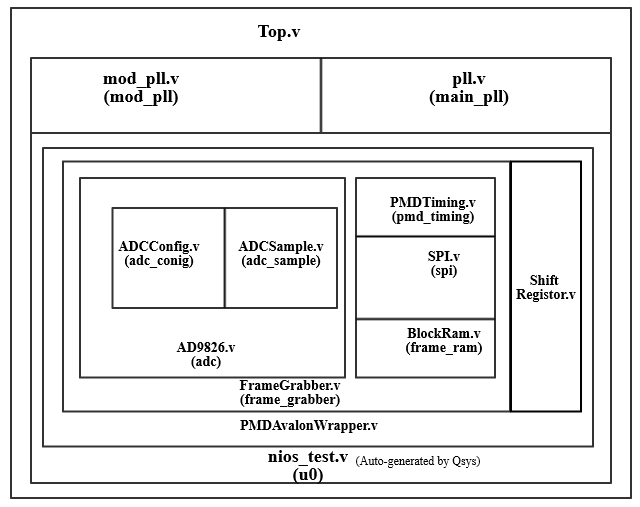
(This part of the document depicts various modules/functions/codes which control the overall data and control flow within the scope of the project.)

**Verilog modules:**

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

**(Fig2: Verilog Module Flow)**

* **ADCConfig.v**

This FSM configure the FPGA and ADC for a bi-directional data flow. It has three states,

1. STATE\_ADDRESS -- Sends the R/W bit, and the three address bits
2. STATE\_DELAY -- Pause the operation for few cycles
3. STATE\_DATA -- Sends out the 9-bit data word

* **ADCSample.v**

This FSM controls the Sampling of ADC and the Readout process. It has three states

1. STATE\_IDLE -- Waits for the sample\_en signal to go high
2. STATE\_SAMPLE -- Sample and Hold
3. STATE\_READOUT -- Pushes the data into the Bus

* **AD9826.v**

Instantiates the ADCConfig.v & ADCSample.v and controls the whole ADC bi-directional flow with FPGA.

* **PMDTiming.v & SPI.v**

These module controls the whole timing operations of the PMD Sensor. They have 16 different states to control the whole flow of IDLE, RESET, INTEGRATION, READOUT, COOLDOWN for PMDTiming, & STATE\_DATA, STATE\_IDLE, STATE\_ADDRESS for SPI . Each of these states meet the timing requirements specified by the PMD Datasheet.

<http://web.media.mit.edu/~achoo/pr/19k3_datasheet.pdf>

* **BlockRam.v**

This module controls the reading and writing of ADC readout in and out of FPGA temporary

memory. The data from ADC is moved into a temporary register and then pushed into the Ethernet for data visualization on PC.

* **FrameGrabber.v**

FrameGrabber instantiates PMD\_timing, SPI, BlockRam & AD9826. This FSM has various states to control,

1. Setting frame size.
2. Grabbing the captured frame from the FPGA temporary memory.

* **ShiftRegister.v**

ShiftRegister module helps in driving the modulating signal to the light source and to the MODSEL terminal of the PMD Sensor.

* **PMDAvalonWrapper.v**

FrameGrabber is instantiated in this module. It interacts with the C-codes via the address bus and helps to set the Integration time & modulation mode. Since the C-code and the PMDAvalonWrapper are in unison through the address bus, the above specified values can also be set during the compilation of C-codes.

All the above specified modules are fed into the Qsys Library, and when the Qsys is generated, it automatically compiles all the Verilog codes and provides a single interfacing code named **nios\_test.v.**

**Qsys can be accessed by going into Tools→ Qsys**

* **PLL.v**

This is a module that is auto generated by using the **Megawizard plug-in manager Wizard**.

This can be accessed by going into Tools → Megawizard plug-in manager. The interactive GUI guides through the process of configuring the PLL for user requirements.

Here we use PLL.v to generate clock signals of different frequencies, but same phase(i.e. 0). These clocks are used to drive various blocks in the FPGA.

* **mod\_pll.v**

This is a module that is auto generated by using the **Megawizard plug-in manager Wizard**. Here we use mod\_pll, to generate two output clocks of same frequency, but one clock signal configured for dynamic phase reconfiguration. The latter one is fed into the light source, and the former into the MODSEL signal of PMD sensor.

* **top.v**

This is highest in the hierarchy, and instantiates three modules, i.e. PLL, mod\_pll & the nios\_test. This acts as a wrapper for the whole Verilog code base.

**MATLAB Code & functions:**

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

* **pmd\_connect.m**

This function establishes a tcp/ip connection with the FPGA at a specified port and ip address.

* **pmd\_read\_image.m**

This function reads the image data via the tcp port, and shapes it to a specified frame rate and stores it in a variable named **rawCorr**.

* **pmd\_phase\_step.m**

This function writes through the ethernet, specifying the FPGA to increase/decrease the phase of the Light Source.

* **pmd\_fpn.m & pmd\_flat.m**

These two functions help to capture a fixed pattern noise & flat frames during the calibration of camera. The number of frames to capture for creating a master fpn/flat frame can be specified in this code.

* **pmd\_600.m**

This matlab code calls the pmd\_read\_image and thereby captures the frames. The number of frames to be captured can be set in this code. After the frame capture is done, code automatically gets takes the FFT of the frames, and stores it in a variable named rawCorrFFT, and then it plots the angle and the amplitude images using the **mview.m** function.

* **mview.m**

This matlab function, inputs the frames captured and provides an interactive GUI to move through all the frames using a mouse scrollbar, and also provides an option to play it at specified frame rates.

**C-codes:**

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.

There are mainly two project database to handle the C codes.

* **pmd\_tcp\_bsp**

This project helps to generate the Board support packages (BSP) & helps to re allocate the memory changes made in the Verilog code. In order to change update the BSP,

Goto Project Explorer → Right click on **pmd\_tcp\_bsp** → Nios 2 → Generate BSP.

* **pmd\_tcp**

This is the C project which controls the whole interface operation. **hello\_ucosii.c** is the main code that can be used to update the integration time. This code, dumps the Memory write and retrieves the code into the soft core processor and establish a bidirectional communication between FPGA and Computer.