DESIGN OF EMBEDDED NIOS II PROCESSOR BASED SYSTEM FOR IMPLEMENTING BRIDGING ALGORITHM OF PROFILING APPLICATION

by

VINAY S ASHI

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 ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2012

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge the enthusiastic supervision of Dr. Roger walker who has been a constant source of motivation for me and guiding me with his invaluable advice during my research work. I am really thankful for his patience, cheerful disposition and positive feedback.

I would like to thank Dr. Jonathan Bredow and Dr. W Alan Davis for showing interest in my research and taking the time to be a part of my thesis committee.

I sincerely appreciate all the help and encouragement received from my lab mate Ashwin Arikere. Also I would like to thank my friends and roommates for their moral support.

Last but not the least; I would like to thank my parents and family for their support throughout my studies and their faith which made it possible for me to complete my work.

November 7, 2012

ABSTRACT

DESIGN OF EMBEDDED NIOS II PROCESSOR BASED SYSTEM FOR IMPLEMENTING BRIDGING ALGORITHM OF PROFILING APPLICATION

Vinay Ashi, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Roger Walker

The Texas Department of Transportation (TxDOT) uses multiple instruments for quality assurance of new and existing pavements. These instruments measure the different characteristics of the pavement such as the longitudinal profile, transverse profile, texture, rutting, etc. to determine the smoothness and ride quality of the pavement. These instruments mainly consist of different sensors such as lasers, accelerometers, distance encoders, etc., to calculate the ride quality. Various ride statistics such as International Roughness Index (IRI), Mean Profile Depth (MPD), and Present Serviceability Index (PSI) are then calculated and used to monitor the conditions of the roads.

The Roline laser system is used to calculate the road profile. The profiling module is configured and controlled by a program running on a windows operating system. A certified profiling system with the wide line laser is now being successfully used on roads. Also, a dual laser profiling system, which can collect data from two profiling modules simultaneously, is being used.

The objective of this thesis is to investigate and develop an embedded NIOS II processor based system on Cyclone IV FPGA (DE2-115 development board) from Altera Corp for implementing a bridging algorithm. This algorithm is used to preprocess the free mode data from the Roline line laser and compute the bridge values which will be used in profile computation. For computing bridge values, processing steps include data qualification, tilt compensation and averaging. A memory interface has to be designed for this NIOS II processor based system, so that the Roline laser data to be used for bridge mode data calculation and computed bridge mode values can be saved in a flash memory. Memory system design consists of a combination of components such as on-chip memory, DMA controller, SDRAM and SD card Interface. FPGA resource usage is also monitored for the system design.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

A high-end embedded system usually has a processor and I/O peripherals to perform general user interface functions and special hardware accelerators to handle computation-intensive operations. These components can be integrated into a single integrated circuit, commonly referred to as SoC (system on a chip). As the capacity of FPGA (field-programmable gate array) devices continues to increase, the same design methodology can be realized in an FPGA chip and is known as SoPC (system on a programmable chip) or PSoC (programmable system on a chip).

Designing a conventional embedded processor based system requires examination of functional requirements and then selecting a processor, external I/O peripherals, and application specific standard product devices to construct the hardware platform. Because of the fixed-sized processor architecture and the cost of manufacturing printed circuit boards, the hardware configuration is usually rigid and the desired system functionalities are performed by customized software.

An FPGA device contains logic cells and interconnects that can be programmed to perform a specific function. The desired hardware functionalities are usually described in Hardware Description Language (HDL), which is then synthesized and implemented in the FPGA device. Because of the programmability of FPGA devices, customized hardware can be incorporated into the embedded system as well. The processor configuration can be tailored, custom I/O interface can be created or existing I/O peripherals can be used to implement the complete hardware system. The SoPC-based embedded system provides flexibility in customizing the hardware and software and match the system specific needs.

1.1 Objective

The main objective of current research at the Embedded Systems and Instrumentation Lab (ESI) at the University of Texas at Arlington (UTA) is to develop a portable embedded system on programmable chips (SoPC) that can be used for computation of profiles in real-time. The research conducted as part of this thesis is to develop an embedded NIOS II processor based system on a Cyclone IV FPGA chip for implementing a bridging algorithm to run on this embedded system

1.2 Organization of Thesis

The next chapter introduces road profiling and the inertial profiling technique, which provides a basic knowledge of road profilers. Chapter 3 provides an Introduction to Altera's DE2-115 development board and NIOS II embedded processor which is the main component of the system development. It also provides an overview of embedded SoPC development flow. Chapter 4 describes the operational modes of the Roline laser used in a profiler and also gives a description of the bridging algorithm which is used to preprocess the free mode data from the Roline laser. Chapter 5 describes the system implementation and configuration used in the NIOS II processor based embedded portable profiler. Chapter 6 provides detailed analysis of bridging algorithm execution on different system designs and FPGA resource usage of these designs. Implementation results are compared and concluded in Chapter 7.

CHAPTER 2

BACKGROUND

2.1 Road Profiling

Road profile measurements are used to provide statistical information regarding road surface roughness. A profile is a two-dimensional slice of the road surface, taken along an imaginary line and is obtained by combining three parameters: A reference elevation, height relative to the reference and longitudinal distance

Instruments such as Dipstick, Rod and Level, and Inertial Profilers can be used to compute the profiles. Current technologies use inertial profilers to collect profile data, typically at highway speeds. An accelerometer provides the inertial reference. The instantaneous distance is measured using a non-contacting sensor such as a laser. The longitudinal distance of the instruments is obtained using a distance encoder. Combining the outputs from these three devices, profile can be computed and various statistics can be obtained by processing the profile. Depending on the nature of the statistic desired, various techniques may be applied to the profile data to analyze road conditions. For example, International Roughness Index (IRI) is one such statistic which can be obtained from the profile data.

2.2 High Speed Inertial Profiler Systems

There are three basic components in an inertial profiler as shown in Figure 2.1 [2]

1. An accelerometer provides inertial reference and measures the vertical acceleration of the vehicle as it moves along the road. The instantaneous height of the vehicle body with respect to the reference can be computed using the acceleration measurement.

2. A non-contacting sensor is used to measure the height of the ground relative to the reference. Lasers, ultrasonic or infrared transducers are used for this purpose.

3. A distance encoder is used to measure the longitudinal distance traveled by the vehicle.

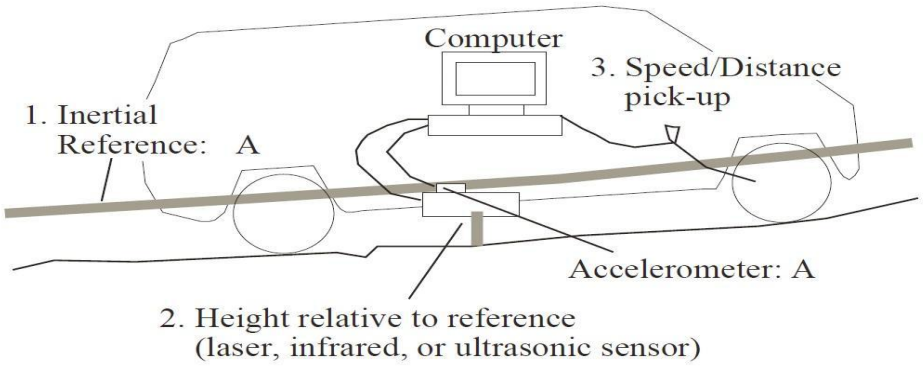


Figure 2.1 Road Profiler Systems

The main advantage of using 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 the longer wavelengths (typically 100 meters and greater) are filtered out because of errors in the measurement process. However, once filtered, accurate and reliable profile data and corresponding statistics can be obtained.

The profiler developed at ESI mainly uses an accelerometer, a laser and a distance encoder as the sensors. Once data is collected from these sensors, the road profile is constructed from the readings according to equation (2.1)

$$p(t) = \int \int [a(t)dt]dt - H(t) \qquad (2.1)$$

where,

H(t) is height of the accelerometer from the road measured by laser,

a(t) is vertical acceleration.

The accelerometer values are integrated twice to yield vertical displacement of the vehicle body. A two pole high pass filter is applied at this stage to remove the effect of long wavelengths (low frequency) on the profile. These wavelengths represent the overall road

curvature and underlying grade and are thus attenuated using filters. The result is added to the laser values and passed through another two pole high pass filter to obtain an accurate road profile.

A portable profiler module was developed as part of TxDOT Research Project 0-6004. The portable profiler computes road surface profile using the same technique as an inertial profiler. Except for the distance encoder which is attached to the vehicle wheel, all sensors including laser and accelerometer, power, and signal conditioning are housed inside the profiler instrument module that is placed on the front or rear bumper of the profiler vehicle. Data from these sensors are then converted to digital values using a data acquisition board (DT9816) and sent to a notebook PC running on Windows OS in the vehicle. The UTA Ride console program interfaces with the data acquisition board and retrieves accelerometer and distance encoder data from it and stores it in a text file. The Roline laser data is received in the form of ethernet packets and stored in a separate file as comma separated values. A separate program then computes the profile using laser and accelerometer readings. This profile however is phase shifted because of the use of an IIR filter in the profile computation process. A reverse filter is applied on the profile to remove the non-linear phase characteristics present in the profile.

CHAPTER 3

EMBEDDED SYSTEM DEVELOPMENT FOR PROFILING

In Chapter 2, the basic inertial profiling techniques and the profiler module developed as part of TxDOT research project were discussed. During measurements using profiler module, the data collected from various sensors, are converted to digital values and then sent to a notebook PC located in the vehicle for computing profile. All communications between the sensors and the PC is done via a USB cable. The instrument module is attached to the power, PC (via the USB), distance encoder, and the infrared start signal via four connectors. The UTA Ride console program interfaces with the data acquisition board, Line laser and stores the raw sensor data in a text file. A separate program is used to computes profile from the raw laser and accelerometer readings.

A real-time program can be developed that processes the data collected from various lasers immediately, producing the profiles in real-time. A Single Board Computer (SBC) or an embedded processor based board mounted inside the instrument box can be used to collect and process the data. With the SBC, or embedded processor in the module, interfaced with Roline laser and Data Acquisition module (DAQ), profile may be computed by the module itself rather than having a separate notebook PC inside the vehicle. The embedded processor system along with peripherals may be designed to start data collection and process it.

Embedding a processor inside an FPGA has many advantages. Specific peripherals can be chosen based on the application, with unique user-designed peripherals being easily interfaced to the system. A variety of memory controllers enhance the FPGA embedded processor system's interface capabilities.

An FPGA embedded processor system offers many exceptional advantages including:

1. Customization: Design of an FPGA embedded processor system has complete flexibility to select any combination of peripherals and controllers. Application specific peripherals can be designed that can be interfaced directly to the processor.

2. Component and cost reduction: With the versatility of the FPGA, previous systems that required multiple components can be replaced with interfacing to a single FPGA chip provided with various peripherals. For example, both data acquisition board and laser can be interfaced to the FPGA based processor and this system can be used to compute the profile. By reducing the component count in a design, board size and inventory will be reduced which will save design time and cost.

3. Hardware acceleration: The most important reason to choose an FPGA embedded processor is the ability to make tradeoffs between hardware and software to maximize efficiency and performance. If an algorithm is identified as a software bottleneck, a custom co-processing engine can be designed in the FPGA specifically for that algorithm.

4. Obsolescence mitigation: FPGA soft processors are an excellent solution in this case since the source HDL for the soft-processor can be purchased. Ownership of the processor's HDL code may fulfill the requirement for product lifespan guarantee.

Altera and Xilinx are two major companies that produce FPGA families which embed a physical processor core into the FPGA silicon. In the next section we will discuss the features of DE2-115 development board from Altera which has Cyclone IV FPGA and provides NIOS II soft processor based development. This development board will be used for developing embedded NIOS II processor based system for implementing an algorithm to preprocess the Roline laser data.

3.1 Introduction to DE2-115 Development Board

A top view of DE2-115 board is shown in Figure 3.1[10]. It depicts the layout of the board and indicates the location of the connectors and key components.

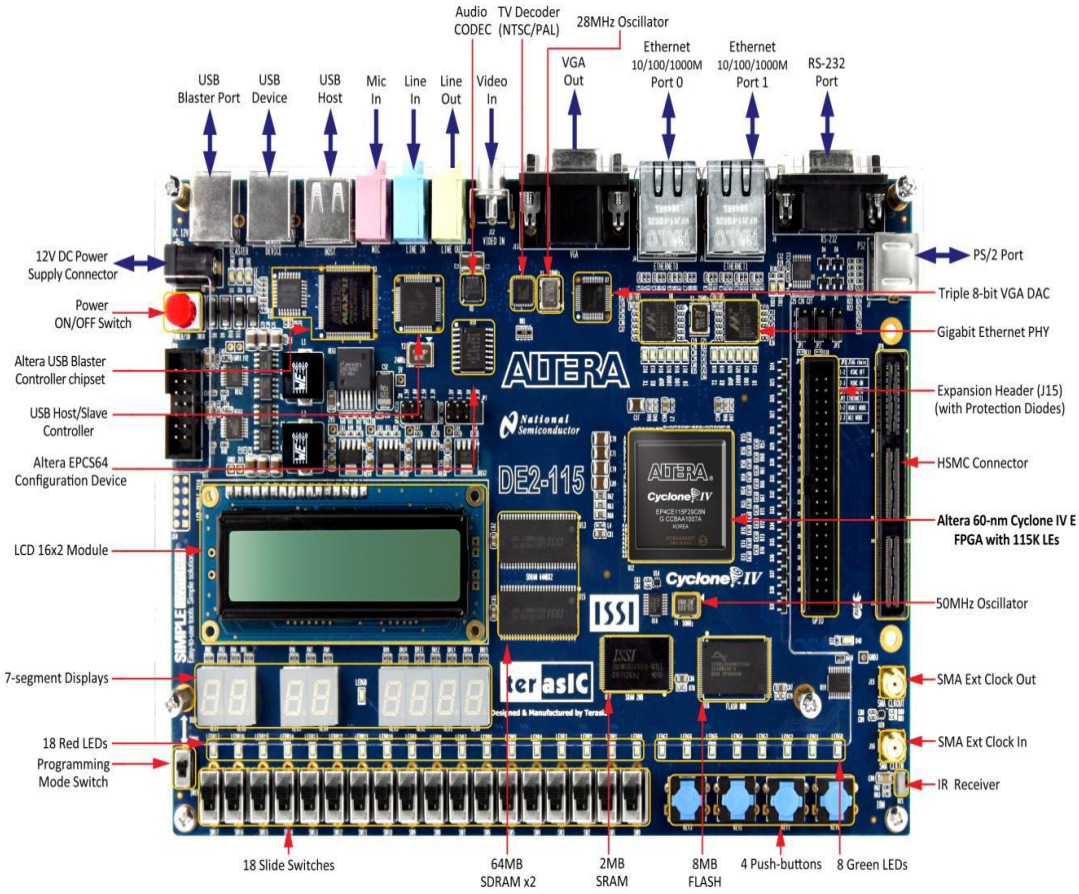


Figure 3.1 DE2-115 development board (Top View)

The DE2-115 board has many features and peripherals that allow users to implement a wide range of designed circuits, from simple circuits to various multimedia projects. To provide maximum flexibility for the user, all connections are made through the Cyclone IV FPGA device. Thus, the user can configure the FPGA to implement any system design.

DE2-115 board (bottom view) showing the SD card socket is shown in Figure 3.2 [10]

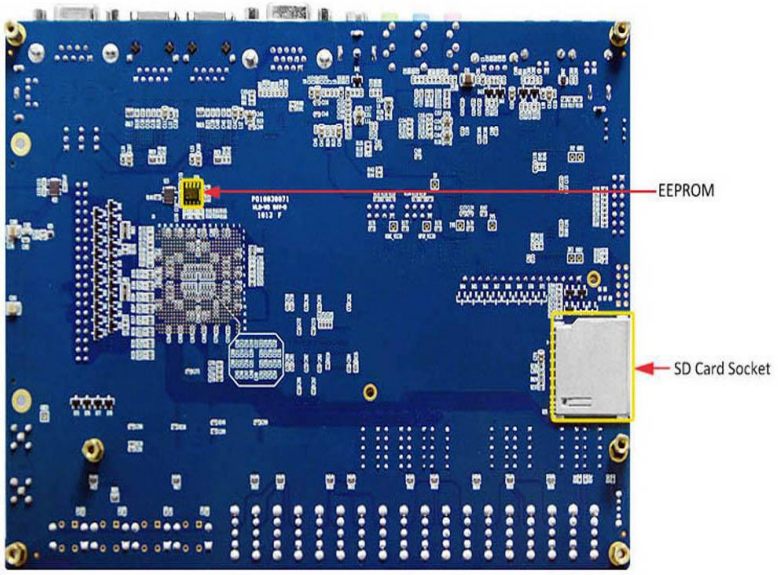


Figure 3.2 DE2-115 development board (Bottom View)

The components and peripheral available on the DE2-115 development board is given in table 3.1

Table 3.1[10] DE2-115 board components and peripherals

DE2-115 Board Information	
Feature	**Description**
FPGA	Cyclone IV EP4CE115F29C7 with EPCS64 64-Mbit serial configuration device
I/O Interfaces	Built-in USB-Blaster for FPGA configuration
	Line In/Out, Microphone In (24-bit Audio CODEC)
	Video Out (VGA 8-bit DAC)
	Video In (NTSC/PAL/Multi-format)
	RS232
	Infrared input port
	PS/2 mouse or keyboard port
	Two 10/100/1000 Ethernet
	USB 2.0 (type A and type B)
	Expansion headers (one 40-pin header)
	HSMC high-speed header

Table 3.1 - *Continued*

Memory	128 MB SDRAM, 2 MB SRAM, 8 MB Flash
	SD memory card slot
Displays	Eight 7-segment displays
	16 x 2 LCD display
Switches and LEDs	18 toggle switches
	18 red LEDs
	9 green LEDs
	Four denounced pushbutton switches
Clocks	50 MHz clock, External SMA clock input, External SMA clock output

3.2 Introduction to NIOS II Processor

The NIOS II processor is a configurable soft Intellectual property (IP) core, as opposed to a fixed, off-the-shelf microcontroller. It is possible to add or remove features on a system-by-system basis to meet performance or price goals. Soft means the processor core is described in HDL code and later synthesized by using Altera Cyclone IV FPGA's generic logic cell.

The NIOS II processor is a general-purpose RISC processor core with the following features:

- Full 32-bit instruction set, data path, and address space, 32 general-purpose registers and 32 interrupt sources
- External interrupt controller interface for more interrupt sources
- Single-instruction 32 × 32 multiply and divide producing a 32-bit result
- Dedicated instructions for computing 64-bit and 128-bit products of multiplication
- Floating-point instructions for single-precision floating-point operations and single-instruction barrel shifter
- Access to a variety of on-chip peripherals, and interfaces to off-chip memories and peripherals
- Hardware-assisted debug module enabling processor start, stop, step, and trace under control of the NIOS II software development tools
- Optional memory management unit (MMU) to support operating systems that require MMUs and Optional memory protection unit (MPU)

- Software development environment based on the GNU C/C++ tool chain and the NIOS II Software Build Tools (SBT) for Eclipse
- Integration with Altera's Signal Tap® II Embedded Logic Analyzer, enabling real-time analysis of instructions and data along with other signals in the FPGA design
- Instruction set architecture (ISA) compatible across all NIOS II processor systems
- Performance up to 250 DMIPS

There are three different versions of NIOS II processors:

- NIOS II/f: The fast core is designed for optimal speed performance. It has a 6-stage pipeline, Instruction cache, data cache, and dynamic branch prediction.
- NIOS II/s: The standard core is designed for small size while maintaining good performance. It has a 5-stage pipeline, instruction cache, and static branch prediction.
- NIOS II/e: The economy core is designed for optimal size. It is not pipeline and contains no cache.

3.3 Embedded SoPC Development Flow

The basic NIOS II based development flow is shown in Figure 3.3[1]. The four important steps in embedded SoPC development [1] are elaborated in the following subsections

1. Software-Hardware partition
2. Hardware development flow
3. Software development flow
4. Physical implementation and test

Software-Hardware partition:

An embedded application usually performs a collection of tasks. In an SoPC-based design, a task can be implemented by hardware, software or both. Based on the performance

requirement, complexity and hardware core availability, we can decide the type of implementation accordingly.

Hardware development flow:

Custom hardware can be divided into 3 categories

- NIOS II processor and standard I/O peripheral ("NIOS configuration" in figure 3.3): Altera provides the soft cores of the processor and a collection of frequently used I/O peripherals. We can select the system specific I/O peripherals and configure the basic NIOS II system.
- User I/O peripherals and hardware accelerators ("User I/O & HA" in figure 3.3): Pre-designed core may not exist or cannot satisfy the performance requirement for few specialized I/O functions or computation-intensive tasks. We must design the hardware from scratch and integrate it into the NIOS II system as a custom I/O peripheral.
- User logic: Some portion of the hardware may be separated from the NIOS II system. It is not attached to the NIOS interconnect structure and does not interact directly with the processor.

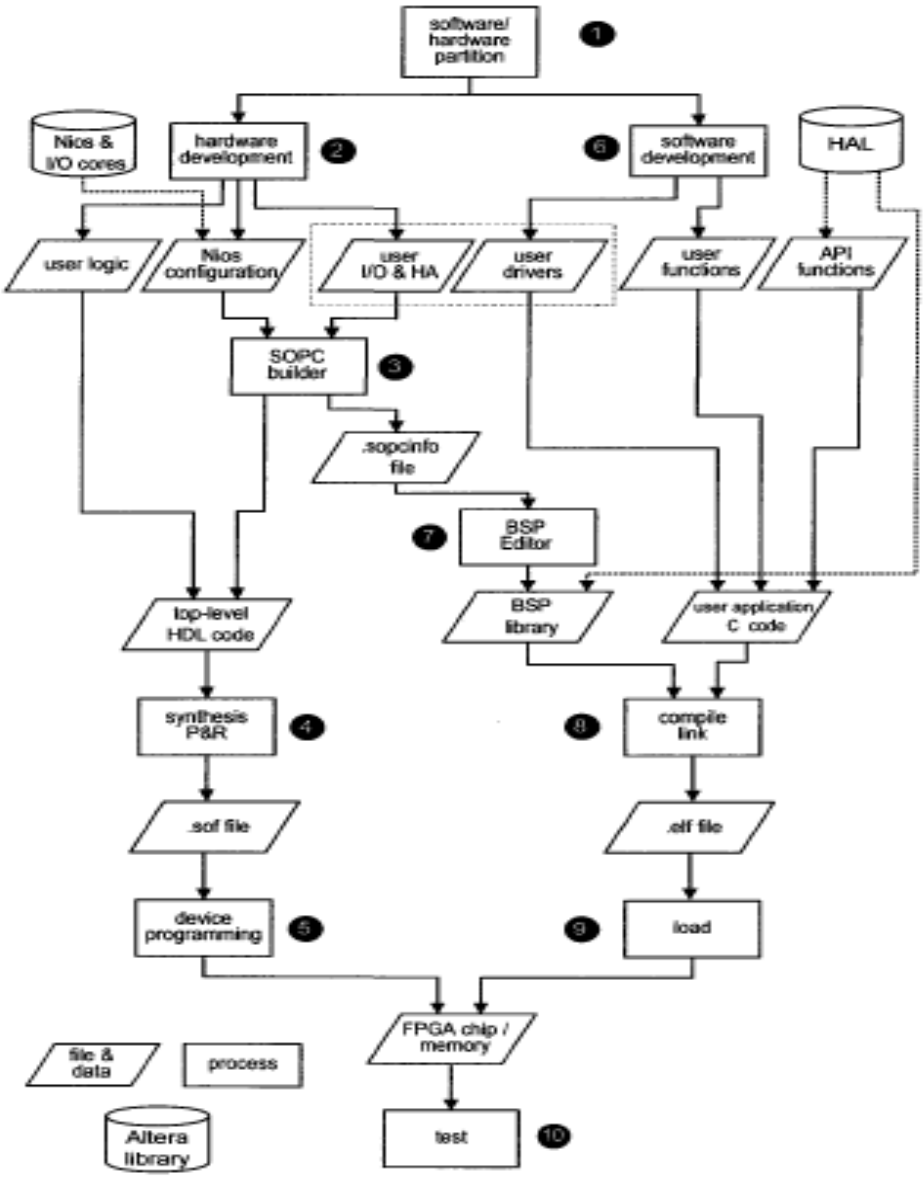


Figure 3.3 Development flow of a system with NIOS II EDS

Altera's SOPC Builder software can be used to configure the processor, select the desired standard I/O cores, and incorporate the user designed I/O peripherals. SOPC Builder then generates the HDL codes for the customized NIOS II system and also generates the .sopcinfo file that contains system configuration information. Auto-generated code for NIOS II system can be combined with the other use logic codes to form the final top-level HDL

description. The top-level HDL code contains the description of the complete hardware. The Quartus II version 11.1 tool is used to perform synthesis, placement, routing and generate the FPGA configuration file.

Software development flow:

Altera provides a software library, which is integrated into its HAL (hardware abstraction layer) platform, for the NIOS II system. It consists of I/O device drivers, which are low-level routines to access I/O peripherals, and a collection of high-level functions in an application programming interface (API).

From the hardware-software interface's point of view, the software code can be divided into three categories:

- API functions: These are the functions from the Altera HAL platform.
- User I/O drivers: When designing a custom I/O peripheral or hardware accelerator, we also need to develop software I/O routines to control its operation and to exchange its data with the processor.
- User functions: These implement the needed functionalities for the embedded application.

These drivers and functions can be used to construct the application program. When a NIOS II system is created, the processor and I/O configuration is recorded in the .sopcinfo file. BSP Editor Software program examines this file, extracts the needed device drivers from the HAL library, and builds up a BSP (Board Support Package) library to support the system. Compilation and linking the software routines, BSP library results final software image file (i.e., the .elf file).

Physical implementation of the system and testing:

Physically implementing the system involves two steps. First download the FPGA configuration file to the FPGA device (i.e. "program" the device), and then load the software

image into NIOS IPs memory. The physical system can be tested afterwards by running the application program.

The NIOS II EDS (embedded design suite), is provided by Altera for software development. It has been customized for the NIOS II processor environment. SBT (Software Build Tools) GUI is based on the Eclipse open development environment and customized for the NIOS II software development flow.

The next chapter provides a brief introduction to the Roline laser and the modes of operation. The Bridging algorithm used to preprocess the laser's free mode data sample is also explained in detail.

.

CHAPTER 4

BRIDGING ALGORITHM

4.1 Roline Laser and Modes of Operation

The RoLine 11xx family of lasers is a new generation of high speed, high density 3D lasers that can be used for road profiling. A laser line projector projects a 2.6-5.4" wide laser footprint. A digital camera mounted at an angle to the laser plane acquires images of the reflected light pattern created on the target. The distance to the target is calculated from the images taken by the digital camera based on the position of the laser line in the image. Figure 4.1 [4] shows the measurement principle of the laser

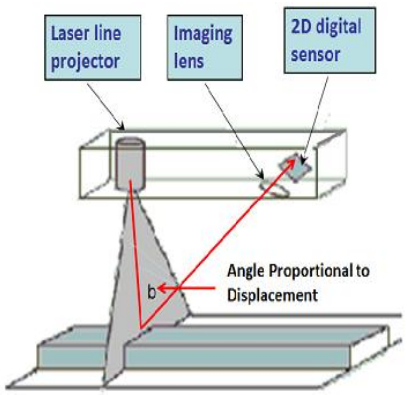


Figure 4.1 Distance measurement principle of RoLine Laser

The laser has a scanning rate of 3000 Hz and operates at 48VDC. The measurement range (MR) of this laser is 200 mm, with a field of view (FOV) of 100 mm or 4.0" at the center of its measurement range. The output of the RoLine laser is in a digital format. The elevation from the clearance distance is represented as a 16-bit number with a resolution of 0.01 mm. RoLine 1145 has an option of delivering the output in either Ethernet packets or Selcom serial format.

The RoLine 11xx lasers also outputs an opto-isolated 3000 Hz pulse while it is

collecting data for synchronizing external devices. The profiling algorithm should get the accelerometer and laser data at the same instant and this sync pulse is used to synchronize with the accelerometer. The rising edge of the sync pulse coincides with the start of the laser scan and will be used to synchronize the accelerometer value at that instant.

There are two basic modes of operation for the RoLine 11xx family.

Free Mode: **I**n the free mode, the data for each point in the scan is sent from the laser. Each packet is configured to contain 100 scans of the laser. Each scan of the laser is configured to contain 80 points. These parameters can be changed by writing a new settings file to the laser. The default mode for the RoLine laser is the free mode. The format of the free mode data is shown in Figure 4.2[4]

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

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).

Figure 4.2 Free mode data message format.

Bridge Mode: In this mode, laser sends a single value per scan. This reduces the high-resolution, high-density full profile data to an averaged value in that scan. The bridge values contain the distance to the target and the attribute field contains the sync index and tracking mode information. Each packet is configured to contain 100 scans of the laser and each scan will contain one bridged value. The format of the Bridge mode data is shown in Figure 4.3 [4]

Bridge Message

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 bride values grouped in message
reserved[7]	64s	Reserved for internal use
attributes[count][3]	64s	Attributes (defined below)
bridgeValue[count]	16s	Bridged value arrays (in 0.01mm)

Bridge Message 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).

Figure 4.3 Bridge mode data message format.

4.2 Bridging Algorithm Steps

The profiling algorithm requires one data point per scan to compute the profile. If free mode is selected for operation, then the laser output file contains the entire raw data and a bridged value for each scan has to be calculated before it is used for computing the profile. Laser output will contain only the bridged values if the bridge mode is set in the configuration, which can then be used directly for generating the longitudinal profile.

The laser manufacturer specifies the steps in calculating the bridge mode values but does not provide the source code for computing the bridge values from free mode operation because of the Intellectual Property (IP) concerns. At ESI, the bridging algorithm is implemented from the steps specified by the laser manufacturers and have obtained bridge mode equivalent values using free mode data.

The International Roughness Index (IRI) values obtained from running the ESI version of bridging algorithm are lower than those obtained by using the laser's bridge mode. TxDOT wants to investigate the differences because these results affect the operation of TxDOT for evaluating the conditions of the road in terms of the profile.

The bridging algorithms involve the following steps[4]:

1. Data qualification – Scan should not contain more than a user-defined number of invalid points. Invalid points are represented by a value of -32767 and occur when the laser line is not captured by the camera sensor. These invalid points are discarded before computing the bridged value.

2. Tilt Compensation (Optional) - Vehicle movement (roll) and rutting may modify the perceived contact point and may be accommodated. If compensating for tilt, the center point of the scan is used for compensating the tilt. Figure 4.4 [4] illustrates this step.

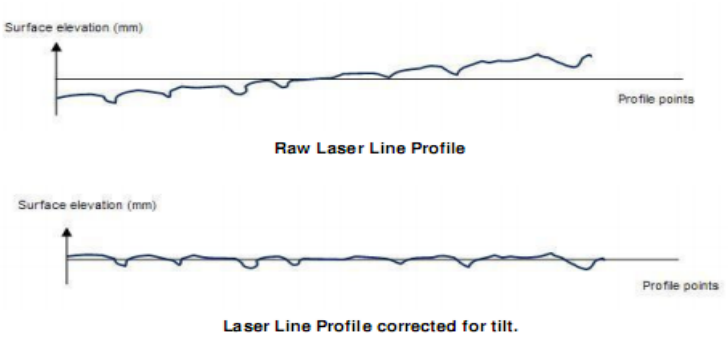


Figure 4.4Tilt Compensation for Bridging Algorithm

3. The points are then sorted according to their elevation for selection. The highest and lowest values can then be removed depending on user defined parameters called "window skip" and "window size". This is done to remove the outliers.

4. The average of the remaining points is calculated and the resulting value is the bridged value for that scan.

CHAPTER 5

SYSTEM IMPLEMENTATION AND DESCRIPTION

The Altera's DE2-115 prototyping board is used in the development of hardware design. It contains an FPGA device from Cyclone IV family. The development board has 128 MB SDRAM, 2 MB SRAM and 8 MB flash memory. The Roline laser data and Analog to Digital (A/D) data from the Data Acquisition (DAQ) module need to be stored for intermittent bridge mode data calculation and profile computation. The Roline laser data and the A/D data collection from multiple runs could exceed a size of 2 GB. The Flash memory provided on this development board does not satisfy the requirement needs of profiling application.

There is a need to interface a flash memory SD card to the NIOS II Processor so that the Roline laser and A/D data samples can be stored for computation of the bridge mode data. The SD memory card slot has been provided on the development board and will be used for this purpose. Then to handle the data, a Direct Memory Access (DMA) needs to be included in the design. The complete system will be a combination of FPGA on chip components and a few onboard components.

Roline laser data in free mode is saved on the SD card as a text file. An application program running on NIOS II processor computes the bridge mode values using the bridging algorithm. The computed bridge values are then written to a second text file on the SD card by the same program.

This chapter will discuss the main components used to design the FPGA based embedded NIOS processor system for implementing the Bridging algorithm.

5.1 NIOS II Processor and Memory Interface using DMA Controller

A Block diagram consisting of NIOS II processor and memory interface with a DMA controller is shown in Figure 5.1

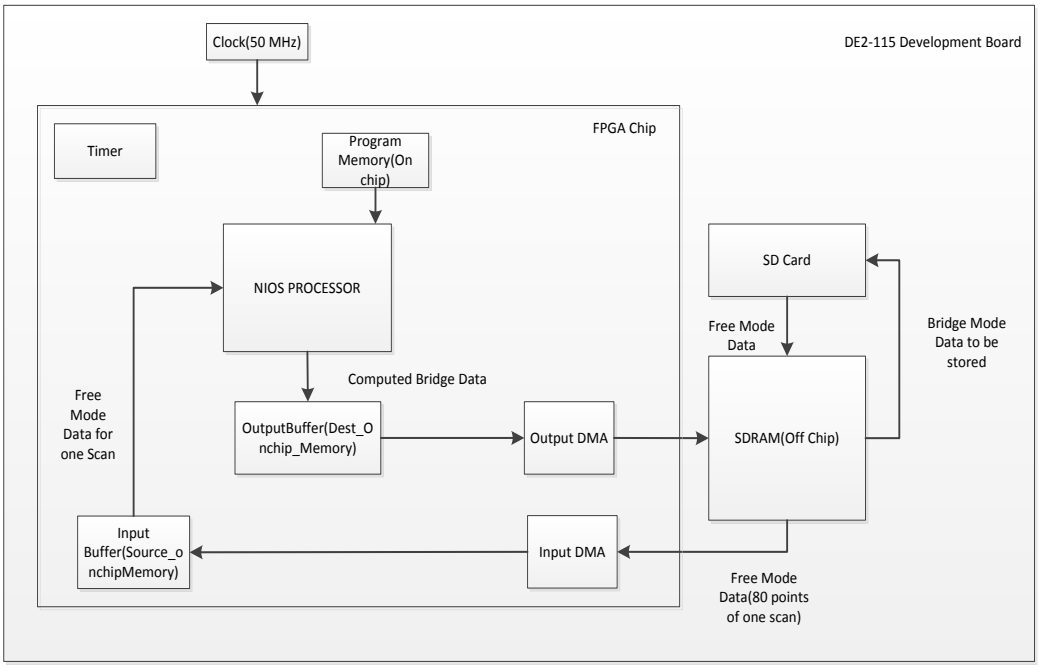


Figure 5.1 NIOS II Processor and Memory Interface using DMA Controller

In this design, the NIOS II processor is interfaced to flash memory (SD card) via SDRAM and DMA controller. The components used in this system design are integrated into SOPC builder-generated system. These components are internal to the FPGA chip.

NIOS II/f Processor Core: The configuration and features of the NIOS II/f processor is given below

- Has separate optional instruction and data caches
- Provides optional MMU to support operating systems that require an MMU
- Provides optional MPU to support operating systems and runtime environments that desire memory protection but do not need virtual memory management
- Can access up to 2 GB of external address space when no MMU is present and 4 GB when the MMU is present

- Supports optional external interrupt controller (EIC) interface to provide customizable interrupt prioritization
- Supports optional shadow register sets to improve interrupt latency
- Supports optional tightly-coupled memory for instructions and data
- Employs a 6-stage pipeline to achieve maximum DMIPS/MHz
- Performs dynamic branch prediction and provides optional hardware multiply, divide, shift options to improve arithmetic performance
- Supports the addition of custom instructions
- Supports the JTAG debug module
- Performance Max. FMAX (2) =185 MHz and the processor is connected to 50 MHz clock using DE2-115 onboard oscillator.

On Chip Memory:

- Cyclone® IV devices feature embedded memory structures to address the on-chip memory needs of Altera® Cyclone IV device designs.
- Embedded memory structure consists of columns of M9K memory blocks that can be configured to provide various functions, such as RAM, shift registers, ROM, and FIFO buffers.
- M9K block has the following features:
    - 8,192 memory bits per block (9,216 bits per block including parity)
    - Independent read-enable (rden) and write-enable (wren) signals for each port
    - Single-port and simple dual-port modes support for all port widths
    - True dual-port (one read and one write, two reads, or two writes) operation
    - Byte enables for data input masking during writes
    - Two clock-enable control signals for each port (port A and port B)
    - Initialization file to pre-load memory content in RAM and ROM modes

- Single-port mode supports non-simultaneous read and write operations from a single address
- IORD and IOWR API's are used to read and write data from on-chip memory location.

Interval Timer:

- Interval Timer core with Avalon® interface is an interval timer for Avalon-based processor systems, such as a NIOS® II processor system
- The core provides the following features:
    - 32-bit and 64-bit counters.
    - Controls to start, stop, and reset the timer.
    - Two count modes: count down once and continuous count-down.
    - Option to enable or disable the interrupt request (IRQ) when timer reaches zero.
    - Optional watchdog timer feature that resets the system if timer ever reaches zero.
    - Optional periodic pulse generator feature that outputs a pulse when timer reaches zero.
- Interval Timer is used as a Timestamp driver in this design to count the number of clock ticks and then calculate the time spent in particular method/function.
- Rate is same as the hardware frequency of the NIOS II processor system (i.e., 50 MHz)

Input/ Output DMA:

- Direct memory access (DMA) controller core with the Avalon® interface performs bulk data transfers, reading data from a source address range and writing the data to a different address range.

- Avalon Memory-Mapped (Avalon-MM) master peripheral (CPU) offloads memory transfer tasks to the DMA controller.
- DMA is configured to transfer 32 bytes of data for every transaction in this design.
- DMA controller is SOPC Builder-ready and integrates into SOPC Builder-generated system.

DE2-115 board components including SDRAM and SD card socket are used to provide the memory support to the entire system.

SDRAM:

- DE2-115 board features 128MB SDRAM, implemented using two 64MB SDRAM (IS42S16320B from ISSI).
- IS42S16320B Memory chip is organized as 8M x16x4 Banks .Each 134,217,728-bit bank is organized as 8,192 rows by 1024 columns by 16 bits.
- Each 512Mb SDRAM is a high speed CMOS, dynamic random-access memory designed to operate in 3.3V Vdd and 3.3V Vddq memory systems containing 536,870,912 bits
- Each device has separate 16-bit data lines connected to the FPGA, and share control and address lines.

SDRAM Controller:

- SDRAM controller core with Avalon® interface provides an Avalon Memory-Mapped (Avalon-MM) interface to off-chip SDRAM
- Core presents an Avalon-MM slave port that appears as linear memory (flat address space) to Avalon-MM master peripherals.
- Controller performs refresh operations, open-row management, and other delays and command sequences.
- Avalon-MM slave port supports peripheral-controlled wait states for read and writes.

- Interface signals must be connected externally to the SDRAM chip(s) through I/O pins on the Altera device
- Clock for the SDRAM chip (SDRAM clock) must be driven at the same frequency as the clock for the Avalon-MM interface on the SDRAM controller (controller clock). On-chip phase-locked loop (PLL) can be used to alleviate clock skew between the SDRAM controller core and the SDRAM chip

SD card Interface:

Altera DE-series boards also have an SD card port. It allows a SD card to be connected to an FPGA-based design on these boards. Altera University Program (UP) SD Card IP Core is used to access the files on the SD card.

Features of Altera University Program (UP) SD Card IP Core are listed below:

- Hardware circuit designed by Altera that enables the use of an SD card on the Altera DE2-115 development board
- SD Card IP Core included in an SOPC Builder design with a Nios II processor, can be accessed and controlled from software.
- Hardware Abstraction Layer (HAL) device driver designed for the Altera University Program SD Card IP Core provides an easy way to access data stored on an SD card.
- Hardware Abstraction Layer (HAL) device driver makes a SD card to appear as a 16-bit File Allocation Table (FAT16)-based portable drive.
- Driver functions as a File Allocation Table (FAT) reader/writer, allowing users to access data on the SD card that has been saved in FAT16 format.
- Current version of the driver supports only FAT16.

The system implemented in the SOPC builder is as shown in the Figure 5.3

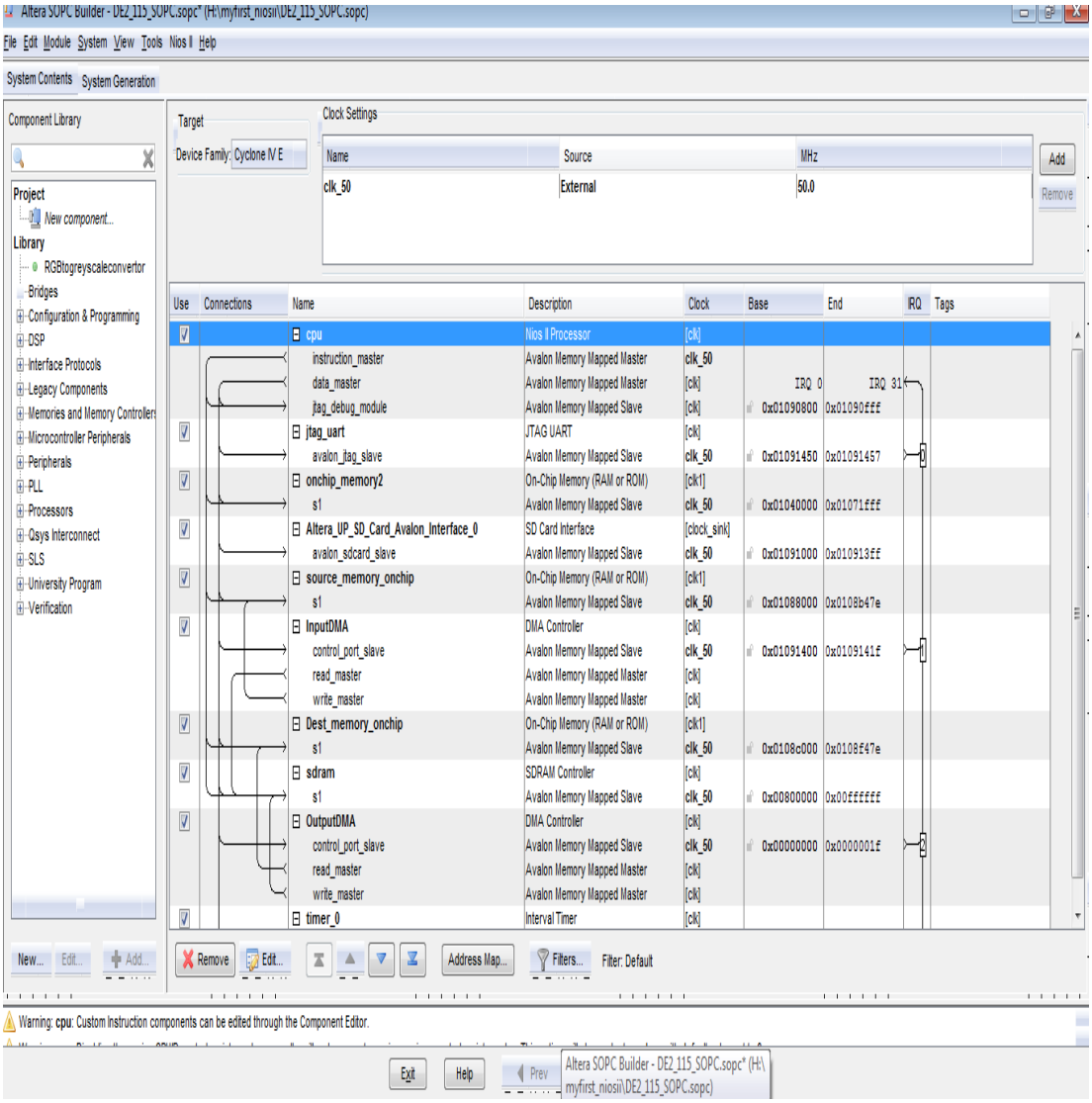


Fig 5.3 NIOS II Processor and Memory Interface using DMA Controller in the SOPC builder

The main idea behind using a DMA controller is to perform parallel computation of bridge values in different tasks (threads) and schedule them using some real time operating system.

An application program running on this processor would compute the bridge mode value from free mode data of Roline laser. The flowchart in figure 5.2 illustrates the flow of the application program implementing the bridging algorithm for the NIOS II processor and memory interface using a DMA controller.

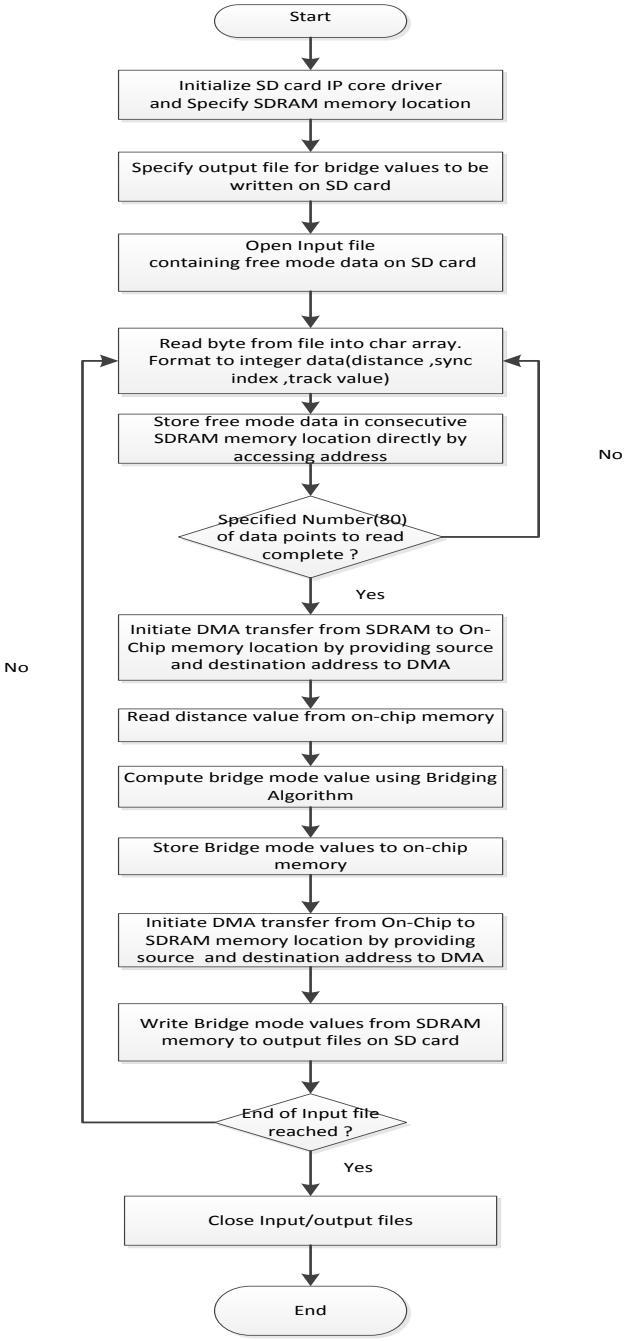


Figure 5.2 Application program implementing the bridging algorithm for NIOS II Processor and Memory Interface using a DMA controller

The SD card driver is initialized and SDRAM memory location for storing the free mode data is specified. API functions provided by HAL device driver are used to read data from the file on the SD card. The free mode data (distance value, sync index and track information) for 80 points of one scan read from file are stored into SDRAM memory by accessing its address location. The address range of the SDRAM and on-chip memory used is provided by the SOPC builder tool and is available in a system library file of the project created using the SOPC information file. Input DMA is instructed to transfer a set of data points for one scan from SDRAM to on-chip memory locations. The processor then computes bridge mode value using the bridging algorithm and stores the result in the on-chip memory. The Output DMA then transfers this result back to SDRAM. Finally the bridge values are written from the SDRAM to a text file on SD card by this application program.

5.2 NIOS II Processor and Memory Interface without DMA Controller

A block diagram consisting of NIOS II processor and memory interface without the DMA controller is shown in the Figure 5.4

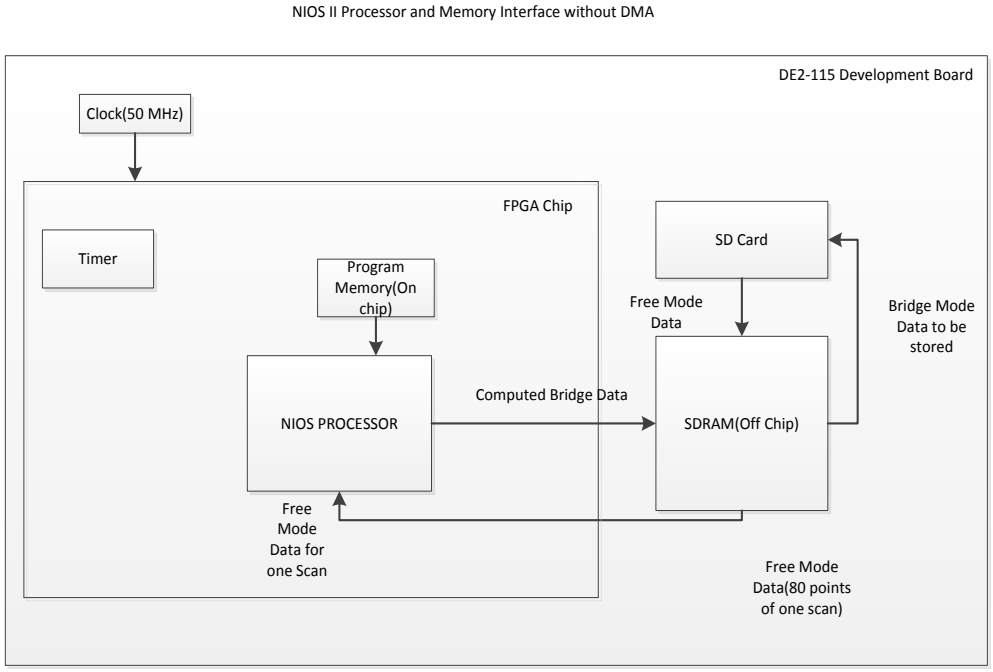


Figure 5.4 NIOS II Processor and Memory Interface without DMA Controller

In this design, NIOS II processor is interfaced to flash memory (SD card) via SDRAM only. The configuration of the components used is similar to that used in the section 5.1.

The system implemented in the SOPC builder is as shown in the Figure 5.5

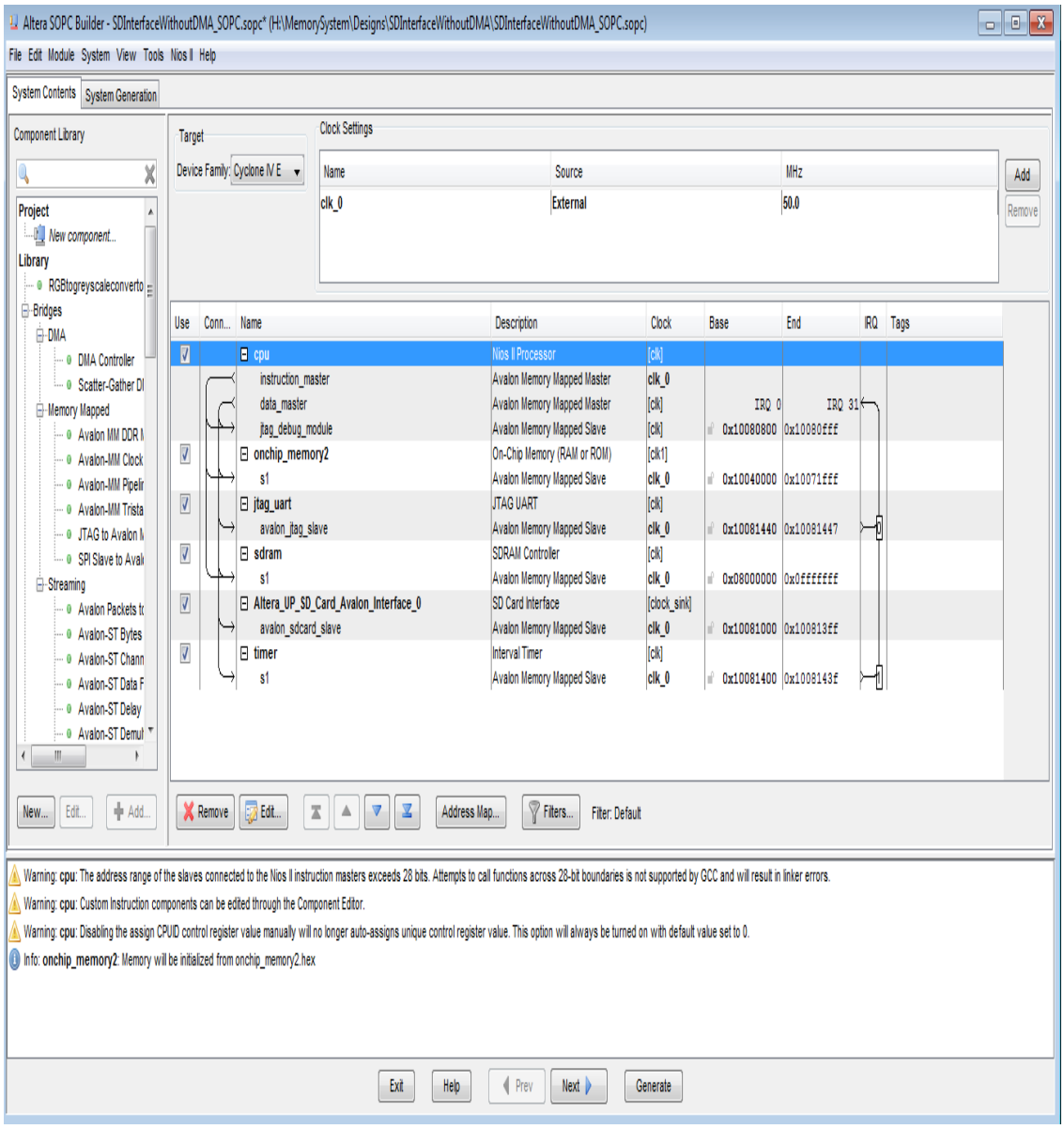


Figure 5.5 NIOS II Processor and Memory Interface without DMA Controller in SOPC builder

The flowchart in figure 5.6 illustrates the flow of the application program implementing the bridging algorithm for the NIOS II processor and memory interface without using a DMA.

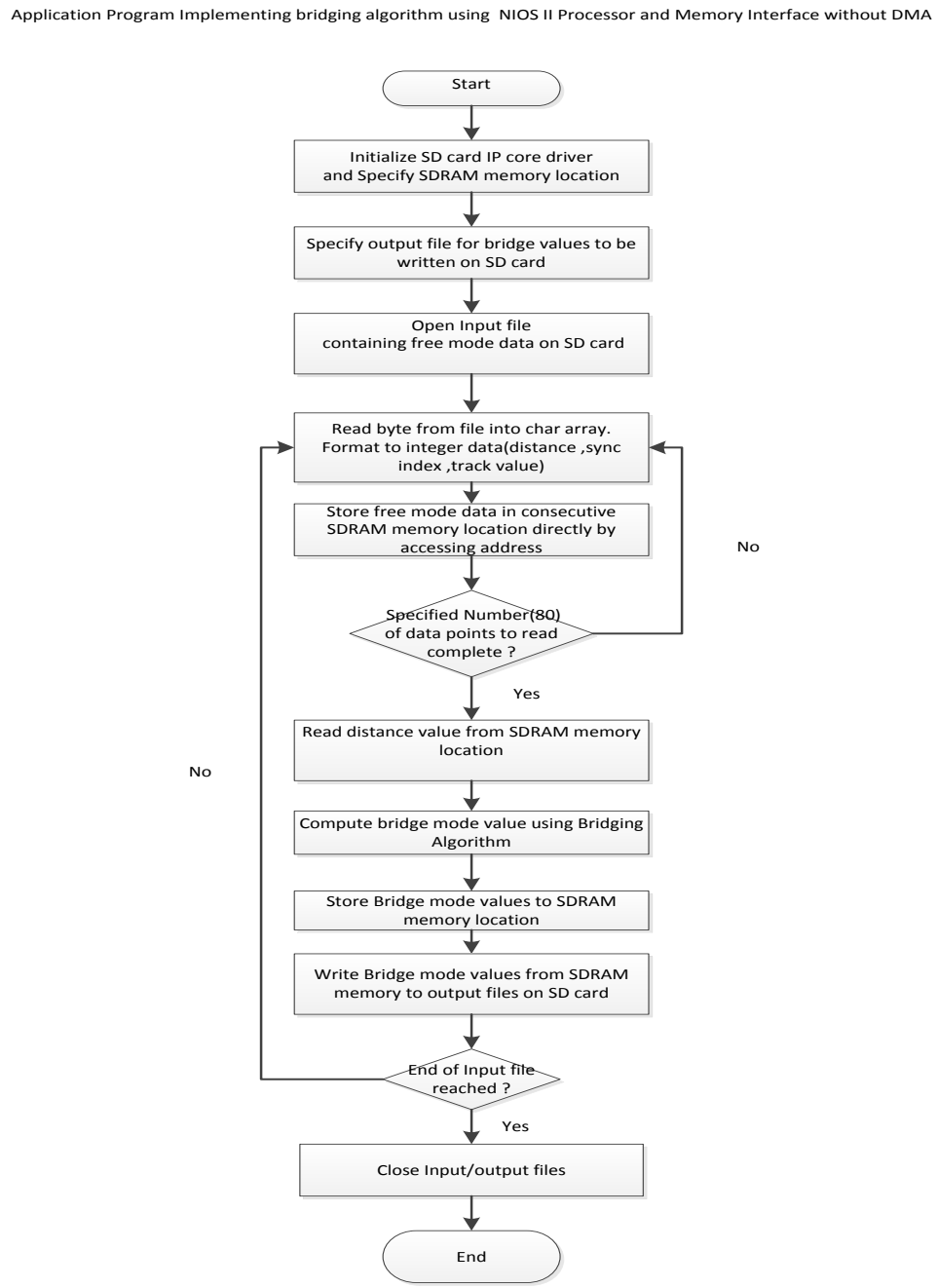


Figure 5.6 Application program implementing the bridging algorithm for NIOS II Processor and Memory Interface without using a DMA controller

Free mode data (80 points) of one scan saved on the text file in the SD card is initially read into SDRAM memory by the application program. The bridging algorithm reads the free mode values from SDRAM memory and computes the bridge mode values. The bridge mode values are stored to SDRAM initially. Then the bridge value is written from the SDRAM to a text file on the SD card by the application program.

5.3 NIOS II Processor and Direct SD Memory Interface

A block diagram consisting of the NIOS II processor and a direct SD card memory interface is shown in the Figure 5.7

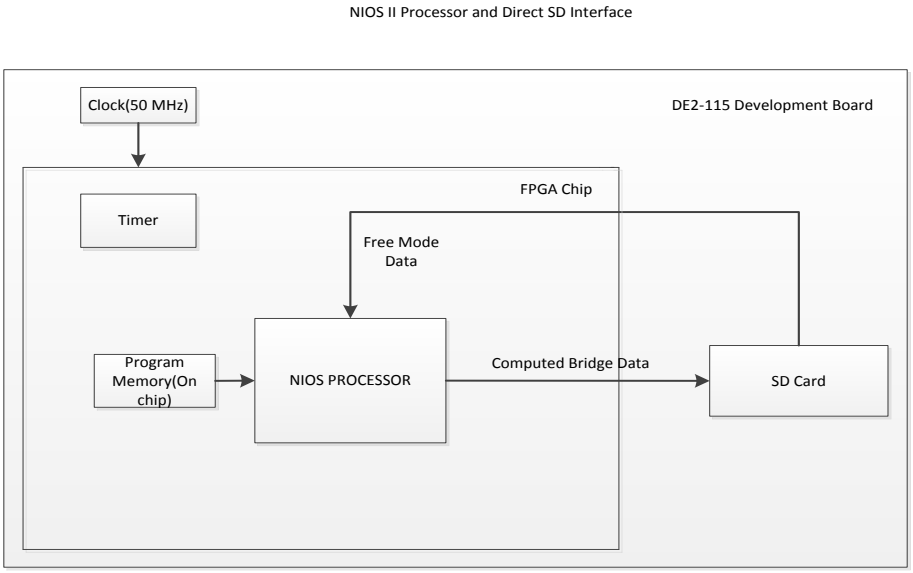


Figure 5.7 NIOS II Processor and Direct SD card memory Interface

In this design, NIOS II processor is interfaced to flash memory (SD card) directly and does not involve SDRAM memory or DMA in its operation of computing the bridge mode values using the bridging algorithm. The configuration of the components used is similar to that used in the section 5.1

The system implemented in the SOPC builder is shown in the Figure 5.8

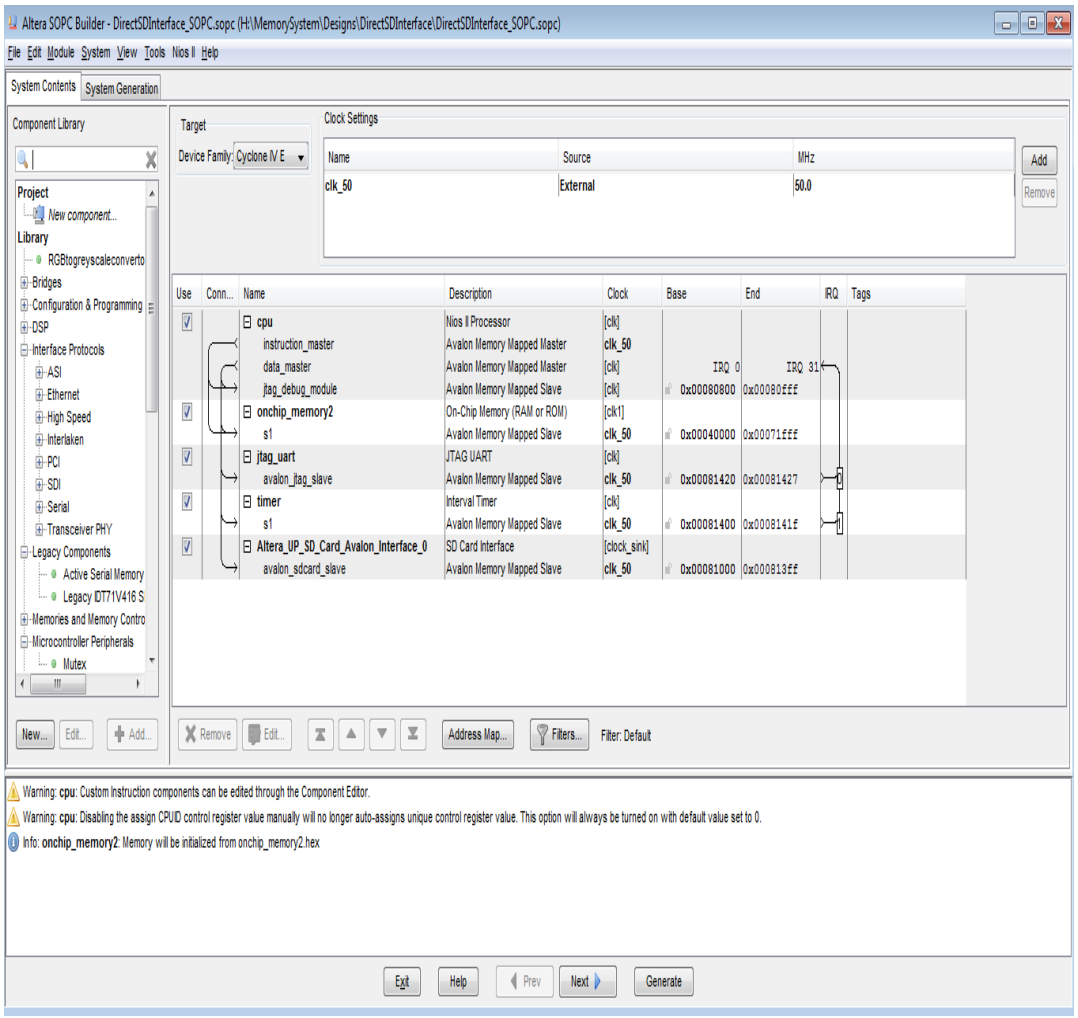


Figure 5.8 NIOS II Processor and Direct SD card memory Interface in the SOPC builder

The flowchart in figure 5.9 illustrates the flow of the application program implementing the bridging algorithm for NIOS II processor and direct SD card interface. Free mode data (80 points) of one scan of Roline laser saved on a text file in the SD card is read into data memory of the processor by the application program. The bridging algorithm computes the bridge mode value using the free mode data from data memory. The computed bridge values are then written to a text file on the SD card by this program.

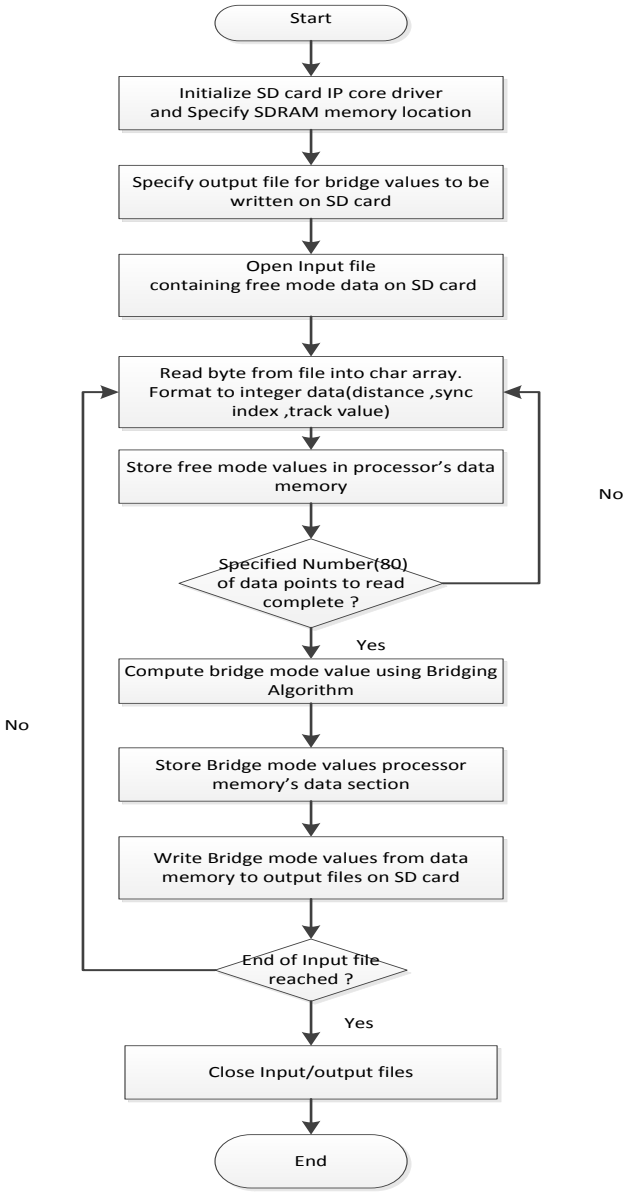


Figure 5.9 Application program implementing the bridging algorithm for the NIOS II Processor and Direct SD Memory Interface

CHAPTER 6

SYSTEM IMPLEMENTATION RESOURCE USAGE AND EXECUTION TIMING ANALYSIS

The different hardware designs which were discussed in Chapter 5 were implemented on the DE2-115 development board and are analyzed for throughput and resource usage in this chapter. Throughput and resource usage are two of the most important issues to be considered in the design of the system. Therefore, an analysis of FPGA resource usage and execution time for each design executing the bridging algorithm was made.

6.1 System Implementation Resource Usage

The Quartus II tool used to program the FPGA device (EP4CE115F29C7) provides an estimate of the resources on the chip that is needed for the system design. An important resource available on the FPGA chip is the logic cell. The current design is used for implementing the bridging algorithm used in profile computation. An effort has been made to analyze resource usage because as system complexity increases, resource usage becomes a critical parameter in implementing the entire profiling algorithm on the FPGA. The number of FPGA on chip resources consumed is dependent on the components that are used in each design. Each subsection provides an overview of the number of resource elements used for the design.

6.1.1 NIOS II Processor and SD Memory Interface using DMA

This design approach interfaces the processor and the SD card memory via SDRAM memory (Off-Chip) and a DMA controller. Altera's pre-designed SDRAM controller core is used on chip to generate the required control signals for SDRAM access. Also the DMA controller core which is used to transfer data blocks between SDRAM memory and on-chip memory is part of the FPGA. Due to the additional SDRAM controller core and DMA controller in this design, there is an increased usage of logic cells and memory bits when compared to the other designs which are discussed in the next section.

Information about the number of elements of FPGA resources used for this design is shown in table 6.1

Table 6.1 FPGA resource usage for NIOS II Processor and Memory Interface using DMA Controller

FPGA Resources	Number of Elements
Logic Cells	6679
Dedicated Logic Registers	3630
I/O Registers	68
Memory Bits	1924480
M9Ks	251
DSP Elements	4
Pins	62
LUT-Only LCs	3049
Register-Only LCs	665
LUT/Register LCs	2965

6.1.2 NIOS II Processor and SD Memory Interface without using DMA

This design approach interfaces the processor and SD card memory via SDRAM memory (Off-Chip) only. A DMA controller is not used in this design. As in the previous design Altera's pre-designed SDRAM controller core is used on chip to generate the necessary control signals for SDRAM access. SDRAM controller in this design has increased use of logic cells when compared to the Direct SD card interface design which is discussed in the next section. And since there is no DMA controller in the design, the number of resources consumed is less compared to the previous design. Information about the number of elements of FPGA resources used for the design in which the NIOS II processor and SD Memory is interfaced without using a DMA component is shown in table 6.2

Table 6.2 FPGA resource usage for NIOS II Processor and Memory interface without DMA Controller

FPGA Resources	Number of Elements
Logic Cells	5293
Dedicated Logic Registers	3126
I/O Registers	68
Memory Bits	1707392
M9Ks	217
DSP Elements	4
Pins	62
LUT-Only LCs	2167
Register-Only LCs	642
LUT/Register LCs	2484

6.1.3 NIOS II Processor and Direct SD Card Memory Interface

This design interfaces the processor and SD card memory directly without using SDRAM memory and DMA. Since there is no SDRAM controller core as well as any DMA, the number of elements used is the least compared to other two designs. Information about the number of elements of FPGA resources used for the design in which NIOS II Processor and SD Memory is interfaced directly is shown in table 6.3

Table 6.3 FPGA resource usage for NIOS II Processor and Direct SD Card Memory Interface

FPGA Resources	Number of Elements
Logic Cells	4704
Dedicated Logic Registers	2789
I/O Registers	0
Memory Bits	1706432
M9Ks	217
DSP Elements	4
Pins	5
LUT-Only LCs	1915
Register-Only LCs	583
LUT/Register LCs	2206

6.2 Bridging Algorithm Execution Timing Analysis

The Roline laser scan data samples in free mode format are saved on a text file and stored in the SD card for this analysis. An application program written in the C programming language would read the set of data points from the laser scan stored on the SD card and feed these set of values to the bridging algorithm for calculating the bridge mode value. Once the bridge mode values are computed, the bridge mode value is saved on a text file in the SD card.

The total time taken by the application includes reading the free mode data from SD card memory, computing the bridge mode value using bridging algorithm and then saving the bridge mode values to a text file on the SD card. The total execution time can be represented by the equation 6.2a given as

Total Execution Time = Read laser free mode data from SD card + Compute bridge mode values using bridging algorithm + Write bridge mode values to SD card ---------- (6.2a)

Also, total execution time can be calculated using the formula given in equation 6.2b

Total Execution Time = Total number of clock cycles * Clock time period---------- (6.2b)

The system clock is running on 50 MHz in each design using the on-board oscillator. The clock time period will be 20 ns. An interval timer is used to count the total number of clock cycles. This timer also runs using the system clock i.e. 50 MHz.

6.2.1 Execution timing analysis for NIOS II processor and Memory Interface using DMA

Initially data points for a single scan of the Roline laser are copied from a file on the SD card to SDRAM memory. These data points are transferred to the FPGA on-chip memory by the DMA. The DMA is configured to transfer 32 bytes of data for every transaction. On chip memory access has a typical access latency of one clock cycle. Since the clock latency is low, the processor can access data at a faster rate as compared to the rate if the data is available in the off chip SDRAM memory. The Bridging algorithm reads the data from on-chip memory to compute and then save the bridge mode values into on-chip memory again. Then the bridge mode values are transferred from on-chip memory to SDRAM using DMA. Finally the bridge

mode values are written from SDRAM memory to a file on SD card. These steps are repeated for each scan and execution time is computed. The total number of laser scans equals 10, 20, 30, 40 and 50.

The experiment was run five times for the varying number of laser scans and table 6.4 shows the average total execution time of the bridging algorithm with the SD card interface using the DMA.

Table 6.4 Total execution time and total number of clock ticks for system using DMA

Number of Laser Scans	Number of Clock Cycles	Total Execution Time (in sec)
10	21349150	0.426983
20	44242600	0.884852
30	59185230	1.1837046
40	75696100	1.513922
50	92327820	1.8465564

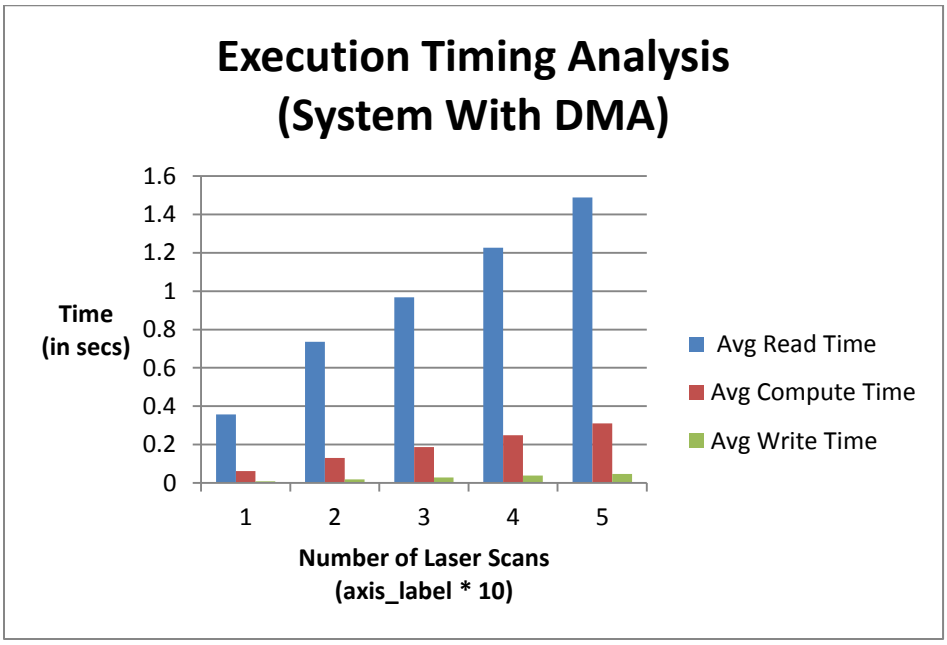
The breakup of the total execution time of bridging algorithm including reading of laser free mode data, compute bridge mode values and writing the values to SD card memory for different number of scans is shown in table 6.5.

Table 6.5 Breakup of total execution time for read, compute and write time for system using DMA

Number of Laser Scans	Read Time (in sec)	Compute Time (in sec)	Write Time (in sec)
10	0.3564852	0.0616828	0.0088148
20	0.7357274	0.130562	0.0185622
30	0.9676594	0.186569	0.029476
40	1.2265298	0.2490326	0.0383598
50	1.4883088	0.3110168	0.0472306

The read time, compute and write time is shown as a graphical representation for SD card interface using DMA in Figure 6.1

Figure 6.1 Graphical representation of read time, compute and write time for NIOS II processor system using DMA controller



From Table 6.5, time spent in reading the laser free mode data from the SD card is very high again, compared to the time spent in bridge mode calculation and time spent in writing the bridge mode values to the SD card.

This design also shows that read time is comparable to the initial system design without using DMA. In this design, DMA offloads the task of data transfer from the processor but the processor has to wait until the DMA completes the transfer of a set of data points so that the processor can run the bridging algorithm on these set of data points. The system performance can be greatly improved if the processor can compute the bridge mode values for the set of data points already in on-chip memory while the DMA is transferring a new set of data points to different locations of the on-chip memory. Such a feature would require a real time operating system to control the functions.

6.2.2 Execution timing analysis for NIOS II processor and Memory Interface without using DMA

Data points for a single scan are copied from a file on the SD card to SDRAM memory for processing. The bridging algorithm reads the data from the SDRAM memory to compute and save the bridge mode values into SDRAM again. The bridge mode values are then written from SDRAM to a file on the SD card. These steps are repeated for each scan and execution time is computed for varying number of laser scans.The experiment was run for five times for the varying number of laser scans and the table 6.6 shows the average total execution time of bridging algorithm for SD card interface without using DMA.

Table 6.6 Total execution time and total number of clock ticks for system without DMA

Number of Scans	Number of Clock Cycles	Total Execution Time (in sec)
10	20572310	0.4114462
20	42398190	0.8479638
30	56483020	1.1296604
40	72162840	1.4432568
50	87836680	1.7567336

The breakup of total execution time of bridging algorithm including reading of laser free mode data, compute bridge mode values and write the values to SD card memory for different number of scans is shown in table 6.7

Table 6.7 Total execution time break-up for read, compute and write for system without DMA

Number of Laser Scans	Read time (in sec)	Compute time (in sec)	Write time (in sec)
10	0.344909	0.0582944	0.008243
20	0.7075808	0.1230892	0.017294
30	0.9277564	0.17415	0.027754
40	1.1738174	0.2334328	0.0360062
50	1.4209122	0.2914884	0.0443328

The read time, compute time and write time for SD card interface without using DMA is shown in figure 6.2 as a graphical representation.

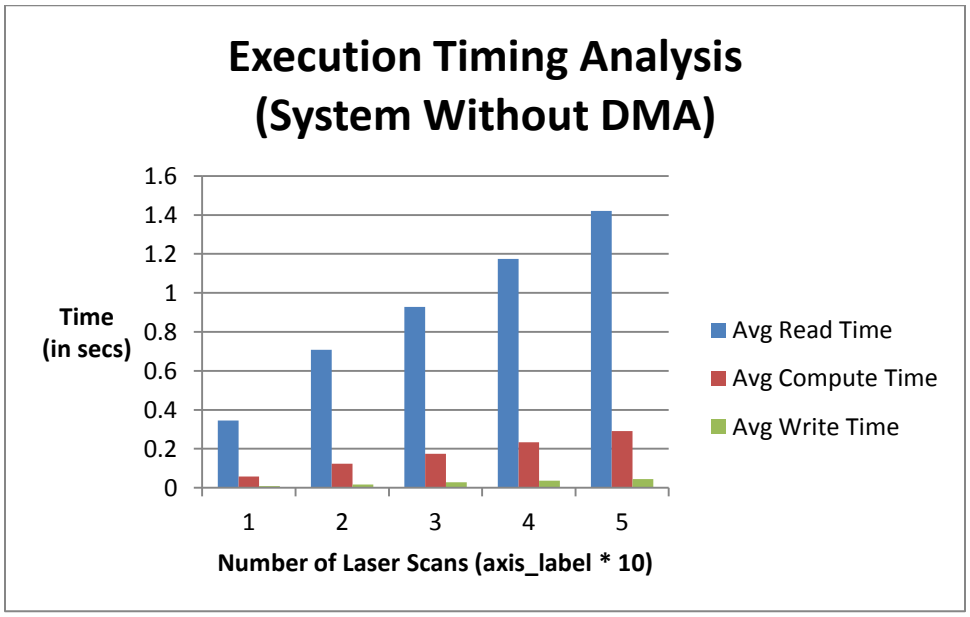


Figure 6.2 Graphical representation of read time, compute and write time for NIOS II processor system without using DMA controller

From Table 6.7, time spent in reading the laser free mode data from SD card is very high compared to the time spent in bridge mode calculation and time spent in writing the bridge mode values to SD card.

Three main factors affecting the read time are:

1. Altera University Program (UP) SD Card IP core is used to provide commands and data to the SD card. The Hardware Abstraction Layer (HAL) device driver designed for the Altera University Program SD Card IP core provides an easy way to access data stored on an SD card. API functions from the device driver are used for reading and writing data to a file on SD card. The read function reads a byte from a file on SD card and returns an integer (ASCII) value of the character read. The characters read have to be converted into appropriate integer data format of the laser scan. All the values for eighty

points of scan have to be read byte wise by using the read function and then used for computing the bridge value. This is one of the main reasons why read time is more compared to bridge mode value compute time and write time. The write function from the device driver also writes only one byte per write function call. The computed bridge mode values (one for 80 data points) to be written on the SD card is less than the number of data points to be read, hence the total write time is lower compared to the total read time.

2. Each scan has 80 points and each point has three attributes to be read from the file on the SD card i.e. distance value, sync index and track mode information. All the attributes are represented by 32 bit integers. For ten scans, a total of 800 data points need to be read from the file. As the number of scans increase the read timing also increases.
3. An SDRAM controller is used to provide the control signals for SDRAM access. The CAS latency for SDRAM in this design is three clock cycles. Three attributes of each scan point has to be read from SDRAM memory. Since the latency is higher compared to on-chip memory, the read time for read operation from SDRAM memory is high.

To improve the efficiency i.e. speed of the system, the total read time has to be decreased. The next section is simplified by directly interfacing the SD card to processor without using SDRAM memory or the DMA and analyses the execution time for the bridging algorithm.

6.2.3 Execution timing analysis for NIOS II processor and Direct SD Memory Interface

Data points for a single scan are read from the file on the SD card and stored in data memory of the processor (on-chip). The bridging algorithm then computes the bridge mode values and writes them back to a text file on the SD card. This process is repeated for varying number of laser scans.

The experiment was run for five times and table 6.8 shows the average total execution time of the bridging algorithm for a different number of scans for Direct SD card interface to the system.

Table 6.8 Total execution time and total number of clock ticks for system with direct SD card Interface

Number of Laser Scans	Number of Clock Cycles	Total Execution Time (in sec)
10	20101830	0.4020366
20	41541940	0.8308388
30	55378260	1.1075652
40	70713800	1.414276
50	85864770	1.7172954

Table 6.9 shows the breakup of total execution time of the bridging algorithm including reading of laser free mode data, computing the bridge mode values and writing the values to SD card memory for different number of scans

Table 6.9 Total execution time for read, compute and write for system with direct SD card Interface

Number of Laser Scans	Read time (in sec)	Compute Time (in sec)	Write time (in sec)
10	0.3322518	0.0616576	0.0081272
20	0.6842244	0.1295524	0.0170618
30	0.8936954	0.1865534	0.0273162
40	1.1303164	0.2484366	0.0355226
50	1.3636734	0.3099038	0.0437182

The graphical representation of read time, compute and write time for NIOS II processor system and direct SD card interface is shown in figure 6.3

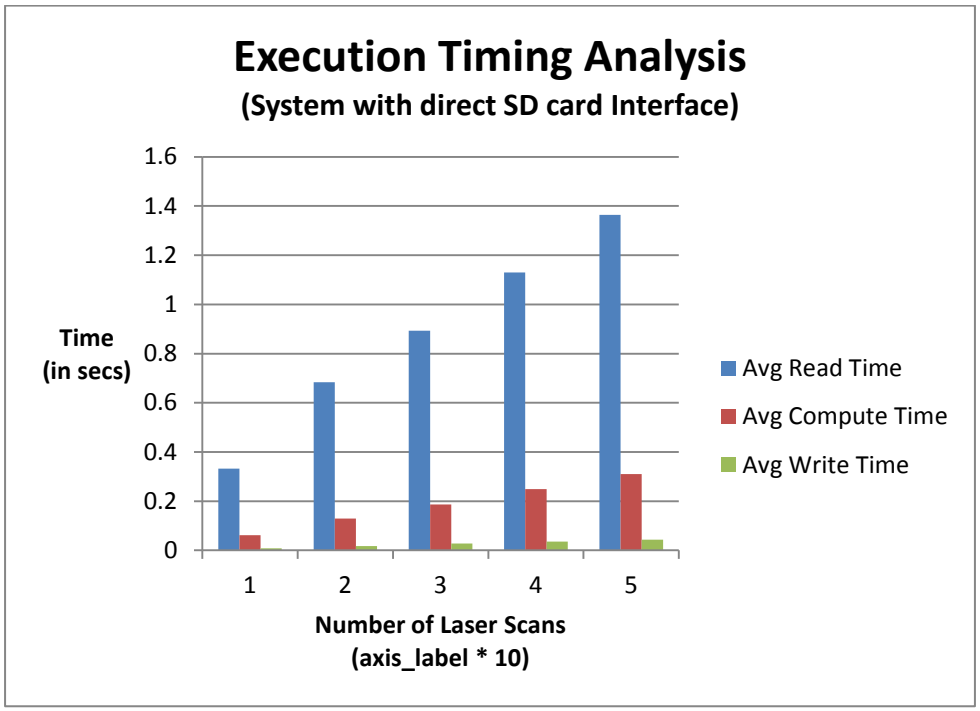


Figure 6.3 Graphical representation of read time, compute time and write time for system with direct SD card interface

From the results it is clear that this design approach reduces the read time compared to the initial system design. The number of memory read, writes and data transfers are minimal compared to the other two designs. Also the design complexity is reduced since there is no SDRAM controller and DMA controller. Even though the speed is increased compared to the initial design, the overall performance of the system is slow if the same bridging algorithm is run on a system running on Windows OS using a GHz processor. The reason for this is that the speed of the NIOS II Processor and the IP core used to send commands to SD card is 50 MHz only.

6.2.4 Execution timing analysis for NIOS II processor and On-chip memory access

Each of the designs discussed in the previous sections were successfully able to compute the bridge mode values from free mode data stored on the SD card. All three designs show high read time because of the SD card memory access even though there are some variations in each design. A different approach is necessary for faster read access. This approach uses the same hardware design as in section 6.2.3 but the timing is calculated with a different technique. This analysis was carried out assuming that Roline laser interfaced to the NIOS II processor would store the free mode data on the on-chip memory. Data points for a single scan are read directly from the on-chip memory location instead of the SD card. The bridging algorithm then computes the bridge mode values and writes it back to the SD card. This process is repeated for varying number of laser scans from 10 to 50.

Table 6.10 show the total execution time of bridging algorithm for different number of scans for reading the free mode laser data from the on-chip memory location

Table 6.10 Total execution time and total number of clock ticks for system with direct on-chip memory access

Number of Scans	**Number of Clock Cycles**	**Total Execution Time (in sec)**
10	3542480	0.0708496
20	7427990	0.1485598
30	10842230	0.2168446
40	14377800	0.287556
50	17922340	0.3584468

Table 6.11 shows the breakup of total execution time of the bridging algorithm including reading of laser free mode data, computing the bridge mode values, and writing the values to an on-chip memory for a different number of scans

Table 6.11 Break-up of total execution time for read, compute and write for system with direct on-chip memory access

Number of Scans	Read time (in sec)	Compute time (in sec)	Write time (in sec)
10	0.001795	0.060919	0.008136
20	0.003579	0.1278762	0.0171042
30	0.00526	0.184252	0.0273332
40	0.006874	0.245046	0.035636
50	0.008701	0.3059404	0.0438052

Figure 6.4 shows the graphical representation of read, compute and write time for the NIOS II processor system for accessing the free mode data from on chip memory and writing the bridge mode values to SD card.

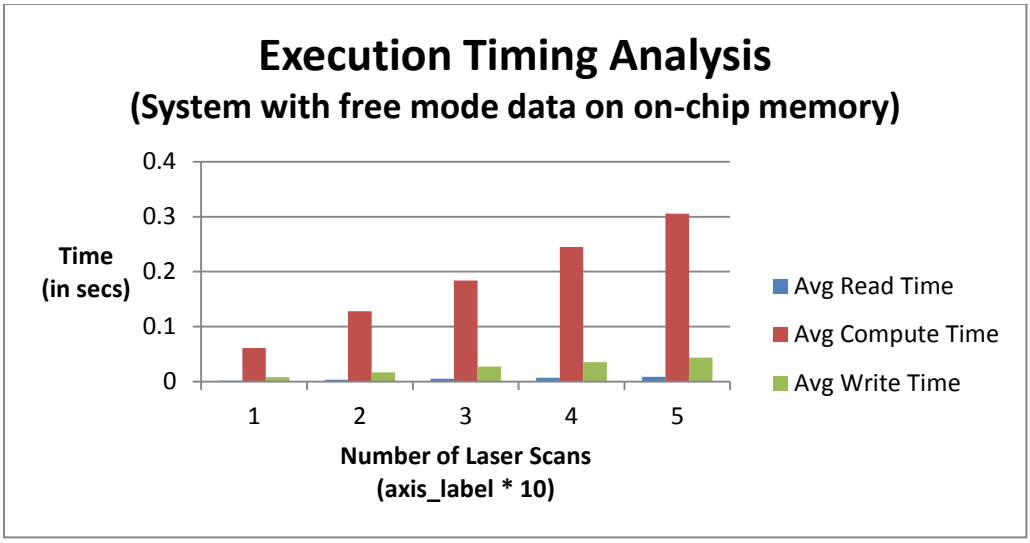


Figure 6.4 Graphical representation of read time, compute time and write time for system with direct on-chip memory access

This analysis shows that the read time is decreased to a great extent because a set of data points are now available on chip memory and the clock latency in low. There is no overhead of reading the data from SD card since it has been assumed in this design that Roline laser interfaced to NIOS II processor system will write free mode data to on-chip memory. Initial approach of reading the free mode data stored on the SD card for computing bridge mode

values would be a time consuming task. The Bridging algorithm can now read the free mode data from on-chip memory and compute the bridge mode values. The values are then written to the SD card. The write time is comparable with the other designs. The final design approach may even compute the profile using the profiling algorithm and write the profile values to SD card.

The next chapter compares the four designs in terms of resource usage, execution timing and also provides a brief idea for the possible future work that can be done on this embedded NIOS II system to improve the performance and achieve enhanced speed.

CHAPTER 7

SYSTEM IMPLEMENTATION COMPARISON AND CONCLUSION

7.1 System Implementation Comparison

The three designs discussed in Chapter 5 were implemented on the DE2-115 development board and a comparison of FPGA resource usage was made. Table 7.1 shows the comparison.

Table 7.1 FPGA resource usage comparison for three designs

FPGA Resources/Design Type	**Design1 (With DMA)**	**Design2 (Without DMA)**	**Design3 (Direct SD Card)**
Logic Cells	6679	5293	4704
Dedicated Logic Registers	3630	3126	2789
I/O Registers	68	68	0
Memory Bits	1924480	1707392	1706432
M9Ks	251	217	217
DSP Elements	4	4	4
Pins	62	62	5
LUT-Only LCs	3049	2167	1915
Register-Only LCs	665	642	583
LUT/Register LCs	2965	2484	2206

Performance comparison of the three systems in terms of a number of clock cycles consumed to execute the bridge algorithm was made for a different number of laser scans for a given set of data. Table 7.2 shows this comparison. The table also includes the analysis results of number clock cycles used for system (Design 4) in which the free mode data is assumed to be available on the on-chip memory directly.

Table 7.2 Comparison of number of clock cycles to execute the bridge algorithm for three designs.

Number of Scans	Design 1 (With DMA)	Design 2 (Without DMA)	Design 3 (Direct SD Card)	Design 4 (System with on-chip memory)
10	21349150	20572310	20101830	3542480
20	44242600	42398190	41541940	7427990
30	59185230	56483020	55378260	10842230
40	75696100	72162840	70713800	14377800
50	92327820	87836680	85864770	17922340

Figure 7.1 shows the graphical representation of comparison of clock cycle utilization for the different design approaches. The design approach in which the free mode data will be available on-chip memory directly will have the least clock cycle consumption.

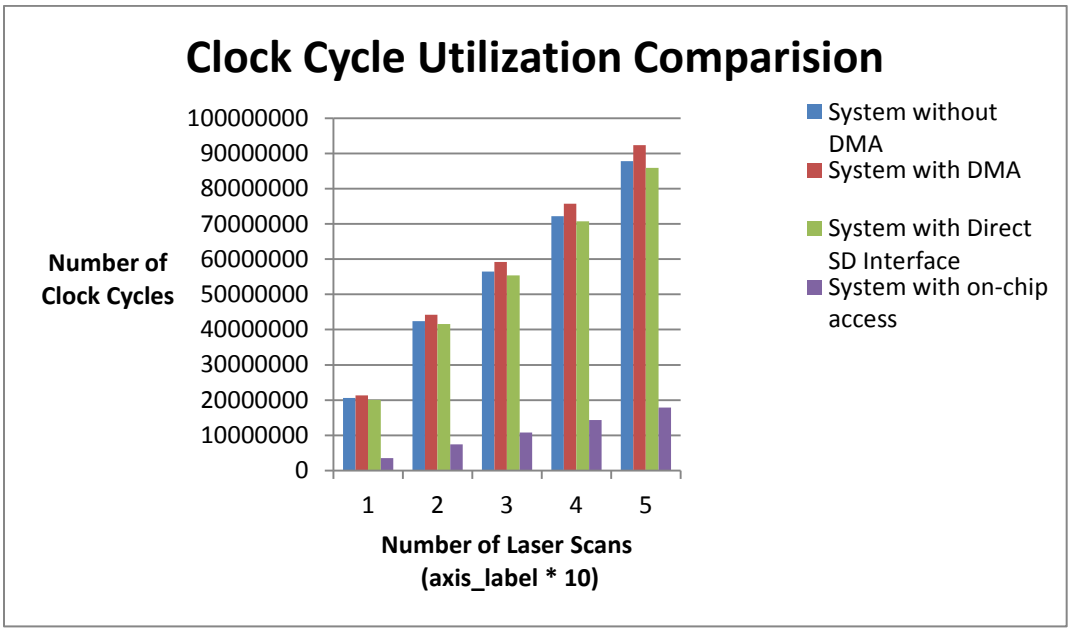


Figure 7.1 Graphical representation of clock cycle utilization comparison

Table 7.3 shows the comparison for total time taken by the system to computing the bridge mode values including reading the free mode data from SD card memory, computing the bridge mode value using bridging algorithm and then save the bridge mode values in text file on SD card. The table also includes the analysis results for system (Design 4) in which the free mode

data is available on the on-chip memory and time taken by the system to compute the bridge mode values.

Table 7.3 Comparison of execution time taken for bridging algorithm to compute the bridge values for all the four designs (all time units in seconds)

Number of Laser Scans	Design 1 (With DMA)	Design 2 (Without DMA)	Design 3 (Direct SD Card)	Design 4 (System with on-chip memory)
10	0.4248984	0.4457164	0.4041258	0.0706736
20	0.8808088	0.9260236	0.836273	0.1481152
30	1.1759644	1.2391028	1.115673	0.2160814
40	1.5053102	1.5886554	1.4231784	0.2868654
50	1.834115	1.9351358	1.7270286	0.3573028

The break-up of total execution time into read time, compute time and write time for each design are calculated as per the formulas mentioned in the appendix A.

The graphical representation for comparison of read time for the different design approaches is shown in figure 7.2

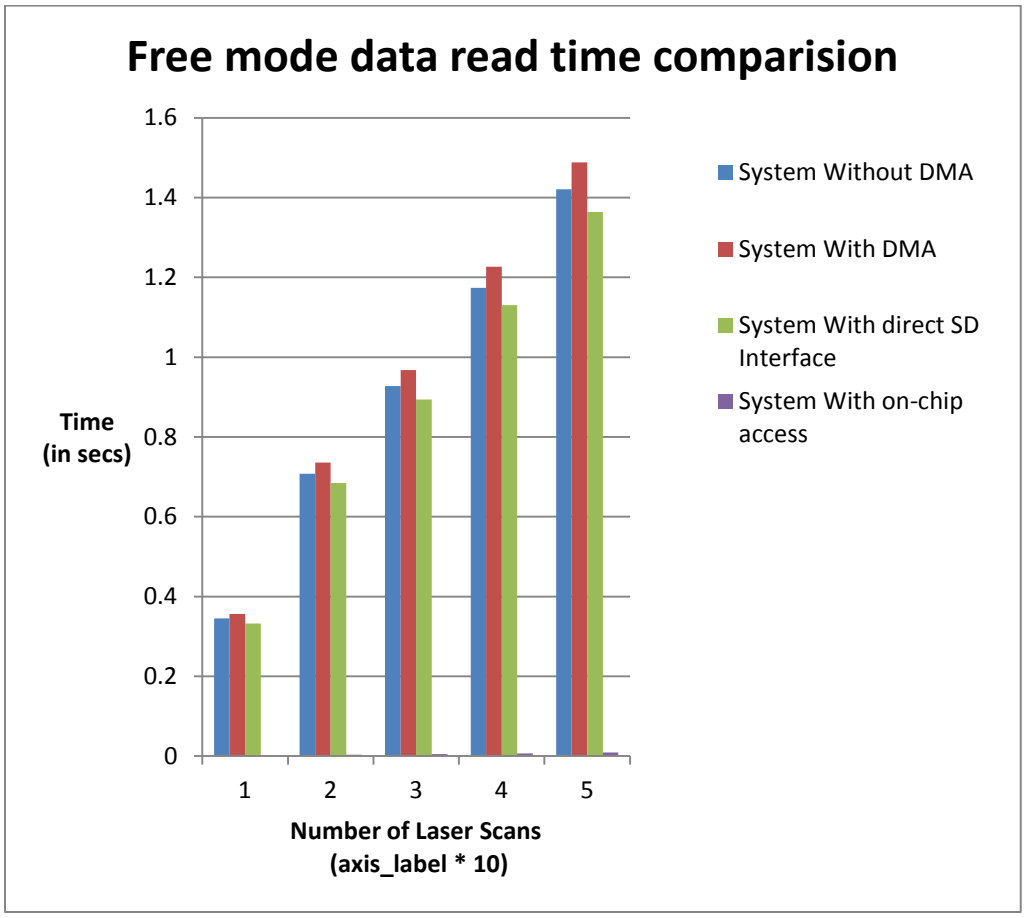


Figure 7.2 Graphical representation of free mode data read time comparison

From the graph in figure 7.1.1, it is clearly visible that the read time is minimum compared to other designs if we can follow the design approach in which the processor can directly read free mode data from on-chip memory. It would be a good design if the Roline laser can be interfaced to a processor such that it can write free mode data to on-chip memory. This design approach indicates that the NIOS II processor system can be used to compute the bridge mode values at a faster rate. The other designs read the free mode data from SD card for every scan. The method of accessing the SD card and its access time results in high read time.

The graphical representation for comparison of write time for the different design approaches is shown in figure 7.3

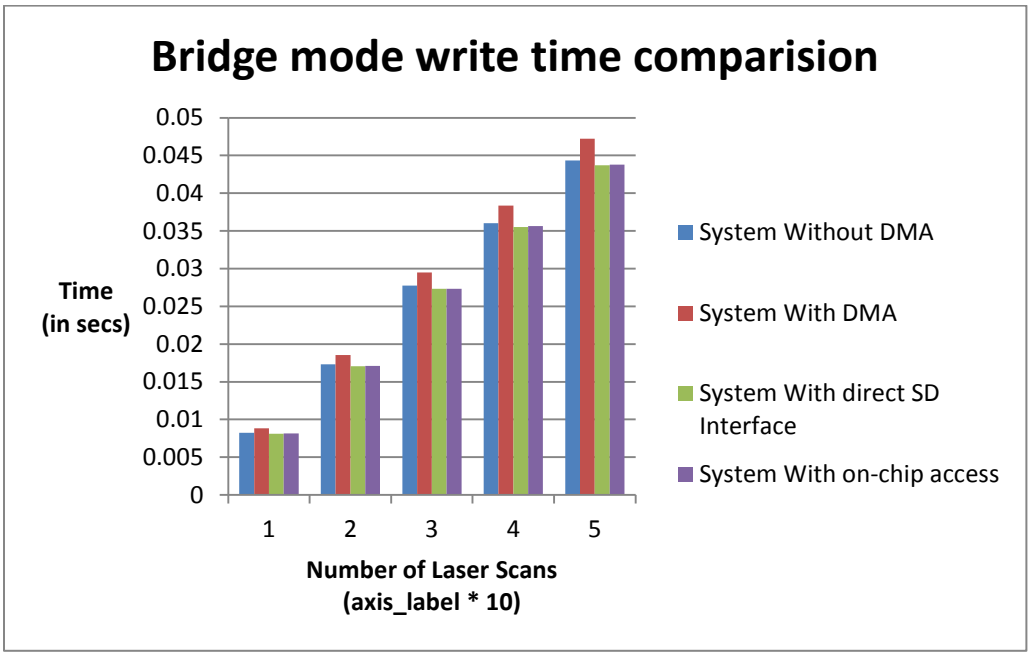


Figure 7.3 Graphical representations of bridge mode data write time comparison for different design approaches

From the graph in figure 7.3, it is clear that the write time is comparable for each of the designs. It is necessary to store the bridge mode values for further processing in profiling algorithm. Therefore the data has to be written to SD card. If design changes can be made such that once the Roline laser and DAQ modules are interfaced to NIOS processor system, the profile can be computed and only the profile values can be directly written to the SD card.

From the design comparisons made, it can be concluded that the design in which the SD card is interfaced directly to the processor has minimal resource usage, and if the free mode data is available on chip directly, then the read time for free mode data would be less. The principal reason for this is the reduced time access from on-chip memory and reduced number of writes with the direct SD card interface. Using DMA has certainly reduced the load on the NIOS II processor but the processor has to wait until a set of data points has been transferred to on chip memory before using it to compute the bridge mode results. The NIOS II processor can

compute the bridge mode values while the DMA is transferring if support is provided for the design for using a real time operating system.

7.2 Conclusion

Embedded NIOS II processor based system was designed to implement bridging algorithm to preprocess free mode data of laser stored on SD card and save the bridge mode values to SD card memory. This system can be extended to interface the Roline laser and DAQ modules to the processor and compute the profile in real time.

The NIOS II processor can run at maximum frequency of 185 MHz and the IP core used to interface the SD card can run at 50 MHz. To improve the system performance and efficiency the system should target using a 1 to 2 GHz processor. Development boards like the Kontron nanoETXexpress-SP COM module based on Intel Atom Z510 1.1 GHz CPU or the Beagle Board based on the TI ARM Cortex A8 processor running at 1 GHz are available in the market and can be used for design enhancements to increase the processing speed. Also a Serial Peripheral Interface (SPI) can be implemented to provide an interface to the SD card to write a block (usually 512 bytes) of data directly.

APPENDIX A

DETAILED ANALYSIS OF READ TIME, BRIDGE MODE VALUE COMPUTE TIME AND WRITE TIME FOR DIFFERENT SOPC DESIGNS

1. **Total execution timing analysis for NIOS II processor and memory Interface without using DMA**

Total Read Time = Read time for laser free mode data from SD card byte wise to SDRAM + Read time from SDRAM memory to data memory section of processor.

Total Compute Time = Time taken to compute bridge values from free mode data available in data memory section.

Total Write Time = Write time for bridge mode values to SDRAM memory + Write time for SDRAM memory to SD card byte wise.

2. **Total execution timing analysis for NIOS II processor and memory Interface using DMA**

Total Read Time = Read time for laser free mode data from SD card byte wise to SDRAM +DMA transfer time from SDRAM to separate on-chip memory in design + Read time for on-chip memory to data memory section of processor.

Total Compute Time = Time taken to compute bridge values from free mode data available in data memory section.

Total Write Time = Write time for bridge mode values to separate on-chip memory in design + DMA transfer from on-chip memory to SDRAM memory + Write time for SDRAM memory to SD card byte wise.

3. **Total execution timing analysis for NIOS II processor and Direct SD card memory Interface based system**

Total Read Time = Read time for laser free mode data from SD card byte wise to data memory of processor

Total Compute Time = Time taken to compute bridge values from free mode data available in data memory section.

Total Write Time = Write time for bridge mode values to data memory + Write time for data memory to SD card byte wise.

4. **Total execution timing analysis for NIOS II processor and On-chip memory access**

Total Read Time = Read time for laser free mode data from data memory of processor

Total Compute Time = Time taken to compute bridge values from free mode data available in data memory section.

Total Write Time = Write time for bridge mode values to data memory + Write time for data memory to SD card byte wise

REFERENCES

[1] Pong P. Chu, Embedded SoPC Design with NIOS II Processor and Verilog Examples.

[2] M. W. Sayers and S. M. Karamihas, The Little Book of Profiling. University of Michigan, Ann Arbor, Transportation Research Institute,1998.

[3] 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

[4] RoLine 11x0 User’s Manual, LMI Technologies, Inc, 2010.

[5] Cyclone IV Device Handbook Volume 1 from Altera’s website documentation section.

[6] NIOS II Processor Reference handbook from Altera’s website documentation section.

[7] NIOS II software developer’s handbook from Altera’s website documentation section.

[8] Embedded peripherals IP user guide from Altera’s website documentation section.

[9] Embedded design handbook from Altera’s website documentation section

[10] DE2-115 development board datasheets from Altera’s website documentation section.

[11] FPGA Embedded Processors: Revealing True System Performance, Embedded Systems Conference San Francisco 2005

[12] Akshay Joshi, “Design and development of a general purpose embedded acquisition system for transportation applications”, July 2011

BIOGRAPHICAL INFORMATION

Vinay Ashi was born in Karnataka, India in 1985. He completed his bachelor's in Electronics and Communication engineering from Visvesvaraya Technological University, India in 2007. He then worked for three years as project associate at Cognizant technology solutions private limited. He came to University of Texas at Arlington to pursue his Master's degree in Electrical engineering. His areas of interest include real time embedded system design and System on programmable chip development.