

Version

1

Camera Culture, MIT Media Lab

Time of Flight – Ultra Fast Imaging Camera

User Reference Manual

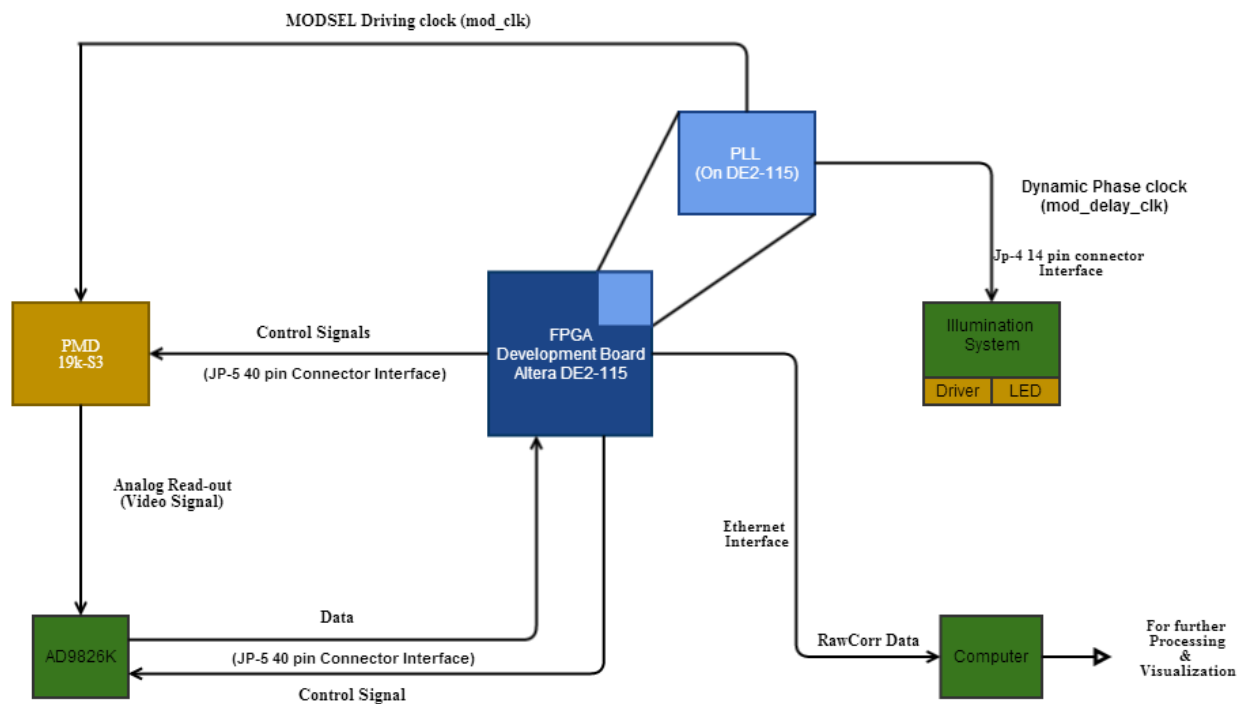
Table of Contents:

- 1. System Architecture**
 - 1.1. Block representation
 - 1.2. Brief Overview – Block wise
 - 1.3. Code database & Structure – Language wise
- 2. Printed Circuit Board (Sensor & Light Source)**
 - 2.1. Overview on PCB design files
 - 2.2. Circuit Diagrams & Schematics
 - 2.3. Eagle file Database
 - 2.4. Ordering from Sunstone
 - 2.5. Interfacing the PCB's to FPGA
- 3. Setting up Quartus ii Verilog programming Environment**
 - 3.1. Quartus Download & Setup
 - 3.2. Archived project – Importing and Setup
 - 3.3. Compilation and Verification
- 4. Setting up Nios ii hardware interface tool**
 - 4.1. Archived project – Importing and Setup
 - 4.2. Compilation and Verification
- 5. Setting up MATLAB programming Environment**
 - 5.1. Code Database – Scripts & Functions
 - 5.2. Setting the parameters
 - 5.3. Workflow & Representation
- 6. Calibration and Capture**
- 7. Appendix 1-2**

System Architecture

This part of the document explains the whole system architecture and signal flow within the Camera.

1.1. Block Diagram Representation



(Fig1: Block diagram of various sub elements)

1.2. Brief Overview – Block Wise

As seen in the Block diagram, the whole system comprises of 3 main blocks. In this section we will briefly describe each one of them and provide additional resources.

1.2.1. Altera DE2-115 Development kit:

The Altera® DE2-115 Development and Education board was designed by professors, for professors. It is an ideal vehicle for learning about digital logic, computer organization, and FPGAs. Featuring an Altera Cyclone® IV 4CE115 FPGA, the DE2-115 board is designed for university and college laboratory use. It is suitable for a wide range of exercises in courses on digital logic and computer organization, from simple tasks that illustrate fundamental concepts to advanced designs.

Official link to website and purchase information for Academic research:

<http://www.altera.com/education/univ/materials/boards/de2-115/unv-de2-115-board.html>

1.2.2. PMD 19K – S3:

The PMD PhotonICs® 19k-S3 is the first commercially available Time-of-Flight 3D chip. The optical sensor enables the real-time capture of distance and gray scale information. Due to the outstanding sensitivity and improved performance of the new PMD PhotonICs® 19k-S3, high accuracy with extremely low power consumption is achievable. Read-out clocks of 15MPixel/s are possible. Thanks to the integrated SBI (Suppression of Background Illumination) technology, a well-known PMD feature, this innovative product can be deployed in indoor and outdoor environments.

Official link to website:

http://www.pmdtec.com/products_services/pmd_photonics.php

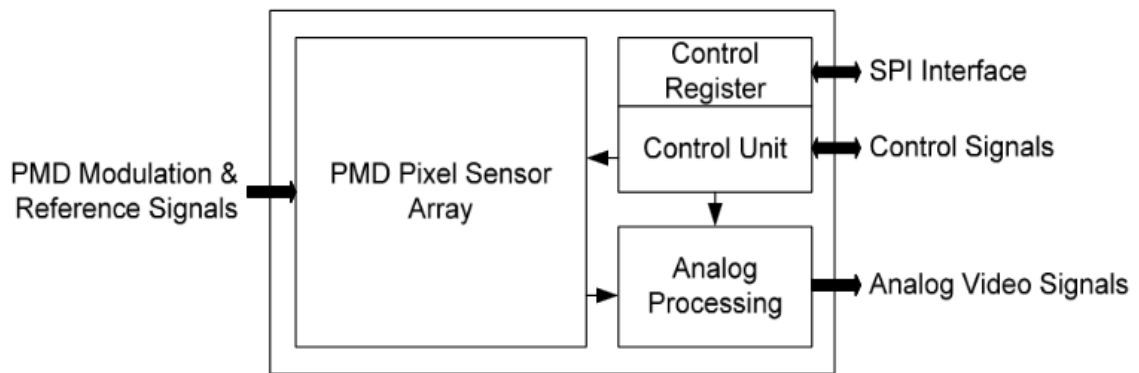


Figure 1: Architecture overview

1.2.3. Illumination System:

Illumination forms a very important part of time of flight imaging, and we are using a Laser source to drive our Camera system. We have used a single, LPC-826 Laser diode from Mitsubishi. These are High speed lasers which serves our purpose of bright illumination.

It is very important to choose an efficient driver to handle high current requirements of Laser diode. We used the iC- HG HG1D evaluation board from iC – Haus to efficiently drive the Laser.

1.2.4. Other peripherals and blocks:

- Other than these three blocks, AD9826K is an Analog to Digital converter used to achieve sampling and readout from the Sensor into the FPGA. Timings and operation is again controlled by the DE-115 board.
- Onboard PLL (DE2-115), is used for achieving phase and frequency modulations, thereby providing a phase lock system for the operation of Camera.

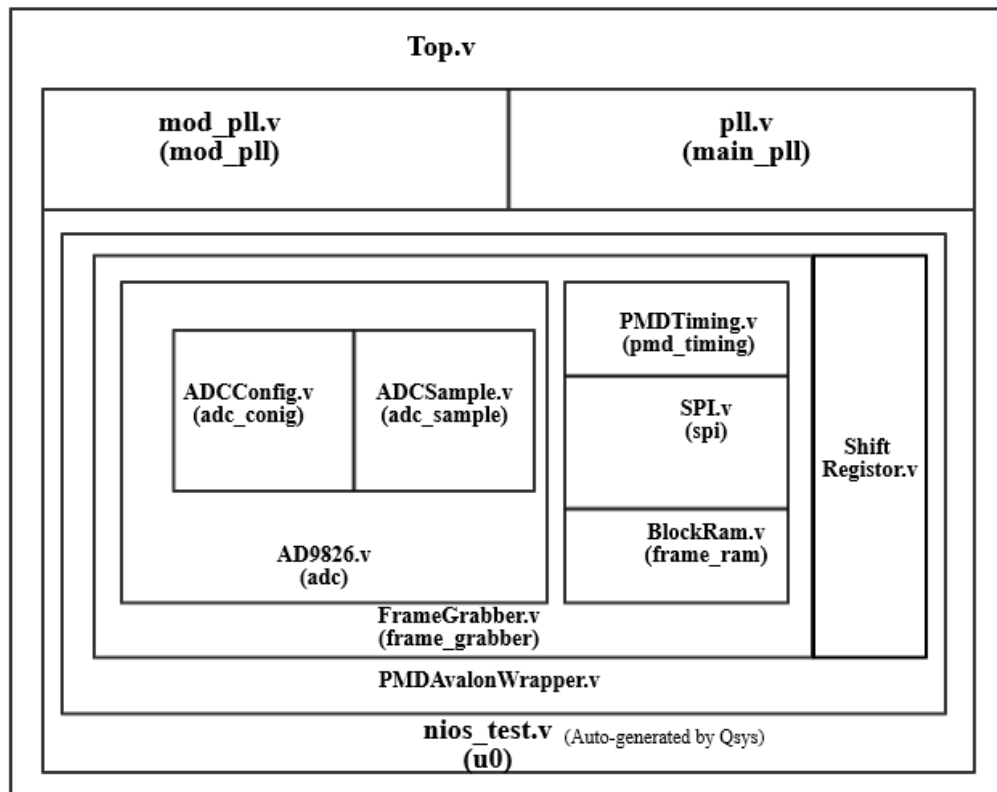
- A computer system forms a medium to configure all the peripherals and also to visualize the data captured from the camera. We use mainly three software packages for working with our camera.
 - a. Quartus ii FPGA programming interface
 - b. Nios ii Hardware tool package
 - c. MATLAB

1.3. Code Database:

This part of the document depicts various modules/functions/codes which control the overall data and control flow within the scope of the project.

1.3.1. Verilog modules:

Verilog modules establish an interface between PMD Sensor, ADC and FPGA board and controls their whole operation.



(Fig2: Verilog Module Flow)

1.3.2. Matlab scripts/Functions:

Matlab codes and functions helps in reading the data from the FPGA ethernet adapter into the Computer and Visualization of the read data.

1.3.3. C-codes Nios interface:

C-codes builds a framework to establish a TCP connection between the FPGA board and Computer. These codes program to the soft core processor onboard DE2-115 & help in memory allocation. Alongside that, these codes also helps user to set the Integration time without disturbing the Verilog codebase.

Here is a link to brief explanation of each module/ function:

<http://tinyurl.com/jguunla>

All the project codes are available here: Download as zip and unzip the files,

<https://github.com/vjs347/coded-time-of-flight-camera>

Printed Circuit Board

(Sensor & Light Source)

This part of the Document illustrates the PCB layouts & Circuit diagrams of the light source and Sensor used for the Camera. This forms the first step towards building your own time of flight imaging device.

2.1. Overview:

Circuit Diagrams/Schematics & PCB Design forms the first step towards building any electronic gadget. In our Camera, we have two important PCB designs,

2.1.1. Sensor board Design:

The PMD 19K-S3 sensor from PMD Photonics, is a 40 pin IC. The terminals are interfaced with the ADC and the FPGA to control the timing and readout. It requires precision design capabilities and ideas in designing the whole sensor board. We have used AD9826K ADC for the readout.

2.1.2. Light Source design:

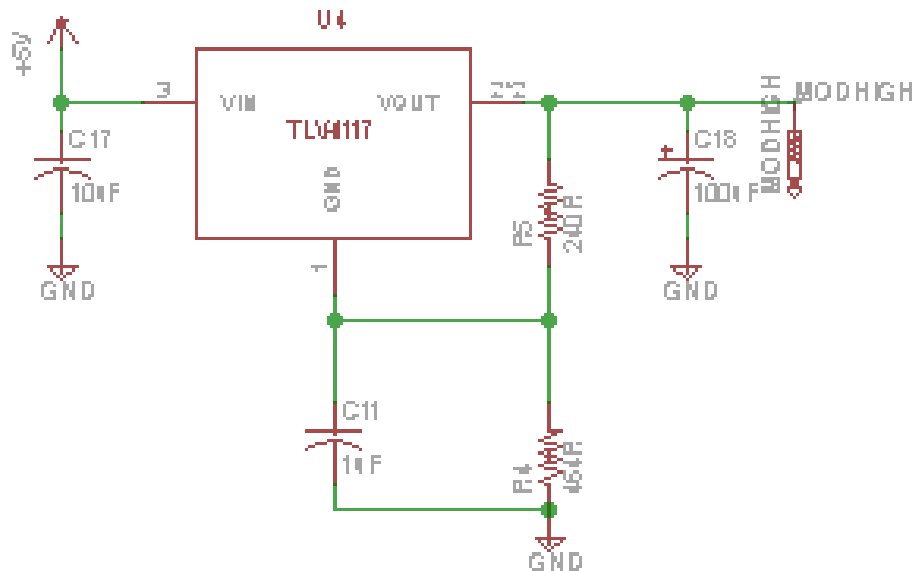
We have used a single, LPC-826 Laser diode from Mitsubishi. These are High speed lasers which serves our purpose of bright illumination. iC- Haus HG1D evaluation board is used to drive these High power Lasers.

2.2. Circuit Diagrams and Schematics:

In this section we briefly describe the Circuits/ Schematics used in our design. Appropriate images are added for reference.

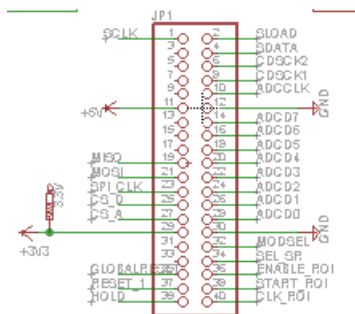
2.2.1. Sensor Schematic:

a. Voltage Regulator:



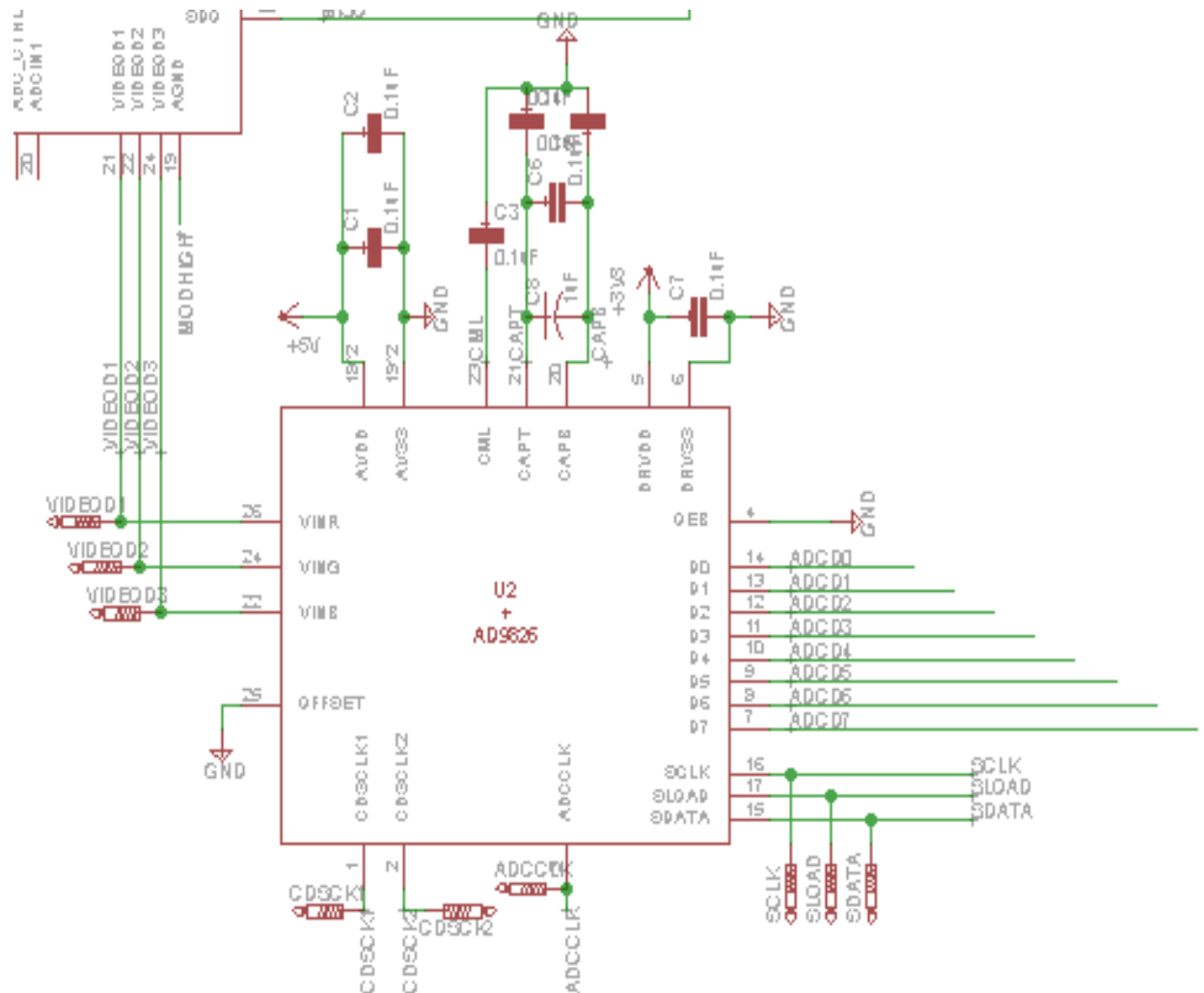
This forms the Voltage regulator circuit to drive/power the PMD Sensor board from the FPGA.

b. 40 Pin connector:



This forms the 40 pin Connector which is the main interface bus between DE2-115 and Sensor board.

c. ADC Circuit:



ADC here works in Sample and hold mode and helps in sampling the analog readout from the sensor and pushes the data into the FPGA through the 40 pin connector.

In the figure the terminals VIDEO01, VIDEO02, VIDEO03 are the input signals into the ADC and the terminals named ADC00.....ADC07 frames the output of the ADC data.

The whole Schematic image could be found at:

Vijay Sadashivaiah | Diony Rosa | Achuta Kadambi

2.2.2. Light Source Schematic:

The Schematic and the mapping for HG1D evaluation board can be found here.

http://www.ichaus.de/upload/pdf/HG_datasheet_B2en.pdf

2.3. Eagle File Database:

All the schematics and board design files (both schematic and board) can be downloaded at:

https://github.com/vjs347/coded-time-of-flight-camera/tree/master/_PCB/Light_Source/New

2.4. Ordering from Sunstone:

Sunstone Circuits is one of the most experienced PCB manufacturers in the USA. All the orders can be placed online by uploading the Schematics and the board files.

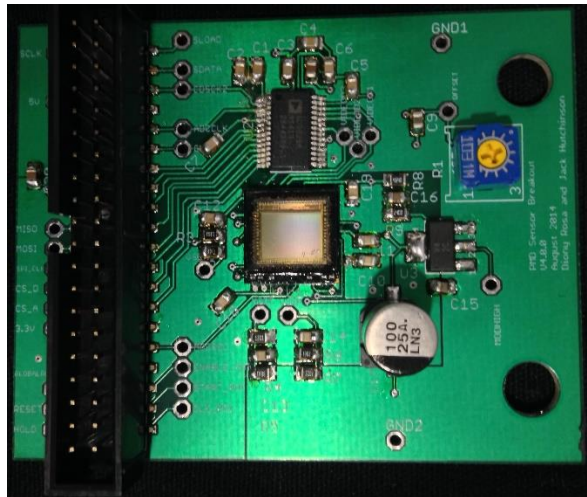
Official link: www.sunstone.com

2.5. Interfacing the PCB and FPGA:

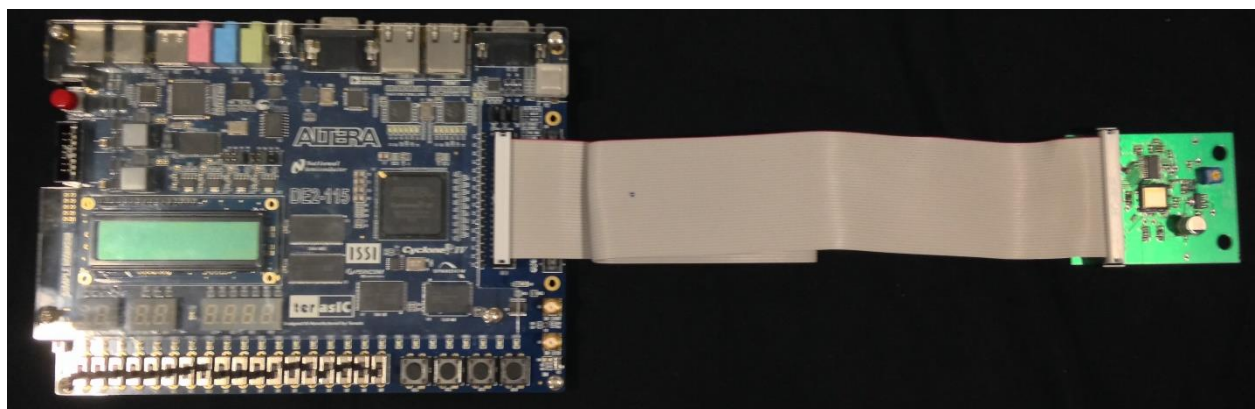
This section illustrates the steps to be followed to wire up the Sensor board & Light source to the FPGA.

Steps to be followed:

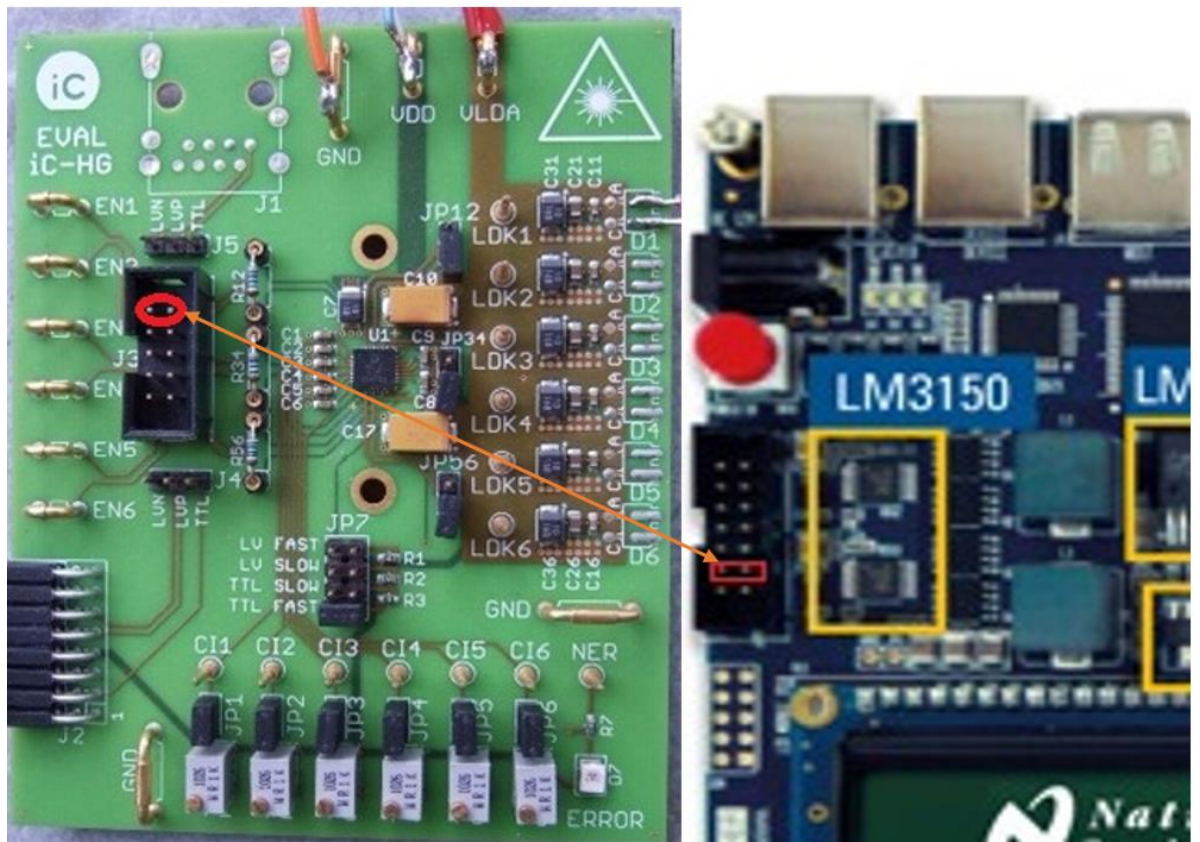
1. Once the PCB's are fabricated, and soldered with all the components as in the schematic, the Sensor should look something like this.



2. Now attach the 40 pin connector to the port in the image and connect the other end to the 40 pin connector on FPGA Development kit. After the connection is done, the system should look like,



3. Once this is done, now we need to hook up the light source, to do this, connect the terminals circled in red on Light source board, to the pins on DE2-115 marked red in the adjacent diagram.



Power up the Evaluation board with plugging VDD and VLDA to ~4V. This completes the whole process of integrating light source and Sensor board to the FPGA.

Setting up Quartus Environment

(Verilog HDL programming)

This section deals with setting up the whole Verilog environment on Quartus ii software package. Quartus ii is a software platform to develop Verilog projects and provides all the toolbox for its integration with Altera FPGA kits.

Here we use Altera DE2-115, Cyclone 4 E FPGA which performs these important tasks,

- Control all the timing requirements of PMD 19K-S3 and ADC.
- Readout data from ADC and push it into the Computer.
- PLL to synthesize the required modulated clocks.

3.1. Quartus Download and Setup / Compilation & Verification

This section helps in downloading the required Quartus version and installation. We prefer using Quartus ii 12.0 edition, since it is stable. Newer versions can also be used.

3.1.1. Quartus ii is a licensed software, but we can use a web edition for our requirements. To download Quartus ii web edition, go to link:

<https://www.altera.com/download/software/quartus-ii-we/12.0sp2>

ALTERA
MEASURABLE ADVANTAGE™

Search [] [Q]

Download Center Documentation myAltera Account

Devices Design Tools & Services End Markets Technology Training Support About Buy

Design Software
Quartus II Subscription Edition
Quartus II Web Edition
MegaCore IP Library
ModelSim-Altera
ModelSim-Altera Starter
Nios II EDS Legacy Tools
DSP Builder
Altera SDK for OpenCL
OS Support

Embedded Software
SoC RTOS and HWLIBs Support
SoC EDS

Archives
Service Packs
Design Software

Licensing
Get and Manage Licenses
Licensing FAQ
License Daemon Software

Programming Software
Quartus II Programmer
Jamt™ STAPL Software

Drivers
Cable & Adapter Drivers

Board System Design
JNEye
HSPIICE/IBIS-AMI Models
IBIS Models
Power Distribution Network

Board Layout and Test
BSDL Models
Schematic Review Worksheets
Layout Review Worksheets
Cadence PCB Libraries
Mentor Graphics PCB Libraries
Gerber Files

Legacy Software
MAX+PLUS II
Other Legacy Software

Please give us feedback

Quartus II Web Edition Software

Home > Support > Downloads > Quartus II Web Edition Software

Release date: August, 2012

Quartus II Web Edition v12.0 Service Pack 2

Select a previous version of Quartus II: 12.0 SP 2

Download Option 1: Individual Files

Quartus II Web Edition	Platform	File Name	Size
Quartus II Web Edition Service Pack 2	Windows	12.0sp2 quartus free windows.exe MD5: 3aa4e0871aaa29ffae5ecd3b42dd6a7b	2.8 GB
Quartus II Web Edition Service Pack 2	Linux	12.0sp2 quartus free linux.tar.gz MD5: eacce40008d33b5eca7277b1f822e621	3.7 GB

[What the MD5 sum value is and what it is for](#)

Other Individual Download Files:

- [Quartus II Subscription Edition](#)
- [ModelSim-Altera](#)
- [ModelSim-Altera Starter](#)
- [Nios II EDS](#)
- [DSP Builder](#)
- [Programming Software](#)

☒ Rate This Page

On the webpage download the required file depending on the platform. In this documentation we follow the installation on Windows PC. Although Linux will have a similar setup procedure.

3.1.2. Ways to setup Quartus environment:

There are two ways of setting up the whole environment,

1. Experience the taste of Verilog by setting up everything from Scratch! (We give the codes though).

To setup everything from scratch, follow this link:

Use the Setting up Verilog from Scratch.docx

2. Using the archived project to directly set up the environment.

In this document we depict the procedure to setup the Verilog environment from the Archived project.

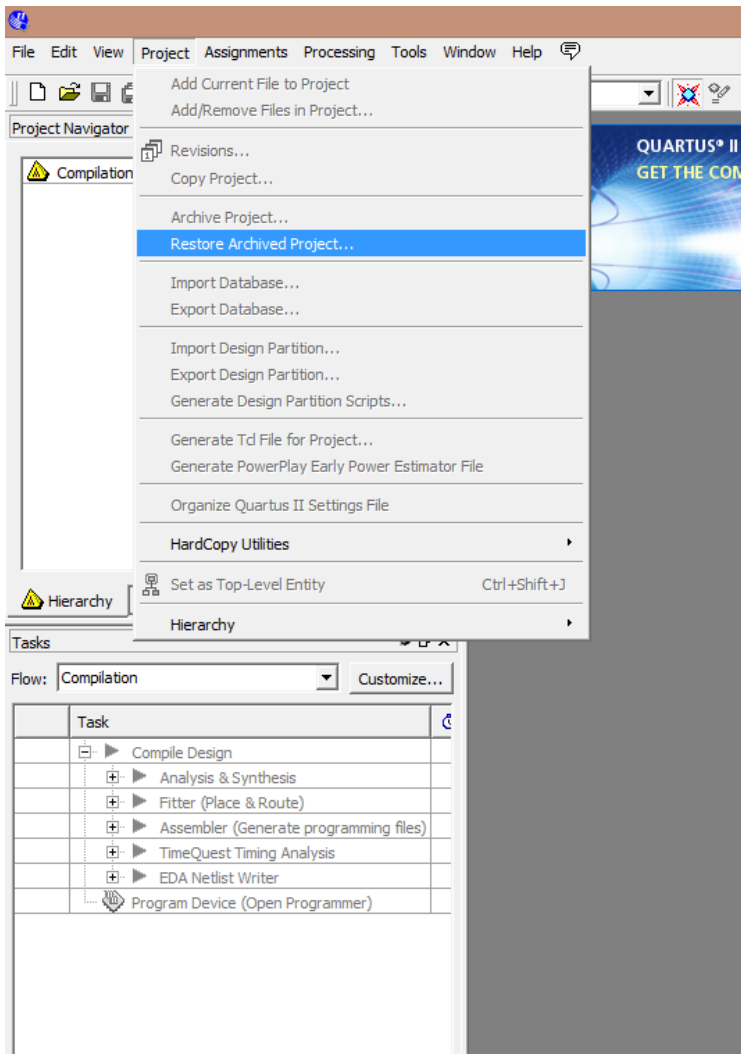
3.2. Archived Project – Import & Setup:

To ease the process of setting up the Verilog environment on Quartus ii, we have created an archive file of the project. This section will illustrate how to download → Import → Setup the project database.

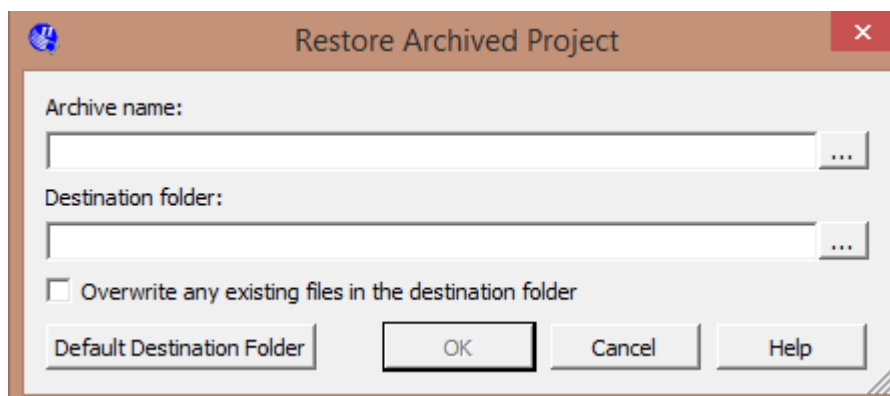
Steps to be followed:

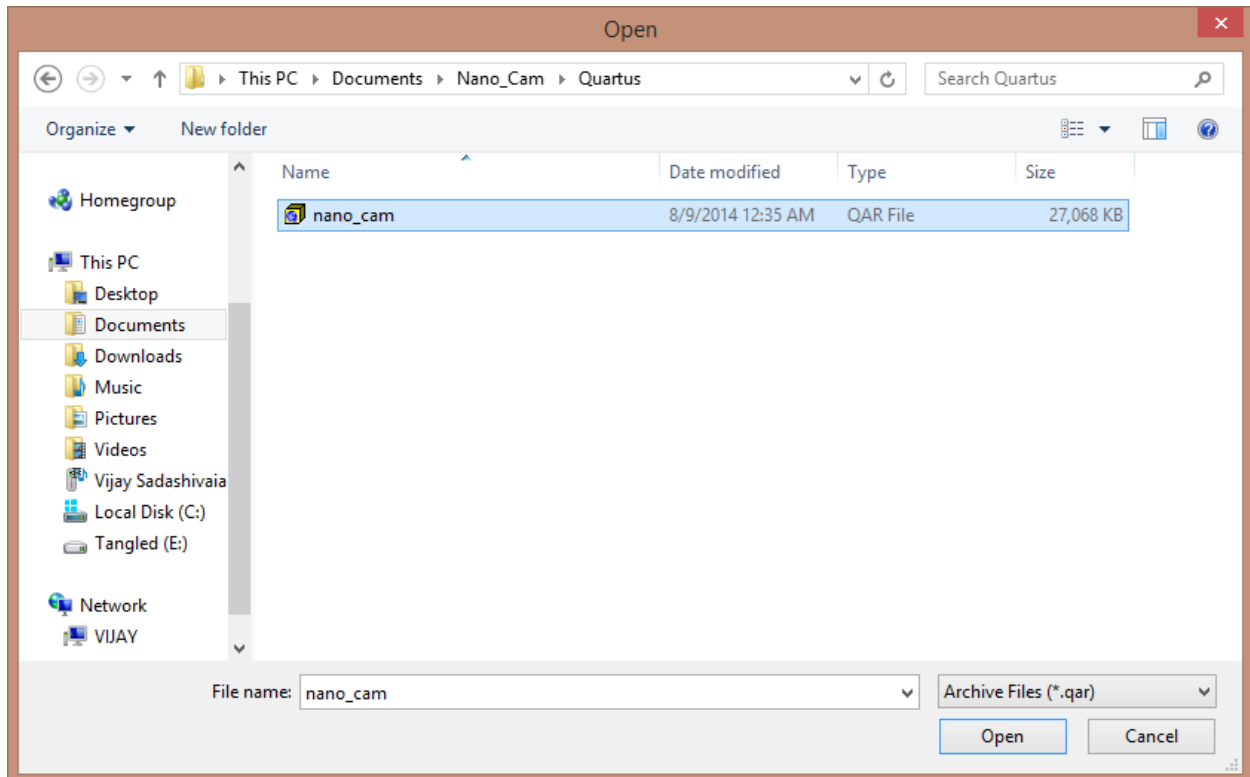
1. To download the archive project, navigate to the below link
https://github.com/vjs347/coded-time-of-flight-camera/tree/master/_Codes/_Verilog
2. Now download all the contents that are present in the “**Quartus**” and “**Verilog_files**” directories and place them on any of your computer directory.
3. After this is done, now open the Quartus ii Software from your computer.
4. Now close the small pop-up that appears asking you to create a new project/open a project.
5. Now navigate to project tab on the top left,

Project → Restore Archived project

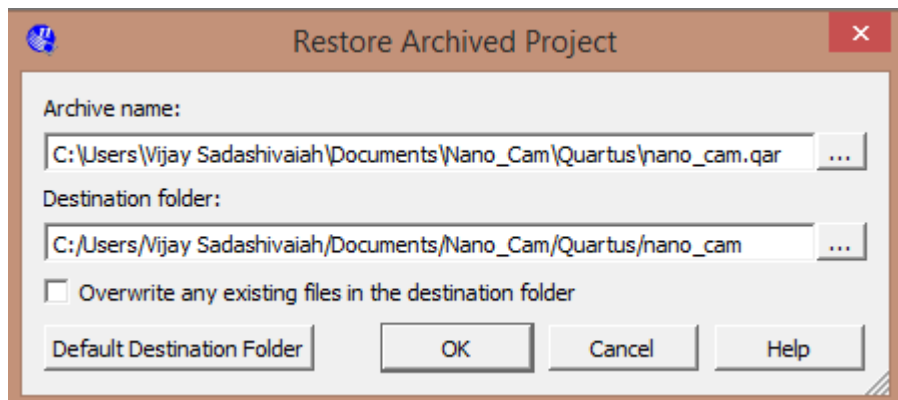


6. Now navigate to the path where you have downloaded the archive files.

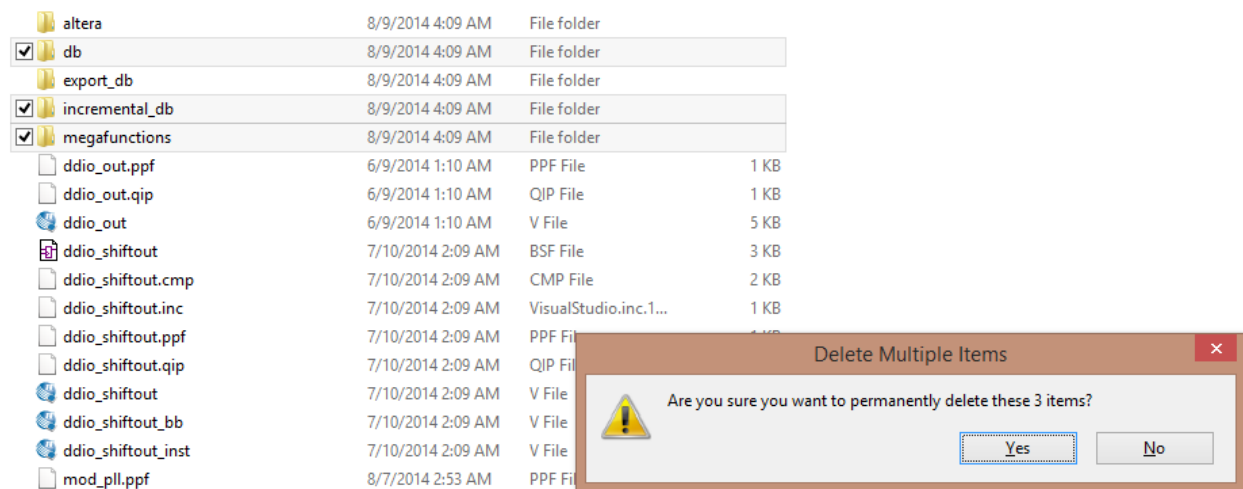




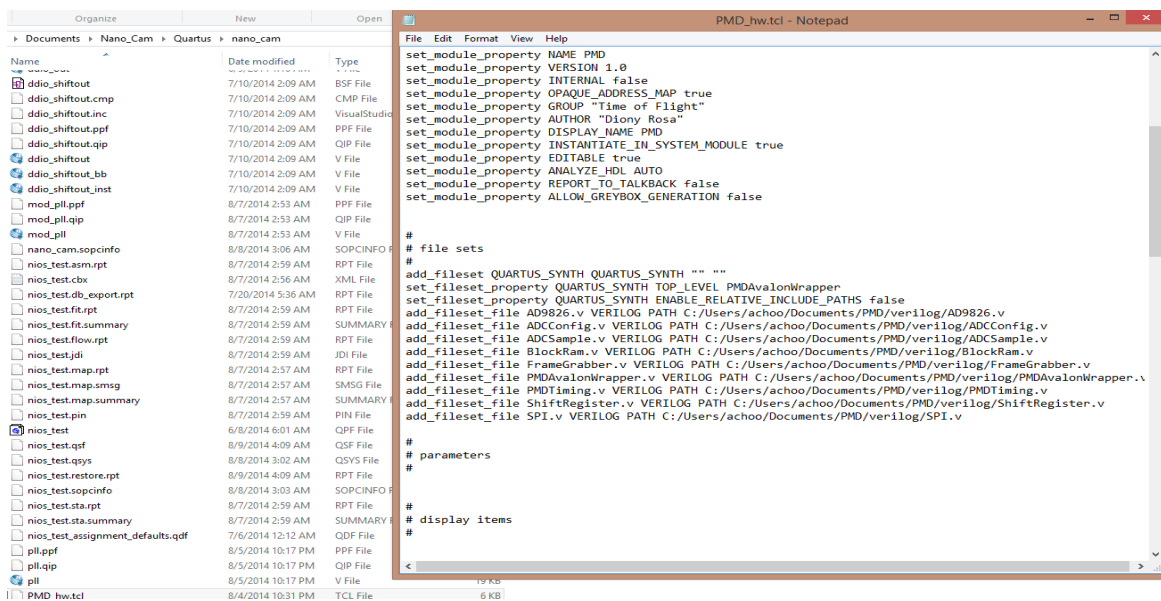
7. Now specify the required output directory and name on this window.



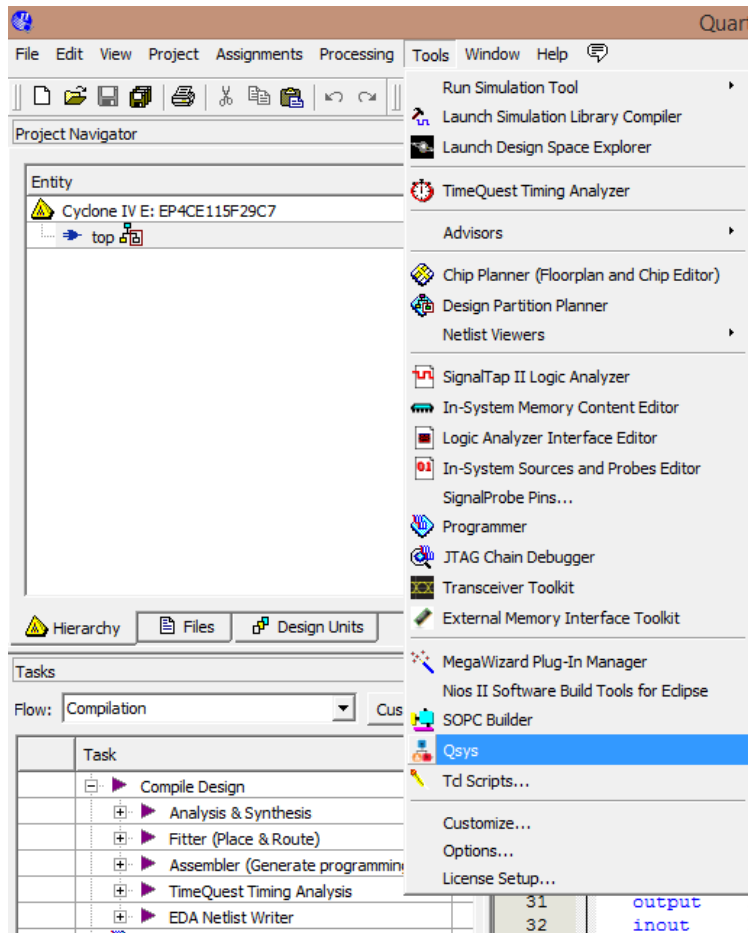
- After the project restoration is complete, now go to the path where you restored the project and delete the folders named “db”, “incremental_db” & megafunctions. (Note that you need to close Quartus window to delete these folders).



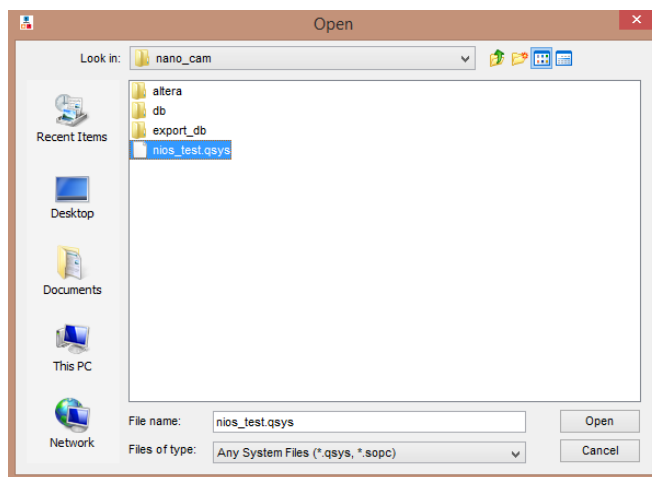
- In the same directory, open the file named “PMD_hw.tcl” and change the directory of files specified, to the path where you have saved the Folder “Verilog_files” from the google drive download link. (Note that you should use ‘/’ symbol and not ‘\’).



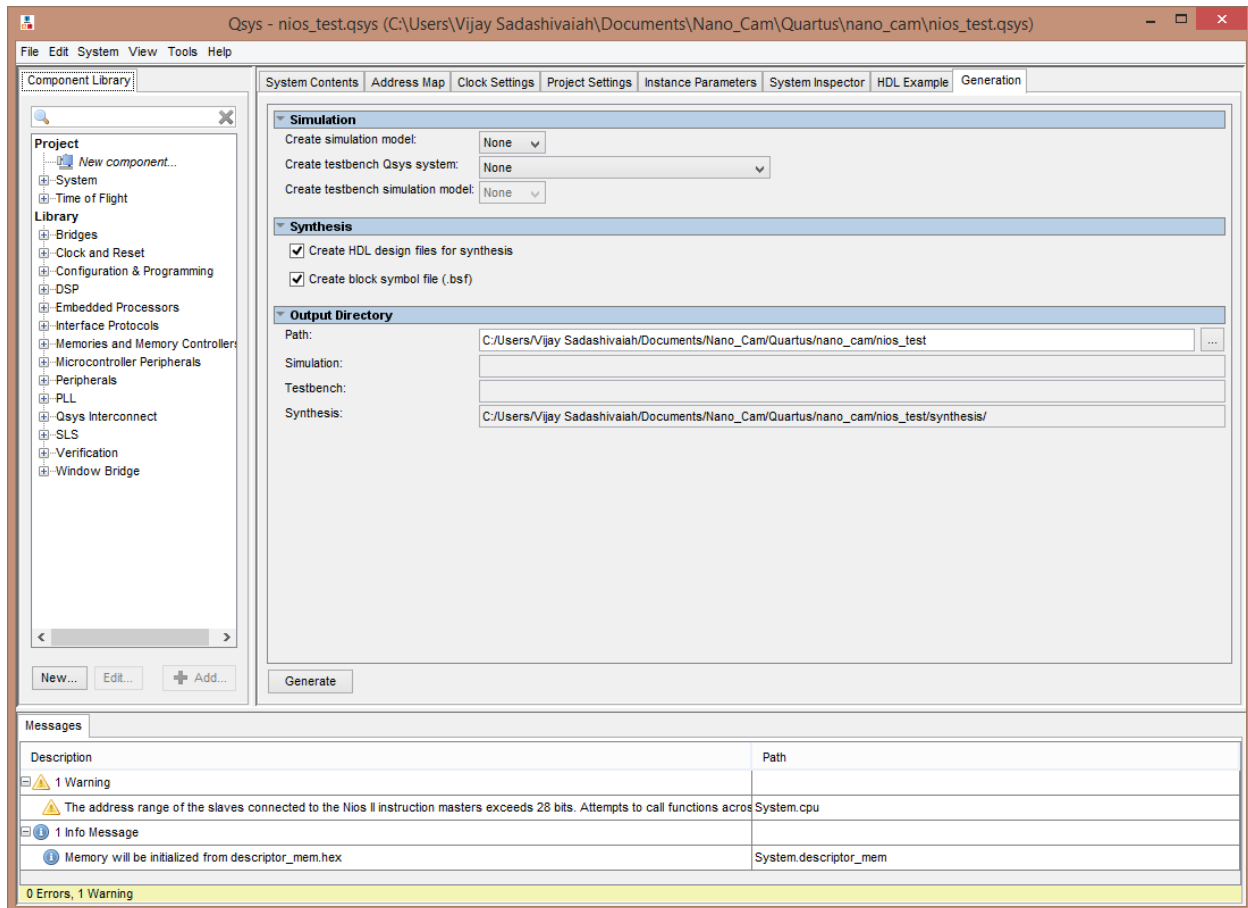
10. Now close that file and open the Quartus project again, now go to Tools → Qsys



11. Select the nios_test.qsys file from the window and click open.



12. Close the completion dialog and navigate into the Generate tab in Qsys.



13. Once generate is complete, save the System with (ctrl+S), and close Qsys.
14. Now compile the Verilog project by navigating into, Processing → Start Compilation.

Wait for the whole project to get compiled and synthesized. (This usually takes 5~6 minutes).

15. Once compilation is complete you can run the programmer by going to Tools → Programmer.
16. After opening the programmer, you can follow the instructions on this page to setup the hardware.

<http://www.altera.com/download/drivers/dri-quartus.html>

There might be some issues with the JTAG Server installation. To overcome this follow instruction on,

http://www.altera.com/support/kdb/solutions/rd02212013_169.html

(Make sure you start the JTAG Server by using command `jtagserver --start`)

➔ Finally even after all this, if the Hardware is not detected, Go to Control Panel ➔ Device Manager ➔ And make sure the USB-Blaster driver is installed. If not, install the drivers by navigating to the drivers folder in `C:/Altera/12.0/quartus/drivers`.

17. After the hardware is setup and detected, click on Start to program the FPGA!

These are the steps involved in setting up the whole Verilog environment. Now we can move on to programming the soft-core processor on Nios ii.

Setting up Nios ii Environment

(C-codes to program soft – core processor)

This section of the document depicts the procedure to setup the Nios ii platform to program the Soft – core Processor. Nios ii EDS package usually comes with the Quartus package itself.

Here the main purpose of Nios is to work as a medium to,

- Temporarily save the readout data from Sensor.
- Push this data via Ethernet from FPGA to the Computer.
- Also creates an active bus to pass few parameters from matlab programming console.

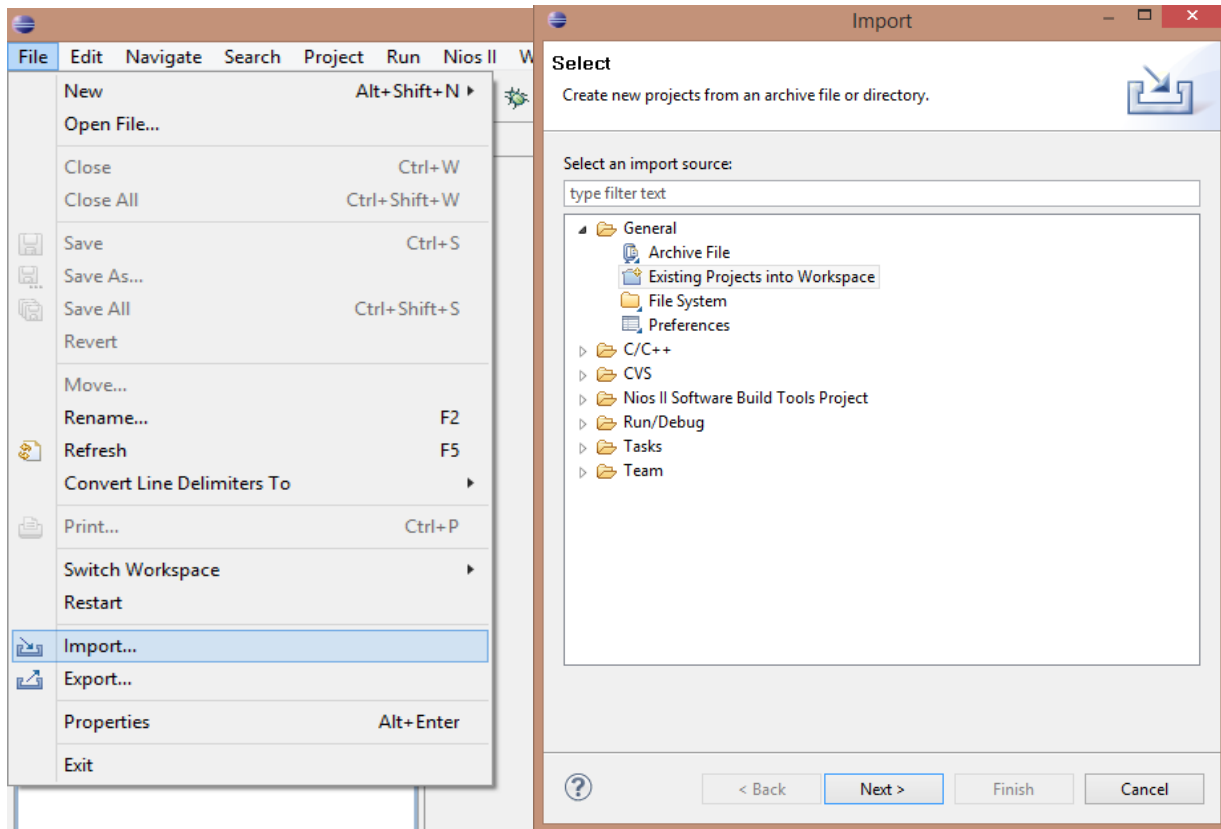
4.1. Archived Project – Import & Setup/Compilation

This section deals with the importing of Archived C code from the code database and setting up the Nios ii environment.

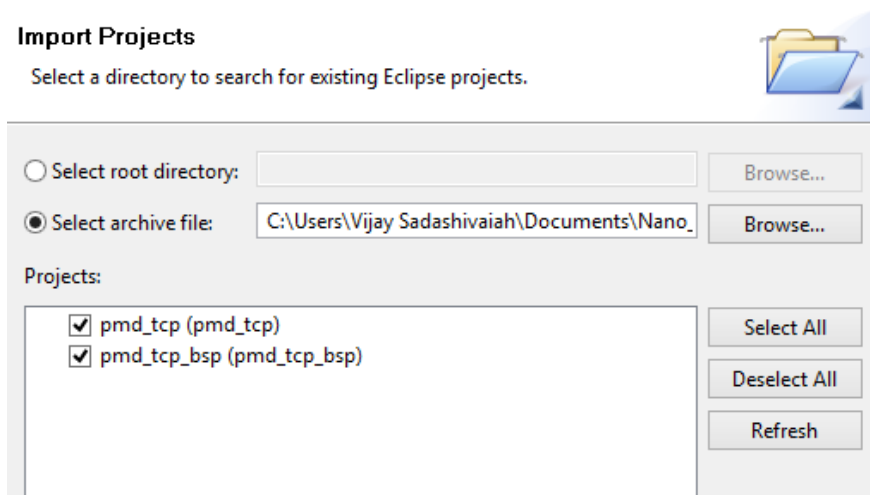
Steps to be followed:

1. Archived project is available on the link:
https://github.com/vjs347/coded-time-of-flight-camera/tree/master/_Codes/_C_Codes
2. Download the zip file named “Cfiles.zip”.
3. Now open the Nios ii from the Computer and specify your workspace to some path.

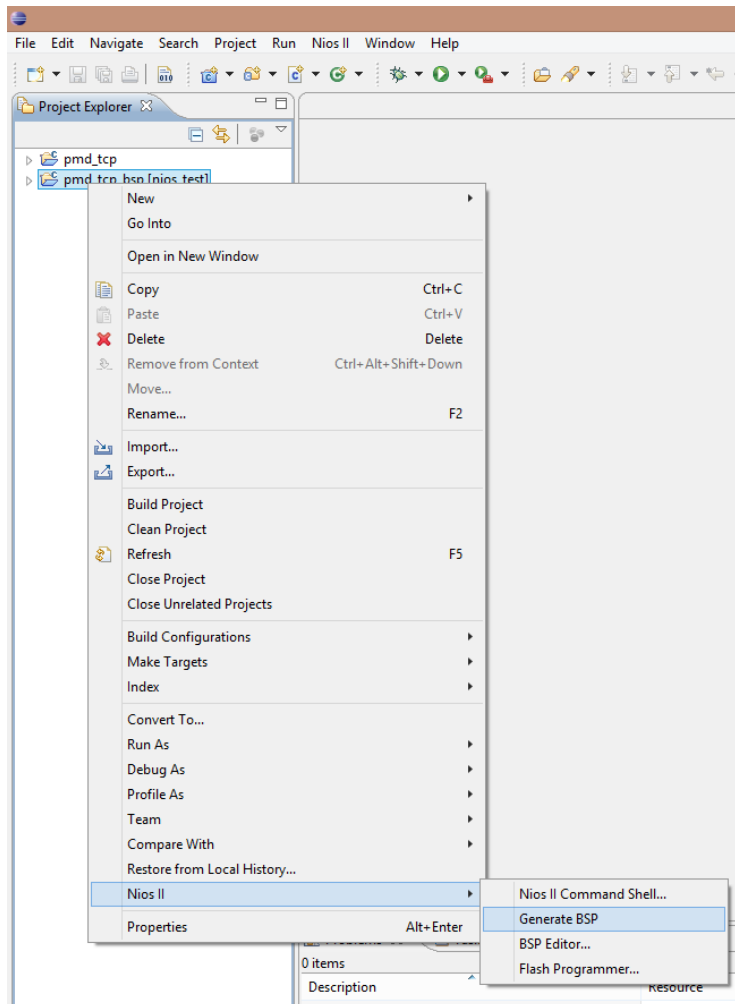
4. Navigate to File → import → General → Add existing projects to Workspace.



5. Now choose to import from Archive project and navigate to the path where Cfiles.zip was downloaded and click open.



6. Once the projects are imported, right click on pmd_tcp_bsp → Nios ii → Generate BSP.



7. Right click on the pmd_tcp_bsp → Build Project. Once the build is complete now, repeat this step to build even the pmd_tcp project.
8. After building pmd_tcp project, we need to run the pmd_tcp project, to do this, click on Run → Run. If a pop up asks for Run as, choose Nios ii Hardware.

➔ If there is an error popping out goto Run ➔ Run Configurations ➔ Nios
2 ➔ Target Devices Tab And check ignore mismatch system id and system
timestamp.

After these steps, the Nios ii will compile and dump the required code
into the soft – core processor and establishes a connecting with FPGA.
Now we can call the matlab functions / Scripts to capture some data!

Setting up MATLAB Programming Environment

(Matlab Scripts and Functions)

This section of the document depicts the last software package to be setup for the data capture. MATLAB is a programming environment created by MathWorks. We have written few MATLAB functions and Scripts to set parameters and capture data from the Camera.

MATLAB is a licensed software, and can be directly downloaded from the official website,

<http://www.mathworks.com/products/matlab/>

5.1. Code Database: Download and Setup:

All the matlab codes required for the Camera are available at the web link:

https://github.com/vjs347/coded-time-of-flight-camera/tree/master/_Codes/_Matlab

. Download all the files and save them in a directory.

5.2. Setting the Parameters:

This is an important section which determines where to set the parameters like Integration time, Custom code, Number of steps etc.

Here we depict this using code snippets and function calls,

5.2.1. Setting the Integration time:

```
function pmd_set_integration_time(conn, integration_time)

    fwrite(conn, 'W');
    fwrite(conn, 1, 'uint8');
    fwrite(conn, integration_time, 'uint32');

end
```

To set integration time, we can call the `pmd_set_integration_time` function from the command prompt. Above snippet indicates its operation.

5.2.2. Setting the Phase step size and number of averages:

```
function [frames] = pmd_600(steps, num_avgs)

if nargin<1
    steps = 10;
    disp( sprintf('Using %g Steps') );
end

if nargin<2
    num_avgs = 1;
end

conn = pmd_connect();
tmp = pmd_read_image(conn);

frames = zeros(size(tmp,1),size(tmp,2),steps);

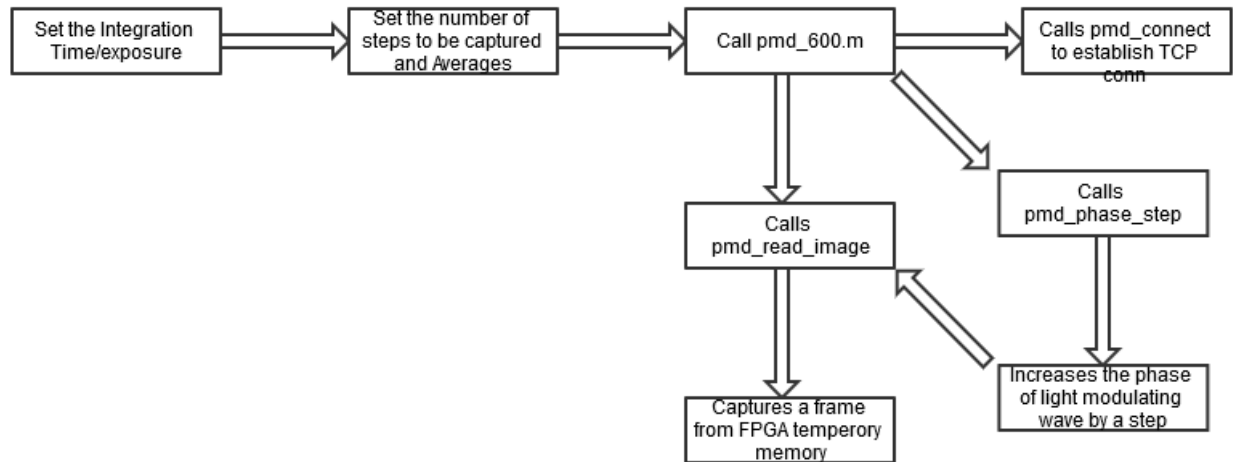
for ii = 1:steps
    pmd_phase_step(conn, 1);
    thisFrame = zeros(size(frames,1),size(frames,2),num_avgs);
    for jj=1:num_avgs
        thisFrame(:, :,jj) = pmd_read_image(conn);
    end

    thisFrame = mean(thisFrame,3);
    frames(:, :,ii) = thisFrame;
    disp(ii);
end
```

To set the Phase step size and the number of averages of every frame we can use this function, where in we can specify these values in the function call.

5.3. Workflow and Representation:

This section will indicate how the actual capture of data happens across the software packages. We will start with calling the matlab function `pmd_600.m`.



(Figure: The data and control flow within matlab)

With all the above procedure and setup, the Ultra-fast Camera is ready for a capture!

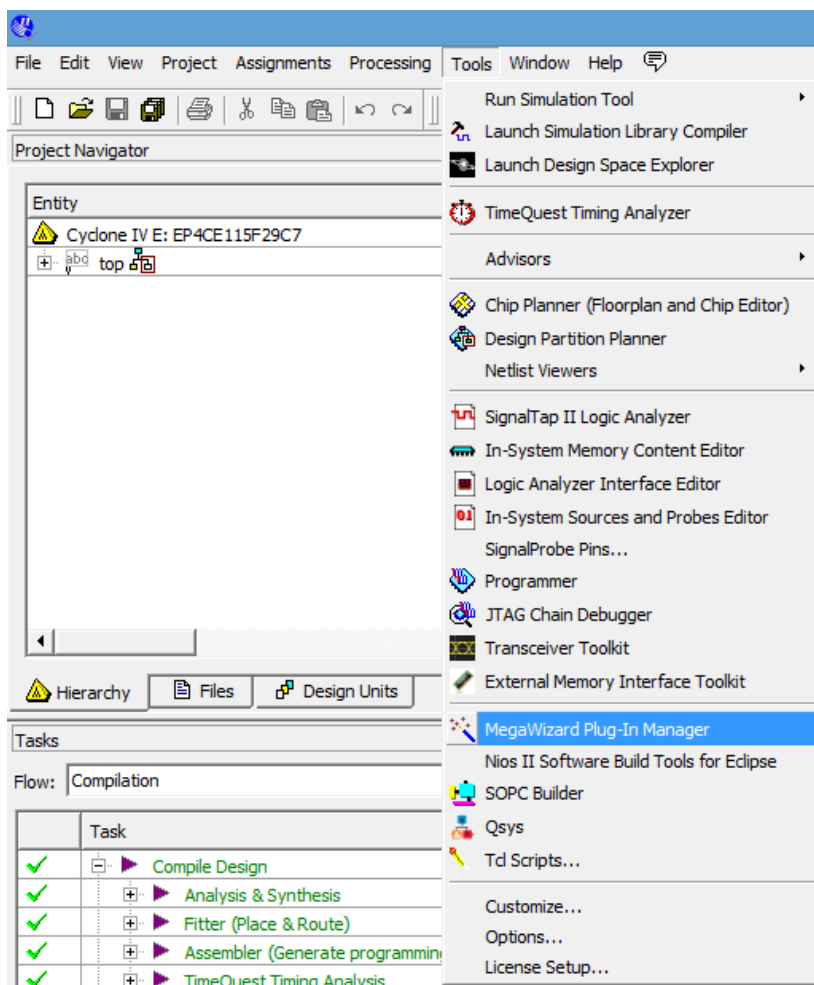
Appendix 1:

Setting the Modulating frequency:

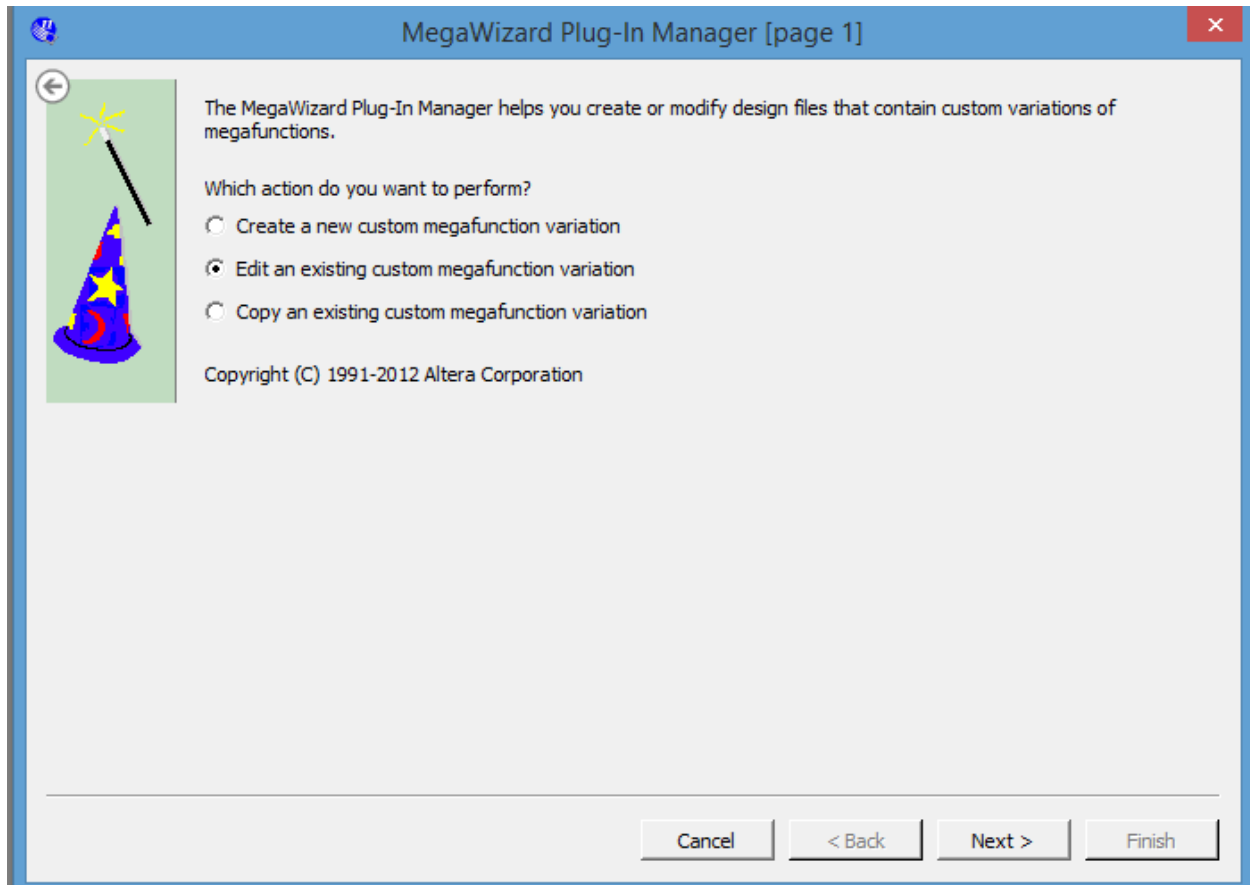
To set the modulation frequency of the signal driving light source and PMD in phase lock, we have to edit the Mega wizard plugin manager.

To do this,

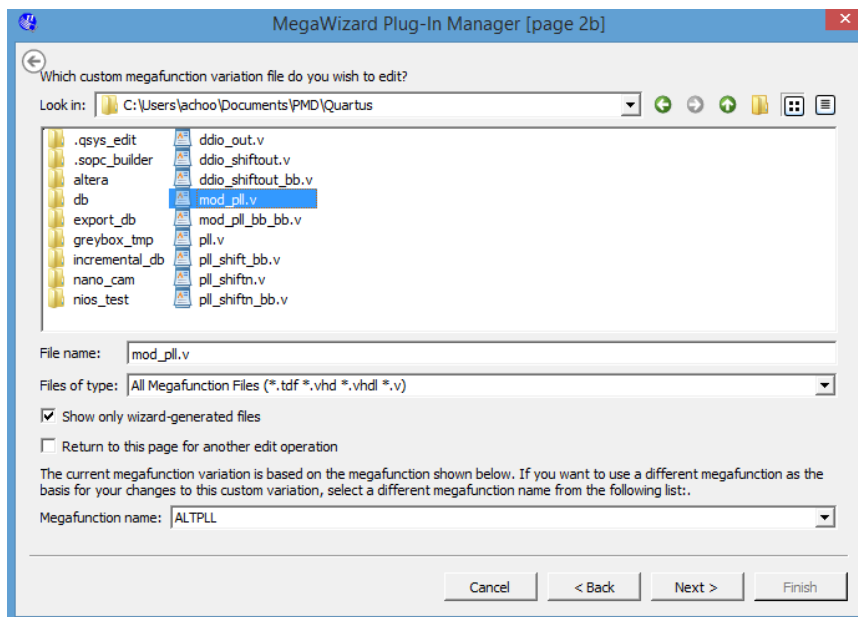
1. Open the Quartus project → Tools → Mega Wizard Plugin Manager



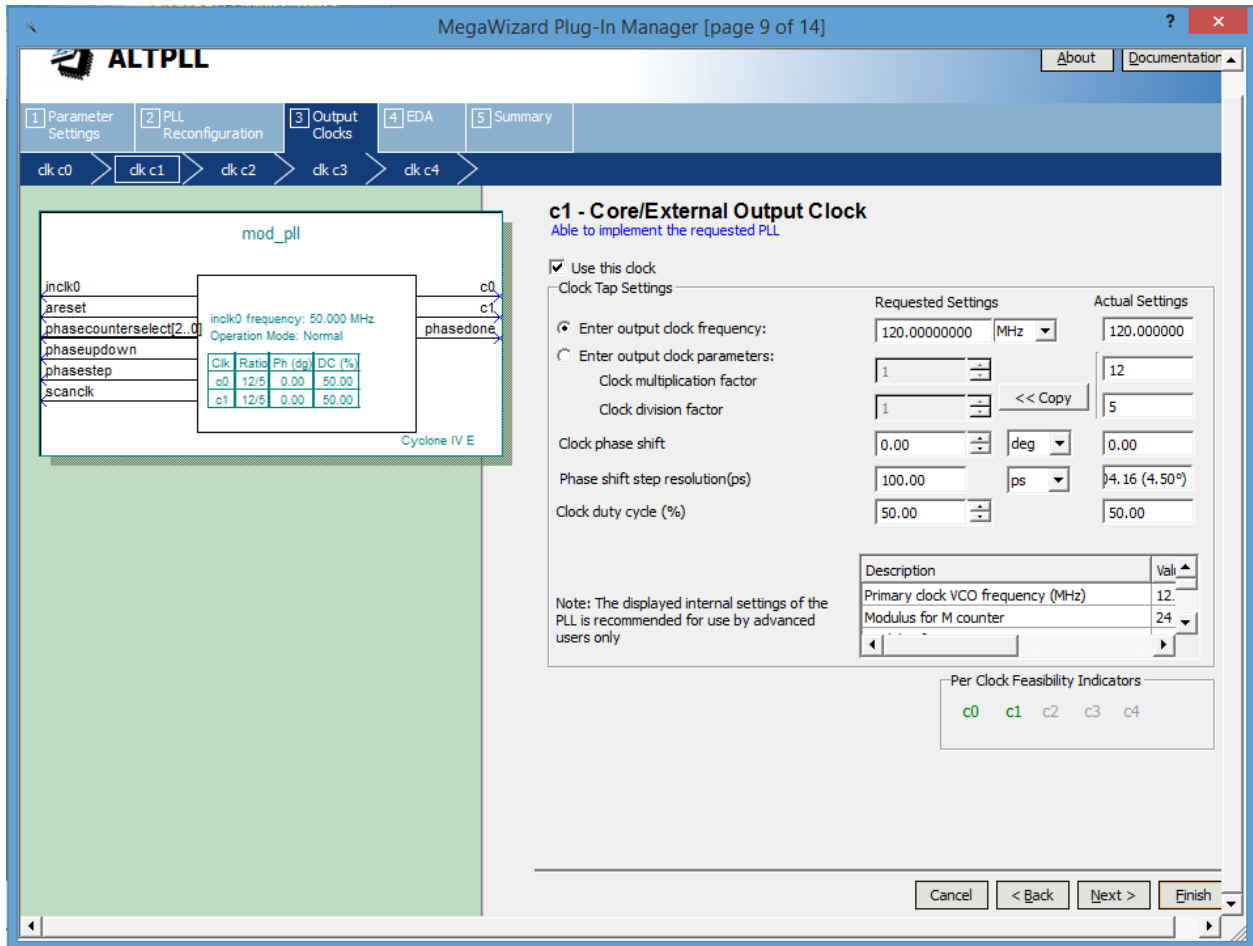
2. Edit existing custom megafunction variation.



3. Choose modpll.v from the list of design,

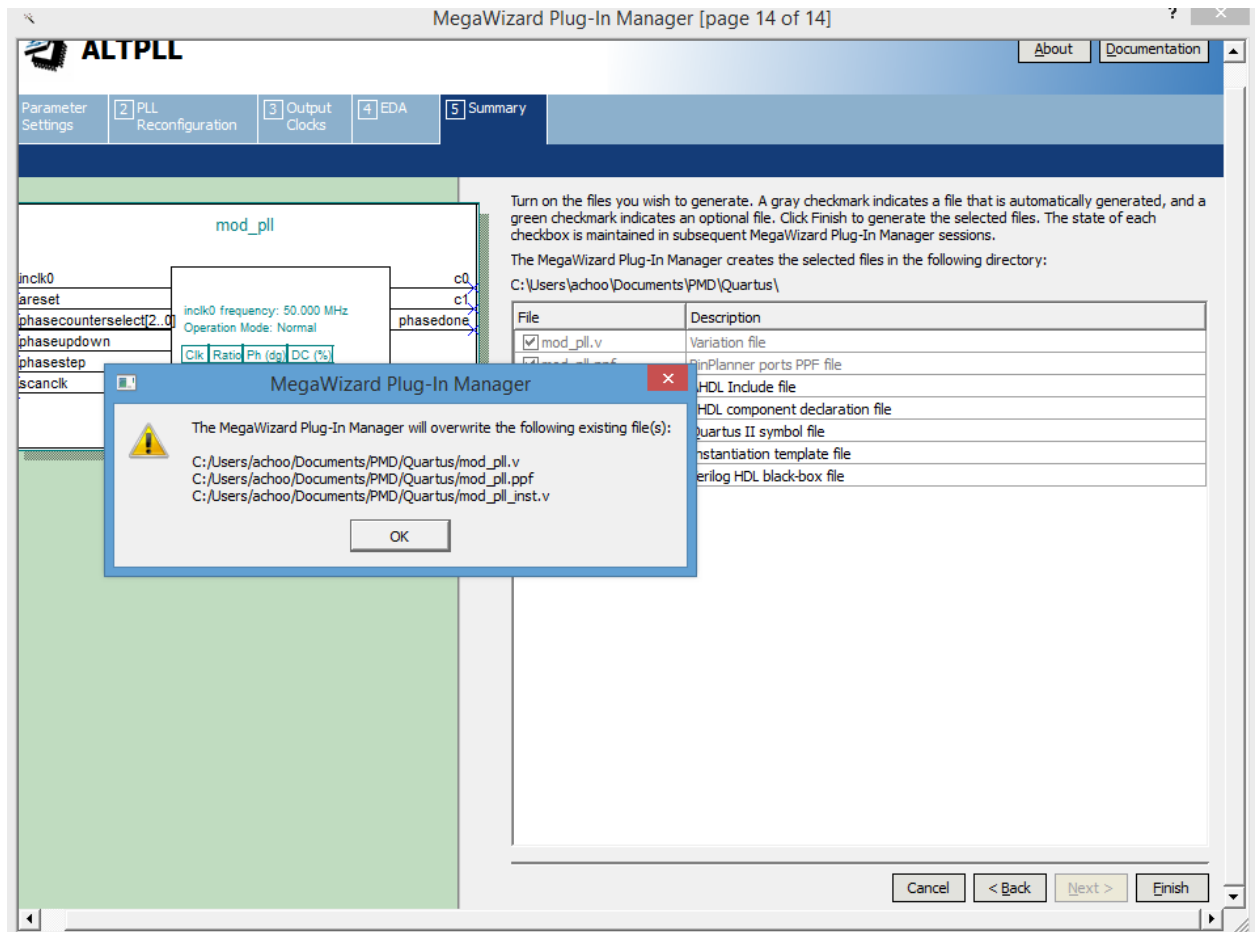


- Now navigate to the output clocks tab and substitute the value of frequency required. (Note that you need to give the same value to both c0 and c1 which drives MODSEL and Light modulation code respectively.)



(The value entered is twice the frequency required. i.e. if you need 60 MHz of modulation, you enter 120 MHz here).

- Click on finish and confirm the screen which says few files will be modified.



Once the modification is done, you need to recompile the Verilog design and upload it onto FPGA.

Appendix 2:

Setting the custom Modulation code:

To set the required modulation code, we will have to convert the binary code of 1's and 0's into hex notation and divide it into two equal halves. Say the custom code required is,

01011101100011111001101001000010. To easily simplify, go to www.wolffromalpha.com

And ask for the hex version of this code by writing in the box,

“01011101100011111001101001000010 to hex”. The output will be,

➔ 5d8f9a42₁₆

Dividing this into halves, we get 5d8f & 9a42.

To change the code into this custom value, navigate to the C code, hello_ucosii.c

```
void pmd_task(void* args) {  
    volatile PMDRegs* pmd_regs = (PMDRegs*)PMD_BASE;  
    volatile INT16U* pmd_mem = (INT16U*)(PMD_BASE + 0x8000*2);  
    set_integration_time(pmd_regs, 100000*1);  
    printf("Integration time: %lu\n", get_integration_time(pmd_regs));  
  
    pmd_regs->modulation_code_length = 31;  
    pmd_regs->modulation_code[15] = 0x5d8f;  
    pmd_regs->modulation_code[14] = 0x9a42;
```

In the above part of snippet, place the two codes accordingly and recompile the c code.