Cost effective Design and Development of Hyper Spectral camera using CMOSIS sensor

A Thesis by

Muhammad Haseeb Tayyab

2013-MS-ENR-06

Supervisor

Dr. Faheem Gohar Awan

Centre for Energy Research and Development

University of Engineering and Technology, City Campus, Lahore

Session 2013

Cost effective Design and Development of Hyper Spectral camera using CMOSIS sensor

Thesis submitted at the University of Engineering and Technology Lahore (City Campus) for partial fulfillment of the requirement for the Degree of

Master of Science
In
Energy Engineering

Dr. Faheem Gohar Awan	
(Internal Examiner)	(External Examiner)
Dr. Nadeem Ahmad Mufti	**Dr. Waqar Mahmood**
(Dean)	(Chairman)

Centre for Energy Research and Development
University of Engineering and Technology, City Campus, Lahore

December 2017

Table of Contents

List of Figures

List of Tables

Acknowledgements

In the name of Allah, the Most Gracious and the Most Merciful

All praises to Almighty Allah Who is a supreme and ultimate ray of hope in distress. I would like to express my earnest appreciation to my supervisor Dr. Faheem Gohar Awan for being a constant and result oriented source of knowledge throughout my research project. His guidance introduced me the innovative means of thinking about technology. Without his help and advices this research and thesis could not have been completed.

Besides my supervisor, I would like to thank Mr Nasir Mehmood ,Muhammad Faizan Aziz and M. Mudassar who helped me a lot in my research work.

Dedication

To my parents…

Abstract

Despite the rapid growth of Hyper spectral cameras in recent years, its components including slits, gratings, collimators and focus lenses are still very expensive and large in size as compared to those on which Fabry perot filters integrated sensors used. Cameras available in market with integrated filters are very expensive in order to reduce costs and save more space, integration technology is required to develop by using CMOSIS sensor. The major parts of Hyper spectral camera are slits, gratings, collimator and focus lenses which are used to break down the light into hundreds of bands, collimate it and focuses on the sensor of Hyper spectral camera. These parts of camera are very expensive, bulky and heavy. It should be cheaper and cover less space without compromising in efficiency. This research work proposes the development progress and optimization of Hyper spectral camera by using CMOSIS sensor containing integrated Fabry perot filters. In this research work, we aim to design small size and cost effective Hyper spectral camera with high resolution.

Key Words: Hyper spectral; camera; CMOSIS sensor; Fabry Perot filter.

Acronyms and Abbreviations

HSIP	Hyper spectral Imaging System
LVDS	Low Voltage Differential Signaling
CMV	CMOSIS Vision
AFE	Analogue Front End
SPI	Serial Parallel Interface
FPGA	Field Programmable Gate Array
DCM	Digital Clock Manager
SR	Shift Register
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MPPT	Maximum Power Point Tracker
MPP	Maximum Power Point
PWM	Pulse Width Modulation
SCR	Silicon Controlled Rectifier

1 Introduction

1.1 Hyper spectral Camera

History of Digital camera instigated at Jet propulsion laboratory by F.Lelly, and idea was utilized by Willis Adcock's in his first filmless camera in 1972. Before this, Analog cameras use films to produce images. Digital camera came into being with the concept of digitizing images on scanners with arrays of sensor elements. Digital camera invention takes usage of camera to new heights from using cameras in our daily life to Remote Sensing Satellites (RSS). Remote Sensing Satellites use technology of acquiring information of earth surface (land or sea) and atmospheric conditions by using the sensors onboard. Remote sensing cameras consist of passive and active sensors[1]. Passive sensors are used to measures energy which is naturally available in form of visible wavelengths from sun or emitted through thermal infrared source. Passive sensors can only acquire energy when it is naturally available as it cannot emits light whereas active sensors produce radiation by their own source , emits the light towards the target and investigates its response. Remote Imaging with passive sensors is further of three types, panchromatic imaging, multispectral imaging and hyper spectral imaging. Panchromatic imaging consist of single channel detector that gather the information of broad wavelength in visible region, information consists of physical quantity of brightness of the target[2].

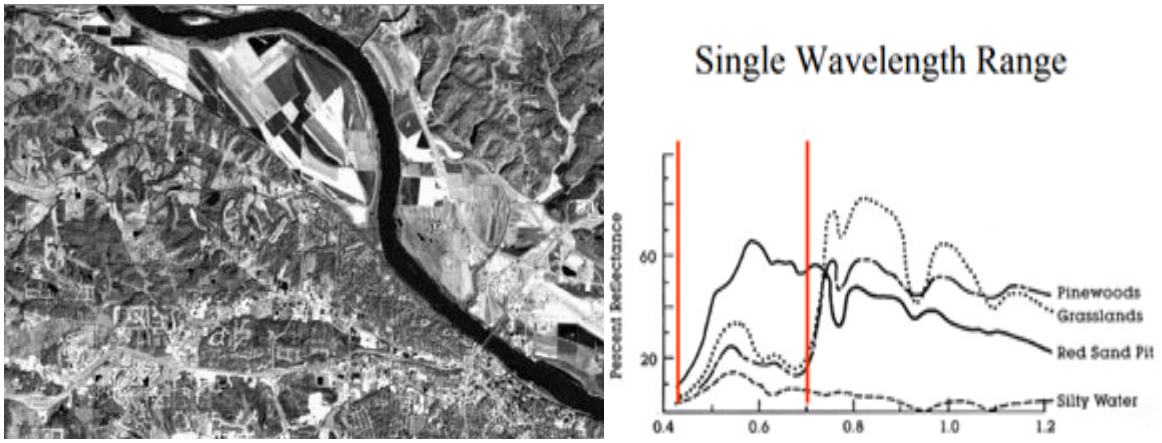


Figure 1.1 Panchromatic Imaging

Panchromatic imaging is generally rendered in black and white. Multispectral detectors contain few spectral bands; each spectral band channel contains information of narrow wavelength bands. Multispectral imaging consists of RGB (Red, Green, and Blue) bands with in visible regions of wavelength and it gathers the information of brightness and color of the target.

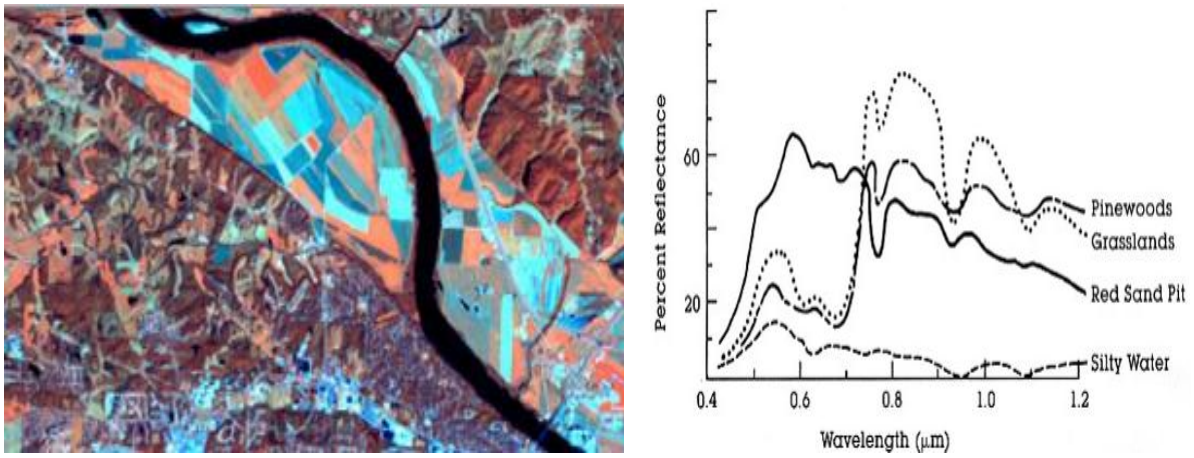


Figure 1.2 Multi Spectral Imaging

Hyper spectral sensors collects image data with hundreds of small narrow bands ,with large amount of each spectral band information it generates continuous spectrum for each image cell, Then this information is post processed by some atmospheric or sensor calibration and correction software, e.g. Adjustments like sensor, atmospheric and terrain effect. Spectral response then compared with spectrums of different vegetation's and minerals available at USGS library to recognize the map surface.

In recent years, Hyper spectral cameras have attracted a lot of attention in the field of RSS (Remote Sensing Satellite) as the telescope of RSS, Hyper spectral camera is main part of the satellite that can produce real time imagery during its continues rotation. Panchromatic and Multi spectral cameras are old versions that can produce only single band (black and white) and multi bands (Red, Green, Blue) images simultaneously, whereas Hyper spectral camera can produce image of hundred or more bands that can break the light from visible to infrared region , properties of different materials can be detected easily[3]. Hyper spectral imaging is a combination of two technologies, Real time imaging and spectroscopy, Spectroscopy is process of gathering information of the light emitted and reflected back from the targeted image. It is deviation of energy received in the reflected light with all spectrum

of wavelength. Spectroscopy in remote sensing is study of light gathered through earth surface as shown in Figure 1.3.

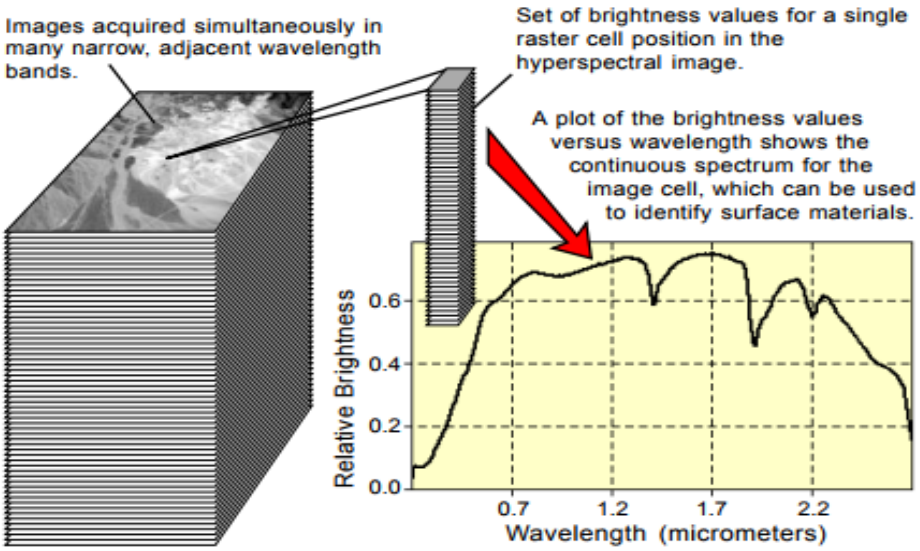


Figure 1.3: Hyper spectral Imaging

Hyper spectral camera has following main components like focusing mirror, slits, collimator, grating lens optics and detector. Internal structure of conventional hyper spectral camera is described in the figure below.

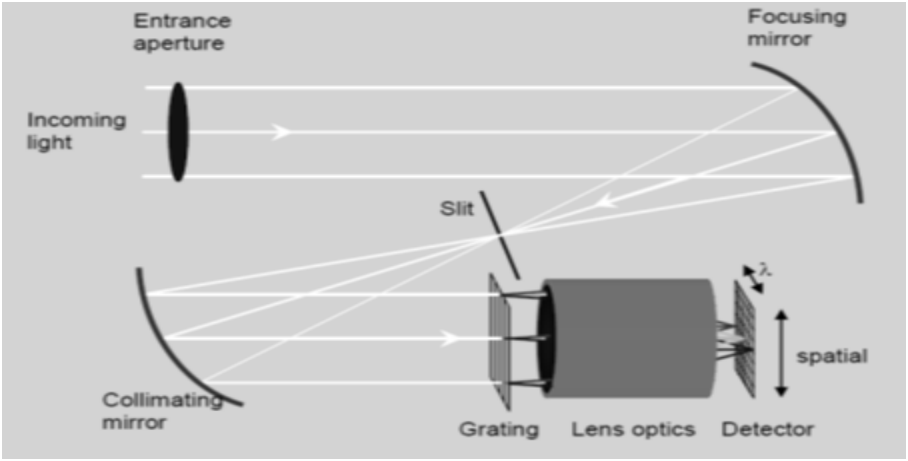


Figure 1.5: Hyper spectral camera optics

Focusing mirror of hyper spectral camera is a fore optics which produces scene of the target and focus it on slit, Slits has number of grooves that passes the narrow lines of the focused scene which then further passes through the collimator. Collimator is a major component of hyper spectral camera that produces parallel beams of rays or radiations these parallel beams of radiations of different wavelengths are then focused on detector[4]. Net effect of optics of hyper spectral camera is to produce parallel beams for each image pixel of the detector.

Sensor has columns of detectors on the array which produce a slice of a hyper spectral image, with spectral information in one direction and spatial information (image) in the other.

Recent progress in hyper spectral imaging is spectral filters which are deposited directly onto a sensor chip. Advancement is made on sensor by using linear variable filter concept which infect in field of optics allows the placement of spectral filters on the image sensor, this wafer level filter integration can influence on the high end equipment which is used for modern image sensor production. It eliminates the requirement for other assembly of optics, alignment of different steps and other glue layers. The filter layer are in microns and very thin as compared to gratings used before.

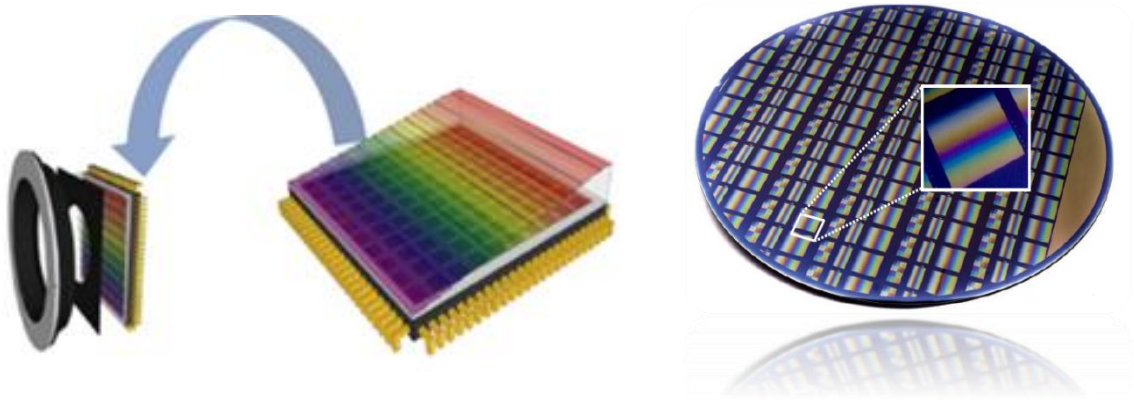


Figure 1.6 Integrated filters on sensor chip

1.2 Fabry Perot Filters

Fabry perot filters in the field of optics have same working principle as electronic or passive (RLC) filters have in electronics. RLC filters distinguish the required narrow frequency from bands of frequency applied to the system whereas fabry perot filters in the same way determines the required wavelength of the light to focus on corresponding pixel of the sensor from large bandwidth of approaching light[5].

Fabry perot filters are working on the phenomena of interference, diffraction or absorption.

1.3 CMOSIS Sensors

CMOSIS sensors are most advanced and latest technology in the field of optics. It is the most advance image sensor used for the real time imaging. CMOSIS imaging sensor has many advancements that make it most suitable for hyper spectral imaging systems such as:

1. Compactness.
2. Cost Effective.
3. Energy Efficient.
4. Large number of bands.

The main feature that makes it more suitable for imaging e.g. airborne imaging where camera is required to be placed at the bottom of air craft is its compactness. The old versions of the camera required different elements to break the light beam into hundreds of narrow bands or wavelengths and focus it simultaneously on particular pixel of the sensor .e.g. Slits , grating's and focusing mirrors which make camera heavy and bulky[1].

CMOSIS sensor use most modernized Fabry Perot filters to break light and focus desired wavelengths precisely on the pixels of the sensor that eliminates the uses of heavy elements and make it more compact and suitable for imaging

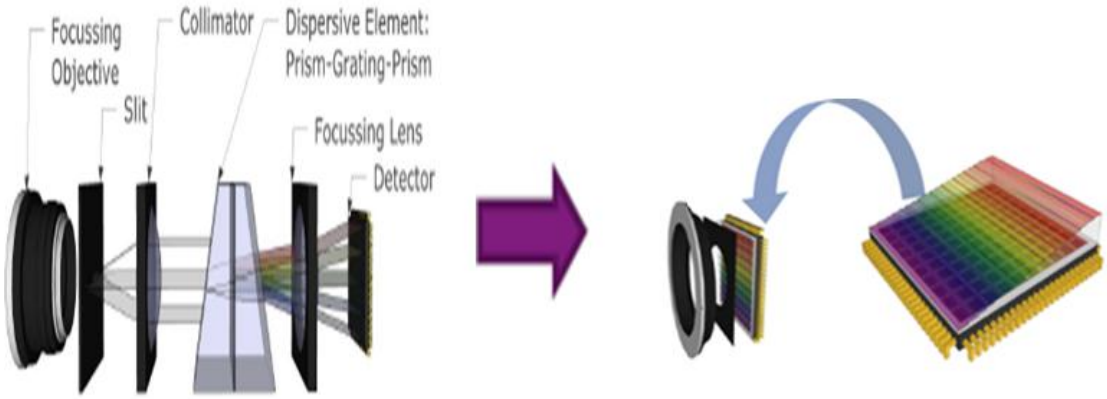


Figure 1.7: CMOSIS Technology

Slits, gratings and focusing mirror and all elements are most expensive so the other versions are very expensive as compared to CMOSIS. Due to CMOSIS sensor only power board and controller contains FPGA and Frame Grabbers are required to produce the image and eliminate the cost of other elements. CMOSIS sensors have different features like global and rolling shutters that increase the field of view of the sensor, High frame rate due to high speed ADC (Analogue to Digital Converters) on a single chip, low noise and high dynamic range.

CMOSIS sensor has 16 LVDS outputs and option to reduce the output to 2, 4 and 8 that reduces the resources used in FPGA and even simple low grade FPGA can be used to process the data , Eliminating the external elements and modes of operation makes it more suitable energy efficient and cost effective.

1.4 Thesis Objectives

Main objective of the thesis research is to produce compact and cost effective Hyper spectral imaging camera by using latest wafer level Fabry-Perot filters integrated on single chip sensor CMOSIS CMV 2000.

- Reducing the extra external optical elements.
- Wafer level filter integrated sensor with high speed ADC to make it compact and cost effective.
- Reduce LVDS output channels to 4 to reduce the resources of FPGA, and use simple low cost FPGA and frame grabber card to reduce power consumption and make it cost effective.

1.5 Thesis Outline

This thesis has been constructed in five chapters. Chapter 1 is of introduction. Literature review has been discussed in detail in chapter 2 that discusses all the existing technologies regarding Wafer level filter integration on a single chip, Hyper spectral camera system, CMOSIS 2000 sensor architecture and programing techniques, Frame grabber card to from the real time image on the device, Buffer IC's . Chapter 3 has been dedicated to methodology that explains the methods adopted to process the data receive from 4 LVDS (Low Voltage Differential Signaling) by low cost FPGA and reduce its resources. FPGA process the LVDS data from the sensor and converts the serial combination of data to parallel and send to buffer IC's to use it in Frame grabber card. Chapter 4 discusses the results and output waveforms. Chapter 0 concludes the thesis and gives research direction for future work in this research area.

2 Literature Review

This chapter provides an summary of the literature of Fabry Perot filters integrated on sensor chip for breaking lights into hundreds of bands and choose the required wavelength on desired pixel simultaneously, CMV 2000 sensor its internal techniques and structure, FPGA and its modeling technique, impact of solar filters on, CameraLink and frame grabber for generation of image on system.

2.1 Fabry Perot Filters Integrated On Sensor

Fabry–Perot Filters are interferometers that produce interference of numbers of multiple reflection in the middle of two thin plates and condition of interference is 2d sinθ= nA, where n is an integer and d is the thickness of the plate The Fabry-Perot interferometer is based on of numbers of multiple reflection in the middle of two thin plates. All wavelengths that are multiple of 2π satisfy the condition[2].

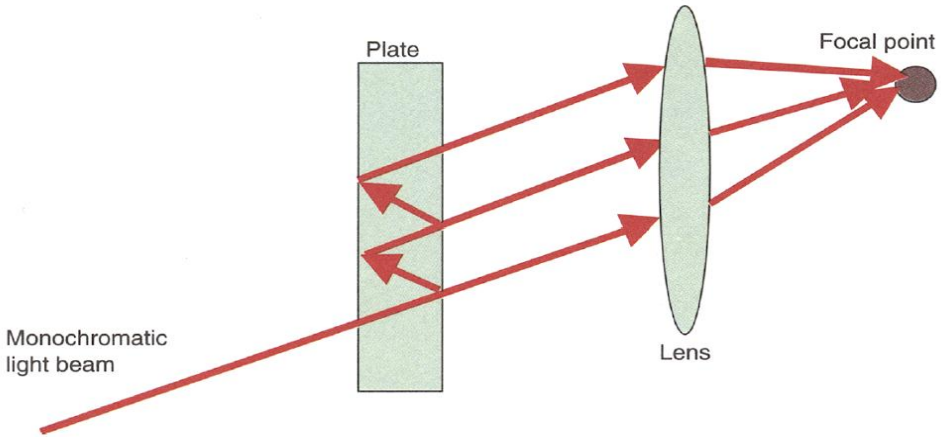


Figure 2.1: Interference of Febry perot filter

Fabry-Perot filters have two shining thin plates which has reflection of about 99% has reflective inner surfaces with the coefficient of 0.94.Due to highly reflective plane plates mounting parallel to each other infinite numbers of parallel beams are generated, these beams are distinguished by each other by the number of runs inside the reflective plates and constructive interference in produced. As the reflections of beams takes palace with each reflection a phase shift of αR is produce and when the beams leaves the interferometer from

right plate a large phase difference between two wavelengths are generated and this phenomena of band separation is explained mathematically.

First Beam:

$$E1 = Eo(\varphi)\sqrt{T}ei^{\alpha T} \cdot \sqrt{T}ei^{-\alpha T}\ e.^{i2\pi\sigma n\frac{a}{cas\varphi}} = \ TEoe.^{i2\pi\sigma n\frac{a}{cas\varphi}}\ ; \ldots\ldots\ldots 1$$

Second Beam:

$$E2 = E1\sqrt{R}e{-}i^{\alpha T}\ e{-}i^{2\pi\sigma\delta}\sqrt{R}e{-}i^{-\alpha R}\ e.^{i2\pi\sigma n\frac{a}{cas\varphi}} = \ TEoe.^{i2\pi\sigma n\frac{a}{cas\varphi}}\ Re{-}i^{(2\alpha R+2\pi\rho\delta)}$$

Sum of these in geometrical series:

$$\frac{E\varphi}{Eo(\varphi)} = \ Te.^{i2\pi\sigma n\frac{a}{cas\varphi}}\left(1 + Re{-}i^{(2\alpha R+2\pi\rho\delta)} + \left[Re{-}i^{(2\alpha R+2\pi\rho\delta)}\right]^2 + \cdots\right) = \frac{Te.^{i2\pi\sigma n\frac{a}{cas\varphi}}}{1 - Re{-}i^{\emptyset}}$$

Where $\emptyset$ is = $2\alpha R + 2\pi\sigma\delta$.

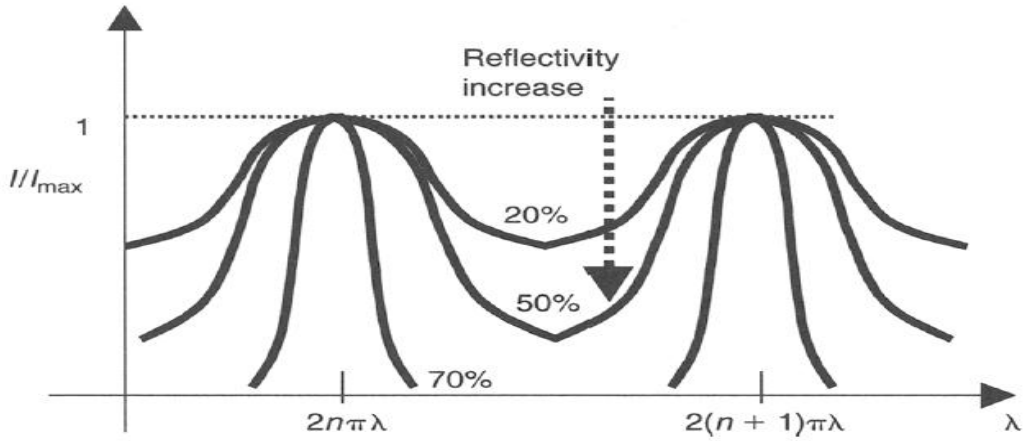


Figure 2.2 Interferometer at maxima

To achieve such constructive wavelengths the reflective surfaces of the plates must be plane, coating of the reflective areas of the plate's must contain reflection co-efficient which is more likely close to one and light focused on the reflective plates should be parallel. At maxima point the intensity is Imax=E^2 [5].

2.2 CMOSIS CMV 2000

The imaging sensor used in hyper spectral camera design with fabry perot filters integrated on chip is CMOSIS CMV2000. CMOS commercial image sensor contains 2048 by 1088 pixels which are shielded up with 2/3 optical inch for vision applications, sensor contains array of thousands of micro level pixels with the size of 5.5µm x 5.5µm which are pipelined with the global shutter commercially available to increase the coverage of the focused target.

The CMOSIS CMV2000 sensor contains sixteen channels to transmit 10 and 12 bits of LVDS (Low Voltage Differential Signaling) data. The sensor has a gain amplifier which can be programed and offset controller that can produce attainable amplification of desired signal of that target and minimum desired offset for required output. CMV 2000 has 16 channel LVDS output and 2 LVDS channels for control and synchronization each channel has 480 Mbps maximum attainable speed which generates 340 frames per second on full resolution and high frame rate, which is obtained when sensor is used in its row-windowing operational mode or row-subsampling operational mode. These modes can be used during Serial parallel SPI interface. On –board sequencer can produce internal exposure and all readout timings [6].

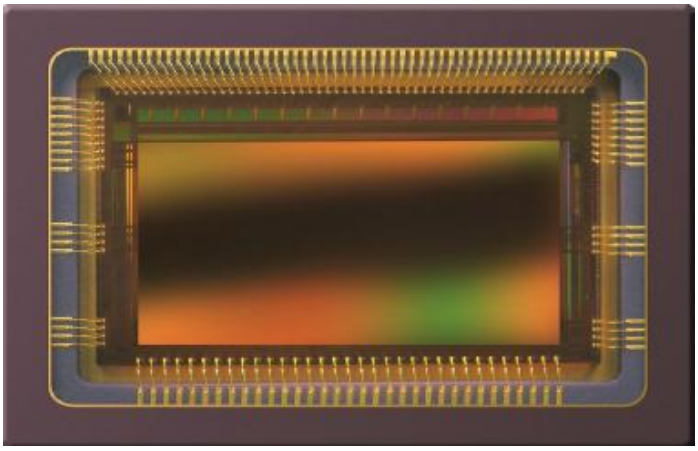


Figure 2.3 CMOSIS Sensor

2.2.1 Features and Specifications

CMV 2000 has 8 different output windows of operation with horizontal and vertical mirroring. It has multiplex able different output channels: 2, 4,8,16 with LVDS control and DDR (Dual Data Rate) LVDS clock and selectable ADC (Analog to Digital) resolution at different frame rate of 10 or 12 bits. It has on-chip temperature sensor and sensor control via SPI (Serial Peripheral Interface).

CMV 2000 has following specifications:

Table 2.1 CMV 2000

Full well charge	13.5 Ke-
Sensitivity	5.56 V/Lux
Dark noise	13e- RMS
Gain	60dB
Dark current	125e-/s
Power consumption	550mW

2.2.2 Sensor Architecture

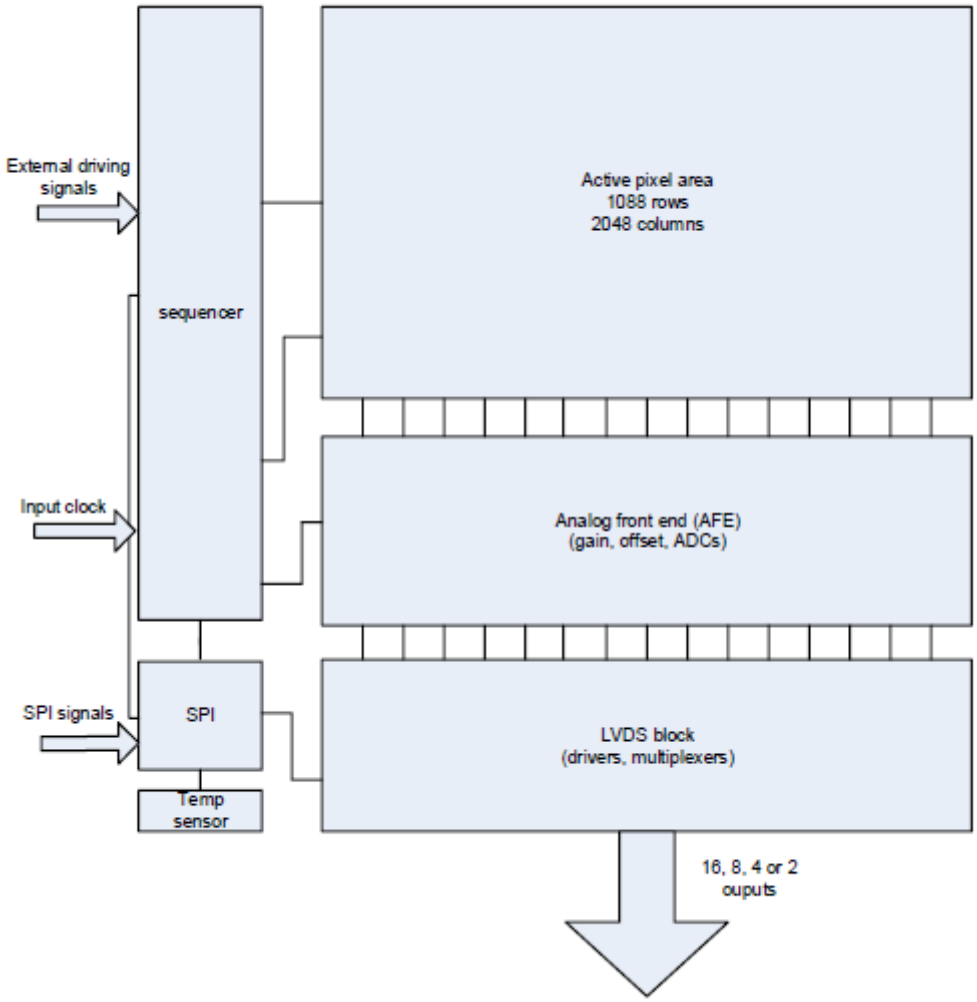


Figure 2.4: CMV2000 Internal Architecture

The internal architecture of CMV 2000 as shows in Figure 2.4, that the internal sequencer uses some external input signals and a master clock to generate the required commanding signals desired for the acquisition of targeted image. The information of image is stored in the micro pixel of the sensor (global shutter) available in active pixel area of the sensor on which the target image is exposed and it has 1088 rows and 2048 columns, all the data stored in the pixels, and required gain is applied by the sequencer programmed through SPI are then read out consecutively row by row. Output of the pixels are in analog form is then fed to Analog front end (AFE) on the sensor which contains ADCs (Analog to digital converters) which concerts the analog output data of imager pixels to digital output with amplification of gain and required offsets if necessary. Digital output from ADCs is then read on multiple LVDS channels, each channel has 128 columns and rows in Y-direction for data transfer which is chose able through row decoder for the multiple windowing operation. Control registers are uploaded using SPI interface which are used for the programing of the sensor to generate the control signals and synchronizing operation [3]. Array of pixels in sensor is consists of 2048 x 1088 no of pixels with the size of approximately 5.5μm are designed to achieve maximum level of sensitivity with minimum noise. Analog Front with two important parts, amplifier and ADC block. Amplifier applies the required gain to pixel signal and ADC converts the analog signal to 10/12 bits digital value. A digital offset is also programmed to digital output by using SPI interface. LVDS block converts the ADC data to required serial LVDS data at maximum attainable frequency of 480Mbps on 18 no of LVDS output pairs that has 32 pins for output. LVDS block output contains 16 channels for data and 2 channels for control signal and DDR (Dual Data Rate) clock pulse which has half of the frequency of the frequency of output signal. Data on control channel also contains 10 to 12 bits of data containing words transfer from sensor. Sequencer generates required control signals of the sensor with few input signals which are programed by using SPI interface, it is used to load control register to sequencer containing data which is used for controlling the operations of the sensor. Temperature sensor produces 16 bit output controlled by SPI interface. Clock (CLK_IN) describes rate of output of the sensor as input clock is ten to twelve times slower than output data rate e.g. maximum output data rate of CMV 2000 is 480Mbps so input clock will be 48MHz.

2.2.3 Frame Rate Control

Frame transfer rate of the sensor depends on two major factors, one is exposure time and other is read out time. If the readout time of the sensor is higher than the exposure time and sensor is functioning on default settings with full resolution 10-bits image at 48 MHz at 16 outputs. This determines that the frame rate will only depends on exposure time. Read out time is mainly depends on the following parameters, Output clock speed, ADC bit output mode, Number of lines to be read and number of LVDS output [7].

With 16 output channels the frame rate with constant parameters are 340 FPS (Frames per second). Read time is sum of F-O-T (frame overhead time) and read time shown in Figure 11.

$$FOT = (fot_{length} + \left(2 * \frac{16}{outputs}\right) * 129 * master\ clock \ldots\ldots 1$$

Read out time :

$$Read\ out\ time = \left(129 * master\ clock * \frac{16}{outputs}\right) * nrlines..2$$

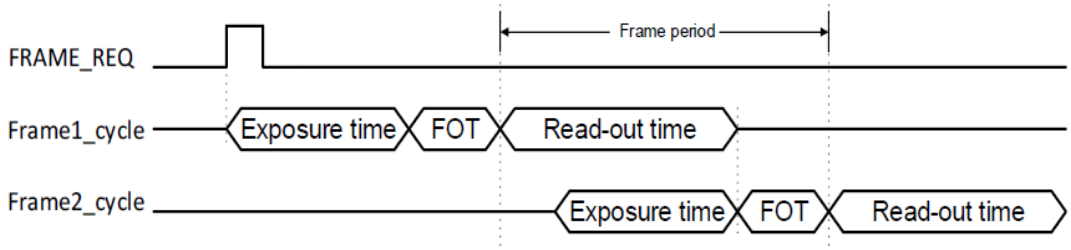


Figure 2.5 Frame Rate

2.2.4 SPI Programming

SPI programing is to write the control information to the on-board registers of the sensor. Control information can be written or read on the registers in simple serial interface (SPI).

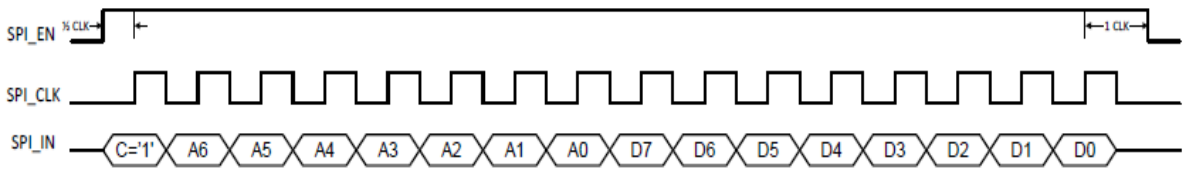


Figure 2.6 SPI Write Control Information

Details of the control information written on the registers are shown in above figure Fig 2.6. CMV 2000 writes data on rising edge of SPI clock signal. Clock signal of (SPI _CLK) operates on frequency of 48 MHz, SPI_EN enable signal required to be high for half of the cycle before first data bit sent and high for a cycle after last data bit. Total control information has 16 bits in which first bit will determine control bit, which indicated zero ‘o’ read or ‘1’ write on SPI interface [7]. Other seven bits are address bits and last eight bits are data bits and are written on MSB first pattern.

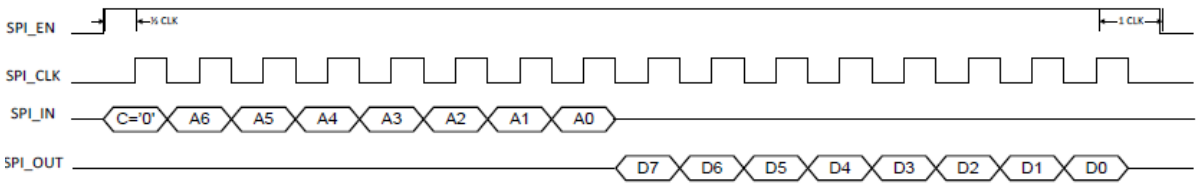


Figure 2.7 SPI Read

SPI read has following procedure as shown in Fig 2.7, Control bit or first bit will be zero for read operation of SPI interface, after the address info is received on MSB format and then data is received on same pattern.

2.2.5 Reading out the Sensor: LVDS Data Outputs

CMV 2000 image sensor provides the image data in form of LVDS (Low Voltage differential signaling).CMV 2000 has 18 LVDS outputs in which 16 outputs are for data output channels and other 2 are used for controlling and synchronizing the image data with other surrounding subsystems.2 pins are used for differential signals of each LVDS output so total 36 pins are available for data and controlling that provides 10 to 12 bits of data each [8].

Control channels provides the control information, that weather the data received is valid or not and clock channel provides clock pulse signal to synchronize the data with frequency half of the output data rate frequency e.g. if Output data rate frequency is 480 MHz the clock frequency will be 240 MHz

2.2.6 READ OUT TIMING

Channels containing the imaging data consists of 128 pixels, each pixel consists of 10 bits of data or 12 bits of data. Pixel period of data bit stream is equals to period of complete one master clock and OH (Over Head) time exist between the two 128 pixels burst of image data, OH has same time as one pixel readout of 10 to 12 bits of data.0000000

2.2.6.2 16 Output Channels

If 16 output channels are used then complete row of image date is sent on a single row of 128 pixel period with maximum frame rate of 135 frames per second.

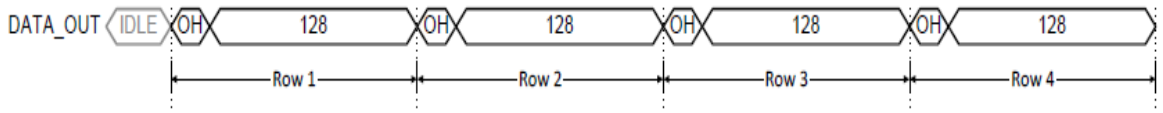


Figure 2.8: 16 Channels Output

2.2.6.3 8 Output Channels

If 8 channels output used for transmitting data then (2 x 128) + (2 x 1) master clock periods used . The maximum frame rate is reduced half as compared to 16 channels.

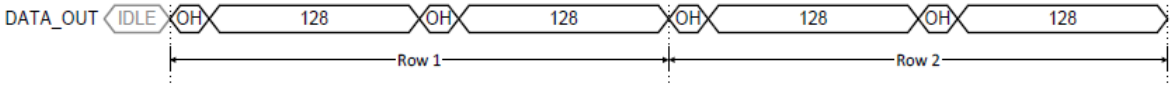


Figure 2.9: 8 Channels Output

2.2.6.4 4 Output Channels

If 4 channels output used for data then, (4 x 128) + (4 x 1) master clock periods used and frame rate reduced with a factor 4.

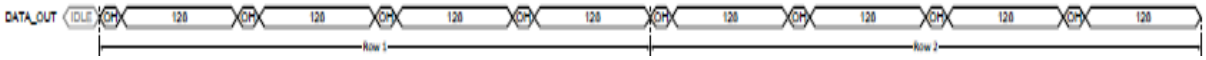


Figure 2.10: 4 Channels Output

2.2.6.5 2 Output Channels

One row takes (8 x 128) + (8 x 1) master clock periods.

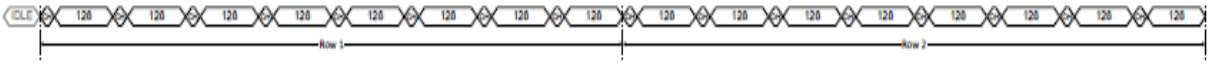


Figure 2.11: 2 Output Channels

2.2.7 PIXEL REMAPPING

Pixels containing the image data are read at different time and channels , so the read out time of the channels determines the no of output channels.

2.2.7.1 16 Output Channels

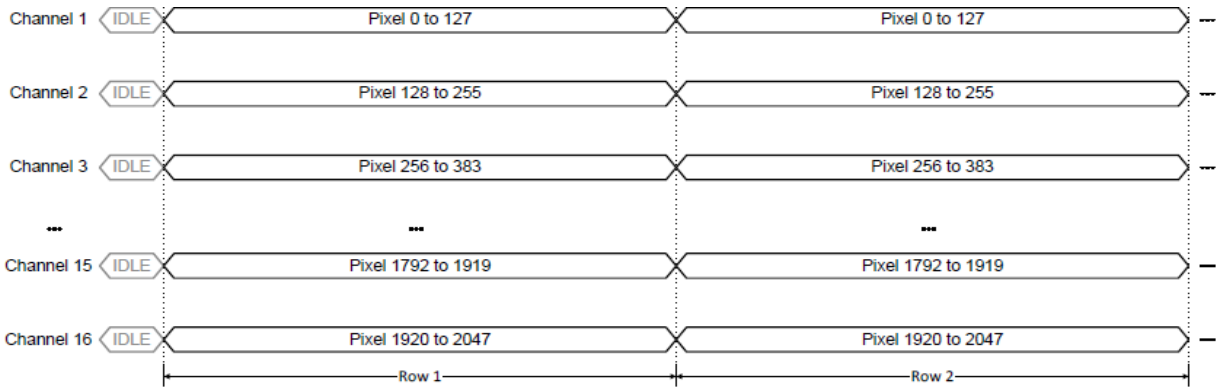


Figure 2.12: 128 pixel burst in single row

Fig 18 shows when 16 output channels are used to send the data on LVDS outputs then , complete data of the single image row will be send through on a 128 burst of pixels [9]. All the 16 output LVDS channels contains data along with the 2 additional LVDS data channels that contain control and clock data for controlling and synchronize function.

2.2.7.2 8 Output Channels

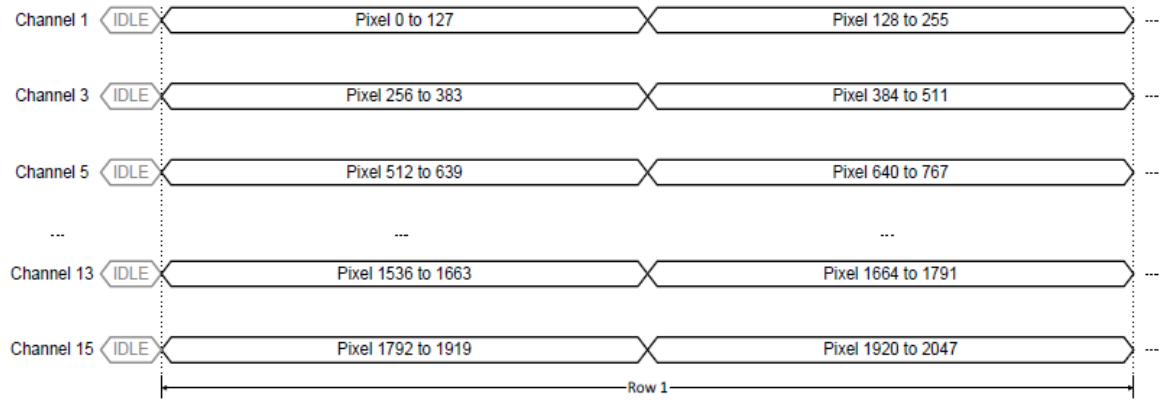


Figure 2.13 128 pixel burst in two rows

Fig 19 shows when 8 output channels transmit the data on LVDS outputs then, complete image data row will be transmit trough 2 sets of 128 burst of pixels. All the 8 output LVDS channels contains data along with the 2 additional LVDS data channels that contain control and clock data for controlling and synchronize function.

2.2.7.3 4 Output Channels

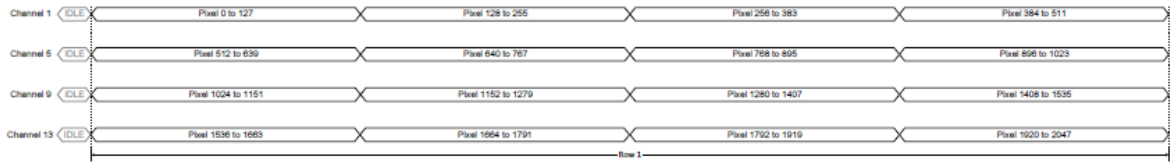


Figure 2.14: 128 pixel burst in four rows

Image data row will be transmit trough 4 sets of 128 burst of pixels. All the 4 output LVDS channels contains data along with the 2 additional LVDS data channels.

2.2.8 Control Channels:

CMV 2000 has only one dedicated output LVDS channel that contains the control information of 10 bits -12 bits. Two pins used for differential signal [10]. Control signal determines the validation of the data transmitted through the output channels provides the information to synchronize the valid data with the clock signal.

Control signals generated by the image sensor CMV 2000 are shown below in Fig 2.15 from bit 0 to bit 11 all 12 bit control signal in LSB format.

Bit	Function	Description
[0]	DVAL	Indicates valid pixel data on the outputs
[1]	LVAL	Indicates validity of the read-out of a row
[2]	FVAL	Indicates the validity of the read-out of a frame
[3]	SLOT	Indicates the overhead period before 128-pixel bursts (*)
[4]	ROW	Indicates the overhead period before the read-out of a row (*)
[5]	FOT	Indicates when the sensor is in FOT (sampling of image data in pixels) (*)
[6]	INTE1	Indicates when pixels of integration block 1 are integrating (*)
[7]	INTE2	Indicates when pixels of integration block 2 are integrating (*)
[8]	'0'	Constant zero
[9]	'1'	Constant one
[10]	'0'	Constant zero
[11]	'0'	Constant zero

Figure 2.15 : Control signals on LVDS Ouptut

2.3 Field-Programmable-Gate -Array

Field-Programmable-Gate-Array are basically consists of interrelated programmable blocks of integrated circuits. Multiple task can be performed by engineer by programed the FPGA according to the desired design [11].

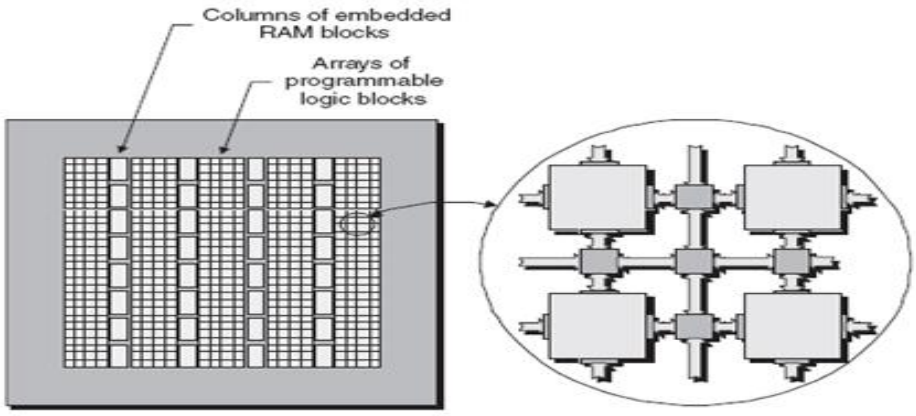


Figure 2.16: Internal Structure of FPGA

The basic building block of FPGA is a Logic Cell, hundreds and thousands of such logic-cells are located in FPGA which is also known as Look –up table. This table can be like act like a RAM. It has at least 4 inputs to form any kind of logic gate.

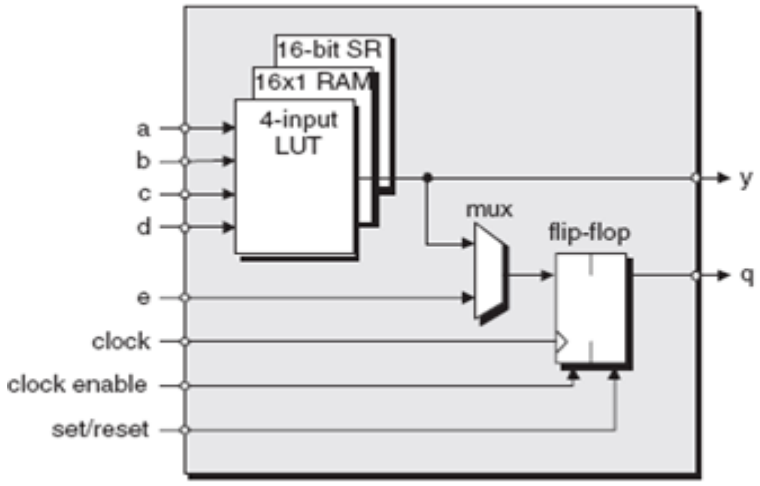


Figure 2.17 Loigic Cell in FPGA

2.3.1 Microsemi A3PE3000 FPGA

Microsemi A3PE3000 is a FPGA of third generation of Flash based FPGA. It has high performance and density. Due to its flash technology it is more safe, energy efficient and single chip device. Due to its reprogrammable technology it is most beneficial for the design engineers and companies require real time implementation of logics. It has 3 million programmable gates that support up to 504Kb and 620 input and outputs [12-13].

Main components are described in the following sub sections:
Microsemi A3PE3000 FPGA is being used as central part of the FPGA board. The FPGA has following important roles in the design:

1. Accept the LVDS data from the sensor over 04 data channels @ up to **240**Mbit/s each channel, 2 LVDS clock channel @ **120**MHz and 1 control channel @ **240**Mbit/s)
3. Performs bit alignment on the LVDS data.
4. Generate all clocks needed by the sensor and Camera Link interface
5. Prepares data for the Camera-Link interface
6. Sends the data to the Camera-Link together with the control bits and clocks

Following design elements are implemented in the FPGA.

Table 2.2 Design Elements Implemented in FPGA

Design Elements	Description
SR_12	12 bit Shift Register
Ofd12	12-Bit Output D Flip-Flop
SR4RE	4-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
SR4RE	8-Bit Serial-In Parallel-Out Shift Register with Clock Enable and Synchronous Reset
CB4CLE	4-Bit Loadable Cascadable Binary Counters with Clock Enable and Asynchronous Clear
COMPM4	4-Bit Magnitude Comparator
OBUF	Output buffer
DCM	Digital Clock Manager

2.4 High Speed LVDS to TTL converter SN65LVDT2

High Speed LVDS (Low Voltage Differential Signaling) is converted to TTL (Transistor Transistor Logic) for the inputs to FPGA [14].

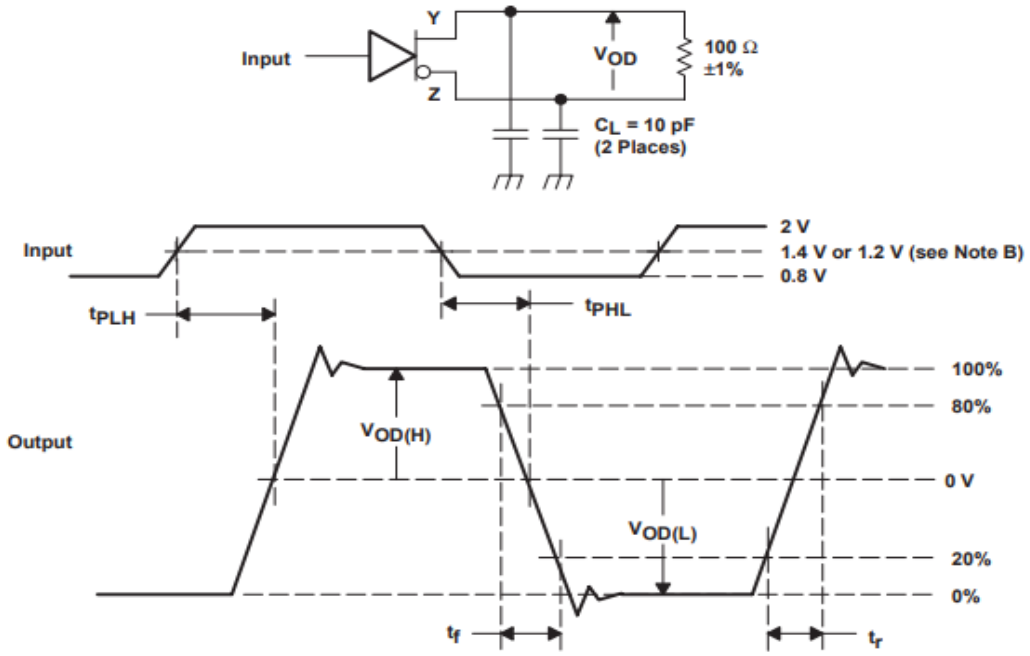


Figure 2.18 LVDS to TTL Logic

3 Methodology

This chapter concisely explains the proposed system: Sensor Board, FPGA Board, and Power I/O Board, Data transmitted to frame grabber over Camera Link Interface (Base + Medium configuration and software used for recover image on computer.

3.1 Description of Design

Electronics design of the system consists of the following main components:

1. Imager module
2. Frame grabber and display module

Imager module captures the image using CMV2000 image sensor, perform necessary operations to align the data required for Camera link Standard with the help of FPGA and then output the data in Base + Medium Camera Link configuration using DS90CR287 Camera Link ICs. It also provides regulated power required for different parts of the design and responsible for performing registers configurations of the CMV2000 Image sensor The frame grabber (NI PCIe-1433) is a standard third party card which plugs into a PC's PCI Express bus that acquires the image data and displays it on a computer screen.

Flow diagram of the design is given below:

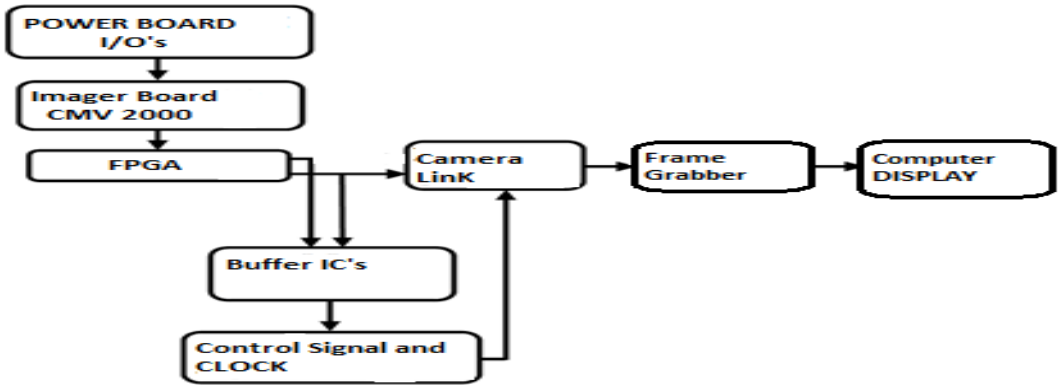


Figure 3.1 Flow Diagram

Main characteristics of our designed electronics part of the system are given below:

1. Consists of Three PCBs i.e. Sensor Board, FPGA Board, and Power I/O Board.
2. Total power dissipation < 3.0W
3. Operating at main clock of 20 MHz.
4. 12bit quantization.
5. Data transmitted to frame grabber over Camera Link Interface (Base + Medium configuration).
6. 1200Mbyts/sec (i.e. 100MP/sec) data throughput.
7. Configurable gain, offset, exposure, and quantization.

3.2 Block Diagram

Block diagram of CMV-2000 CMOS image sensor based camera electronics is shown in figure. Internal structures of the different blocks, data flow through the blocks and their required clocks and input channels are described in the block diagram. Imager module design consists of CMV2000 image sensor, FPGA, power regulation circuit and Camera-Link interface ICs.

To perform the aforementioned tasks, the imager module consists of following electronic boards:

1. Sensor board
2. FPGA Board
3. Power I/O Board

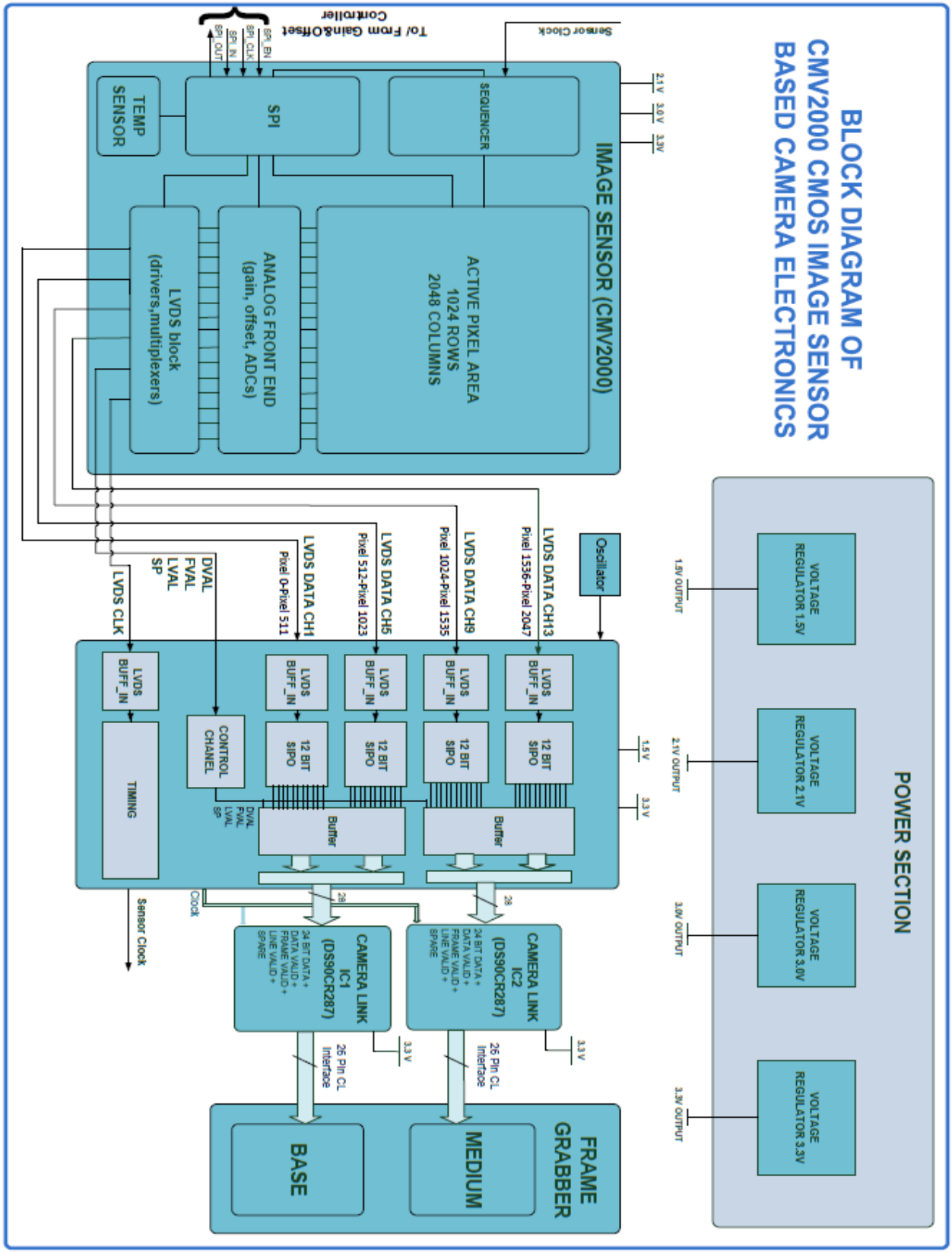


Figure 3.2 Block Diagram

3.3 Imager Module

Imager module is responsible for following operations:

1. Capturing the image using CMV2000 sensor.
2. Align the data from different channels from the sensor and prepare it in a required Camera Link format.
3. Provide connection to send data to Power I/O board.
4. Power regulation.

To perform the tasks, the imager module contains of following electronic boards:

1. Sensor board
2. FPGA Board
3. Power I/O Board

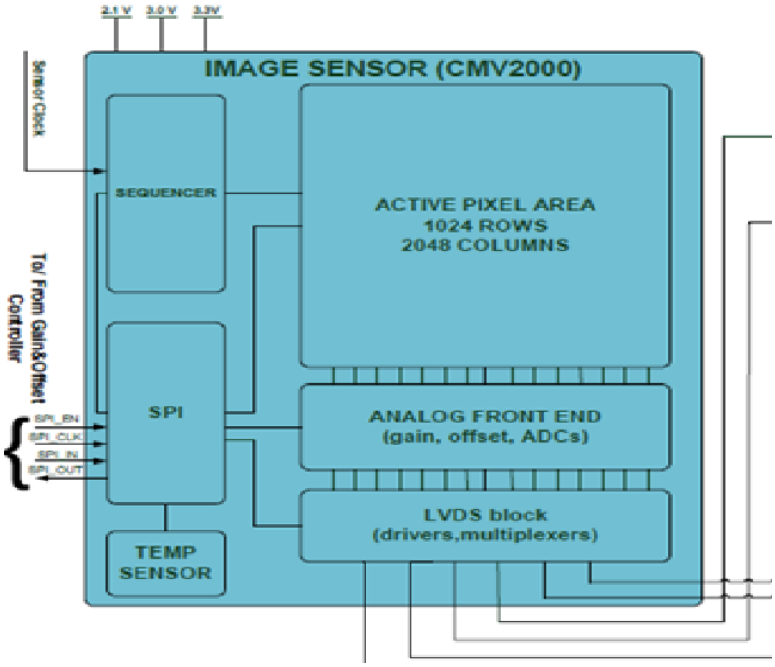


Figure 3.3: Imager Sensor board

3.3.1 Sensor Board

This part of the Imager Module board includes the sensor socket, CMV2000 image sensor and its associated circuit (i.e. decoupling to ground, LVDS circuit and board-to-board connectors). Brief Details on main sections of the sensor board are given in the following sections:

3.3.1.1 Sensor Socket

The sensor can be removed easily from board by using latest ZIF (Zero Insertion Force) socket. Sensor can easily be raised up by using metal handle to 90°. By using this socket no extra force is required to pull up sensor. It can be properly closed by pulling it down as it clicks. The sensor is tightened and cannot fall. The sensor should be placed in the area of the socket as given in the picture below:

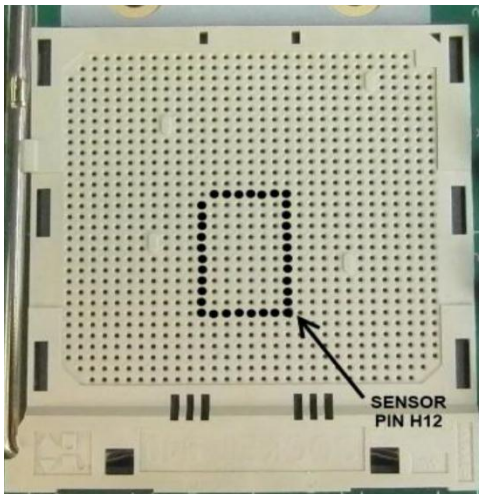


Figure 3.4 Sensor socket

3.3.1.2 CMV 2000 Sensor

The imaging sensor used in hyper spectral camera design with fabry perot filters integrated on chip is CMOSIS CMV2000. CMOS image sensor commercially available consists of 2048 no of pixels in complete row and 1088 pixel in each column, size of each pixels is approximately 5.5µm x 5.5µm

CMOSIS CMV2000 sensor contains sixteen output channels with each channel has 10 bits or 12-bits of data outputs. The sensor also has a gain amplifier which can be programmed and offset controller that can create attainable amplification of chosen signal of that target and minimum desired offset for required output. CMV 2000 has 16 channel LVDS output and 2 LVDS channels for control and synchronization each channel has 480 Mbps maximum attainable speed.

3.3.1.2.1 Data Out put

The CMV2000 produces digital data on 18 output channels contains LVDS- (low voltage differential signaling. 16 Output data from the sensor is then process through FPGA. 1 channel is used for control signals to identify the correct data bits and control the flow of data through the blocks according to desired stream. 1 Clock channel is used to synchronize the flow of data through FPGA, camera link and frame grabber.

Table 3.1 Output Channel Description

No of Channels	Channels	Description	No of pins
16	Data Channels	Used to send data from sensor to FPGA board	32
1	Control Channel	Provides control signal to validate the data.	2
1	Clock Channel	Synchronize the flow of data	2

The 16 data channels transfer data in 10-bit (per pixel) or 12-bit (per pixel) modes. However in this design, data is transfer on 12-bit mode to FPGA board for SIPO (Serial in Parallel Out) processing and then bit streams of data are generated according to protocols required by camera link IC.

The output clock channel contains clock signal, which is used to synchronize the data on the receiving end. Data contains different bit streams passing through four different SIPO processing blocks of FPGA and then through the buffer block so the synchronization of all bit streams are required.

The control channel contains signals of indication for the validation of the data on the data channels. Data on the control channel contains 10 or 12-bits of words that are shifted synchronous to the 16 data channels.

3.3.1.2.2 Input Clock

Input frequency applied to the sensor CMV 2000 determines the output frequency rate at which the output data is processed. Output data rate of the LVDS data is much higher than input frequency. Output data rate LVDS_CLK at 480 Mbps is achieved with the input master clock (CLK_IN) of sensor is 40 MHz

Table 3.2 Relationship b/w CLK_IN and LVDS Clock

CLK_IN	LVDS_CLK (12bit mode)
5 MHz (min)	60 MHz
10MHz	120MHz
20MHz	240MHz
40 MHz (max)	480 MHz

The FPGA A3PE3000 supports upto 350MHz, so CLK_IN is set to 25MHz and the LVDS_CLK would be 300MHz in the design.

3.3.1.2.3 Pixel Readout Timing

The 128 pixels of data from imager board is received per channel. Each pixel contains 10/ 12 bits of data. Pixel period is equals to master clock input and OH (Over Head) time exist between the two 128 pixels burst of image data, OH has same time as one pixel readout of 10 to 12 bits of data.

If 4 channels output used for data then, (4 x 128) + (4 x 1) master clock periods used and frame rate reduced with a factor 4.

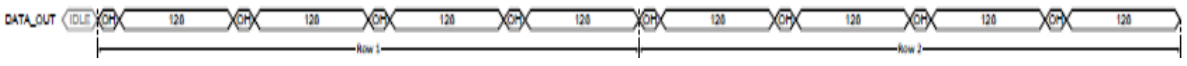


Figure 3.5 4 Channel output readout

Fig 3.5 shows when 4 output channels transmit the data on LVDS outputs then, complete image data row will be transmit trough 4 sets of 128 burst of pixels. All the 4 output LVDS channels contains data along with the 2 additional LVDS data channels that contain control and clock data for controlling and synchronize function.

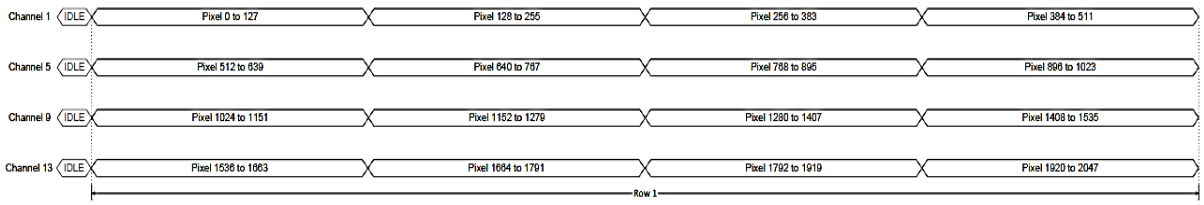


Figure 3.6 Data read Out of 4 channel

3.3.1.2.4 Control channel

CMV 2000 has only one dedicated output LVDS channel that contains the control information of 10 bits -12 bits, with two pins used for differential signal. Control signal determines the validation of the data transmitted through the output channels provides the information to synchronize the valid data with the clock signal.

Control signals generated by the image sensor CMV 2000 are shown below in Fig 21 from bit 0 to bit 11 all 12 bit control signal in LSB format.

Table 3.3 Elements of Control Channel

Bit	Function	Description
[0]	DVAL	Indicates valid pixel data on the outputs
[1]	LVAL	Indicates validity of the read-out of a row
[2]	FVAL	Indicates the validity of the read-out of a frame
[3]	SLOT	Indicates the overhead period before 128-pixel bursts
[4]	ROW	Indicates the overhead period before the read-out of a row
[5]	FOT	Indicates when the sensor is in FOT (sampling of image data in pixels)
[6]	INTE1	Indicates when pixels of integration block 1 are integrating
[7]	INTE2	Indicates when pixels of integration block 2 are integrating
[8]	'0'	Constant zero
[9]	'1'	Constant one
[10]	'0'	Constant zero
[11]	'0'	Constant zero

3.4 FPGA Board

The FPGA board consists of Microsemi A3PE3000 FPGA, DS90CR287 Camera Link interface IC, LVDS networks, decoupling capacitors, interface connectors for Power I/O Board.

Main components are described in the following sub sections:

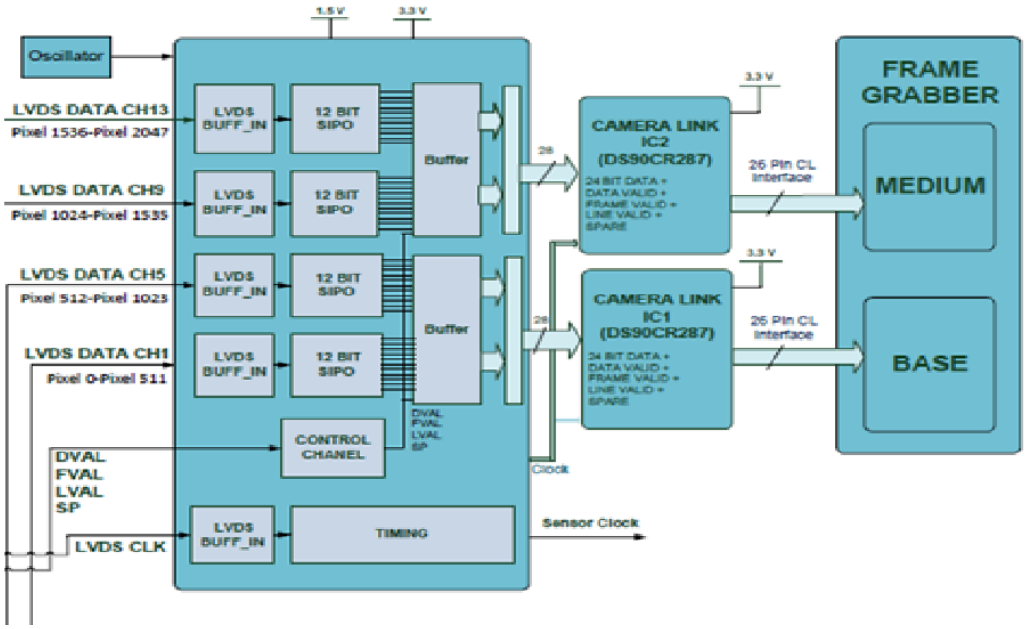


Figure 3.7: FPGA and Frame Grabber

3.4.1 FPGA

Microsemi A3PE3000 FPGA is being used as central part of the FPGA board. The FPGA has following important roles in the design:

1. Accept the LVDS data from the sensor over 04 data channels @ up to 240Mbit/s each channel, 1 LVDS clock channel @ 120MHz and 1 control channel @ 240Mbit/s)

2. Performs bit alignment on the LVDS data.

4. Generate all clocks needed by the sensor and Camera Link interface

5. Prepares the data for the Camera Link interface

6. Sends the data to the Camera Link interface together with the control bits and clocks

Following design elements are implemented in the FPGA.

Table 3.4 Design Elements Implemented in FPGA

Design Elements	Description
SR_12	12 bit Shift Register
Ofd12	12-Bit Output D Flip-Flop
SR4RE	Shift register with serial input and parallel out of 4-Bit. Containing enable clock and Reset.
SR8RE	Shift register with serial input and parallel out of 8-Bit. Containing enable clock and Reset.
CB4CLE	Cascadable Binary loadable 4 bit Counters .
COMPM4	4-Bit Magnitude Comparator
OBUF	Output buffer
DCM	Digital Clock Manager

Implementation of These elements is provided in the table and screen shots of simulation waveforms are discussed in Chapter 4.

3.4.1.1 VHDL Programming Logic in FPGA

Sensor board produces the data contains image on 4 LVDS output data channels and 2 channels containing control signal and clock all the 6 LVDS channels are then processed through converter IC, which is known as LVDS to TTL converter as LVDS data processing through FPGA is very difficult to handle and it will allocate large amount of cells of FPGA that will not only increase the allocation but also complex the situation and decrease efficiency. SN65LVDT2 high speed LVDS to TTL IC is used to convert the LVDS data and then this data is communicated to FPGA board.

FPGA Board process the data to produce data required according to protocol that will be used in Camera Link IC interface, FPGA converts the bits coming from the sensor and perform serial to parallel operation by using shift registers; these serial bits are coming in line in the form of burst with OH (Over Head) as discussed earlier in sensor structure, all four channels of data and control channels contains TTL logic data are converted to parallel bit streams using the shift register in FPGA, and clock channel is used to implement DCM (Digital Clock

Manager). DCM is a logic block implemented in FPGA to generate common clock having frequency double as compared to clock frequency of the sensor for the entire buffer IC's, camera link IC's and frame grabber card. DCM block in FPGA in implemented to synchronize all the data which is processes through different block implementation like in buffer and camera link protocol during the regular changing of bit streams.

Implementation of all the blocks in FPGA using XYLINX software is shown in Fig 3.8.

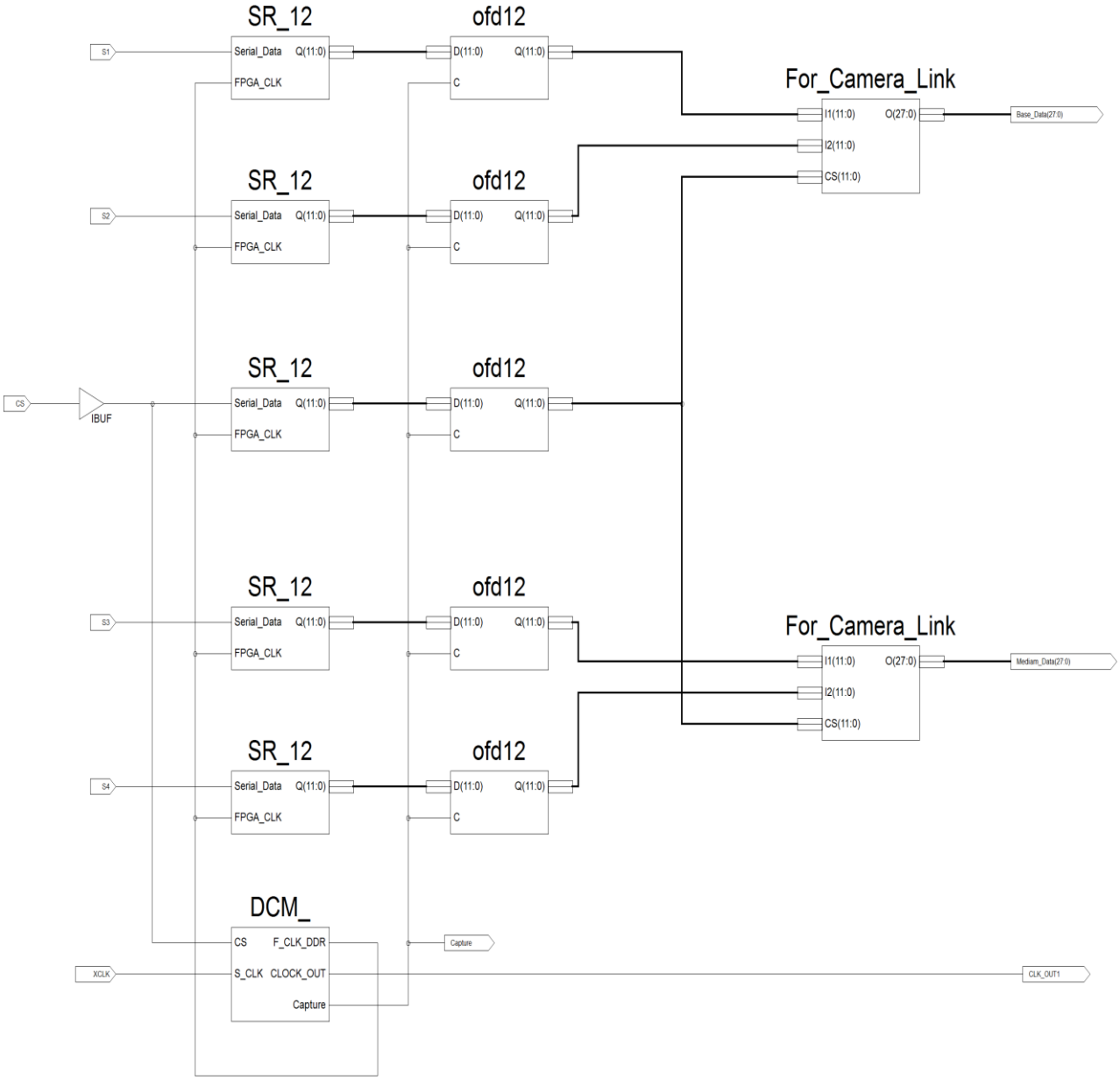


Figure 3.8: VHDL Implementation in FPGA

SR_12 is a Shift Register Block which shows the 4 input channels with control channels with inverted input are processed for SIPO (Serial in Parallel out).

Ofd 12 are D-Flip Flop implemented for PIPO (Parallel in Parallel out) configuration , these blocks are used for buffering of data to generate delay od 1 clock cycle.

DCM (Digital Clock Manager) receives the input of control channel and clock, DCM is used to generate a common clock signal to synchronize all the data throughout the process, DCM produce DDR (Dual Data Rate) output having frequency double as compared to sensor Clock. F_clk output used for clock to produce parallel 12 bit streams, Clock _out is used output DDR is used for further Frame grabber IC and Capture is used to capture or hold 12 bit of data to d- flipflps.

Camera Link block receives 28 bits , 12 bits of data from each data channels and 4 bits of control signal containing the info regarding data valid or line valid. IT transmits data to Frame grabber card in base and medium configuration contains 2 or 4 channels having 14 or 7 bits simultaneously and converts to LVDS.

3.4.1.1.1 SR_12 Shift Register Block

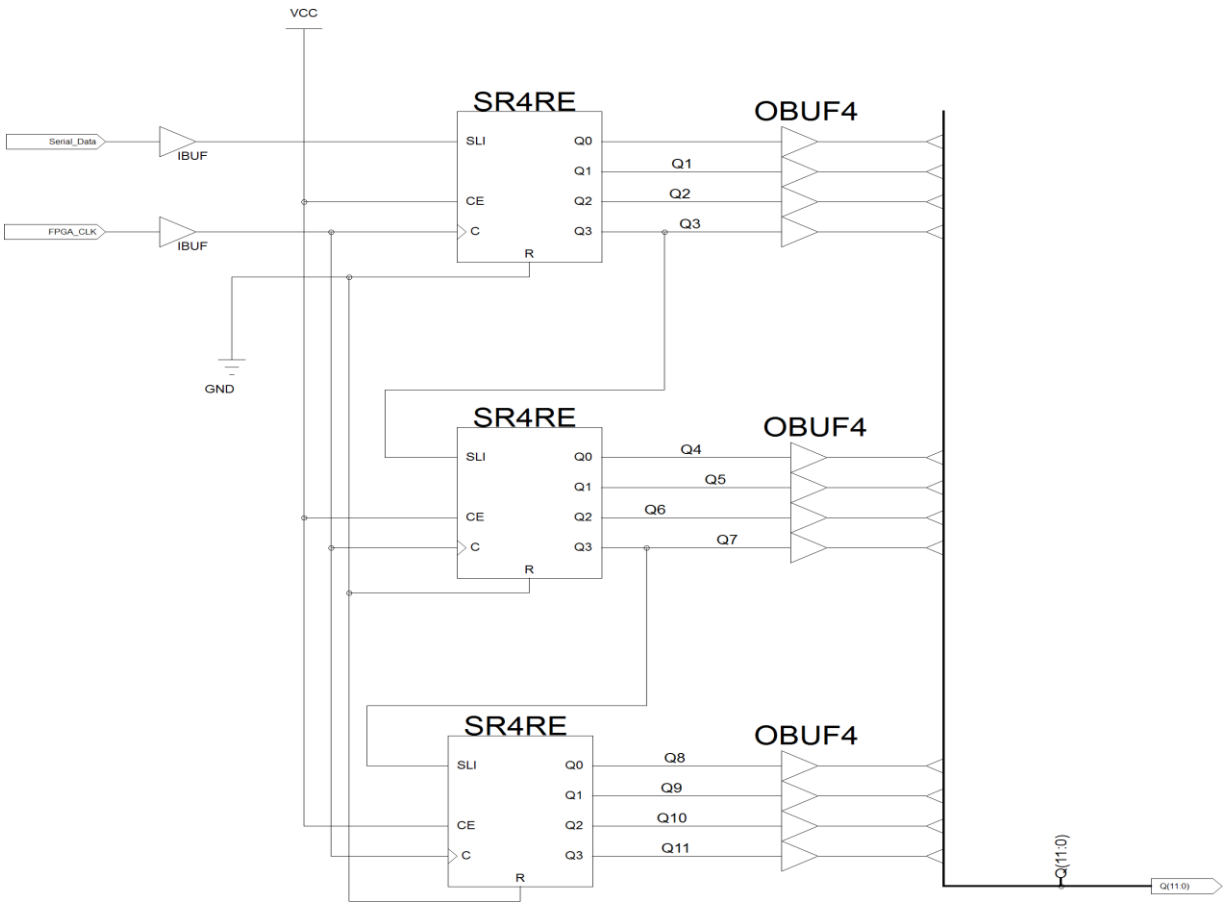


Figure 3.9 : SR _12 Shift Register

SR_12 Block contains 4 bit SR4RE Shift register with Reset enable and output buffer with sink source used to increase the current on pins to connect other devices.

3.4.1.1.2 12 bit Output D Flip-Flop

D Flip-Flops are used to hold the data for 1 Clock cycle .

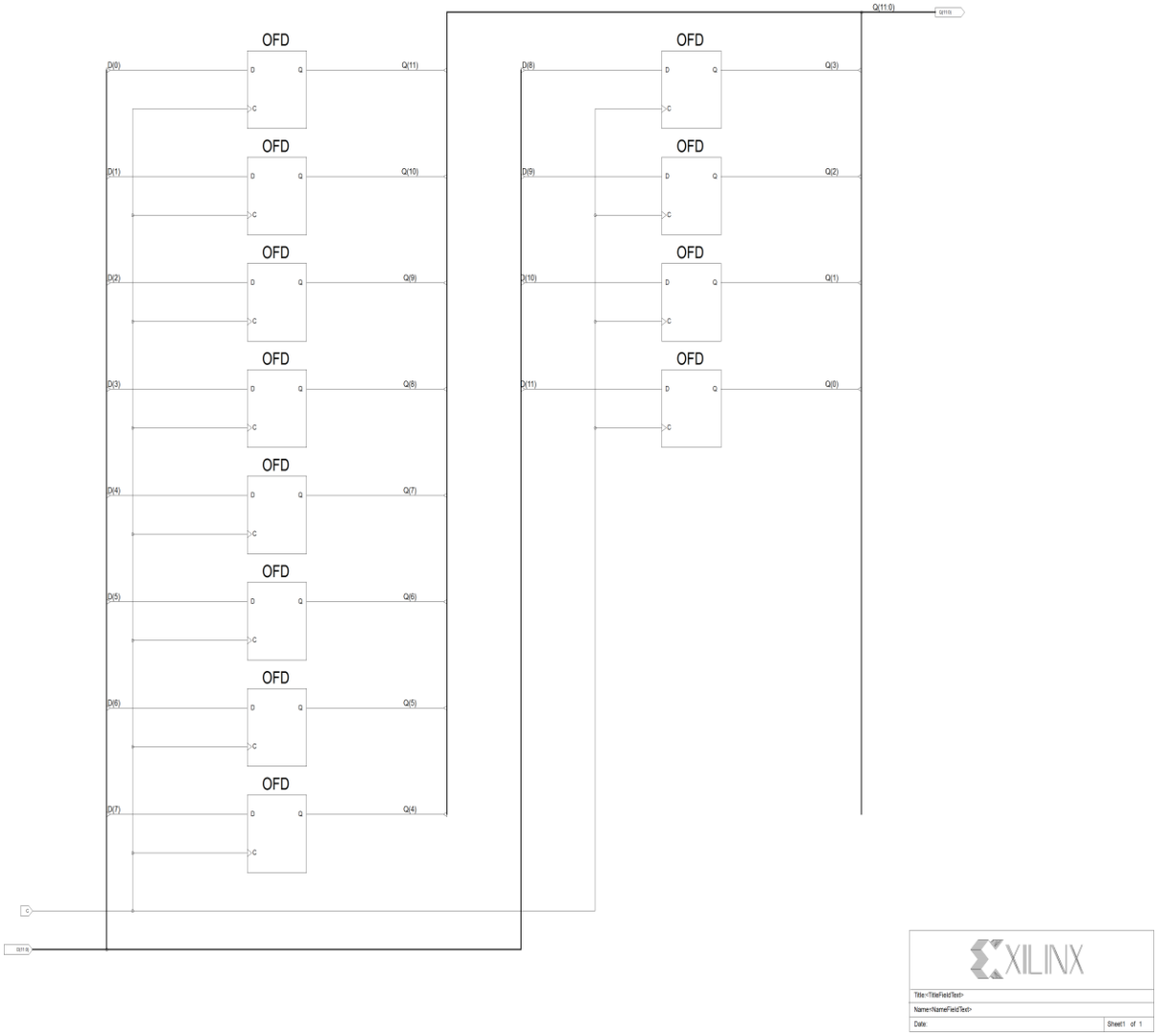


Figure3.10: 12 Bit D Flip-Flop

3.4.1.1.3 Camera Link

Camera Link block receives 28 bits , 12 bits of data from each data channels and 4 bits of control signal containing the info regarding data valid or line valid. IT transmits data to Frame grabber card in base and medium configuration contains 2 or 4 channels having 14 or 7 bits simultaneously and converts to LVDS

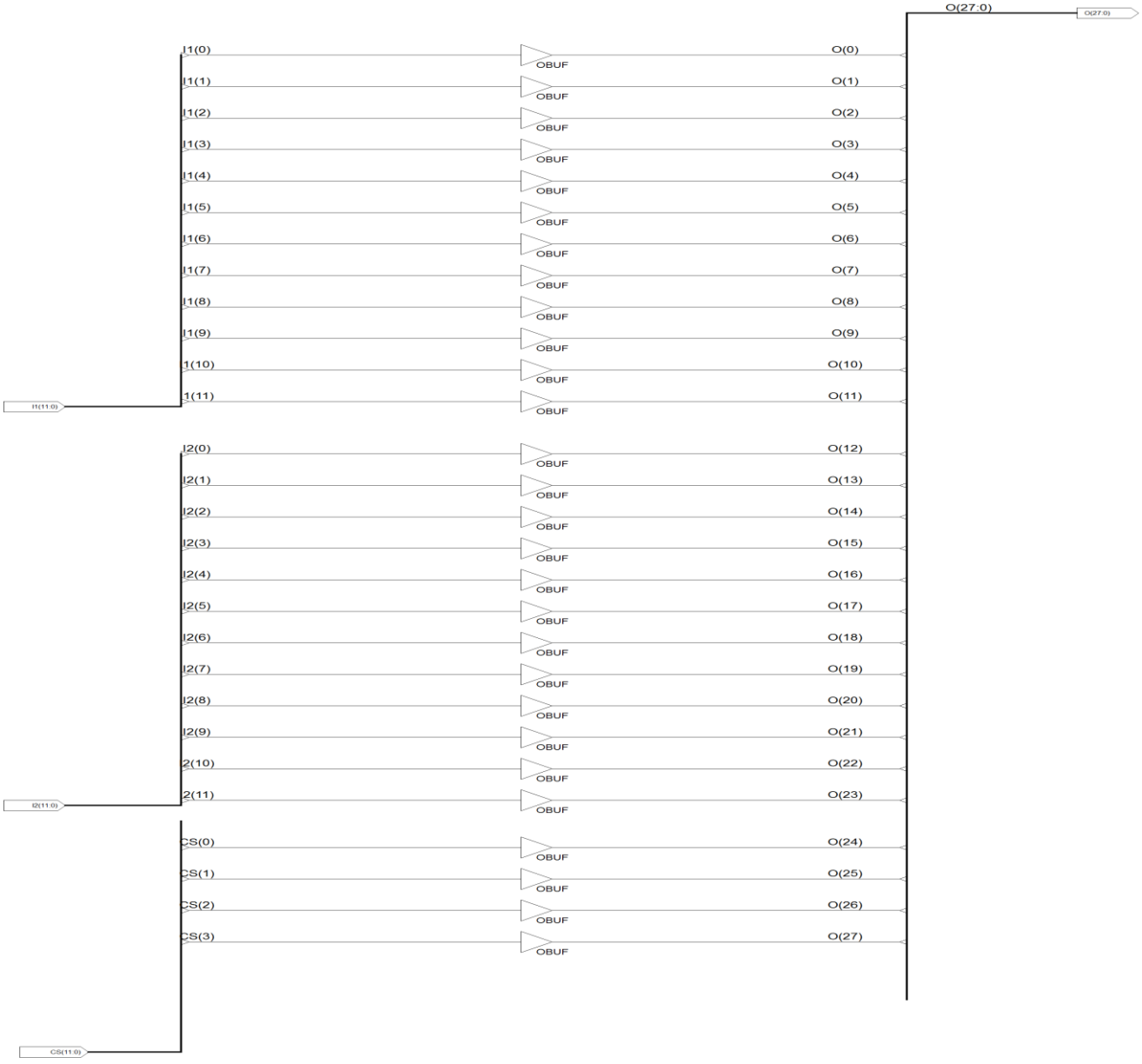


Figure 3.11 Camera Link

3.4.1.1.4 DCM (Digital Clock Manager)

DCM is used to generate a common clock signal to synchronize all the data throughout the process, DCM produce DDR (Dual Data Rate) output having frequency double as compared to sensor Clock. DDR is used to read out the data at same time on both rising edge of the frequency and falling edge of the clock signal. F_clk output used for clock to produce parallel 12 bit streams, Clock _out is used output DDR is used for further Frame grabber IC and Capture is used to capture or hold 12 bit of data to d- flipflps.

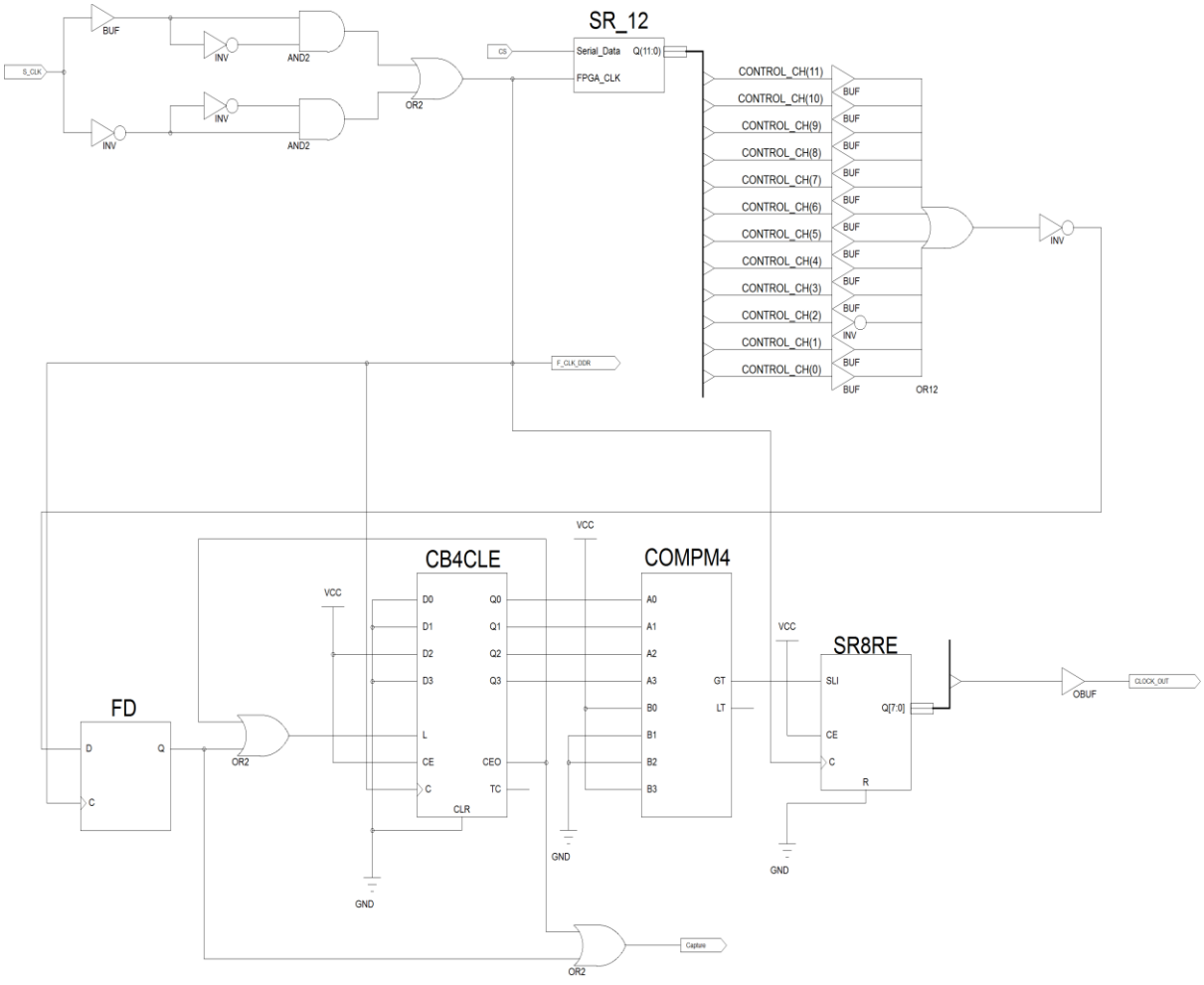


Figure 3.12: Digital Clock Manager

3.4.2 SN65LVDT2 high speed LVDS to TTL

Six SN65LVDT2 are used for high speed (up to 630 Mbps) LVDS to TTL conversion in SOT23 package. These ICs are used as following:

Four ICs for Each LVDS Data (Input to FPGA)

One IC for LVDS Clock to (Input to FPGA)

One IC for LVDS Control Channel (input to FPGA)

3.4.3 DS90CR287 Camera Link Transmitter IC

The DS90CR287 IC is a transmitter that transforms 28 bits (24 data bits + 3 control bits +1 spare bit) LVTTL data to four LVDS (Low Voltage Clock Differential Signaling) data streams. Every cycle of the Low Power Consumption transmit clock 28 bits of input data are sampled. Two ICs are used to implement Base + Medium Configuration of the Camera Link Standard at a rate of 100MPixels/sec with 12 bit quantization.

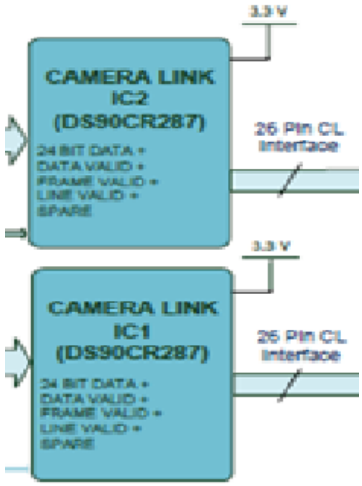


Figure 3.13: Camera Link IC

3.5 Power I/O Board

Power I/O Board receives +5V power and regulates it into +3.3V, +3.0V, +2.1V, and +1.8V to provide required power to all the components on different boards. Power I/O board has a PIC 18L8680 microcontroller for gain and offset settings of CMV2000 image sensor. The microcontroller will also be used for health monitoring of the system. Power I/O board also has MDR-26F interface Connectors to transmit the image data on Camera Link interface over the Camera Link cables to the Frame grabber.

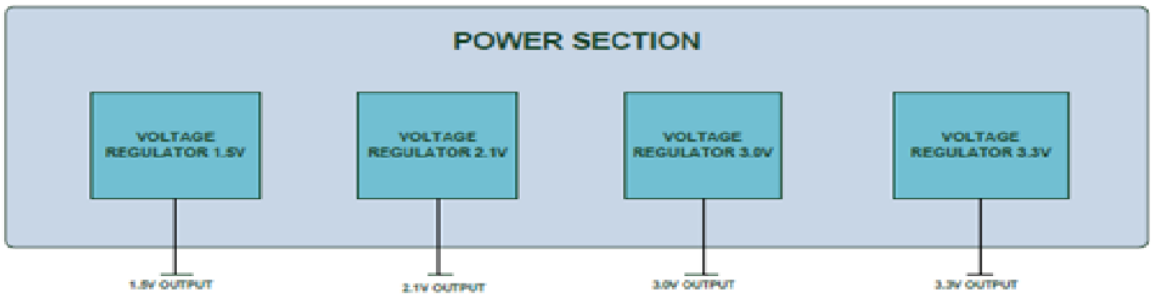


Figure 3.14: Power Board

3.6 Software Implementation

Software used for the image acquisition is Imec system software developed by the OEM which produces imec hyperspectral cameras commercially. Software is purchased from OEM and used for image acquisition from frame grabber card implemented on card.

Raw data of the image is acquired through this Imec software and then this raw data is processed through MATLAB to produce the signatures of different samples available.

3.6.1 Imec software

The imec hyperspectral imager is designed for commercially available CMOS sensor for the vision market. User interface of Imec hyperspectral software is developed for user-friendly imaging operations with implementation of integrated camera and materials on the translation stage; It has translation stage control to adjust the frame rate and scanning speed of camera.

It has easy to use GUI(Graphical User Interface), Frame rate; rotational speed can easily set for the line scanning of the camera .Imec software obtain the DN value of the image and the reflection of light of the image for each pixel and reflection for the white object, then divide the value obtained by white reflection value so we get reflectance values of each image.

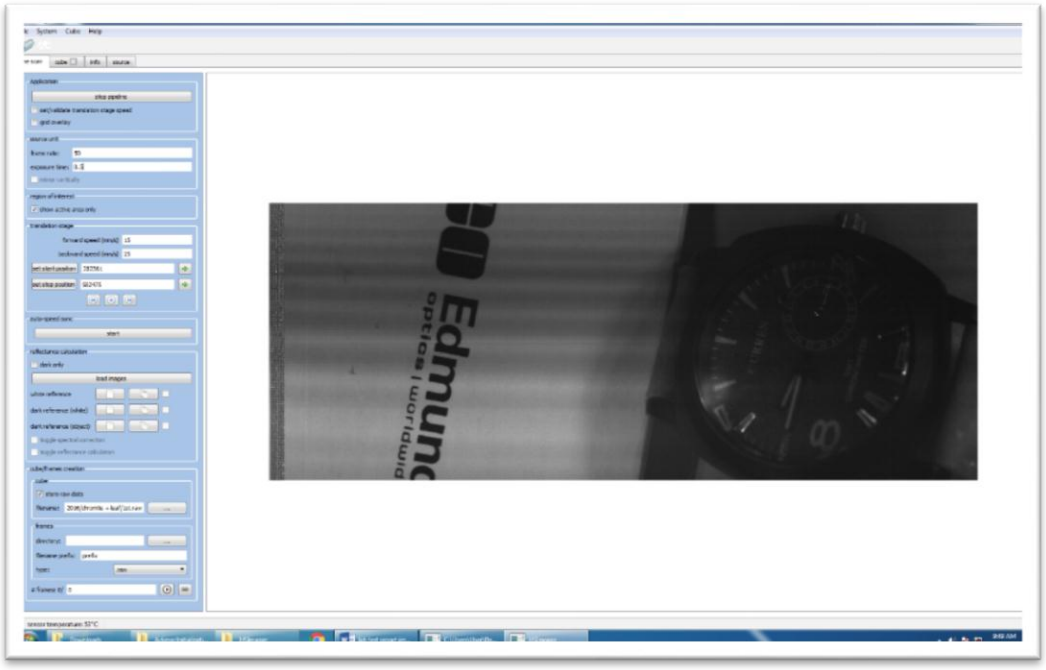


Figure 3.15: GUI of Imec software

3.6.2 MATLAB

MATLAB software is used after raw data of the image is obtained through the imec software. Frame grabber data is read through imec software; in which we select the area, scanning and integration time of the capturing time. Imec software produce the raw data that include the information related to image and reflectance of the light on that point for each pixel of the image.

As each line of the camera include 2048 *1088 pixels and 100 bands so it produce the value of reflection of light for each pixel; 100 bands contains 109 pixels approximately. Raw data through the imec software is then processed through MATLAB to create the spectrograph of image at any point we want.

First of all code has been written on MATLAB to call the raw data of the image,as shown in the Fig 3.16.

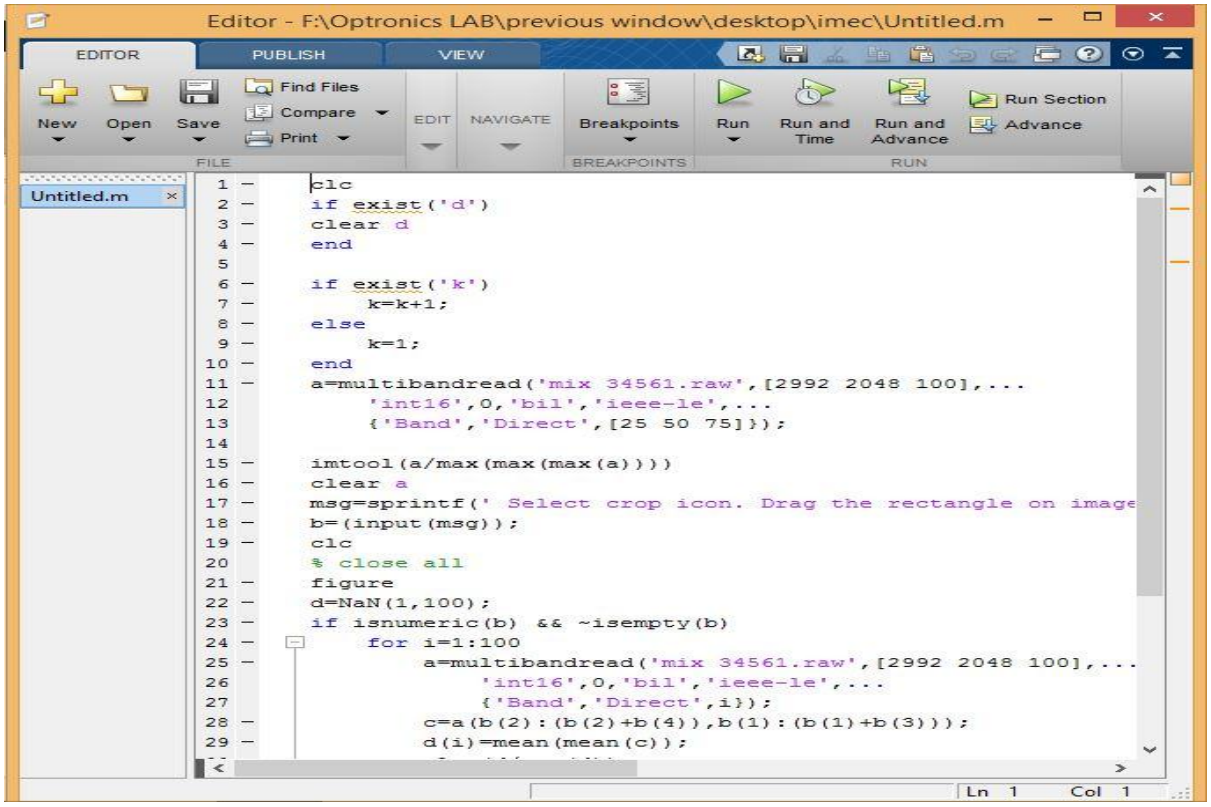


Figure 3.16 MATLAB CODE

Then after the command prompt window appears then we have to select the area of interest to develop the spectrograph of respective region as shown in Fig 3.17.

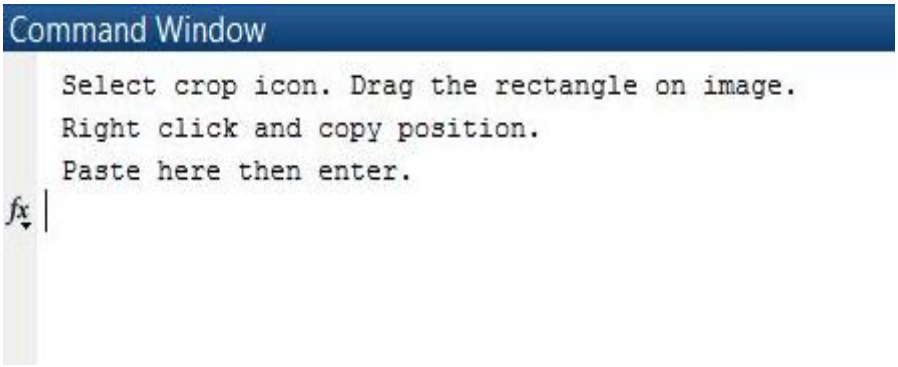


Figure 3.17 Area selection

In Figure 3.18, required are of the image is cropped and paste in the command window.

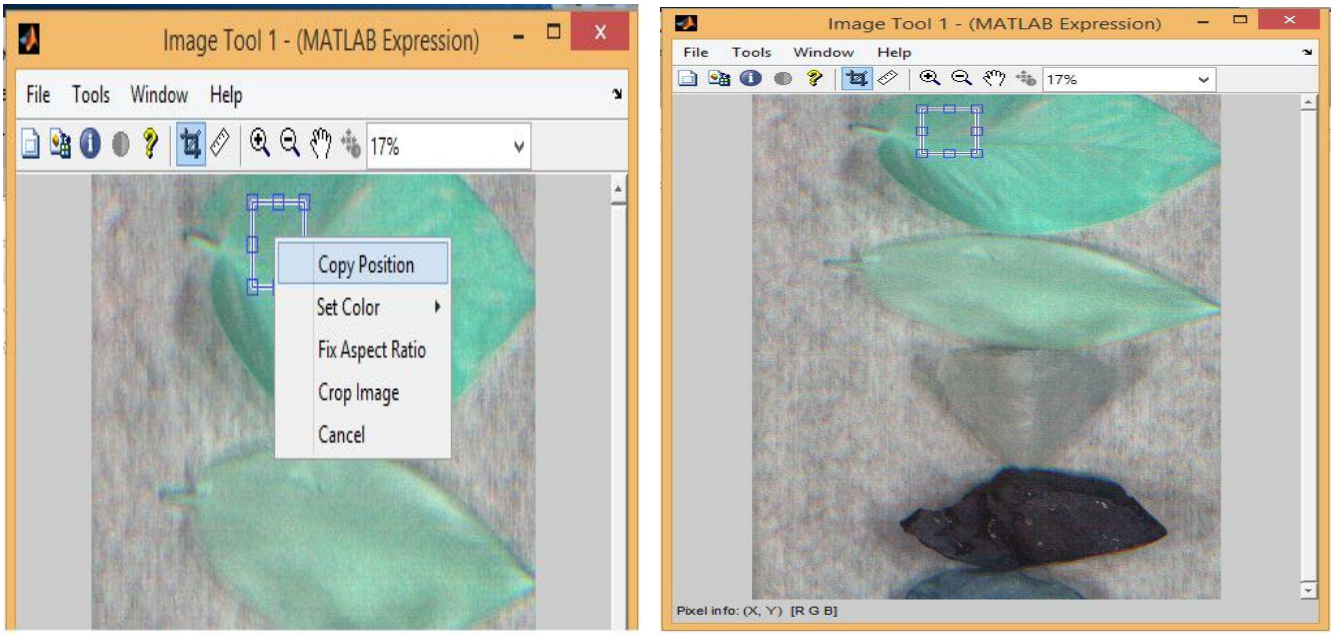


Figure 3.18 Cropping area of image

Figure 42 shows the required coordinates of the position have been pasted in command prompt.

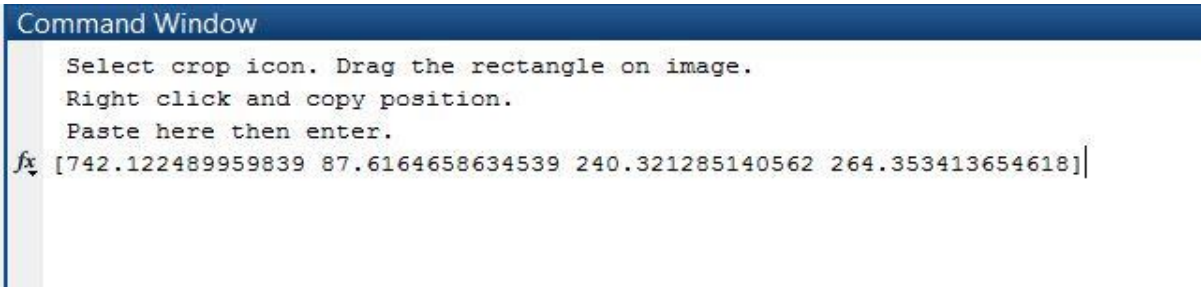


Figure 3.19: Coordinates of required area

Finally the spectrograph of the required image is shown in Figure 3.20.

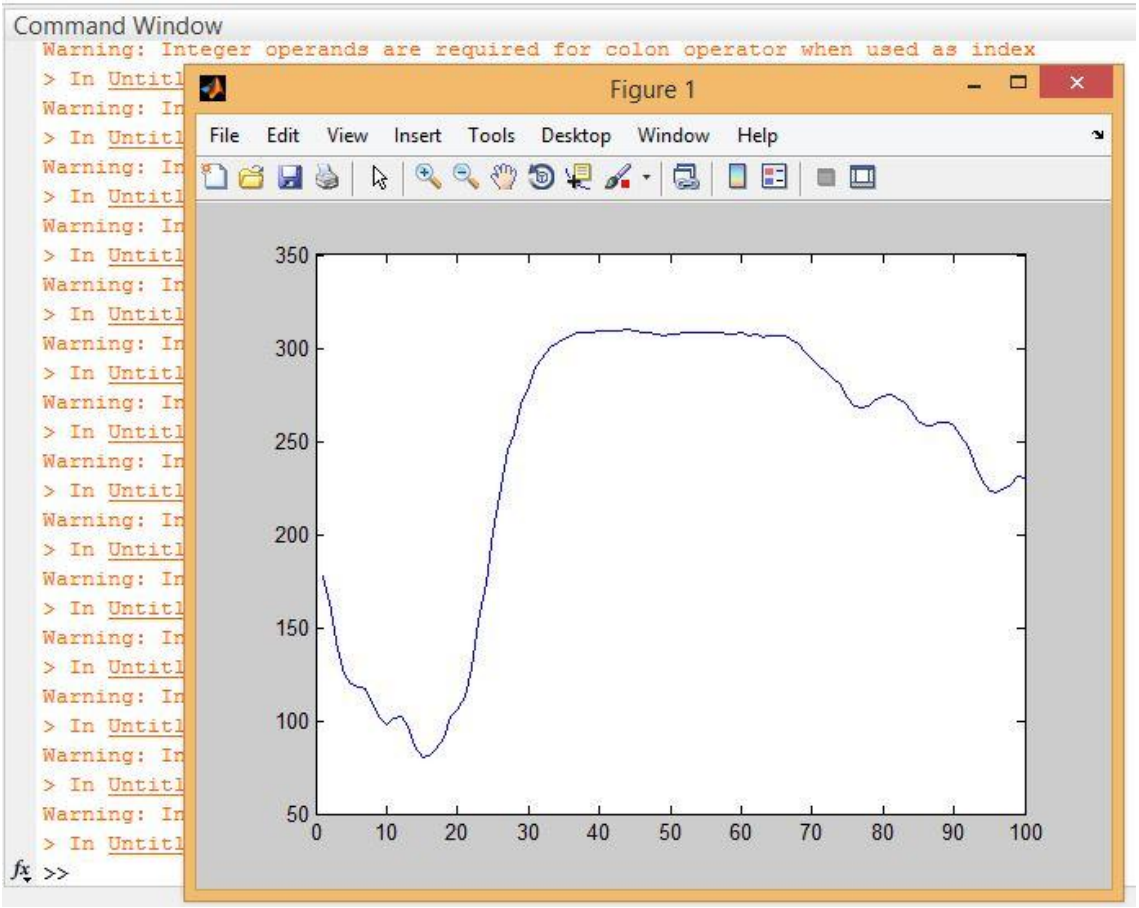


Figure 3.20 : Spectrograph through MATLAB

And code written in MATLAB is attached in Appendix.

4 Results and Discussion

This chapter gives a brief discussion on the results, output waveforms and data analysis by using MATLAB.

4.1 Simulation of FPGA

Camera image sensor CMV 2000 produce the image data on 4 channels LVDS and 2 channels for the control and clock signals and these are also LVDS then these signals are converted to TTL logic from LVDS output and provided to FGPA of the system to convert all the four channel signals SIPO (Serial in parallel out) combination with the control signals too in that format with the clock signal used to produce common clock to synchronize all the data in blocks of the FPGA , camera link IC and Frame grabber card.

To implement the design we have performed the simulation of FPGA input and outputs by using the assumed values of data through the sensor. Inputs of the FPGA are created to check the follow of data through the blocks implemented in FPGA and for that purpose a test bench code is written in XYLINX software to simulate performance of logic. Test bench code is provided in Appendix.

Simulations of the design are carried out by creating 4 inputs S1 to S4 for FPGA. These inputs are assumed as image data from the sensor to block 1 which is described earlier SR_12 is the shift register to carry out SIPO combination of the data, All the inputs and control signal are in hexadecimal system as the code is written for 12 clock pulses for each logic 1 or 0, and output to camera link for frame grabber is in 28 bit format so that we can understand the functionality as shown in Figure 3..

CS is control signal also written in hexadecimal numbers system and in this simulation we have set the value 0111 or 7 in hexadecimal for 12 clock pulses.

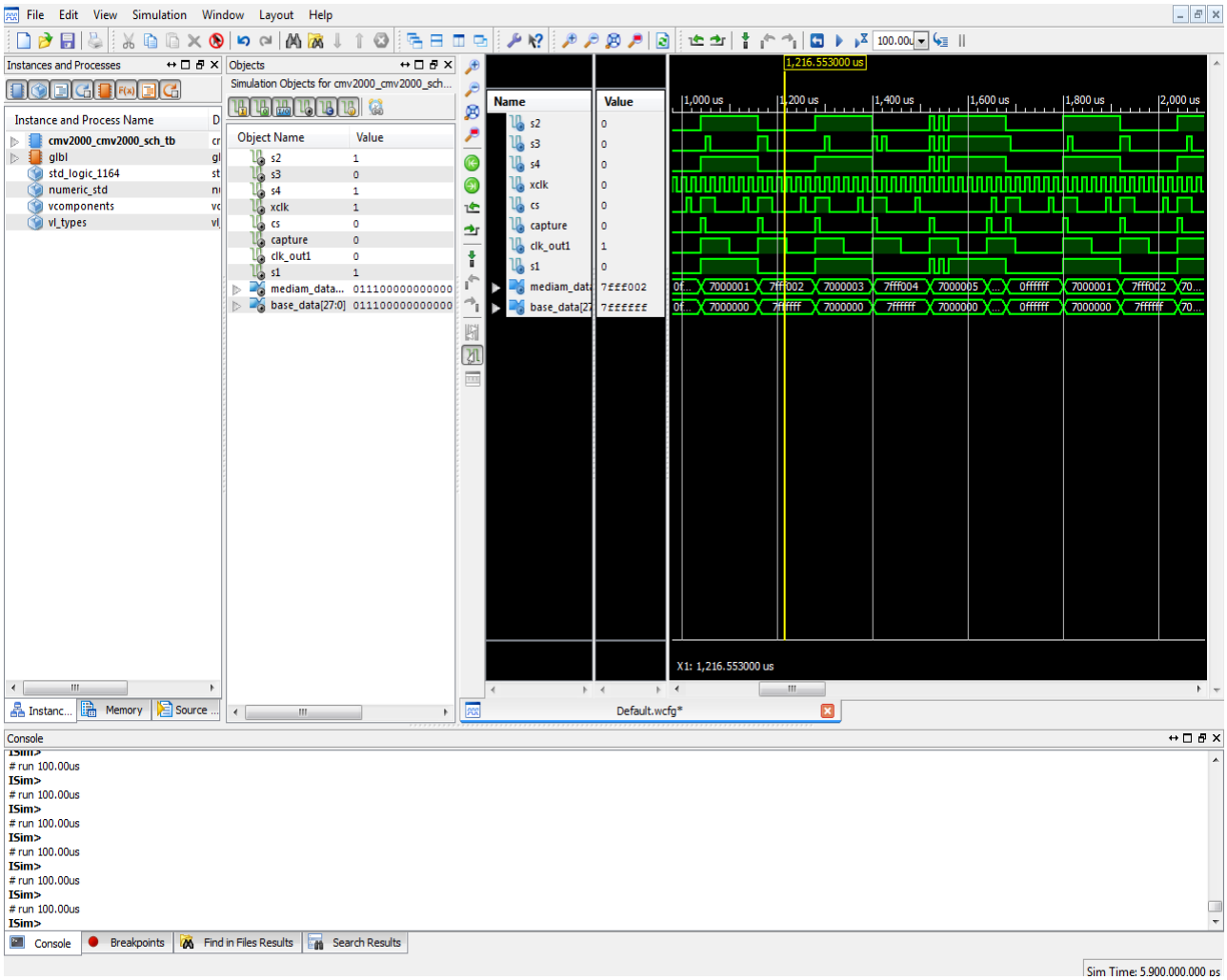


Figure 4.1 Simulation of FPGA

CLK_1 is the clock for the camera link and Frame grabber it has frequency half of the frequency of the data processing at DDR (Dual Data Rate). Capture is high after each 12 clock pulse to hold or capture the data.

S1 to S4 are different combinations of data, Medium and base are the two outputs that are used for camera link IC and frame grabber. Data is simulated in two different combinations in Medium combination which is 28 bit output of FPGA is combination of 4bit control signal then 12 bit S4 input and 12 bit S3 input. S4 and S3 are increasing numbers as shown in simulation.

Base output combination is 4bit control signal then 12 bit S2 and the 12 bit S1 , here the combination will the inverted form for 12 clock pulses values are 0 and next 12 cycles values are F in hexadecimal as shown in Figure 4.1.

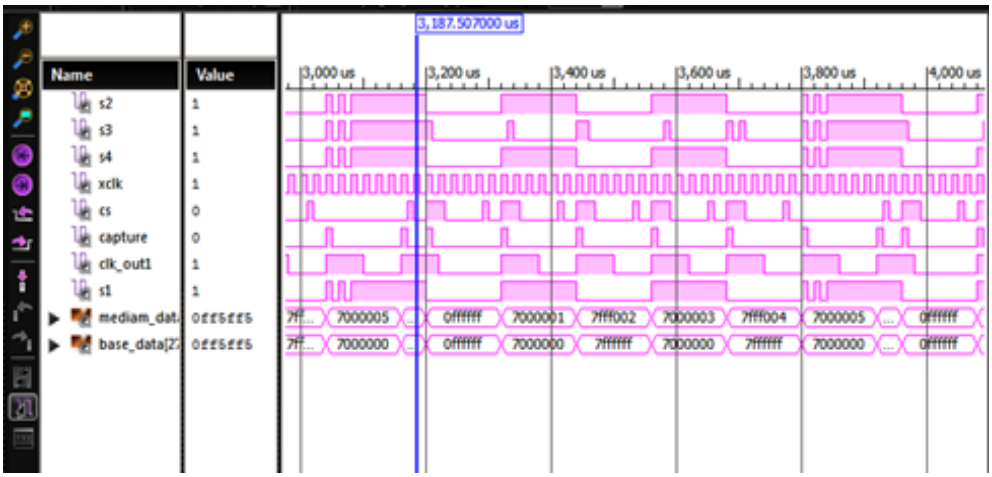


Figure 4.2 Simulation Software

Interrupts are generated at 3.8 usec to check that the system will reset itself on next capture signal, Simulation results shown the system reset itself and correct values again are shown in Figure 4.2.

4.2 Outputs of Imec and MATLAB

Imec hyper spectral imaging software is procured by OEM of CMOSIS sensors. Imec software is used to develop the image of line scan imager camera by choosing the desired values of integration and rotational stage time. Imec software produces image with raw data of image containing value of reflection for each image cell of imager camera containing 2048*1088 pixels. This raw data then finally used in MATLAB to form the signatures of different images of the objects.

Different minerals and leaves of plant are tested in laboratory and use imec software to develop the image of these minerals and raw data which is further used in MATLAB to produce the signature or spectrograph of all the images obtained.

Spectrograph of two different leaves is formed by using these software's which shows that we can easily recognize by using the signatures of different materials during airborne imaging as shown in Figure 4.3 and 4.4.

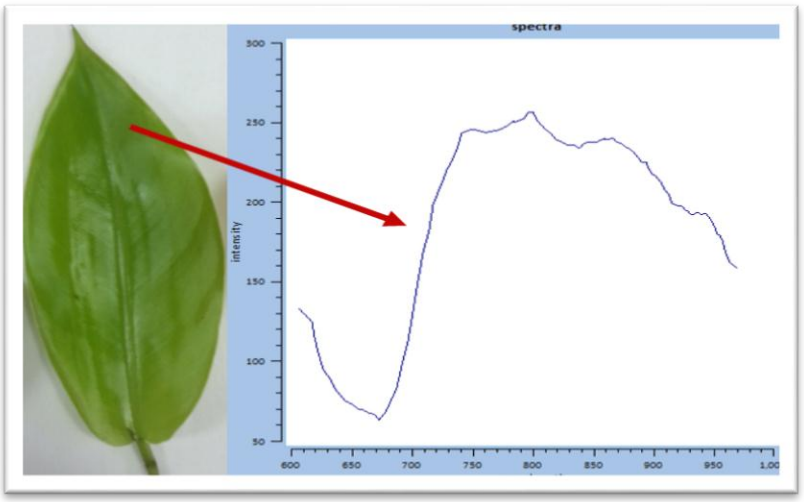


Figure 4.3 Spectrograph of Fresh leaf

Figure 4.3 shows the image and spectrograph of fresh leaf for 100 bands which shown that the reflectance of fresh leaf has dip at visible band and at green band at RGB region.

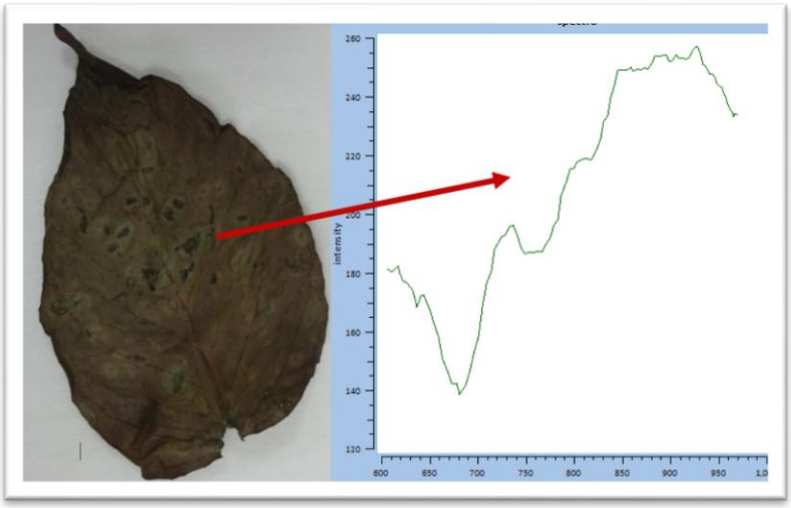


Figure 4.4 Spectrograph of Dead leaf

Spectrograph of dead leaf in Figure 4.4 shows that it has dip in graph at visible band but it has no absorption band or dip at green band , hence we can say that we can easily distinguished between different objects through spectrograph during airborne imaging.

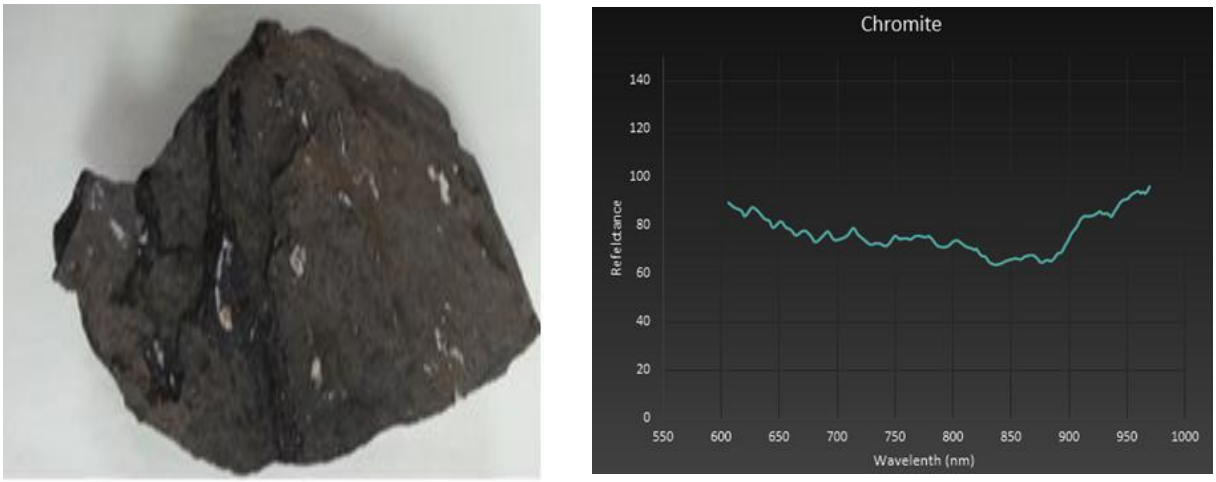


Figure 4.5 Spectrograph of Chromite

Spectrograph of different minerals is also obtained in order to test the results of imaging and spectrograph. Spectrograph of Chromite and salt are shows different structure and dips or absorption at different band wavelengths that shows it is working properly.

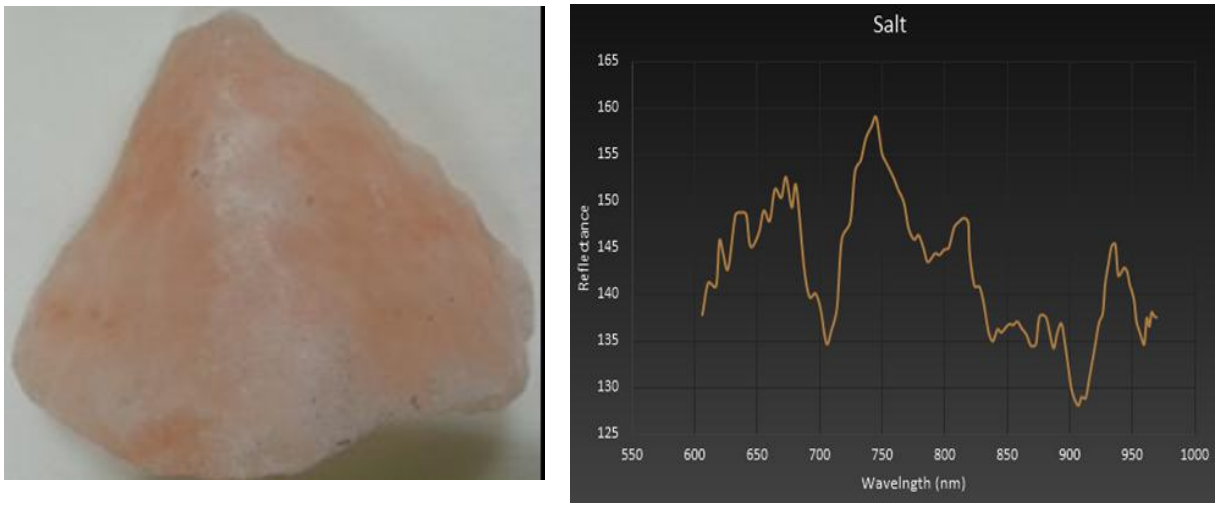


Figure 4.6 Spectrograph of Salt

Figure 4.5 and 4.6 shows the spectrographs of Salt and granite, which shows both have different signature curves for different wavelength bands , so we can easily differentiate between different minerals by using hyper spectral imaging camera during air borne operation.

4.3 Hardware Setup

To verify the functioning of the proposed system it was implemented on hardware. For hardware implementation Sensor board, imager board power board PCB's are designed and then outsource it through Smart PCB's .Smart PCB's designer company developed our PCB's which are shown in Figure 4.7 and 4.8.

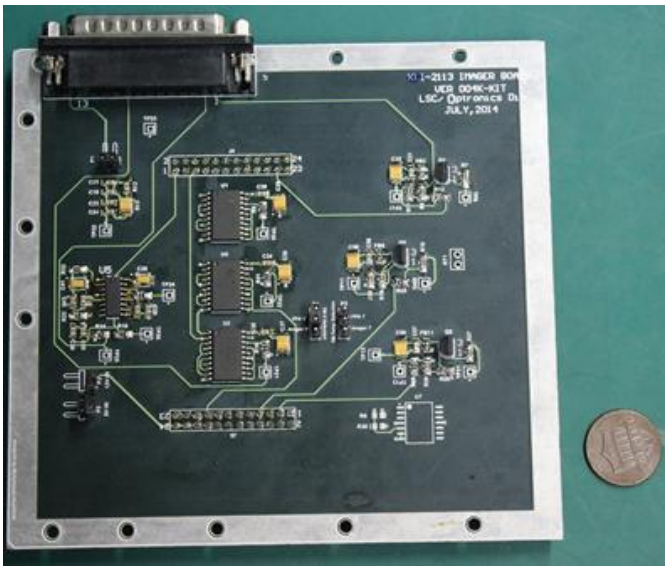


Figure 4.7 Sensor Board

Hardware PCB designed for sensor board is shown in figure 4.7 , Sensor board contains sensor and support IC's that provide required data to the FPGA board to form real time image of the scene .

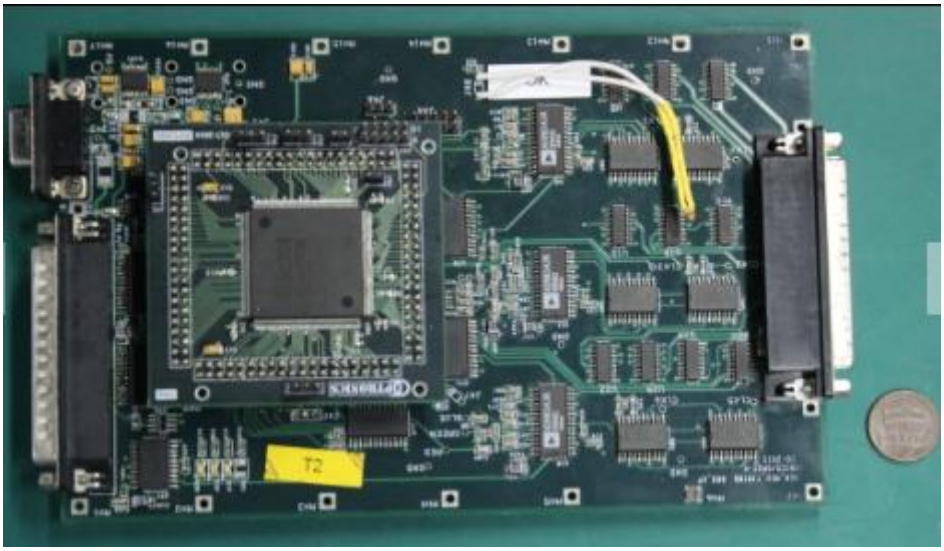


Figure 4.8 FPGA Board

Figure 4.8 describes the implementation of FPGA board that receives data signals in LVDS format then LVDS to TTL IC's are used to convert the format of data. FPGA performs SIPO (Serial in Parallel out) and then the buffer IC's and supporting circuitry produces the required data on the described protocol of Camera Link IC and Frame Grabber card.

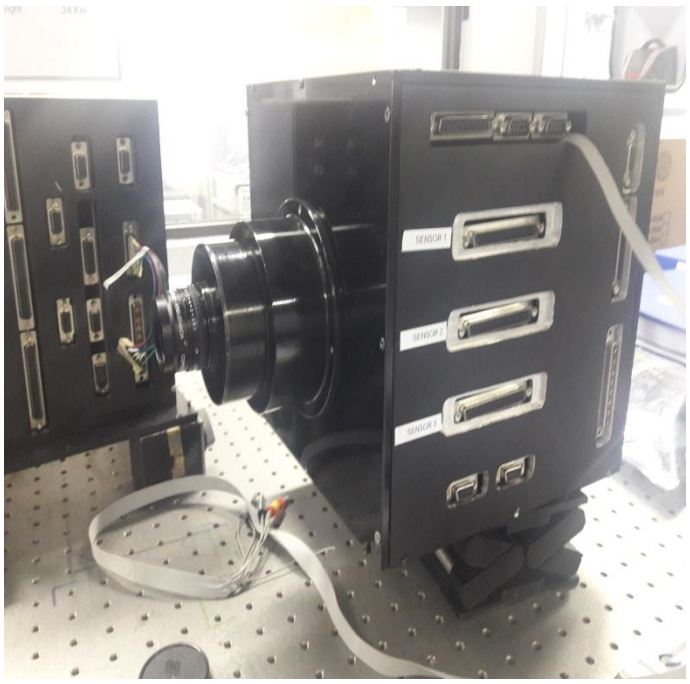


Figure 4.9 Camera Housing with lenses

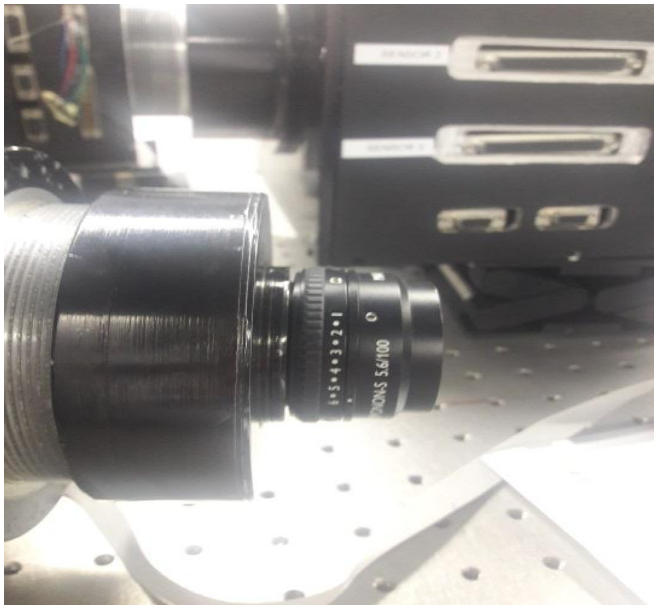


Figure 2.10 Lenses for sensor

These three PCB’s are placed in housing developed for testing facility of the cameras. Camera testing platform in formed containing bi-slide movable system for line scanning of camera using Fresnel lens and high capacity LED lights as shown in Figure 4.9.

Test platform is used to scan minerals and stone, as the hyper spectral is line scan camera, it scans complete area of image line by line so image scanning set up is developed as shown in figure 4.11 in which linear movable bi slider is used to move the minerals below the lens of the camera .

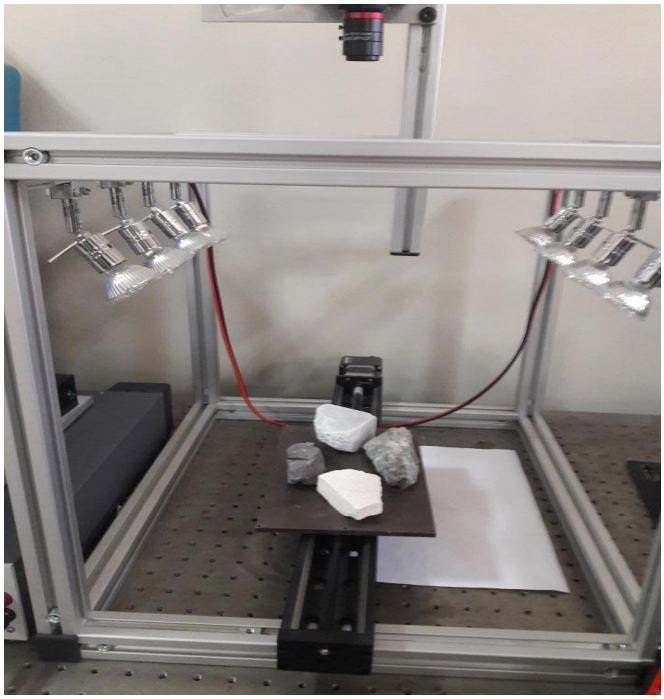


Figure 3.11 Test Setup

5 Conclusion & Future Work

This chapter concludes the proposed work in this thesis and also recommends future research directions in this field.

5.1 Conclusion

Purpose behind this research project was to develop a low cost and compact hyper spectral camera that can be used to plot the spectrograph of all materials e.g. minerals and vegetation with their real time imagery. To develop hyper spectral camera in which extra optical elements will be removed to make it more compact and cost effective by using latest optical filter technique which is Fabry-Perot integrated sensor. The intended purpose was achieved by using CMOSIS sensor containing thin film optical filters to break the light in to several bands and focus the required wavelength of the light to the corresponding pixel of the sensor, output channels of the sensor is restricted to 4 LVDS outputs to minimize the resources of FPGA. FPGA is programmed to convert the incoming data of the sensor to the required format of the data for camera link IC's and frame grabber card to develop the real time image. Imec hyper spectral camera software is outsourced to create the raw data of the image and calculate the reflectance of the image for each pixel of the sensor. At last MATLAB code is developed to create the spectrograph of all the minerals and vegetation available for testing at laboratory.

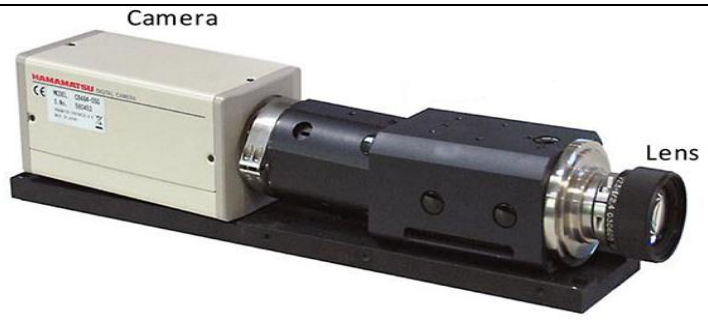
5.1.1 Cost Effective

Hyper spectral cameras with large optical elements are very much expensive due to their large camera structure and design. Cameras available commercially with spectral filters implemented sensors are less expensive as compared to old versions but they are expensive for countries like us which are underdeveloped. Hyper spectral camera developed in our laboratory is very much cheaper as compared to both the version that discussed above.

Table 4.1 Cost Comparison		
Old versions with optical elements.	Camera with spectral filters commercially available	Camera developed
In Rupees (Millions) Approx. 1.5 Million Rupees	In Rupees (Millions) Approx. 0.6 Million Rupees	In Rupees (Millions) Cost for Sensor = Rs 0.13 M Cost for FPGA = Rs 0.034 M Cost for Camera Link =Rs 0.02 M Cost for Frame Grabber =Rs 0.05M Cost for PCB's and Housing/Assembly = Rs 0.038M Total Cost =Rs 0.272M

5.1.2 Compactness

Hyper spectral camera developed at our laboratory is much more compact as compared to the versions available with large number of optical components required for their assembly.

Table 4.2 Size Comparison	
 Dimensions: 66*189*104 mm	 Dimension: 44*66*88 mm

5.2 Future Work

This research project was based on airborne and laboratory version hyper spectral imaging camera. However, this camera can be modified for large applications like payload of remote sensing satellite by using CMOSIS sensor having large swath width (Area scanned by sensor on earth) and properly designed telescope which have better quality and technology for earth sensing. Not only hyper spectral cameras but multi spectral remote sensing satellites can be developed by using CMOSIS sensors developed for only three bands Red, Green and Blue.

In future, functionality of camera link IC can be developed with in FPGA. FGPA can be reprogramed to regenerate the bit streams coming out of FPGA will be in format required by the frame grabber card for two standards which are Base and Medium required one and two serial LVDS lines simultaneously of data containing 24bits. Resources of FPGA can be increased but camera design will be simple, reduce complexity and cost and increase the compactness of camera.

Hyper spectral camera is used in mineral and vegetation detection during air borne applications. Applications of hyper spectral camera can be enhanced by using it with proper additional support systems like LIDAR (Light detection and Ranging) and proper GPS (Global positioning system) which make it more effective to develop the mapping of the area scanned by the camera. Mapping of location will be helpful after imaging to locate the area containing minerals and substances that are useful for us. A proper library of signatures or spectrographs of different minerals is required to be developed like USGS (United States of Geological Survey) library to compare the signatures obtained by camera for minerals, stones and vegetation of our own land.

Bibliography

[1] Pisani, M., Zucco, M., Labate, D., & Molina, M. (2015, June). A new hyperspectral camera concept for space-borne application. In Metrology for Aerospace (MetroAeroSpace), 2015 IEEE (pp. 497-501). IEEE.

[2] Vora, P. L., Farrell, J. E., Tietz, J. D., & Brainard, D. H. (2012). Image capture: simulation of sensor responses from hyperspectral images. IEEE Transactions on Image Processing, 10(2), 307-316.

[3] Wang, L., Xiong, Z., Gao, D., Shi, G., Zeng, W., & Wu, F. (2015). High-speed hyperspectral video acquisition with a dual-camera architecture. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4942-4950).

[4] Zucco, M., Caricato, V., Egidi, A., & Pisani, M. (2015). A Hyperspectral Camera in the UVA Band. IEEE Transactions on Instrumentation and Measurement, 64(6), 1425-1430.

[5] Caricato, V., Egidi, A., Pisani, M., Zucco, M., & Zangirolami, M. (2014, August). A device for hyperspectral imaging in the UV. In Precision Electromagnetic Measurements (CPEM 2014), 2014 Conference on (pp. 706-707). IEEE.

[6] 6.Çırpıcı, U., Karaca, A. C., Ertürk, A., Güllü, M. K., & Ertürk, S. (2014, April). Design of a hyperspectral imaging spectrometer for visible and near infrared region applications. In 2014 22nd Signal Processing and Communications Applications Conference (SIU) (pp. 1407-1410). IEEE.

[7] Lai, K. W. C., Xi, N., Chen, H., Chen, L., & Song, B. (2012, October). Development of 3D hyperspectral camera using compressive sensing. In Sensors, 2012 IEEE (pp. 1-4). IEEE.

[8] Barducci, A., Guzzi, D., Lastri, C., Marcoionni, P., Nardino, V., & Pippi, I. (2012, July). Simulating the performance of the hyperspectral payload of the PRISMA mission. In 2012 IEEE International Geoscience and Remote Sensing Symposium (pp. 5013-5016). IEEE.

[9] Nie, Y., Zhou, J., & Wei, X. (2011, July). Design of a miniature hyper-spectral imager. In Multimedia Technology (ICMT), 2011 International Conference on (pp. 3482-3484). IEEE

[10] Van Nguyen, H., Banerjee, A., & Chellappa, R. (2014, June). Tracking via object reflectance using a hyperspectral video camera. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops (pp. 44-51). IEEE.

[11] Mao, H., Silva, K. D., Martyniuk, M., Antoszewski, J., Bumgarner, J., Nener, B. D., ... & Faraone, L. (2016). MEMS-Based Tunable Fabry–Perot Filters for Adaptive Multispectral Thermal Imaging. Journal of Microelectromechanical Systems, 25(1), 227-235.

[12] Lin, J., Tong, Q., Lei, Y., Xin, Z., Zhang, X., Ji, A., ... & Xie, C. (2016). An Arrayed Liquid Crystal Fabry–Perot Infrared Filter for Electrically Tunable Spectral Imaging Detection. IEEE Sensors Journal, 16(8), 2397-2403.

[13] Mirshafieyan, S. S., Guo, H., & Guo, J. (2016). Zeroth Order Fabry-Perot Resonance Enabled Strong Light Absorption in Ultra-thin Silicon Films on Different Metals and Its Application for Color Filters. IEEE Photonics Journal.

[14] Bogaerts, J., Lafaille, R., Borremans, M., Guo, J., Ceulemans, B., Meynants, G., ... & van der Groen, S. (2016, January). 6.3 105, 65mm2 391Mpixel CMOS image sensor with> 78dB dynamic range for airborne mapping applications. In 2016 IEEE International Solid-State Circuits Conference (ISSCC) (pp. 114-115).

Appendix

TEST BENCH Code and Schematics

Vhdl test bench created from schematic D:\FAIZAN\CMV2000\XILINX\CMV2000_2\CMV2000.sch - Mon Oct 31 09:05:27 2016

```
--
-- Notes:
-- 1) This test bench template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the unit under test.
-- Xilinx recommends that these types always be used for the top-level
-- I/O of a design in order to guarantee that the test bench will bind
-- correctly to the timing (post-route) simulation model.
-- 2) To use this template as your test bench, change the filename to any
-- name of your choice with the extension .vhd, and use the "Source->Add"
--  menu in Project Navigator to import the test bench. Then
-- edit the user defined section below, adding code to generate the
-- stimulus for your design.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY CMV2000_CMV2000_sch_tb IS
END CMV2000_CMV2000_sch_tb;
ARCHITECTURE behavioral OF CMV2000_CMV2000_sch_tb IS

  COMPONENT CMV2000
  PORT( S2      :       IN      STD_LOGIC;
        S3:     IN      STD_LOGIC;
        S4:     IN      STD_LOGIC;
        XCLK    :       IN      STD_LOGIC;
```

```
    CS:         IN      STD_LOGIC;
    Capture     :       OUT     STD_LOGIC;
    CLK_OUT1:           OUT     STD_LOGIC;
    S1:         IN      STD_LOGIC;
    Mediam_Data     :       OUT     STD_LOGIC_VECTOR (27 DOWNTO 0);
    Base_Data       :       OUT     STD_LOGIC_VECTOR (27 DOWNTO 0));
  END COMPONENT;

  SIGNAL S2     :       STD_LOGIC;
  SIGNAL S3     :       STD_LOGIC;
  SIGNAL S4     :       STD_LOGIC;
  SIGNAL XCLK   :       STD_LOGIC;
  SIGNAL CS     :       STD_LOGIC;
  SIGNAL Capture        :       STD_LOGIC;
  SIGNAL CLK_OUT1       :       STD_LOGIC;
  SIGNAL S1     :       STD_LOGIC;
  SIGNAL Mediam_Data:           STD_LOGIC_VECTOR (27 DOWNTO 0);
  SIGNAL Base_Data      :       STD_LOGIC_VECTOR (27 DOWNTO 0);

BEGIN

  UUT: CMV2000 PORT MAP(
                S2 => S2,
                S3 => S3,
                S4 => S4,
                XCLK => XCLK,
                CS => CS,
                Capture => Capture,
                CLK_OUT1 => CLK_OUT1,
                S1 => S1,
                Mediam_Data => Mediam_Data,
```

```
                    Base_Data => Base_Data
  );

-- *** Test Bench - User Defined Section ***

-- **  S1 Data Signal **

-- ** CLOCK ***

  Sensor_CLK : PROCESS
   BEGIN
   XCLK <= '1'; wait for 10 us;
   XCLK <= '0'; wait for 10 us;
   END PROCESS;

-- ** CONTROL SIGNAL **
  CONTROL_SIG : PROCESS
   BEGIN
--  Invalid XX0h
   CS <= '0'; wait for 40 us;
--  200h 207h 207h 207h 207h 207h
   CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
            CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
   CS <= '0'; wait for 10 us;
   CS <= '1'; wait for 10 us;
```

```
CS <= '0'; wait for 10 us;
CS <= '0'; wait for 10 us;

        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
CS <= '1'; wait for 10 us;
CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;

        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
CS <= '1'; wait for 10 us;
CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;

        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
```

```
        CS <= '1'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
CS <= '1'; wait for 10 us;
CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;

        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
CS <= '1'; wait for 10 us;
CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;

        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '1'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
        CS <= '0'; wait for 10 us;
```

```
            CS <= '0'; wait for 10 us;
            CS <= '0'; wait for 10 us;
    CS <= '1'; wait for 10 us;
    CS <= '0'; wait for 10 us;
            CS <= '0'; wait for 10 us;
    END PROCESS;

-- ** DATA SIGNAL 1 **
   S1_Data_Pattern : PROCESS
   BEGIN
--  invalid
    S1 <= '1'; wait for 10 us;
    S1 <= '0'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '0'; wait for 10 us;
--  FFFh 000H FFFh 000h FFFh 000h
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;
    S1 <= '1'; wait for 10 us;

    S1 <= '0'; wait for 10 us;
    S1 <= '0'; wait for 10 us;
```

```
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;

S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;

S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
```

```
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;

S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;
S1 <= '1'; wait for 10 us;

S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
S1 <= '0'; wait for 10 us;
```

```
    END PROCESS;

-- ** DATA SIGNAL 2 **
   S2_Data_Pattern : PROCESS
   BEGIN
--  invalid
    S2 <= '1'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
--  FFFh 000H FFFh 000h FFFh 000h
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;

    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
```

```
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;

S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;
S2 <= '1'; wait for 10 us;

S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
S2 <= '0'; wait for 10 us;
```

```
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;
    S2 <= '1'; wait for 10 us;

    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    S2 <= '0'; wait for 10 us;
    END PROCESS;

-- ** DATA SIGNAL 3 **
   S3_Data_Pattern : PROCESS
   BEGIN
```

```
-- invalid
   S3 <= '1'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
-- FFFh 001H 002h 003h Fh 004h
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;
   S3 <= '1'; wait for 10 us;

   S3 <= '1'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
   S3 <= '0'; wait for 10 us;
```

```
S3 <= '0'; wait for 10 us;
S3 <= '1'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;

S3 <= '1'; wait for 10 us;
S3 <= '1'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;

S3 <= '0'; wait for 10 us;
S3 <= '0'; wait for 10 us;
S3 <= '1'; wait for 10 us;
S3 <= '0'; wait for 10 us;
```

```
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;

    S3 <= '1'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '1'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    S3 <= '0'; wait for 10 us;
    END PROCESS;

-- ** DATA SIGNAL 4 **
   S4_Data_Pattern : PROCESS
   BEGIN
--  invalid
    S4 <= '1'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '1'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
```

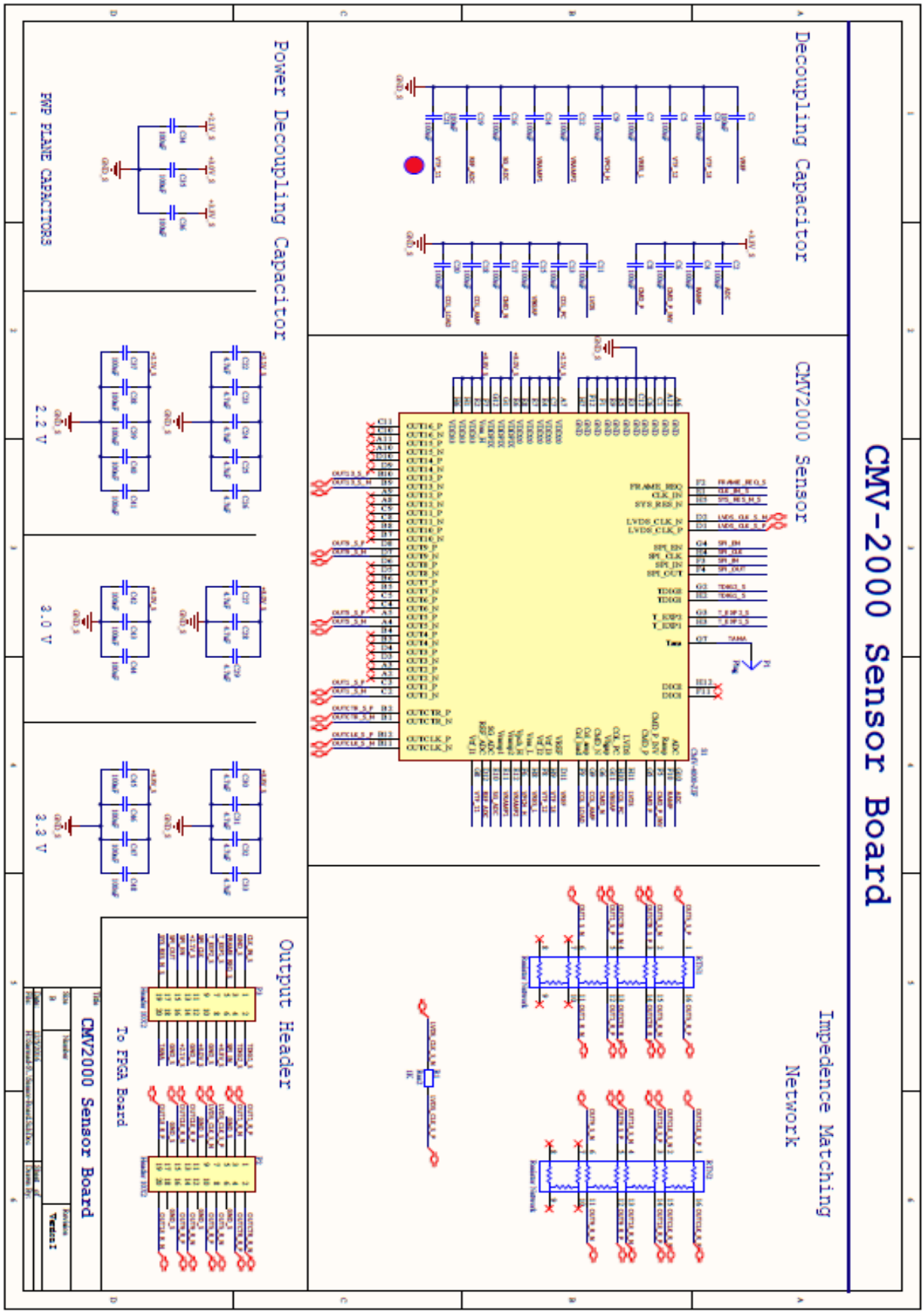
```
-- FFFh 000H FFFh 000h FFFh 000h
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;

   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;
   S4 <= '0'; wait for 10 us;

   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
   S4 <= '1'; wait for 10 us;
```

```
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;

S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;
S4 <= '0'; wait for 10 us;

S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
S4 <= '1'; wait for 10 us;
```

```
    S4 <= '1'; wait for 10 us;
    S4 <= '1'; wait for 10 us;
    S4 <= '1'; wait for 10 us;

    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    S4 <= '0'; wait for 10 us;
    END PROCESS;

-- *** End Test Bench - User Defined Section ***

END;
```

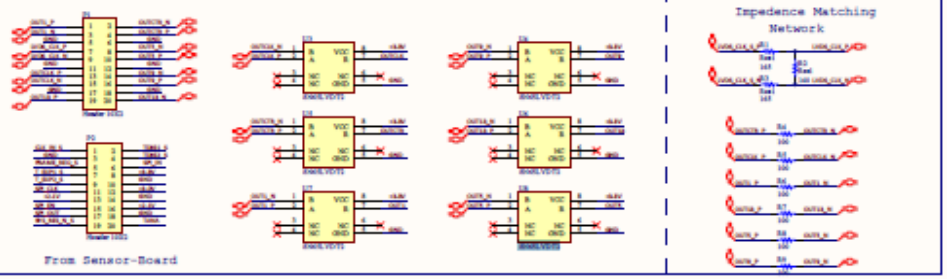
Code for MATLAB

```
clc
if exist('d')
clear d
end

if exist('k')
    k=k+1;
else
    k=1;
end
a=multibandread('mix 34561.raw',[2992 2048 100],...
    'int16',0,'bil','ieee-le',...
    {'Band','Direct',[25 50 75]});

imtool(a/max(max(max(a))))
clear a
msg=sprintf(' Select crop icon. Drag the rectangle on image. \n Right click
and copy position. \n Paste here then enter. \n');
b=(input(msg));
clc
% close all
figure
d=NaN(1,100);
if isnumeric(b) && ~isempty(b)
    for i=1:100
        a=multibandread('mix 34561.raw',[2992 2048 100],...
            'int16',0,'bil','ieee-le',...
            {'Band','Direct',i});
        c=a(b(2):(b(2)+b(4)),b(1):(b(1)+b(3)));
        d(i)=mean(mean(c));
        plot(d/max(d))
        axis([1 100 0 1.2])
        title([num2str(i),'%']);
        drawnow
    end
    title(['Its a normalize value multiply by ',num2str(max(d)),' factor to
get real magnitude']);
end
close all
clear b msg i c a
x(k,:)=d;
plot(x')
```


CMV-2000 Sensor Board
Decoupling Capacitor
CMV2000 Sensor
Impedence Matching Network
Power Decoupling Capacitor
2.2 V
3.0 V
3.3 V
EMP PLANE CAPACITORS
Output Header
To FPGA Board
CMV2000 Sensor Board

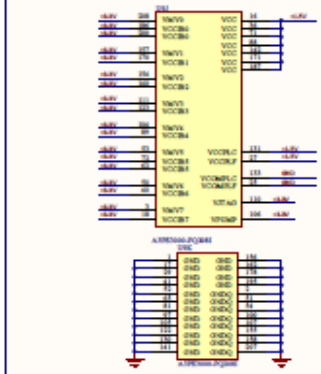
FPGA CONTROL BOARD CMV2000 CAMERA

SENSOR INPUT SECTION

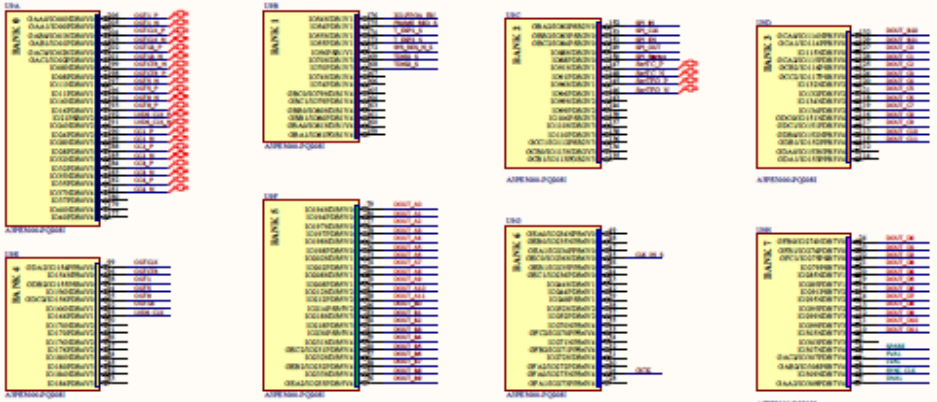


OUTPUT SECTION

FPGA SECTION

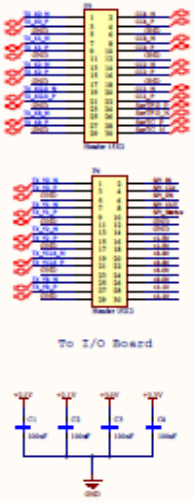


FPGA I/O SECTION

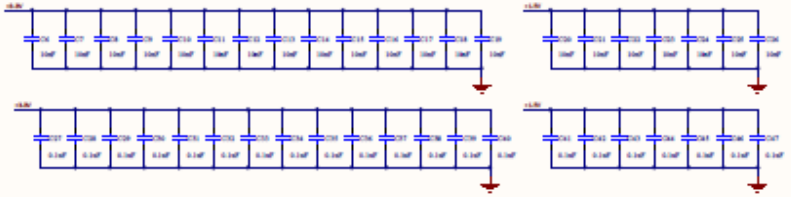


FPGA PROG

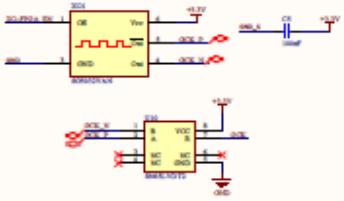
Power and Input/Output Interface



Decoupling Capacitors



Clock

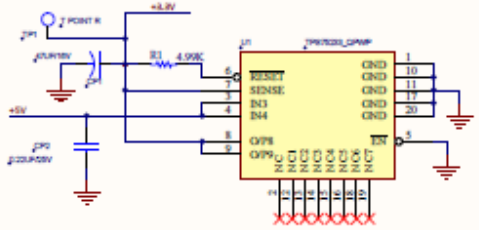


FPGA Control Board CMV2000 Cam

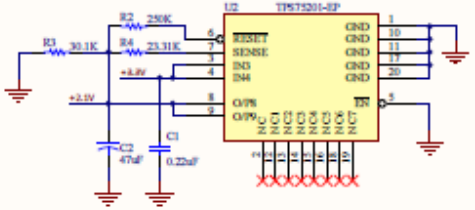
POWER-INPUT&OUTPUT BOARD CMV2000 CAMERA

POWER INPUT & REGULATOR

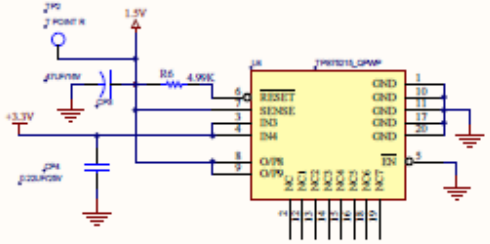
3.3 VOLTS REGULATOR



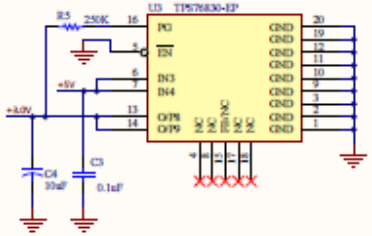
2.1 VOLTS REGULATOR



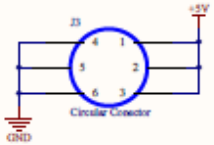
1.5 VOLTS REGULATOR



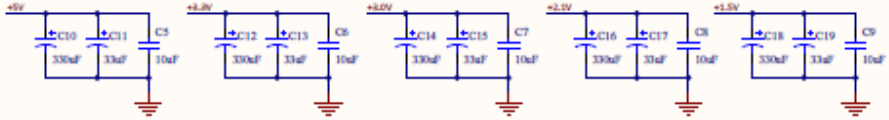
3.0 VOLTS REGULATOR



INPUT POWER

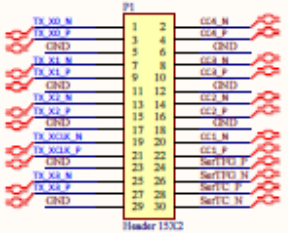


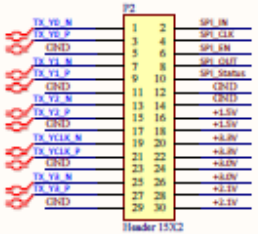
POWER FILTERING CAPACITORS

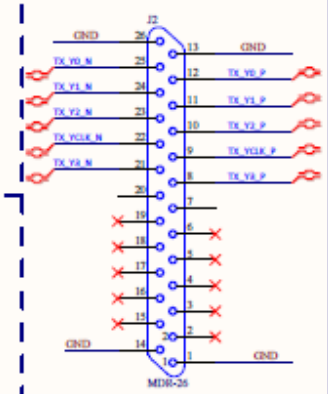


CAMERA LINK OUTPUT

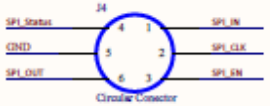
OUTPUT FROM CONTROL BOARD







EXTERNAL SPI COMM



Title: POWER-INPUT&OUTPUT BOARD		
Size: D	Number	Revision: Version I
Date: 12/5/2016		Sheet of
File: E:\Sarina\6\...\Power4D Board.SchDoc		Drawn By: