# Comparators                                          : 2
 33-bit comparator less                                : 1
 4-bit comparator less                                 : 1
# Multiplexers                                         : 1
 1-bit 8-to-1 multiplexer                              : 1
# Xors                                                 : 1
 1-bit xor2                                            : 1

=========================================================================
```
Figure 1.15 HDL Synthesis Report

Device Utilization Summary on FPGA:

Table No. 1.3 shows the Device utilization Summary of the UART Source and Destination.

The summary shows the resources utilized by the UART as only 3% slices, 1% Flip Flops and 3% of LUTs are utilized.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	72	1920	3%
Number of Slice Flip Flops	68	3840	1%
Number of 4 input LUTs	127	3840	3%
Number of bonded IOBs	17	141	12%
Number of GCLKs	1	8	12%

Table No.1.3 Device Utilization Summary

Timing Summary: Figure 1.16 shows the timing summary of the UART source and destination design.

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 6.709ns (Maximum Frequency: 149.059MHz)
   Minimum input arrival time before clock: 1.572ns
   Maximum output required time after clock: 6.216ns
   Maximum combinational path delay: No path found
```
Figure 1.16 Timing Summary of UART source and destination

5. Implementation of UART protocol in FPGA

UART protocol is implemented for the application of fuel measurement in automobile using the Xilinx sparton3 development board.

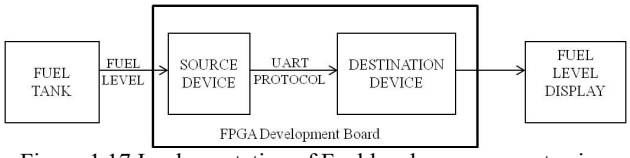

Figure 1.17 Implementation of Fuel level measurement using UART protocol

Figure 1.17 shows the block diagram of UART implementation for fuel level measurement using FPGA Development board.

In this fuel level is measured using fuel level sensor and fuel level is sent to source device. Source device sends the fuel level data to destination device in the UART format. Destination device separates the fuel level data from received data which is in the UART format and sent to display device.

Fuel Measurement: Fuel measurement is made using potentiometer. Potentiometer is connected to ball which floats on fuel.

Figure 1.18 Fuel Measurement

Figure 1.18 shows the arrangement of fuel measurement. It shows that potentiometer is connected to floating ball. Ball floats on fuel and as fuel level vary, ball position changes and it results in change in potentiometer value. Potentiometer value is taken as input data and sent to the FPGA Board.

Data transfer using UART protocol: Fuel level is in the form of analog and ADC0804 is used to convert analog to digital. The digital data is sent to source node designed in FPGA. Source node converts fuel level data into UART format and sent to destination node designed in the FPGA. Destination node receives data in the form of UART format and recovers the fuel level data. This data is sent to Display System.

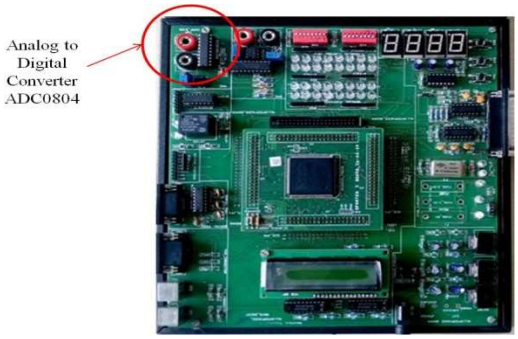

Figure 1.19 Analog to Digital Converter using ADC0804

Display System: Destination node sends the data to the display device. LCD is used to display the fuel level. Fuel level data is converted into equivalent ASCII code to display on the LCD.

Figure 1.20 LCD display

6. Conclusion

The design of UART bus is simplified using verilog VHDL.

VI. ENHANCEMENT AND PERFORMANCE COMPARISON OF I2C, UART AND CAN BUS PROTOCOLS

This chapter gives the performance comparison between Inter Integrated Circuit (I2C), Universal asynchronous receives and transmits (UART) and Controller area network (CAN) protocol.

Data Size: 256 Bytes

Parameters	I2C	CAN	UART
Timing	231.4µS	368µS	239.92µS
Data Rate	1.106Mbps	0.695Mbps	1.066Mbps

Data Size: 512 Bytes

Parameters	I2C	CAN	UART
Timing	461.8µS	736µS	479.54µS
Data Rate	1.108Mbps	0.695Mbps	1.067Mbps

Data Size: 1024 Bytes

Parameters	I2C	CAN	UART
Timing	922.6µS	1472µS	958.77µS
Data Rate	1.109Mbps	0.695Mbps	1.068Mbps

Data Size: 2048 Bytes

Parameters	I2C	CAN	UART
Timing	1844.2µS	2944µS	1917.24µS
Data Rate	1.1105Mbps	0.695Mbps	1.068Mbps

Above table shows the performance comparison of different protocols for various sizes of data.
It shows that there is no much difference in timing and data rate for different data sizes but in comparison of difference protocol it shows that CAN protocol has very low Data rate. I2C has slight higher data rate compared to UART protocol.

VII. ENHANCEMENT IN BUS PROTOCOL

1. I2C Protocol

Present Inter Integrated Circuit has a following frame format.

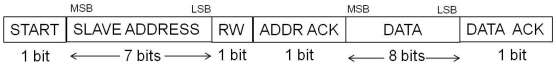

Figure 1.21 Present I2C frame format

In this, it checks only whether slave has received data but it does not have facility of error detection.
The following enhancements can be introduced in I2C protocol for error detection.

1.1 I2C with parity bit addition:

Figure 1.22 I2C with parity addition

Above figure shows the addition of parity bit in the I2C format. With the help of parity bit, receiver can check whether received data has error.

Parameters	I2C	I2C with parity
Timing	922.6µS	1025µS
Data Rate	1.1099Mbps	0.999Mbps

Table 1.4. Timing and Data rate comparison of I2C & I2C with parity bit

Above table shows the comparison of data rate of I2C format with parity bit and without parity bit (Present). It shows that adding parity bit to I2C format does not affects

the data rate but it will provides the facility of error detection.

1.2 I2C with CRC bits addition

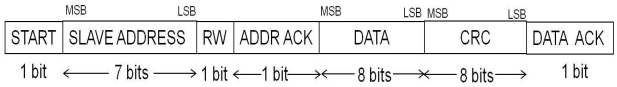

Figure 1.23 I2C with CRC addition

Above figure shows the addition of 8 bits of CRC to the I2C format. 8 CRC bits are sent after every data set.

Parameters	I2C	I2C with CRC
Timing	922.6µS	1844.1µS
Data Rate	1.1099Mbps	0.555Mbps

Table 1.5 Timing and Data rate comparison of I2C & I2C with CRC bits

Above table shows the comparison of data rate of I2C format with CRC bits and without CRC bits. It shows that adding CRC bits to I2C format decreases data rate by half but it adds the advantage of error detection.
It can be used where low data rate is acceptable for the error detection feature.

2. CAN Protocol

Following format shows the present CAN protocol format.

Figure 1.24 Present CAN protocol

It is very popular in automobile applications. It has the advantage of error detection using CRC bits. But data rate of CAN protocol is very low compared to all other protocols.

The following enhancements can be introduced in CAN protocol to increase the data rate.

2.1. CAN without CRC frame

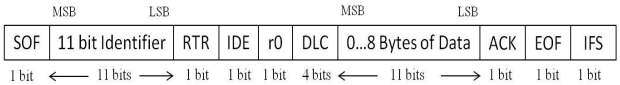

Figure 1.25 CAN without CRC bits

Above figure shows the CAN protocol without the CRC bits. By removing CRC bits in CAN protocol reduces the error detection capability but increases the data rate.

Parameters	CAN	CAN without CRC
Timing	1472µS	1267.2µS
Data Rate	0.695Mbps	0.808Mbps

Table 1.6. Timing and Data rate comparison of CAN & CAN without CRC bits.

Above table shows that by removing CRC bits in the CAN protocol increases the data rate from 0.695Mbps to 0.808Mbps. It can be used where data rate is of prime concern compared to error detection capability.

2.2. CAN with replacing CRC bits with parity bit.

Figure1.26 CAN replacing 16 CRC bits with 1 parity bit.

Above figure shows the CAN frame without the CRC bits but it has added parity bit to provide error detection.

Parameters	CAN	CAN with Parity bit (Without CRC)
Timing	1472µS	1280µS
Data Rate	0.695Mbps	0.800Mbps

Table 1.7 Timing and Data rate comparison of CAN & CAN with CRC replaced by parity bit.

Above table shows that by replacing 16 bits of CRC with one bit of parity increases the data rate from 0.695Mbps to 0.800Mbps. It does not give exact location of error bit but it shows the presence of error in the received data.

3. UART Protocol

Following format shows the UART protocol format.

Figure 1.27 UART frame format

The UART protocol has the parity bit to detect the error in the received bits. The UART protocol can be altered as following to increase the error detection capability by replacing parity bit by CRC bits.

3.1. UART with CRC frame

Figure 1.28 UART with CRC bits

Above figure shows the UART protocol with 8 bits of CRC replacing one parity bit.

Parameters	UART	UART with CRC bits
Timing	958.77µS	1704.248µS
Data Rate	1.068Mbps	0.6008Mbps

Table 1.8 Timing and Data rate comparison of UART & UART with CRC bits

Above table shows that by replacing parity bit with 8 bits of CRC, decreases the data rate but it will increase the single bit and multi bits error detection capability of UART protocol.

VIII. CONCLUSION AND FUTURE WORK

In this thesis we have worked on different bus protocols and modeled in hardware. The simulation results obtained in Xilinx for different bus protocols have been used to compare its timing and data rate. The work presented in this paper gives clear differences between different bus protocols and enhancements in each protocol. The future work can be carried out by concentrating on enhancement in each protocol mentioned in this paper by simulation and implementing the changes in the FPGA board.

BIBLIOGRAPHY

[1] Shivani Mehrotra, Nisha Charaya, " Design And Implementation Of I2C Single Master On Fpga Using Verilog", ISSN 2347-6680 (E), 2015 pp. 001-005.

[2] MR. J. J PATEL, 2 PROF B. H. SONI, "Design And Implementation Of I2C Bus Controller Using Verilog", ISSN: 0975 – 6779| NOV 12 TO OCT 13 | VOLUME – 02, ISSUE – 02, pp. 520-522.

[3] G.KrishnaKishore, K.Shruti, M.Varsha, "Design and Simulation of I2C bus using Verilog", ISSN: 2231-5381, 2014 pp. 244-247.

[4] Christo Ananth, M.Danya Priyadharshini, "A Secure Hash Message Authentication Code to avoid Certificate Revocation list Checking in Vehicular Adhoc networks", International Journal of Applied Engineering Research (IJAER), Volume 10, Special Issue 2, 2015,(1250-1254).

[5] Robert Bosch GmbH, "CAN Specification", Version 2.0, Postfach 50, D-7000 stuttgat 1, 1991 pp. 010-021.

[6] Christo Ananth, M.Priscilla, B.Nandhini, S.Manju, S.Shafiqa Shalaysha, "Reconstruction of Objects with VSN", International Journal of Advanced Research in Biology, Ecology, Science and Technology (IJARBEST), Vol. 1, Issue 1, April 2015, pp:17-20.

[7] Tejaswini Hulawale, Neha Koul, Shivang Gupta, "FPGA Based Can Protocol Controller", ISSN (online): 2348 – 7550, 2015 pp. 581-587.

[8] Vikash Kumar Singh, Kumari Archana, "Implementation Of 'CAN' Protocol In Automobiles Using Advance Embedded System", ISSN: 2231-5381, 2013 pp. 4422-4424.

[9] Tsou, Tung-Hsun, "An Implementation of Controller Area Network Bus Analyzer Using Microblaze and Petalinux" (2013). Master's Theses. pp 174-174.

[10] Amit Kumar Bhadrawat1, Prof. Sourabh Sharma, "DMR Based CAN Bus Control System Implemented into FPGA", ISSN: 2349 – 4689, Volume-04, Number - 01, 2014 pp. 039-042.

[11] Bhavna Mahure And Rahul Tanwar, "Uart With Automatic Baud Rate Generator And Frequency Divider", ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, 2012, pp. 265-268.

[12] G. Bhanu Priya, B. Lakshman Murthy, P. Pragathi, "An Advanced Universal Asynchronous Receiver Transmitter (UART) Design & Implementation By Using VERILOG", ISSN: 2278 – 909X , 2014 pp. 796-800.

[13] Poonam R. Kedia, N.N.Mandaogade, Sneha R. Gade, "A Review Paper on Implementation of UART Controller with Automatic Baud Rate Generator using FPGA", ISSN: 2321-7782, 2014 pp. 280-283

[14] Karl Henrik Johansson,, Martin Torngren,Lars Nielsen, "Vehicle Applications of Controller Area Network", pp. 004-010.