A REAL TIME EMBEDDED FPGA INTERFACE FOR AN ETHERNET BASED LINE LASER

by

SUBRAHMANYA RAMASWAMY

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGNEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2012

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my committee chair, Professor Roger Walker, who has the attitude and the substance of a genius: he continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this thesis would not have been possible.

I would like to thank my committee members, Professor W. Alan Davis and Professor and Chairman Jonathan Bredow.

I would like to thank my parents who raised, supported, taught, loved and inspired me to earn things through hard work and sincerity. I would really like to thank my sisters and brother-in-law's whose support and love, has motivated me to accomplish my goals. I would like to thank my friends for their support and encouragement.

I would like to thank my lab mates for their valuable suggestions during the course of my Thesis. Special thanks to Ashwin Arikere for his guidance and support.

November 7, 2012

ABSTRACT

A REAL TIME EMBEDDED FPGA INTERFACE FOR AN

ETHERNET BASED LINE LASER

Subrahmanya Ramaswamy, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Roger Walker

Profile measurement systems developed at the Transportation Instrumentation and Embedded systems Lab (TIL) at the University of Texas at Arlington are used by The Texas Department of Transportation (TxDOT) to compute the profile of the road. The systems use certain basic sensors such as a laser, start sensor, distance encoder and accelerometer. The data samples from these sensors are acquired and stored for further processing. The data is processed in real-time using a road profiling algorithm to calculate the profile. This research effort investigates the feasibility of implementing the existing system using FPGA components and specifically for interfacing the laser to FPGA components. The laser outputs raw data samples in the form of Ethernet packets using the TCP-IP protocol. The laser operates using a client – server model. Hence, the FPGA design needs to include in the interface a Client-Server based system.

The objective of the thesis is to interface the laser to an Altera DE2-115 FPGA board using a client – server model. The research effort shows that it is feasible to interface the laser to the Altera FPGA board. It also proposes the interface and time complexity analysis of the real time system.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The first high speed inertial profiler measurement system was developed at General Motors Research Laboratory. Since this time, the technology has gone through several iterations of improvements including the use of wide footprint line lasers. Today's profiler uses the same general components, but it now uses different types of sensors and variations of the profiling algorithm. There are a number of manufactures of these instruments which are used to thought out the world to measure t surface quality of pavements. The Materials and Pavements Division of the Texas Department of Transportation (TxDOT) uses profiling technology developed at the Transportation Instrumentation and Embedded Lab (TIL) at the University of Texas at Arlington. The profiler is an instrument which produces a series of numbers related to pavement profile. The current existing system developed at TIL uses three sensors and are

- A laser for road-body displacement measurements,
- A distance encoder, for measuring distance traveled and synchronizing the computed profile to this distance, and
- An accelerometer for measuring vehicle body displacements.

An additional fourth sensor augmenting the primary sensors is an infrared start sensor.

1.1 Organization of the Thesis

The next chapter briefly describes the existing design of the road profiling system. I then discuss the possibility of many of the system components being implemented using the FPGA. Chapter 3 introduces all the basic components in more detail focusing mainly on the interface of the Roline Laser to the Altera FPGA DE2-115 board. Chapter 4 describes the design and implementation of a client server model for obtaining real-time Roline laser data

using Altera FPGA DE2-115 board. Chapter 5 shows the results of the research and verification that the Altera DE2-115 FPGA module can be used. Chapter 6 summarizes the work performed.

CHAPTER 2

CURRENT ROAD PROFILER SYSTEM AND FPGA BASED ROAD PROFILER SYSTEM

2.1 Current System

The current profiling system is described in this Chapter. As noted in Chapter 1, the profile instrument used by states and other various entries for measuring pavement profiles all consists of a similar type of sensors i.e. Accelerometers, laser, and a distance encoder. The accelerometer is a sensor that measures acceleration. A data processing algorithm converts the vertical acceleration measurements to an inertial height. The height of the ground relative to the reference is measured by a laser or other noncontact sensors, and the distance travelled is measured by a distance encoders. This system component is illustrated in Figure 2.1 and Figure 2.2 illustrates the components used in a measurement vehicle.

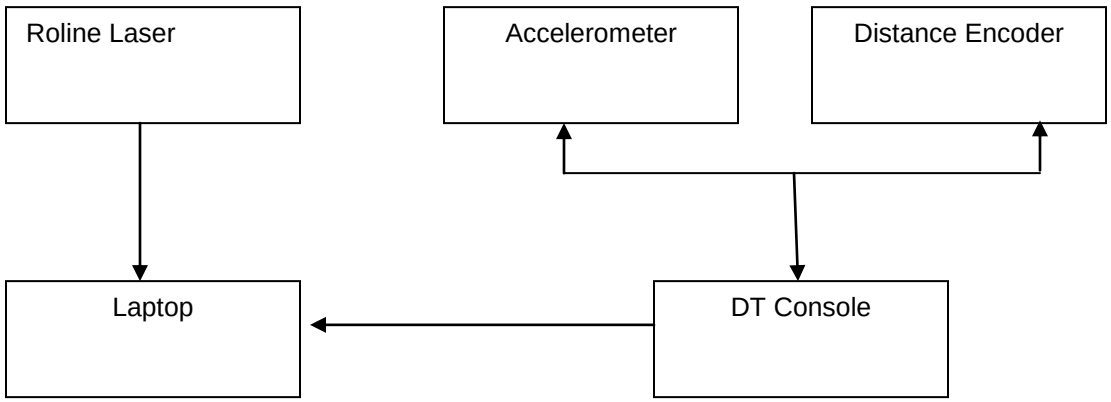

Fig 2.1 Current Road Profiling System

One major advantage of these inertial profilers is that the data collection can be done at highway speeds. The profile computed using an inertial profiler does not look like the true profile since it filters out the long wavelengths that can cause errors in the accelerometer measurement process. However, accurate and reliable band limited profile statistics can be obtained from the inertial profilers.

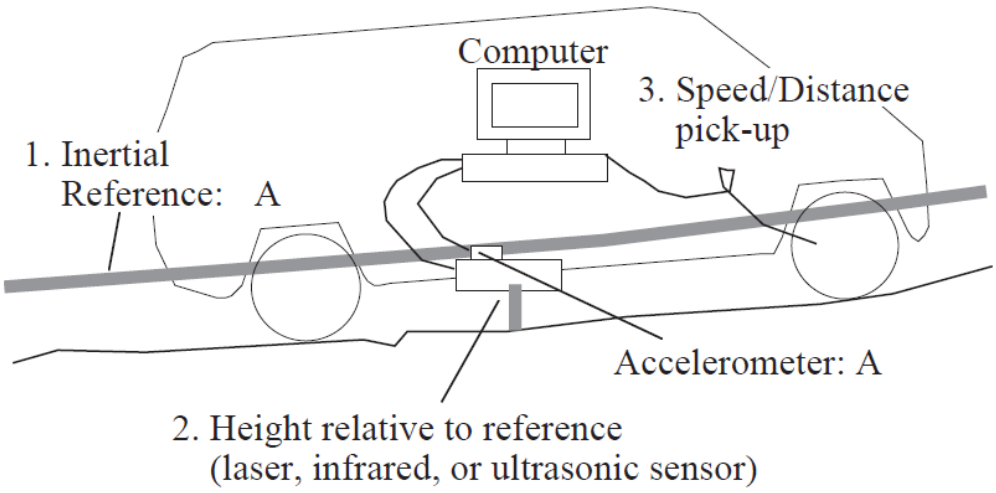

Fig 2.2 Overall Systems on a Car [13]

2.2 Redesign of the Current System

As noted this research focuses on the use of FPGA components in the current system. The real time system should be capable obtaining data from newer and different sensors as they become available. The Altera Nios II processor used is one of the examples of a soft core processor which could be easily reconfigured depending on the requirement. There are multiple existing Altera FPGA boards which use the Nios2 processor. For development purposes, the DE2-115 board has been chosen. The board was chosen since it is a development and educational board and is available at TIL. The DE2 nano board was also considered which can be used for industrial purposes. The Altera DE2-115 board comes with the Ethernet port, SD

Card reader, LEDs, Switches, IO ports etc. The profiler module considered in this research uses a Roline Laser, which sends data samples in the form of Ethernet packets. This combined with the different sensors in the profile module, can all be integrated using IO Ports of the DE2-115 board.

2.3 Laser

The transportation industry uses laser sensors in a variety of applications such as inspection of pavements for highways and airport runways. These applications include profiling of the surface to determine texture and wear, which have an impact on ride quality, traction and noise generation. Figure 2.3 depicts this laser in the UTA portable profiler module.

LMI Technologies Inc. is one of the leading providers of single point and line laser sensors in the transportation industry. Their current Roline sensors are the latest generation of the laser based 3D sensors for transportation applications, to reliably measure complex texture pavements including tined surfaces.

The Roline laser provides data line readings using the Ethernet packets. To receive these packets the system should be equipped with a 100 M (Megabits) Ethernet port to communicate with this sensor. The sensor requires a 48V power supply.

The Roline laser operates in two modes,

1. Free Mode.
2. Bridge mode.

The Free mode outputs the raw data containing the distance value for each of the 198 points in a single scan. The laser outputs a 3K Hz sync pulse which indicates the beginning of each scan and which is used to synchronize it with accelerometer readings.

The Bridge mode outputs a single value for each scan, which is the filtered average distance value of all the data points in the scan.

Fig 2.3 Road Profiler Module with Roline Laser[11]

2.4 Accelerometer

The Accelerometer used in the current UTA module is a Columbia Research Labs SA-107BHP single axis servo accelerometers. The analog output voltage is 1.876 V/g and an input voltage of +/- 15 V for operation. The range is ± 4g.

2.5 Distance Encoder

A Distance encoder is required to associate the distance along the road travelled to each laser-accelerometer measurement. The output of the distance encoder is of the form of analog pulses which are connected to an A/D converter to get a digital distance signal.

2.6 Approach and Focus

As noted, the Altera DE2-115 development and educational board is used for the investigation. It is used to interface the Roline laser, accelerometers, distance encoders and infrared detectors. The Roline laser outputs the data in the form of Ethernet packets; hence it can be interfaced to the real time educational and development board DE2-115 through the Ethernet port available on the board. The other sensors like the accelerometer, distance

encoder and infrared detectors can be interfaced to the DE2-115 board through the IO ports. The Figure 2.4 below shows an example of the complete design and the outline of the road profiling system using the DE2-115 educational and development board.

The DE2-115 educational and development board has only 2 MB (Megabits) of SRAM, two 64 MB (Megabits) of SDRAM and 8MB (Megabits) of flash memory. In order to build and develop the complete road profiling system it requires additional memory space. Hence, the design of the profiler with the FPGA DE2-115 board would require interfacing external memory. The DE2-115 has an SD Card socket; hence in this design we have used the SD Card. The profile output is displayed on to a VGA display Screen using the VGA port from the Altera FPGA board (see Figure 2.4).

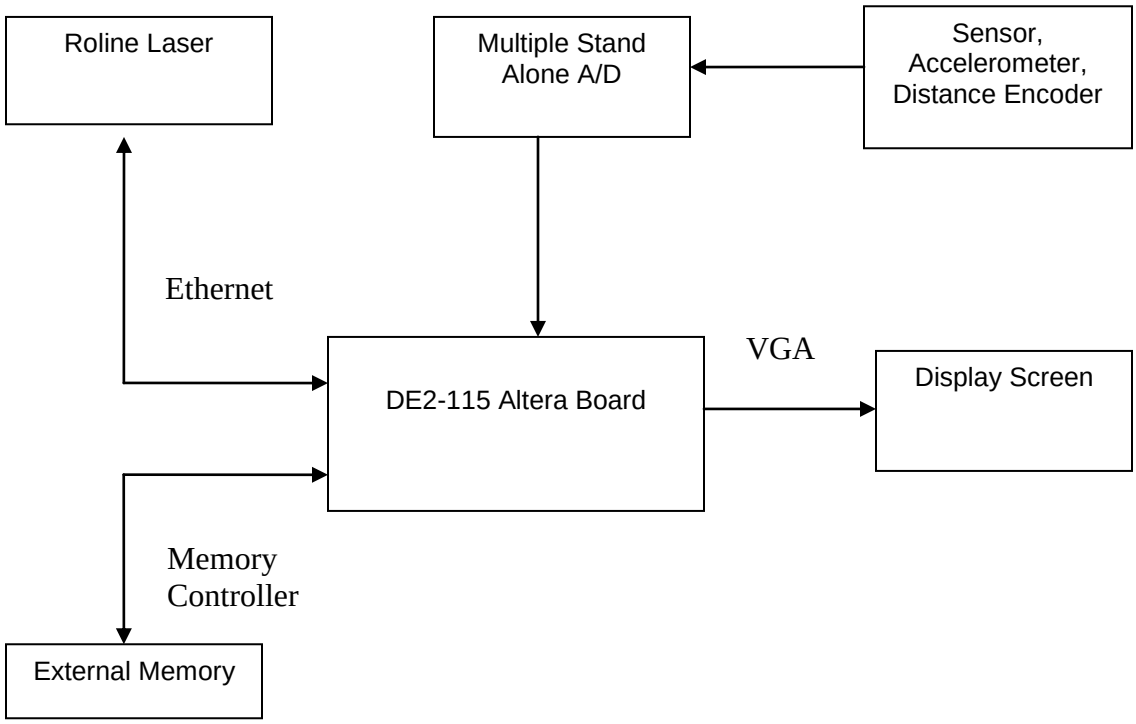

Fig 2.4 Redesign of the Road Profiler using DE2-115

The complete real- time FPGA based profiler system would involve the following design tasks.

- Interface Roline Laser to the DE2-115 board.
- Interface memory to store the calculated profile or interm results and implement Bridging algorithm.
- Interface the sensors, Accelerometer and distance calculator to the multiple stand alone Analog to Digital Converter (ADC).
- Interface the stand alone ADC to the DE2-115 board.
- Synchronize/Process the Ethernet packet from the laser and the ADC output.
- Convert the Walker state machine to the NIOS II processor standard.
- Interface a monitor through the VGA port of the DE2-115 board.
- Display the profile output through VGA.

This thesis report concentrates on developing the interface between the DE2-115 board with the Roline laser.

CHAPTER 3

BASIC FPGA BASED ROAD PROFILER SYSTEM COMPONENTS

As noted, the main objective of the research is to investigate the use of FPGA components for handling the raw data from the sensors and processing it to obtain the road profile. The FPGA provides the flexibility of adding a new peripherals and configuring the device easily for different requirements and sensors types. The raw data from different sensors are obtained and stored in real time. As the raw data is available, the data is processed and the road profile can be computed in real time.

This chapter presents the basic components required for designing and developing the road profiler using the Altera FPGA DE2-115.

3.1 DE2-115 System Diagram

The Figure 3.1 below depicts the Altera FPGA DE2-115 board and indicates the location of the connectors and key components. The Altera DE2-115 FPGA board supports many features using a wide variety of complex circuitry using the soft core processor which is embedded inside the Cyclone IV FPGA chip.

The following hardware is provided on the DE2-115 board.

- Altera Cyclone® IV 4CE115 FPGA device
- Altera Serial Configuration device – EPCS64
- USB Blaster (on board) for programming; both Joint Test Action Group (JTAG) and Active Serial (AS) programming modes are supported

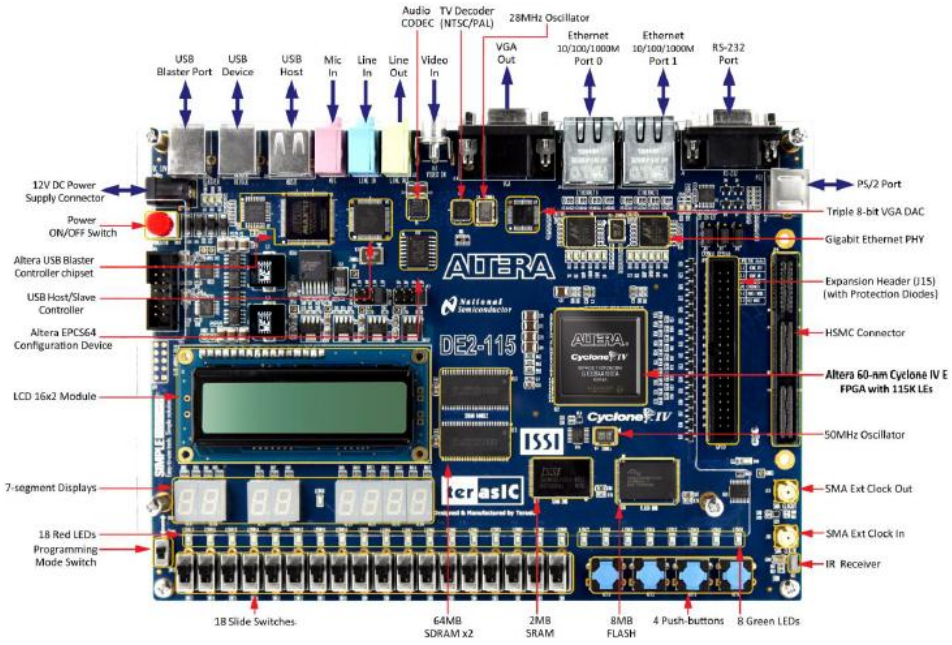

Fig 3.1 DE2-115 Diagram[3]

- 2MB SRAM
- Two 64MB SDRAM
- 8MB Flash memory
- SD Card socket
- 4 Push-buttons
- 18 Slide switches
- 18 Red user LEDs
- 9 Green user LEDs
- 50 MHz oscillator for clock sources

- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- 2 Gigabit Ethernet PHY with RJ45 connectors
- USB Host/Slave Controller with USB type A and type B connectors
- RS-232 transceiver and 9-pin connector
- IR Receiver
- 2 SMA connectors for external clock input/output
- One 40-pin Expansion Header with diode protection
- One High Speed Mezzanine Card (HSMC) connector
- 16x2 LCD module

3.2 Block Diagram of DE2-115 Board

All connections to the peripherals are provided by Altera for the flexibility to the user. The user has to configure the Cyclone IV FPGA device to use the peripherals. Figure 3.2 depicts the DE2-115 board.

For developing the complete real time road profiler the peripherals used include the Ethernet port, LCD for Display interim results, VGA to display the profile output, SD Card Memory to store the results and USB or Input Output pins to connect the Analog to Digital converter of the DE2-115 FGPA.

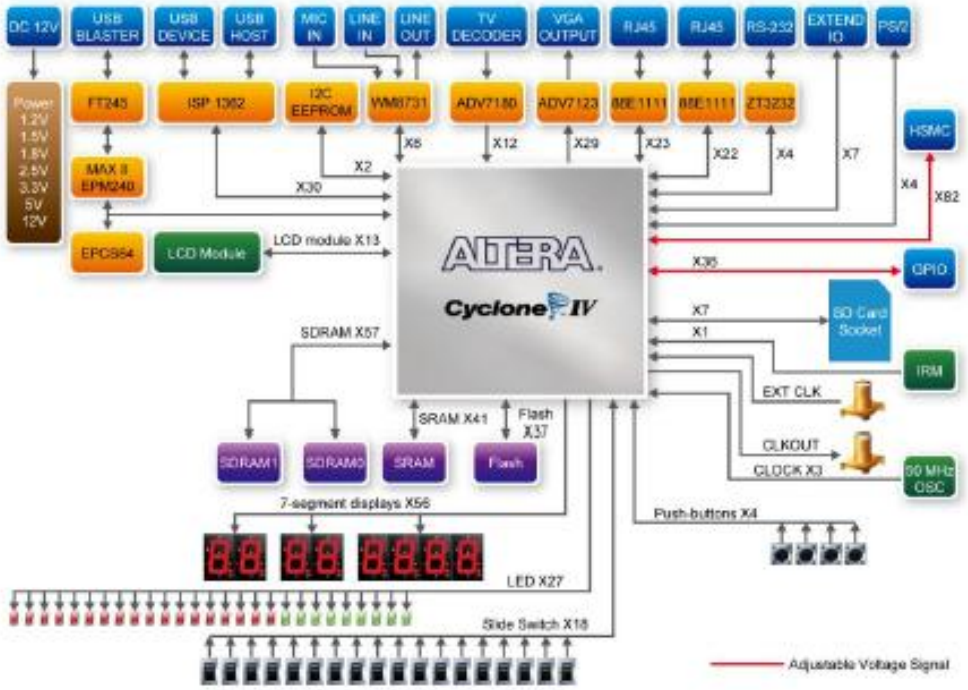

Fig 3.2 Block Diagram of DE2-115[3]

The Nios II soft core processor can be used to develop a variety of application using peripherals available on the DE2-115 board. Here we use the soft core processor to develop the road profiler. Fig 3.2 shows the basic Nios II processor architecture.

The Nios 2 processor connects to various chips on the DE2-115 board using the Avalon switch Fabric. Input/output interfaces are instantiated to provide connection to the I/O devices used in the system. A special JTAG UART interface is used to connect to the circuitry that provides a Universal Serial Bus (USB) link to the host computer to which the DE2-115 board is connected. This circuitry and the associated software are called the USB-Blaster. Another module, the JTAG Debug module, is provided to allow the host computer to communicate and control the Nios II processor.

The processor can be implemented in three different configurations:

1. Nios II/f is a "fast" version designed for superior performance. It has the widest scope of configuration options that can be used to optimize the processor for performance.
2. Nios II/s is a "standard" version that requires less resource in an FPGA device as a trade-off for reduced performance of the processor.
3. Nios II/e is an "economy" version which requires the least amount of FPGA resources, but also has the most limited set of user-configurable features.

The Nios II processor has Reduced Instruction Set Computer (RISC) architecture. Its arithmetic and logic operations are performed on operands in the general purpose registers. The word length of the Nios II processor is 32 bits. All registers are 32 bits long. Byte addresses in a 32-bit word can be assigned in either little-endian or big-endian style. The assignment style is one of the options that the user may select at configuration time. The Nios II architecture uses separate instruction and data buses, which is often referred to as the Harvard architecture.

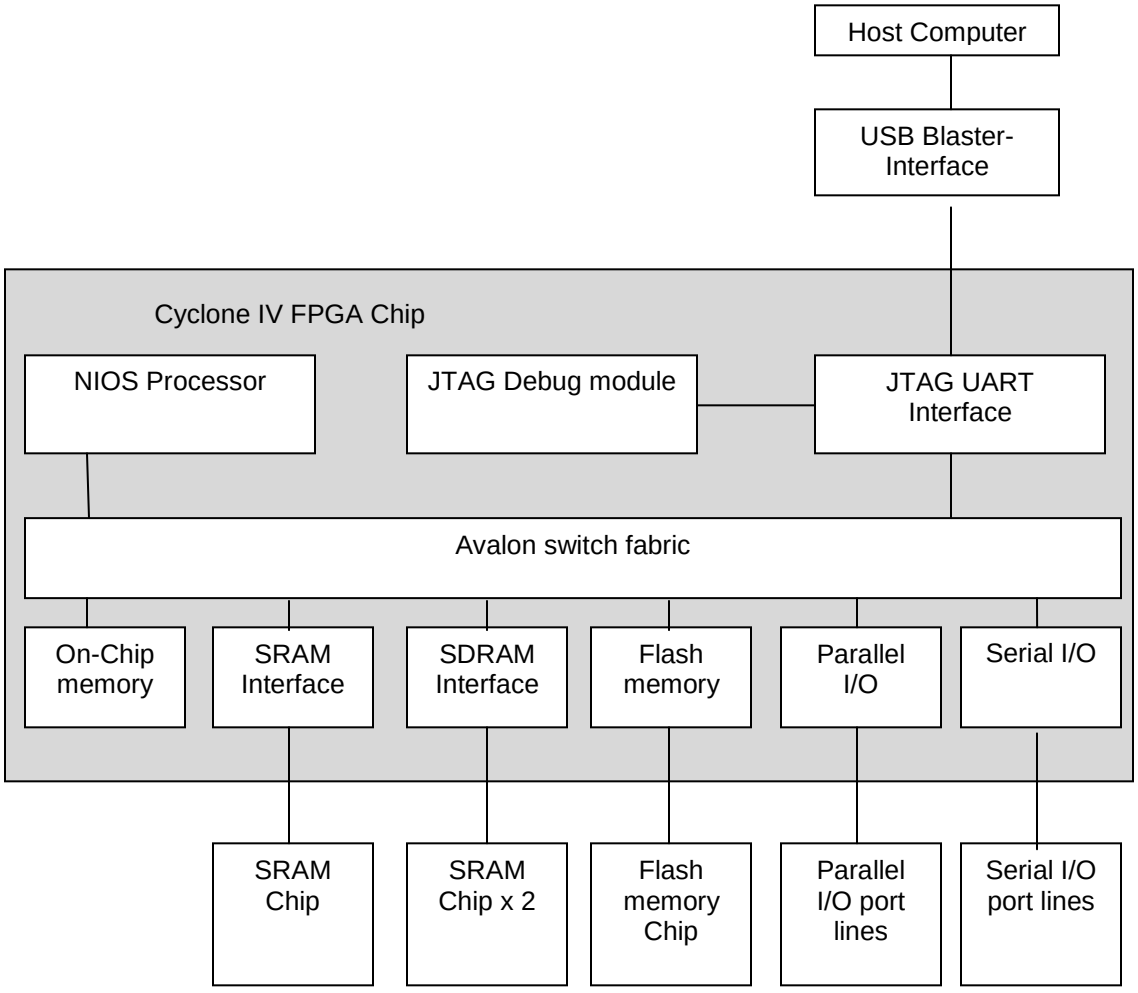

Fig 3.3 Nios II system implemented on the DE2-115 board[4]

3.3 Connections between FPGA and Ethernet

The DE2-115 board provides Ethernet support via two Marvell 88E1111 Ethernet PHY chips. The 88E1111 chip transceiver which supports GMII/MII/RGMII/TBI MAC interfaces. The Media Independent Interface (MII) was originally defined as a standard interface used to connect a Fast Ethernet (i.e., 100 M bit/s) Media Access Control (MAC) block to a Physical Layer (PHY) chip. The MII design has been extended to support reduced signals and increased speeds relative to the MII design.

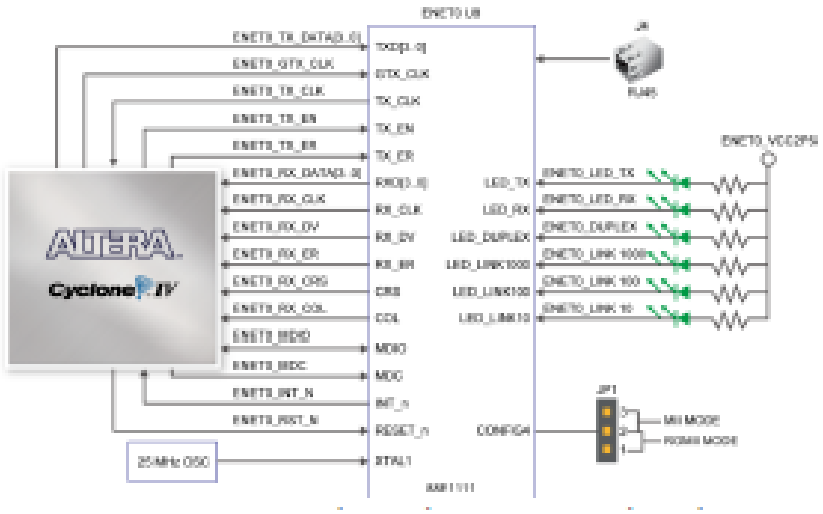

Fig 3.4 Connection between FPGA and Ethernet [2]

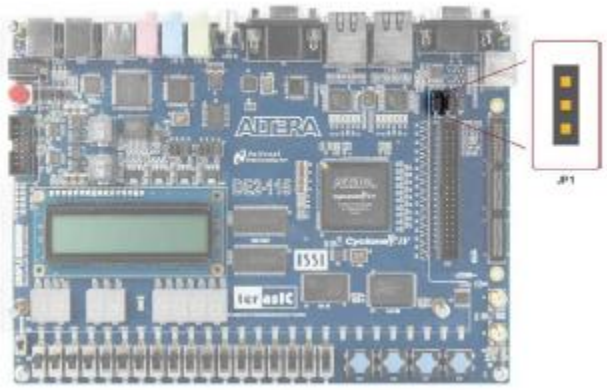

Fig 3.5 Working Mode setup header for Ethernet [2]

Table 3.1 Working Mode setup header for Ethernet

JP1 Jumper Settings	ENET0 PHY Working Mode
Short Pin 1 and Pin 2	RGMII Mode
Short Pin 2 and Pin 3	MII Mode

Table 3.1 shows the jumper settings on the DE2-115 board to configure the Ethernet chip to operate any one of the RGMII or MII Mode.

3.4 88E1111 Integrated 10/100/1000 Ultra Gigabit Ethernet Transceiver

Below are the features of the Marvell 88E1111 Integrated Ethernet chip used in the Altera FPGA DE2-115 board which is configured to accept the incoming packets on the Ethernet port of the DE2-115.

The Marvell chip provides the 802.3 compliant, RGMII mode of operation. It has the feature of generating the packets for transmission. The chip is capable of checking the polarity and auto negotiate with the peripheral. The software programmable LED's gives the ability to identify the incoming packets.

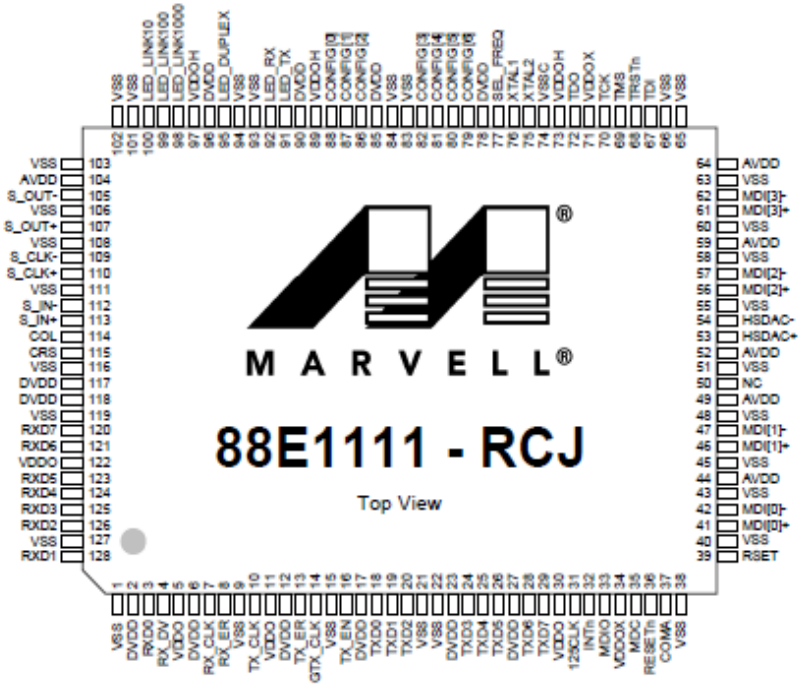

Fig 3.6 88E1111 Integrated 10/100/1000 Ultra Gigabit Ethernet Transceiver [2]

3.5 Roline Laser

The Roline 1145 Laser is used by the road profiler embedded system. The +48VDC power supply, used to power the sensor, must be isolated from AC ground. The Roline laser 1145 provides distance samples of the raw data in the form of Ethernet packets using

Transmission Control Protocol/ Internet Protocol Suite (TCP/IP) or the User Datagram Protocol (UDP). In this thesis, TCP/IP protocol is used to capture the raw data from the laser. The TCP/IP protocol is chosen since it is a reliable and connection orientated.

As noted in Chapter 1, the laser operates in two different modes i.e. Free Mode and Bridge Mode. In Free Mode, the laser outputs 3000 samples per each scan of 80 points where as in Bridge Mode, the laser outputs one distance sample for each scan. The client to which the Laser is interfaced should have an Ethernet adaptor that can be configured for a static IP address and supports 1000 Mb/s operation. The default IP address of the laser is 90.x.x.x where the last two fields are dependent on the sensor's serial number.

3.6 Measurement Principle

The RoLine 1145 sensors function on the principle of structured light triangulation. A semiconductor laser with special optics projects fan of light onto the target. A digital camera mounted at an angle to the laser plane acquires images of the light pattern created on the target. These images contain the basic information needed to compute distances to the target.

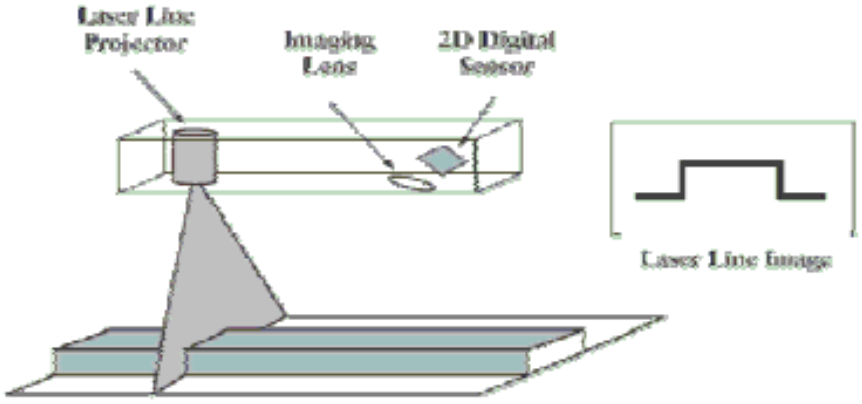

Fig 3.7 Measurement Principle [9]

The laser operates as a server until the Laser is turned ON. Once the Laser light is turned ON, it behaves as a client and tries to establish a connection to the server. The laser operation commands, ON, OFF, Halt, Resume, Set Mode i.e. Free Mode or Bridge Mode etc , which are performed by the LMI's Fire Sync Protocol.

3.7 FireSync Host Protocol

The client should be capable of communicating to the laser via the FireSync Protocol. Each command and reply begins with a Header. The format for the Command Header and Reply Header is given in Table 3.2 and Table 3.3.

Table 3.2 Command Header

Command Header

Each command begins with a header, defined as:

Field	Type	Description
length	64s	Message length in bytes(including header size)
command	64s	Command Identifier

Table 3.3 Reply Header

Reply Header

Similar to command, each reply begins with a header, defined as

Field	Type	Description
length	64s	Message length in bytes(including header size)
command	64s	Command Identifier
status	64s	Status Code

The laser system has four states. The current state of the system determines what commands can be issued to the server. Below are the system states.

Table 3.4 System States

State	Value	Description
Idle	1	The system has started up but has detected errors that prevent the system from being able to run.
Ready	2	The system is ready to run
Running	3	The system is in full- operation mode. The data transfer channel is connected and data is sent when appropriate.
StandBy	4	The system is paused. The data transfer channel remains connected and the system is ready to resume immediately.

3.8 System Operation Commands

The following is a table of the core system commands. These commands are used in typical sensor operations.

Command	Identifier
Start System	0x1000
Stop System	0x1001
Pause System	0x1002
Resume System	0x1003
Set Operation Mode	0x1004
Get Operation Mode	0x1005
Write File	0x1006
Write Volatile File	0x1017
Read File	0x1007
Delete File	0x1008
Enumerate Sensors	0x1009
Get System Current Time	0x100A
Data Channel Enable	0x100B
Data Channel Disable	0x100C
System Run Command	0x100D
System Ping Command	0x100E
Start Event Channel	0x1010
Stop Event Channel	0x1011
Get System Start Delay	0x1012
Backup Storage	0x1013
Restore Storage	0x1014
Get Server Info	0x1015
Get Sensor Info	0x1016

Fig 3.8 System Operation Commands [8]

3.9 Roline Power Sync Board

Roline Laser would require power supply of +48V for operation. Hence, a separate board has been designed and developed to provide the required voltage for the Roline Laser. The board requires an input voltage of +12V and uses a DC to DC Converter to supply the 48V. The board also provides the appropriate interface for the sync signal used to indicate the start line of each line scan.

CHAPTER 4

DESIGN OF FPGA BASED ROAD PROFILER

The design of FPGA based road profiler requires the design and implementation of Hardware and Software components. The Hardware design requires instantiating the NIOS 2 processor. The Software would involve designing the Ethernet stack and controlling all the peripherals using the MicroC/OS II RTOS.

4.1 Hardware Design

A design has been developed using an Altera FPGA in order to interface to the Roline Laser. The system consists of a soft core processor called NIOS II. The Ethernet sockets are interfaced to the Nichestack on MicroC/OS II RTOS using the NIOS II the soft core processor.

The Hardware design of the Nichestack TCP/IP Stack on the DE2-115 board using the NIOS II requires the System On Programming Chip (SOPC) builder from the Quartus II. The System On Programming Chip (SOPC) is a custom made tool from Altera for developing the system. The SOPC system for this design contains NIOS II processor, On-Chip memory, JTAG UART, timer, Triple Speed Ethernet, Clock, Red and Green LED, Keys, SDRAM, LCD and all the other peripherals available on the DE2-115.

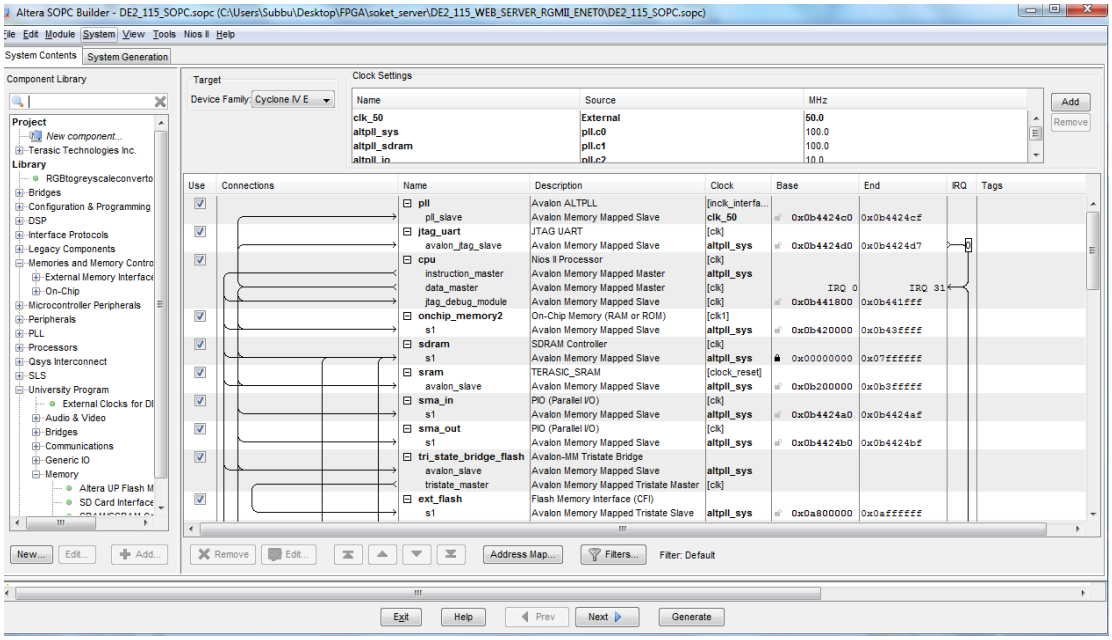

Fig 4.1 Hardware Design

Once the all the peripherals are added using the SOPC Builder, the system is compiled and loaded to the Altera FPGA DE2-115 board. The loaded FPGA board now has the NIOS 2 running on it. Using NIOS II IDE we then create the TCP/IP Nichestack on NIOS II.

4.2 Software Design

This section provides an overview of the software design of the road profiler system. Fig 4.2. Below diagram shows the different layers of the system. Each layer encapsulates the specific implementation details of that layer, abstracting the data for the next outer layer. Each layer is explained below.

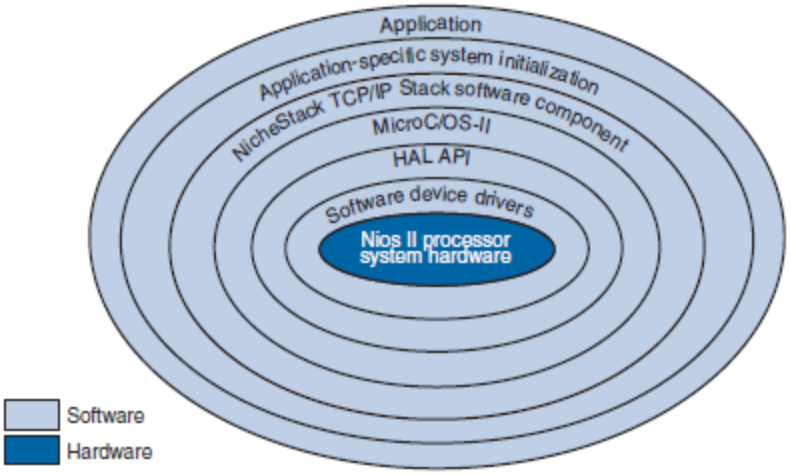

Fig 4.2 System Design [7]

1. NIOS II processor system hardware: The core consists of the Nios II soft core processor and hardware peripherals implemented in the FPGA using the SOPC builder of the Quartus II.
2. Software Device Drivers: The software device driver's layer contains the software functions that manipulate the Ethernet and hardware peripherals. These drivers know the physical details of the peripherals devices.
3. HAL API: The Altera Hardware Abstraction Layer (HAL) application programming interface provides a standardized interface to the software device drivers.
4. MicroC/OS II: The MicroC/OS II RTOS layer provides multitasking and intertask communication services to the Nichestack TCP/IP Networking stack and NIOS II client server.
5. NicheStack TCP/IP Stack software component: The Nichestack TCP/IP Stack software component layer provides networking services to the application layer and the application specific system initialization layer via the sockets API.
6. Application specific system initialization: The application specific system initialization layer includes the MicroC/OS II and Nichestack TCP/IP Stack software component

initialization functions invoked from main(), as well as creates all application tasks, and all the semaphores, queue, and the event flags RTOS inter-task communication resourses.

7. Application: The outermost application layer contains the NIOS II Client Server task.

4.3 NicheStack TCP/IP Stack Initialization

The Nichestack TCP/IP Stack must be initialized from the NIOS II Client Server Socket application code by calling the following Nichestack functions.

1. alt_iniche_init(), called from SSSInitialTask() in iniche_init.c
2. netmain(), called from SSSInitialTask() in iniche_init.c

Three NicheStack functions, get_mac_addr(), get_board_mac_addr() and get_ip_addr() are provided. These functions are called from the iniche_init.c file.

Tasks can be created and managed as the system has incorporated the real time operating system i.e. MicroC/OS II RTOS. An initialization task named SSSInitialTask() calls the alt_inciche_init() and net_main() initialization functions in the proper sequence, and then waits until the NicheStack TCP/IP Stack becomes fully operational before creating the application level task.

4.4 Implementation

A static IP is assigned to the Altera DE2-115 FPGA Board. The static IP "90.0.0.1 " with mask address of 255.0.0.0 is used. The IP address of the laser is obtained from the serial number provided on the Laser. Each Laser has been assigned a unique address and IP address.

The application's main() function (located in iniche_init.c) performs the following actions:

1. Calls OSTimeSet()
2. Calls OSTaskCreateExt which in turn calls SSSInitialTask()

3. Calls alt_uCOSIIErrorHandler()
4. Calls OSStart() to begin multithreading

The Micrium's MicroC/OS-II suggests using a single task to initialize the rest of the application. This technique ensures that stack checking initializes enabled features correctly. The SSSInitialTask() task (located in iniche_init.c) initializes the NicheStack TCP/IP Stack software, initializes the operating system data structures, and starts any user-defined networking tasks and regular tasks. The SSSInitialTask()task performs the following specific actions:

1. Calls alt_iniche_init() to perform pre-initialization of the NicheStack Networking Stack
2. Calls netmain() to initialize and start the NicheStack Networking Stack

Two user defined tasks are created one the system behaving as the Server and other as the system acting as a Client. The Server task is assigned a priority of 5 which is less than the priority of the Client task which is assigned 4. Both the tasks have their own stack and are created using the function TK_NEWTASK (). The Scheduler of the MicroC/OS II RTOS is priority based scheduler and it schedules the tasks based on the priority of the tasks. This RTOS supports 64 tasks of which two of them are used for Ideal and start tasks. The other two tasks i.e. Server and Client are created and used.

4.4.1 Server Task

On Start command to the Laser, the laser starts and tries to connect to the server by switching itself to client from server. The server IP address is provided in the start command issued from the client using the Firesync Protocol. On start of the system, the Altera DE2-115 board having the soft core processor NIOS2 and the MicroC/OS II RTOS creates the server task using the create task function. The created server task would start when the OS starts the RTOS. A socket is created to serve the TCP/IP connection. Below function is used to create the socket.

```
Socket( AF_INET, SOCK_STREAM,0)
```

Below is the code used to create the packet.

```
fd_listen = socket(AF_INET, SOCK_STREAM, 0)
```

The value of AF_INET is “2” which signifies the TCP/IP connection. The SOCK_STREAM signifies the stream socket format. On successful creation of the packet, associates a socket with a socket address structure, i.e. a specified local port number and IP address. When a socket is created it only gives the protocol family and not the address or the port. The association with the address is made using the bind(). Below is the syntax for binding a packet.

Bind(Socket_name, (Struct sockadd *) &addr,sizeof(addr));

Bind takes three arguments and are as below

- a descriptor representing the socket to perform the bind on.
- a pointer to a sockaddr structure representing the address to bind to.
- field specifying the size of the sockaddr structure.

Below is the code to bind the socket to a particular port.

```
struct sockaddr_in
{
  sa_family_t      sin_family;      /* Address family       */
  in_port_t  sin_port;  /* Port number                */
  struct in_addr sin_addr;    /* Internet address          */

  /* Pad to size of `struct sockaddr'. */
  unsigned char   __pad[__SOCK_SIZE__ - sizeof(short int)
                  - sizeof(unsigned short int) - sizeof(struct in_addr)];
};
addr.sin_family = AF_INET;
addr.sin_port = htons(port);
addr.sin_addr.s_addr = INADDR_ANY;
bind(fd_listen,(struct sockaddr *)&addr,sizeof(addr)
```

After the socket is associated with the protocol and an address, listen () prepares it for incoming connections. The prototype of the listen() is as below

Listen (socket_name, 1);

The snapshot of the listen command used in the code is as below:

```
(listen(fd_listen,1)
```

The listen socket is a socket which is waiting for incoming connections. This call to listen will block (i.e. not return) until someone tries to connect to this port. When the application is listening on a particular port, it is notified by an event to accept the connection on request. The accept() function creates a new socket for each connection and removes the connection from the listen queue. The prototype for the accept () function is as below

Accept(socket_name, address, len);

It takes the following arguments:

- the descriptor of the listening socket that has the connection queued.
- a pointer to a sockaddr structure to receive the client's address information.
- a pointer to a socklen_t location that specifies the size of the client address structure passed to accept().

Below is the accept code which is called from the function sss_handle_accept, once there is any connection request.

```
socket=accept(listen_socket,(struct sockaddr*)&incoming_addr,&len))
```

Once the connection is accepted from the client i.e. the Laser, a data channel is created and data is transferred from the laser to the On_chip memory of Altera board having the Nios 2 and the MicroC/OS II RTOS. The data is collected by calling the function sss_handle_receive (). The connection is terminated when the Laser is turned OFF. The connection can be terminated using the close() command which causes the system to release resources allocated to a socket. In case of TCP/IP, the connection is terminated. The prototype of the close command is as below

Close(Socket_name);

Below is the flowchart of the server Task.

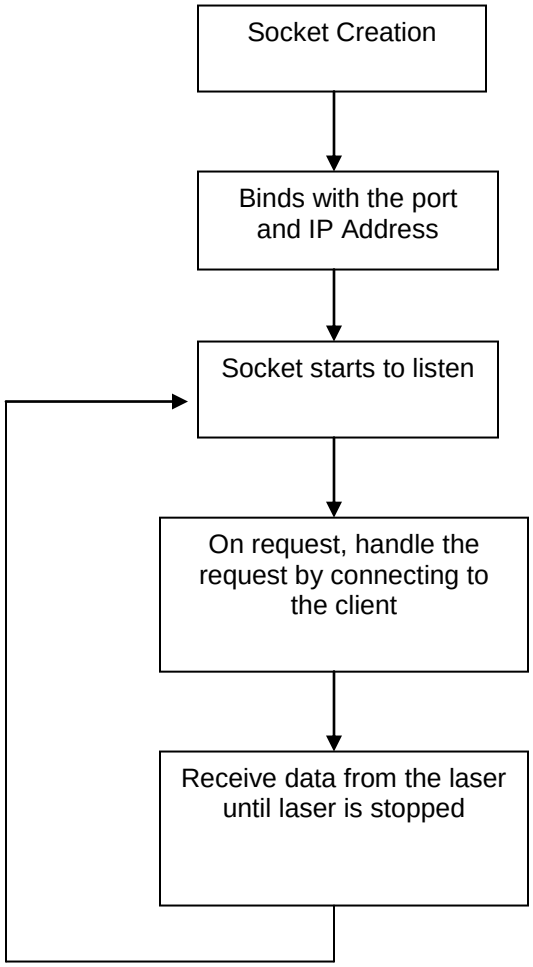

Fig 4.3 Flow Chart of Server Task

4.4.2 Client Task

The Client task is created with a priority of 4 which is of higher priority than the server task. The client task tries to connect to the server on “Connect “ command. The server IP is given as an argument to the connect command. The client task is created using the create task function. The client program would start and waits in the infinite while loop which has many

different FireSync protocol command options. There are many options available for issuing the commands using the Switches of the DE2-115 board. Below are some of the switch options with their corresponding command options.

Table 4.1 Switch Conditions

Switch Conditions	Output
1	Disconnect
2	Connect
4	Start
8	Stop
16	Set Mode
17	Free Mode
18	Bridge Mode
32	Get Mode

When switch conditions is at binary two, then a client program tries to connect to the specified server. On connect command by the switch condition, a socket is created using the function socket (). The prototype for the socket function is

Socket(AF_INET, SOCK_STREAM,0)

The Socket creation for the client task is created as below using the above prototype of creating the socket. The 90.0.13.220 is the IP address of the Roline Laser which is obtained from the manufacturer.

```
clientService.sin_family = PF_INET;
clientService.sin_addr.s_addr = inet_addr("90.0.13.220");
clientService.sin_port = htons(SSS_PORT);
m_socket = socket(PF_INET, SOCK_STREAM, 0);  //IPPROTO_IP
```

Once the socket is created, the program tries to connect to the specified IP address using the connect ().The connect() system call connects a socket, identified by its file descriptor, to a remote host specified by that host's address in the argument list. Certain types of sockets

are connectionless, most commonly user datagram protocol sockets. For these sockets, connect takes on a special meaning: the default target for sending and receiving data gets set to the given address, allowing the use of functions such as send() and recv() on connectionless sockets. The prototype of the connect is as below

Connect(int sockfd,const struct sockaddr *serv_addr,socklen_t addrlen);

Below is the snapshot of the code used in the client task.

```
connect(m_socket,(const struct  sockaddr *)&clientService,
sizeof(clientService));
```

Connecting to the server with the use of connect(), passing a sockaddr_in structure with the sin_family set to AF_INET, sin_port set to the port the endpoint is listening (in network byte order), and sin_addr set to the IP address of the listening server (also in network byte order.)

Once the connection is established with the server, the mode of the laser can be configured by having 16 as the switch condition. The Laser can operate in free mode and Bridge mode. The laser operates in free mode if the switch condition is 17 and in bridge mode if the switch condition on the DE2-115 board is 18. The laser mode is selected using the different combination of switch conditions as explained above. Later, the laser can be turned “ON”. The reading and writing to the packets and sending them to the server can be done using the below two main commands i.e. Read and Write.

The read and write command prototype is

Read(Packet_name,&buffer, sizeof(buffer));

Write(Packet_name,&command, sizeof(command));

The read and write command are used extensively used in the client task. The switch conditions are read constantly and specific task is called to perform the particular task as shown in Table 4.1.

Below is the example of read and write commands as used in the code.

```
write(m_socket,&cmd, (k32u)cmd.header.length));
read(m_socket,&reply,sizeof(reply));
```

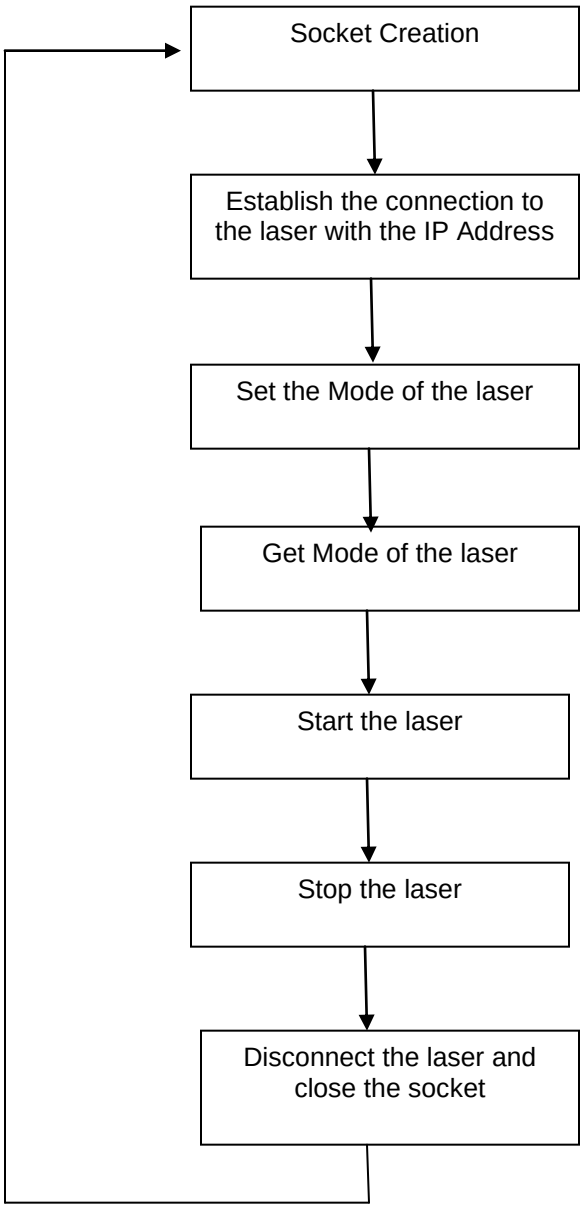

Fig 4.4 Flow Chart of Client Task

The buffer is composed of character arrays where all the data read from the packet are stored. The packet name is specified in the instruction. The command is the data that needs to

be transmitted to the server from the client. The connection can be terminated using the close() command. This causes the system to release resources allocated to a socket. In case of TCP, the connection is terminated.

4.5 Fire Sync Protocol

As noted, the Fire Sync protocol is used by the laser to communicate with the peripherals. The laser can be turned “ON”, turned “OFF”, halted, resumed and can set the different modes of the laser by using the Fire Sync protocol. This protocol is an application level protocol which is implemented on the TCP/IP protocol of the transport layer. Below is the format of the FireSync result.

Field		Type	Description
size		64s	Total size of message
id		64s	Message identifier
attributeCount		64s	Number of attributes
itemCount		64s	Number of items
attributes		64s	first attribute
		...	
		64s	n^{th} attribute (n specified by attributeCount)
itemDescriptors	length*0*	64s	First-dimension length of first item
	length*1*	64s	Second-dimension length of first item
	length*2*	64s	Third-dimension length of first item
	type	64s	Item type of first item
	...		
	length*0*	64s	First-dimension length of n^{th} item
	length*1*	64s	Second-dimension length of n^{th} item
	length*2*	64s	Third-dimension length of n^{th} item
	type	64s	Item type of n^{th} item
itemData	data*0*	block	Content of first item
	...		
	data*n*	block	Content of of n^{th} item

Fig 4.5 Fire Sync Result Format [8]

Each item descriptor describes the block of data at the corresponding index in the data section. The raw data from the laser is read and stored in an array. The length fields of the array contain the lengths in the three dimensions. Only length*0* is valid for a one dimensional array, and length*0* and length*1* for a two dimensional array. The lengths are followed by the item type, which is an identifier for a primitive type, and is defined by the following table.

Type Identifier	Type Description
1	8-bit unsigned integer
2	8-bit signed integer
3	16-bit unsigned integer
4	16-bit signed integer
5	32-bit unsigned integer
6	32-bit signed integer
7	64-bit unsigned integer
8	64-bit signed integer
9	byte
10	single byte character
11	64-bit floating point number

Fig 4.6 Data Format [8]

The Laser operates in two different modes i.e. a free mode where the laser outputs the 80 samples for each scan and a Bridge mode were the Roline captures the raw data and computes a single value for each scan using the LMI Bridging algorithm.

The Free mode message format is defined in Figure 4.7 and 4.8. The invalid profile points are represented by the 16 bit value, -32768 (or 0x8000).

Free Profile Message

Field	Type	Description
messageSize	64s	Total size of message (bytes)
messageId	64s	Type of message (1)
reserved[2]	64s	Reserved for internal use
deviceId	64s	Sensor serial number
endianness	64s	Endianness for the profile data*. Little-endian = 0, big-endian = 1
reserved[4]	64s	Reserved for internal use
count	64s	Count of profile arrays grouped in message
width	64s	Count of range points per profile array
channel		If channel=2, then the profile arrays contain both z- and x-values. If channel=1, then the profile arrays contain only the z-values
reserve[1]	64s	Reserved for internal use
attributes[count][3]	64s	Profile attributes (defined below)
points[count][width][channel]	16s	Profile arrays consisting of (z,x) or only (z), depending on 'channel' (in 0.01mm). Can be little-endian or big-endian

Fig 4.7 Free Profile Message [8]

Free Profile Attribute (Full Profile)

Field	Type	Description
timestamp	64s	Capture time (in microseconds)
syncIndex	64s	Counter of the number of 3kHz clock periods
trackingMode	64s	Indicates whether the range data is obtained from a tracking window (0) or a search window (1).

Fig 4.8 Free Profile Attribute. [8]

The Bridge mode message format is defined in Figure 4.9 and 4.10. The invalid profile points are represented by the 16 bit value, -32768 (or 0x8000).

Bridge Profile Attribute

Field	Type	Description
messageSize	64s	Total size of message (bytes)
messageId	64s	Type of message (2)
reserved[2]	64s	Reserved for internal use
deviceId	64s	Sensor serial number
count	64s	Count of profile arrays grouped in message
reserved[7]	64s	Reserved for internal use
attributes[count][3]	64s	Profile attributes (defined below)
bridgeValue[count]	16s	Bridged value arrays (in 0.01mm)

Fig 4.9 Bridge Profile Message. [8]

Bridge Profile Attribute

Field	Type	Description
timestamp	64s	Capture time (in microseconds)
syncIndex	64s	Counter of the number of 3kHz clock periods.
trackingMode	64s	Indicates whether the range data is obtained from a tracking window (0) or a search window (1).

Fig 4.10 Bridge Profile Attribute.[8]

4.6 Roline System Structure

The processor in the Roline laser 1145 performs both server and client functions interfacing to the FPGA NIOS II processor. The server task is of higher priority compared to the

client Task. Hence, by default on start of the laser system, it operates as a server. The static unique IP is computed and assigned using the unique serial number present on the Laser module. The client can connect to the Laser using this IP. The laser uses the TCP/IP protocol to communicate to the peripherals. The application layer protocol used by the laser is called FireSync protocol. The peripherals can issue commands using the FireSync protocol to turn “ON”, Turn “OFF”, and other operation specified on the laser. On start command request by the peripherals through the Firesync protocol would make the laser to start as a client and tries to connect to the server. The server IP is provided by the peripheral during the start command. If the command to turn “OFF” the laser is issued by the peripheral, then the laser stops it’s client task by terminating the connection established to the server and continues its server task. Below figure 3.5 shows the overview of the Roline System software structure.

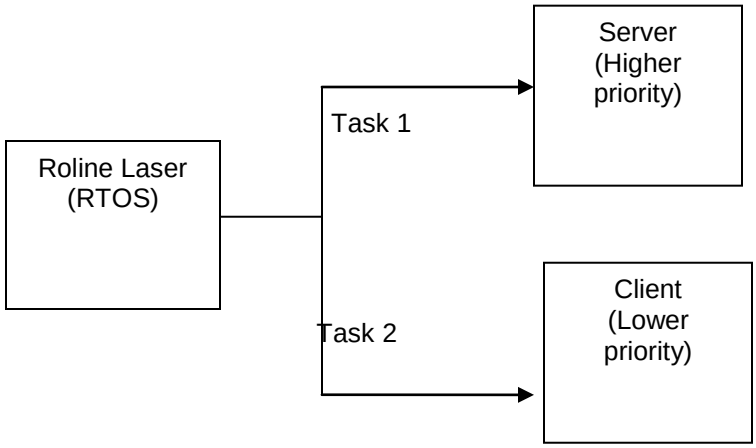

Fig 4.11 Roline laser System Structure

4.7 Altera FPGA System Structure

The Roline laser is a multitasking system having the two main tasks i.e. server and a client. Hence, there is a need to develop a system which is capable of handling both the server and client tasks. The system is designed exactly opposite of the Roline system structure. As previously noted, the Altera FPGA System is designed with the MicroC/OS II RTOS on the Nios

II soft core processor. There exists two task one as a server and other as a client. The client task is of higher priority compared to the server task. The scheduler of MicroC/OS II RTOS is a priority based scheduler. Hence, by default the client would win the priority and run. The client is designed to wait for the switch conditions and issue the commands based on conditions of the switches of the Altera DE2-115 board. The server task is started and listens on the port were the laser tries to connect to the Altera board as a client. As indicated, the complete client/server system is designed and developed based on the TCP/IP protocol. Figure 4.12 shows the overview of the Altera FPGA System software structure.

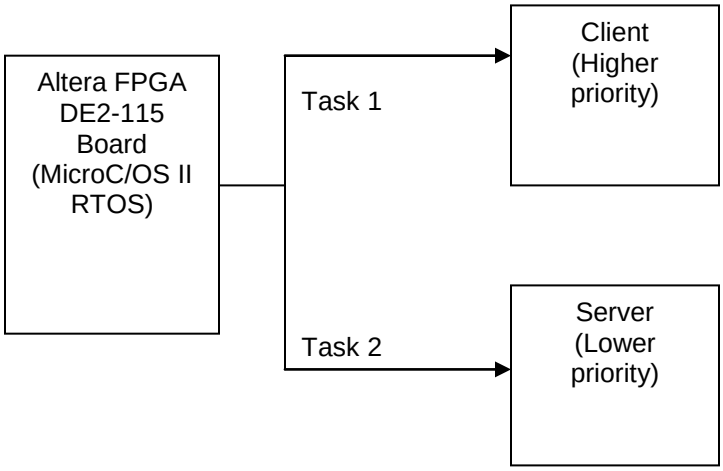

Fig 4.12 Altera FPGA System

CHAPTER 5

RESULTS

This chapter gives the results conducted on the FPGA based Laser system interface. For this chapter, a texture sample using the PC C based data collection program is compared with the FPGA based system. The laser is interfaced to the Altera DE2-115 FPGA Board. As discussed, the laser operates in two modes i.e. Free and a Bridge mode. In the Free mode the laser outputs 80 points for every scan and in the Bridge mode it will output one value for each scan the laser makes.

The laser is pointed at a texture sample which is available in the TIL lab as shown in Figure 5.1. This sample has been used for the testing of both systems i.e. the PC C based data collection program and the FPGA based system.

Fig 5.1 Texture Sample used in TIL for Testing

Table 5.1 PC C Based System Free mode samples

PC C based program	
Current System samples	Scan Number
-32768	109
-32768	109
-32768	109
-32768	109
8748	109
8681	109
8703	109
8703	109
8629	109
8615	109
8608	109
8599	109
8613	109
8760	109
8766	109
8749	109
8683	109
8679	109
8752	109

Table 5.1 shows the results obtained by laser being operated in free mode using the PC C based program. The PC C based program has been developed and tested at the TIL Lab at The University of Texas at Arlington. The program takes the input from the laser and the DT Console board. Figure 5.2 gives the snapshot of the command prompt of the PC C based program.

Table 5.2 shows the sample of results obtained by the laser is operated in free mode using the DE2-115 Altera FPGA Board.

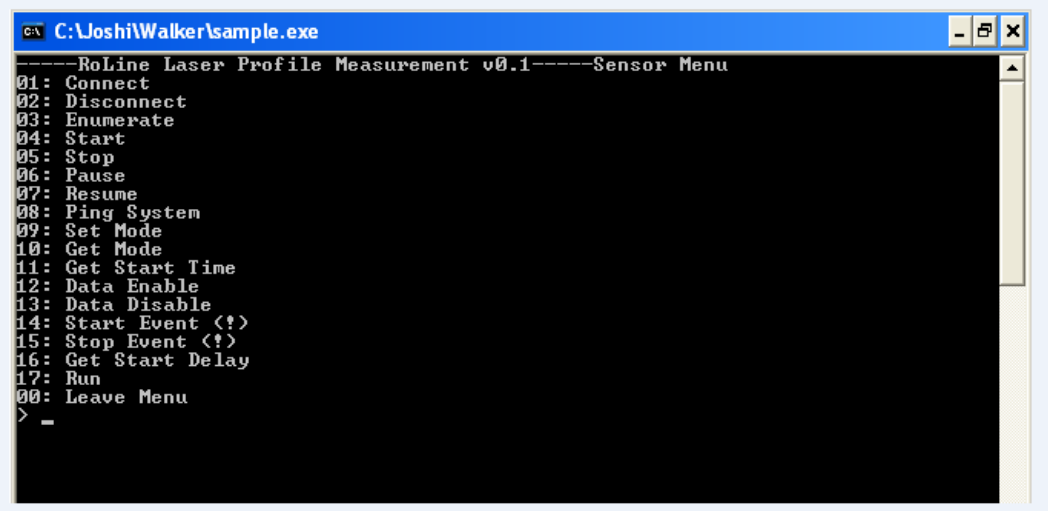

Fig 5.2 Snapshot of the command prompt of the PC C based program.

Table 5.2 DE2-115 Free mode Samples

DE2-115	
DE2 samples	Scan Number
8803	109
8714	109
-32768	109
-32768	109
8739	109
8715	109
8733	109
8690	109
8652	109
8622	109
8617	109
8570	109
8629	109
8714	109
8731	109
8788	109
8684	109
8689	109
8758	109

As seen in the graph Figure 5.3, the raw data obtained from both the DE2-115 and the current systems have the same range of values. The x-axis indicated the sample number and the y-axis indicates the raw data from the laser i.e. the distance between the tip of the laser to the pavement profile.The difference in the values obtained from the two methods can be attributed to the movement of the laser while testing. The value -32,000 indicates that it is invalid and therefore neglected. Hence, it can be shown that the Laser has been interfaced successfully to the Altera FPGA based road profiler.

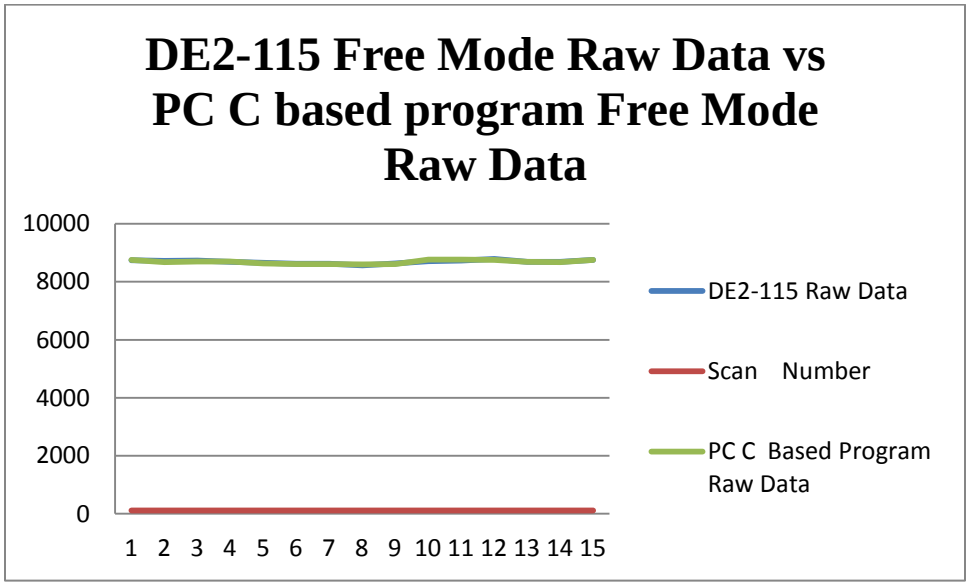

Fig 5.3 DE2-115 Free Mode samples vs Existing System Free Mode samples

The Bridge Mode data samples are obtained from the laser. Table 5.3 shows the bridge mode data samples collected by using both the Altera based FPGA i.e. using DE2-115 board and the existing system.

Table 5.3 DE2-115 and PC C Based System Bridge Mode samples

DE2-115 Altera FPGA Board		PC C based program System	
DE2 raw data	Scan Number	PC C based program raw data	Scan Number
8617	109	8676	109
8616	110	8676	110
8616	111	8676	111
8617	112	8676	112
8616	113	8676	113
8616	114	8676	114
8616	115	8676	115
8618	116	8676	116
8618	117	8676	117
8616	118	8676	118
8617	119	8676	119
8616	120	8676	120
8616	121	8675	121
8617	122	8676	122
8617	123	8676	123
8617	124	8674	124
8617	125	8676	125
8618	126	8676	126
8616	127	8676	127
8618	128	8676	128

Table 5.3 DE2-115 and PC C Based System Bridge Mode samples

Figure 5.4 plots the graph of the Bridge Mode raw data obtained from both the DE2-115 Altera FPGA Board and the current system.

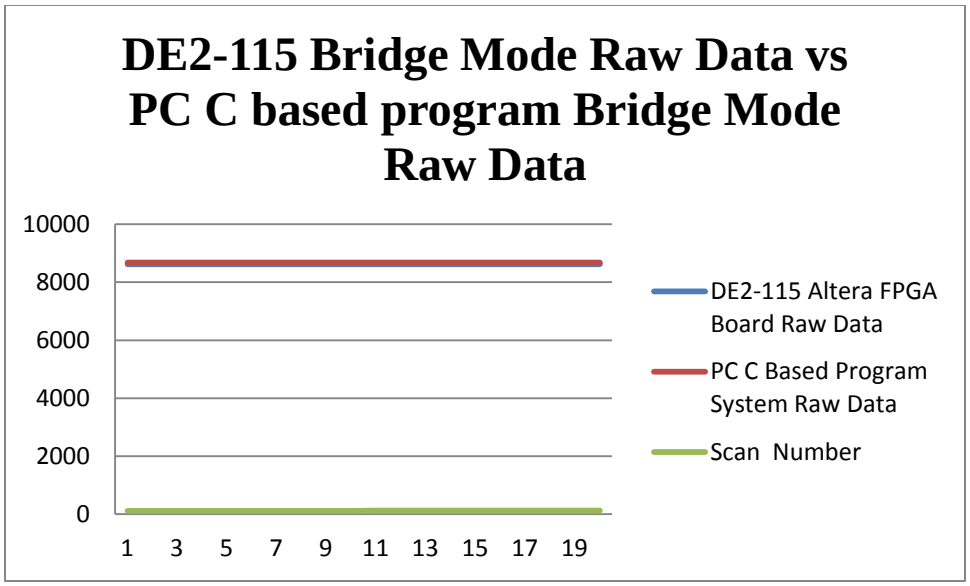

Fig 5.4 DE2-115 Bridge Mode samples vs Existing System Bridge Mode samples

Observing Figure 5.3 and Tables 5.1 and 5.2, the standard deviation of the raw data obtained from the PC C based program and DE2-115 FPGA is 60.93189 and 60.29151 respectively. The standard deviation shows the values are in the same range. Similarly from the Figure 5.4 and Table 5.3, it can be shown that the standard deviation of both the PC C based program and DE2-115 FPGA raw data samples is 0.48936 and 0.48939 respectively. From the above graphs and results, the laser has been interfaced successfully to the Altera based FPGA DE2-115 board in a manner similar to the interface of the laser to current system.

CHAPTER 6

SUMMARY, CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

This research effort was performed to investigate the use of FPGA based system to interface and process real-time data from the laser based distance sensor in order for the usage in a pavement profiling system. During the research, a FPGA system using the NIOS II softcore processor was successfully designed and found suitable for this purpose. The design implemented is based around the Altera DE2-115 development board. The target application for the work was the UTA portable pavement profiler instrument used by the Texas Department of Transportation.

This particular work is focused mainly on the interface of the laser to the Altera FPGA DE2-115 board. The road profiler requires a laser which outputs the distance between the laser and the road and that can be used for a high speed road profile data collection and processing. The line laser used is the LMI Roline laser 1145.

The Roline laser sensor provides distance measurements in the form of Ethernet packets using TCP/IP protocol. An analog signal provided by the sensor provides a pulse to signal the beginning of each line scan.

The laser system design is a multitasking system that operates on the principle of a client-server model. As such the FPGA system has be designed so that is capable of behaving both as a Server and Client in order to interface to the laser. The multitasking of Client-Server is implemented on the Altera FPGA DE2-115 board by using the MicroC/OS II Real Time Operating System (RTOS). The two tasks, one operating as a client and other as a server are created with different priorities. The dedicated real-time road profiler switches its tasks based on the current laser model.

The laser sensor provides distance measurements in two different modes the free mode and the Bridge mode. In free mode, the sensor outputs 80 points for each scan and in the bridge mode it outputs a single point for each scan. The single point is calculated based on the LMI bridging algorithm. The points are read on and stored on the on-chip memory of the NIOS 2 processor.

The points are collected in both the free and the bridge mode of the laser on both the existing system and the newly built dedicated real time FPGA based road system. The data collected are then compared on respective modes of the laser. The Graphs in the chapter 5 shows they are identical. These graphs show that a system is built using the FPGA which can be interfaced to the Roline Laser.

6.2 Future Work

The on-chip memory is small. Hence, a memory controller is still needed to manage the memory using external memory. The complete real time road profiler system would require the design and development of interfacing and synchronizing the data obtained from several other sensors used for the road profiler. Both the memory and additional sensor interfacing is still needed.

The additional sensor like the accelerometer, distance encoder etc outputs the data in the form of analog signals. Hence, there is need to develop a system capable of handling these sensors outputs. An A/D converter as DT Console has to be interfaced and synchronized to the FPGA board.

The real time data collected is fed to the Walker State machine to obtain the road profile. Hence, The Walker State Machine still needs to be designed and developed on the Altera FPGA based system. VGA screen can be interfaced to complete the system to display the results.

REFERENCES

1. Cyclone IV Device Handbook, volume 1 by Altera.
2. “Alaska_88E1111-002” Ethernet data sheet of DE2-115 board by Altera.
3. DE2-115 User manual by Altera.
4. Introduction to the Altera NIOS II Soft Core Processor by Altera.
5. Getting started with the Altera’s DE2-115 Board.
6. Quartus II Introduction Using Schematic Design.
7. “nichestack_tutorial” example by Altera.
8. “15101_4.11.0.36_MANUAL_FireSync_Host_Protocol_Refrence”. FireSync Protocol References Manual by LMI Technologies.
9. “15105-4%5b1%5d.11.0.0_MANUAL_User_RoLine11x0%5b1%5d.” Reference manual for Roline Laser by LMI Technologies.
10. Berkley socket programming by Wikipedia.
11. Design and Development of a General Purpose Embedded Acquisition System for Transportation Application. (Thesis report of Akshay)
12. R. Walker and E. Fernando, “A portable profiler for pavement profile measurements – interim report." Texas Transportation Institute, College Station, TX" Technical Report 0-6004-1, 2009.
13. The Little Book of Profiling by Michael W. Sayers and Steven M. Karamihas.

BIOGRAPHICAL INFORMATION

Subrahmanya Ramaswamy was born in Karnataka, India in April 1987. He completed his Bachelor's in Instrumentation Technology from the Visvesvaraya Technological University, India in 2008. He worked at IBM India Pvt. Ltd., India as a Application Developer for two years prior to joining University of Texas at Arlington, Texas to pursue his Masters in Electrical Engineering. Subrahmanya worked as an Intern at Research in Motion, for five months working with Blackberry's Playbook Team. His research interests lies in various aspects of Embedded systems and Real Time Operating System.