Instrument ShAred Artifact for Computing (ISAAC)

ISAAC Newton 2008 EoY Demo Guide

End of Year FPGA, iBoard Demonstration System Guide

Revision: 1.1

Date: **October 29, 2009**

Demo System Authors: Dmitriy Bekker
Kayla Nguyen
Jason Zheng

Document Authors: Kayla Nguyen

Custodian: Kayla Nguyen

Summary: This document is to guide future users of the ISAAC 2008 end of year demonstration system. This guide will include instructions for EDK (how to set up user cores, ethernet, etc.) and MATLAB (GUI, interface with board, ethernet, etc.).

Paper copies of this document may not be current and should not be relied on for official purposes. The current version is in the ISAAC Project Library at https://isaac

.

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109-8099

Contents

List of Figures

Revision History / Change Sheet

Rev.	Description	Author	Data	Approval
1.0	-Initial Draft	Kayla Nguyen	11/3/08	
1.1	-Added Revision History page -Fixed Part 3, file locations -Removed Ethernet MAC Setup chapter -Added Part II, Chapter 3 to include software download instructions -Make note of the cross-over ethernet cable in Section II, Chapter 5	Kayla Nguyen	10/29/09	

Part I

INTRODUCTION

0.1 Purpose

This document serves as the ISAAC Newton 2008 End of Year Demonstration System Guide. This has sufficient details for future use of the demo system.

ISAAC is a 3-year R&TD task, focusing on reusable modules used by different instruments. Currently there are four subtasks within the ISAAC framework: Asimov, Stern, Singer, and Newton. ISAAC Newton focuses on the radar application, specifically, this year, the on-board processing power which will eventually support the SMAP (Soil Moister Active-Passive) project. The demonstration system is geared toward demonstration the capabilities of the Newton digital filter core (QMFIR).

Chapter 1

Demo System Overall Architecture

1.1 Digital Filter Core (QMFIR)

The main focus of the 2008 End of Year Demo is to showcase the Quadrature Modulation and Polyphase Decimation Filter. The details of this core is given in another document, but the overall architecture of the core will be discussed here. The overall architecture of the QMFIR core is given in figure (1.1).

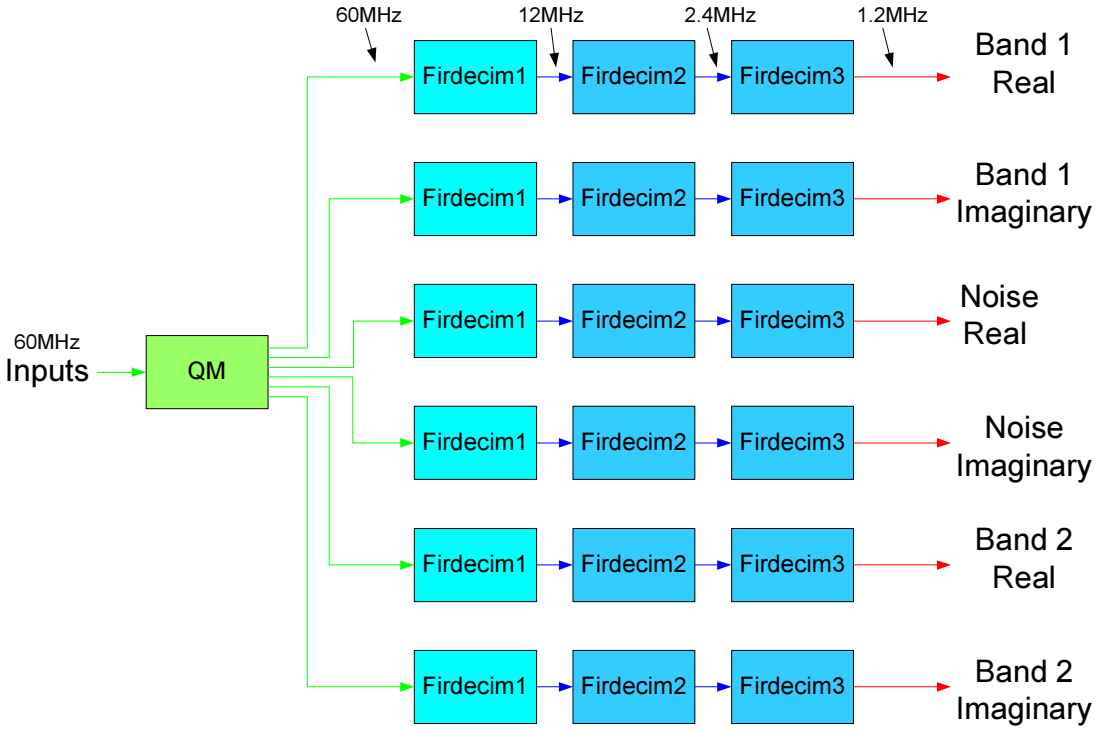

Figure 1.1: QMFIR Overall Architecture

There is one module called QM which will take the incoming signal and demodulates it down to baseband. In the process, it will create 6 streams of data at its output. The data are the real and imaginary part of what we call Band 1, Band 2, and noise band. The 6 streams are then passed through a series of polyphase decimation filters called FIRDecim. The incoming data rate is 60MHz, the output

data rate is 1.2MHz. Through the series of FIRDecim, the data rate has been decimated 50 times.

1.2 Demo System Flow

1.2.1 FPGA Data Flow

The demo system FPGA data flow is given in figure (1.2). Data is first transferred from the host machine through ethernet to the board. The data gets temporarily stored in the DDR memory. When all the data has been stored, the PPC will transfer the data to the FIFO into the QMFIR DSP core. The data gets processed and the result will be written into 3 local Block RAM inside the DSP core. When all the data has been written, there will be an interrupt to let the PPC know that it can then grab the data from the BRAM and transfer it back through ethernet to the host machine.

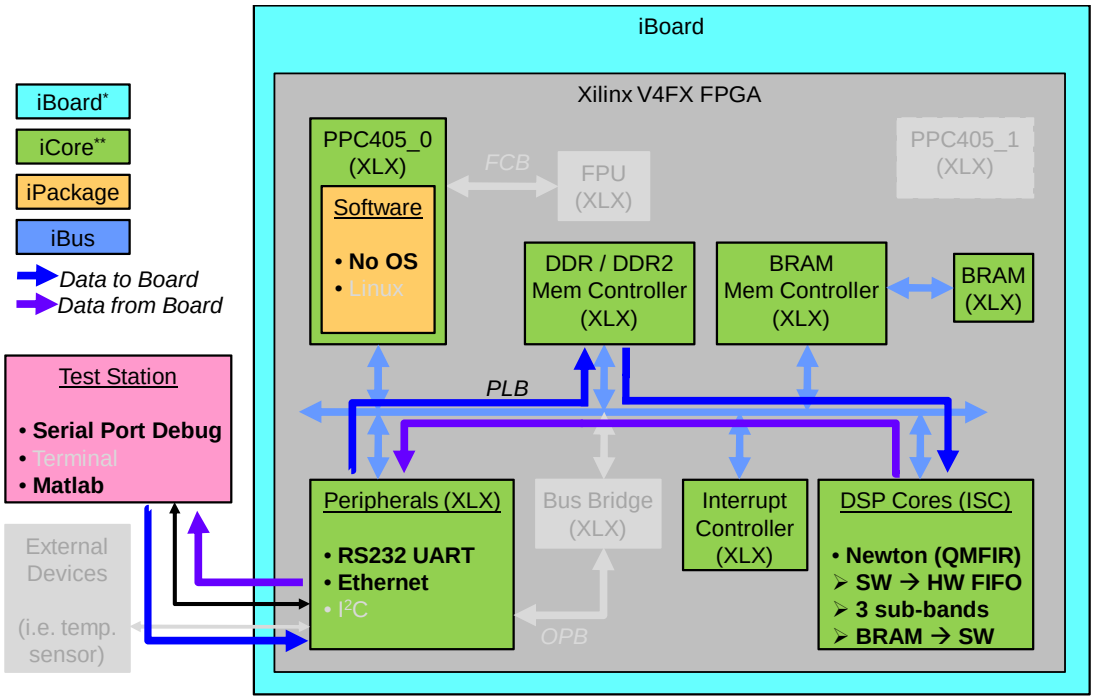

Figure 1.2: FPGA Demo Data Flow

1.2.2 Overall Demo System Data Flow

The overall data flow for the demo system is shown in figure (1.3). This figure shows quite a few things they will be broken down in order to familiarize the user with all the information that will be given in the demo.

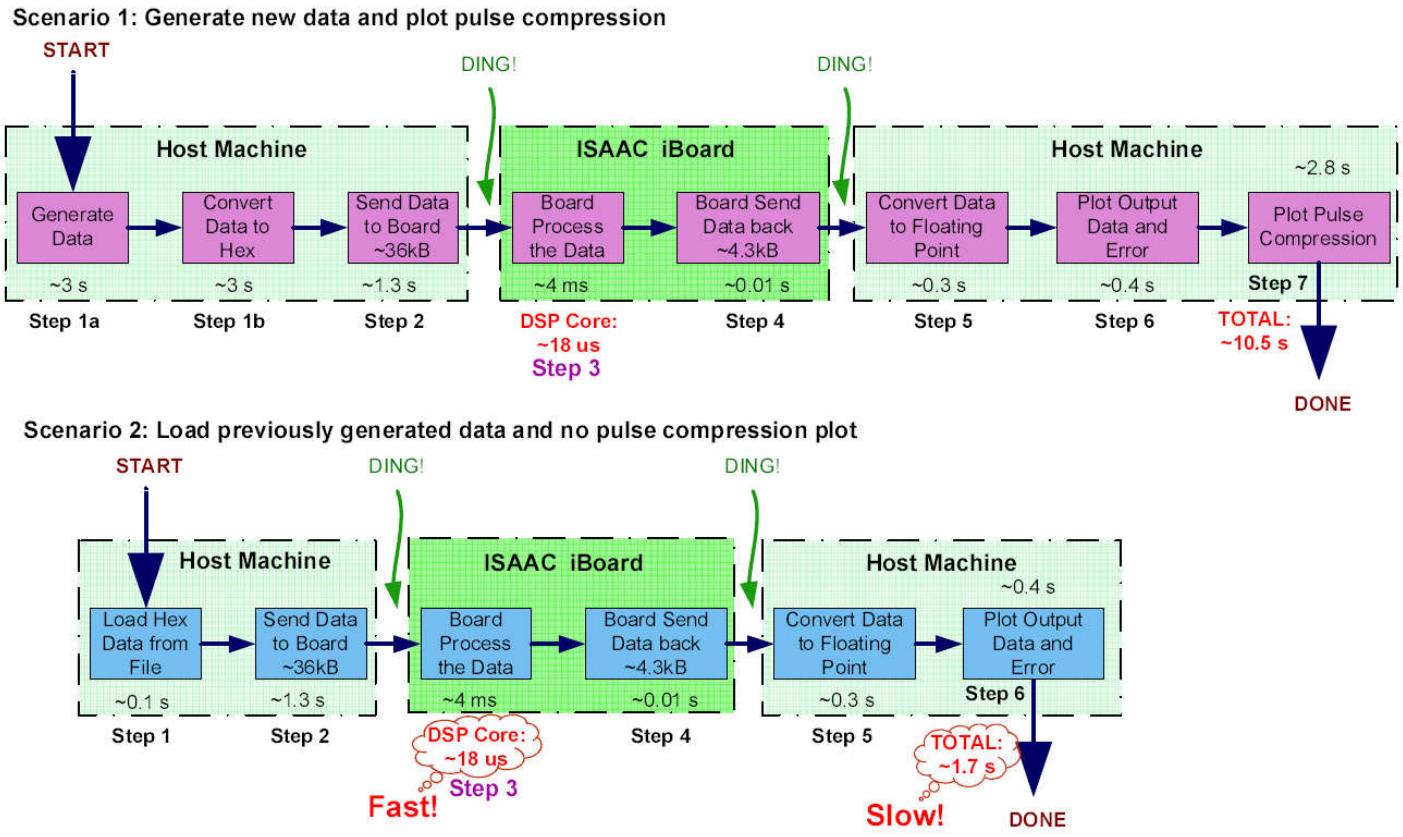

Figure 1.3: Overall Demo System Data Flow

1. The figure shows 2 scenarios in which the demo system can run. These are not the only 2 scenarios but are chosen for their contrast. Scenario 1 demonstrates that the demo system is able to process data on the fly with unknown, new data generated in real-time by MATLAB. This is the most accurate scenario because on a flight mission the data will be unknown and will be processed in real-time. This scenario will also plots the pulse compression from the data that has been processed by the FPGA. Scenario 2 will attempt to depict the speed of the DSP core. This scenario will use previously generated data and will depict a moving target. This scenario will not plot the pulse compression.

2. The figure breaks down which steps are computed by the host machine (MATLAB, data movement to the board, etc.), and which steps are computed by the iBoard (DSP core, data movement on the board, etc.).

3. The figure links what each step is doing to a step number, which will then be depicted on the MATLAB GUI.

4. The figure gives an approximate time each step takes to execute, either in MATLAB or on the iBoard.
5. The figure gives a total amount of time that can be expected for each scenario to play out.

Part II

DEMO SYSTEM GUIDE

Chapter 2

EDK

2.1 Integrating CoreGen IP into EDK

2.1.1 Introduction

This section serves to guide users with the appropriate steps to integrate a Xilinx CoreGen IP core into EDK. CoreGen is a GUI tool provided by Xilinx to help FPGA designers generate commonly used cores without writing a Verilog/VHDL code from scratch. The tool provides the user with a blackbox design of the core through the following files:

1. -.asy Graphical symbol information file.
2. .ngc *Binary Xilinx implementation netlist file.
3. -.sym Schematic symbol file used to instantiate a generated core into the ISE schematic editor
4. .v *Verilog wrapper file provided to support functional simulation.
5. -.veo VEO template file containing the code that can be used as a model for instantiating a CORE Generator module in a Verilog design.
6. -.xco CORE Generator input file containing the parameters used to regenerate a core.
7. -.edn EDIF wrapper file generated for a core when the Generate netlist wrapper with IO pads project option is enabled.
8. -.tcl ISE Project Navigator interface file.

The two files denoted by the star symbol (*) needs to be transferred from the CoreGen project into the EDK project. This document will show the appropriate steps that needs to be taken.

Similarly, EDK is a tool provided by Xilinx to help in the integration of the CoreConnect system on chip bus, the embedded processor, memories, CoreGen IP Cores, users' own cores, etc. This tool has a GUI interface which can be used to create bus connections, address assignments, and more. One disadvantage with the EDK tool is in the integration of CoreGen IP Cores with the rest of the system. Since CoreGen only provides a blackbox design with a netlist given, there are several changes that must be made to EDK before it will recognize the core to synthesize it.

2.1.2 Changes in EDK

This section will show the steps to be taken after the EDK project has been started, and after the Peripheral has been created for the CoreGen core. The guide assumes that the user has read through Xilinx documents instructing on the steps to creating an EDK project and creating a Peripheral for the core.

1.) In the EDK project directory, there will be a directory called /pcores/qmfir_plb_v1_00_a/ after the Peripheral for the core has been created.

   a.) Create a /netlist/ folder in the /pcores/qmfir_plb_v1_00_a/ directory.

   b.) Copy the BRAM.ngc file from CoreGen into the /netlist/ directory that was just created.

2.) Go to /pcores/qmfir_plb_v1_00_a/hdl/verilog/ folder.

   a.) Add the BRAM.v file from CoreGen into the /verilog/ directory.

3.) Open file /pcores/qmfir_plb_v1_00_a/data/qmfir_plb_v1_0_0.mpd (This file name may be different than the Peripheral folder name in the /pcores/ directory).

   a.) Add the following lines:
   OPTION STYLE = MIX

   b.) Make sure that the **OPTION HDL = MIXED** if the core is written in Verilog. Otherwise **OPTION HDL = VHDL** is ok.

4.) Open file /pcores/qmfir_plb_v1_00_a/data/qmfir_plb_v1_1_0.pao.

   a.) Add the following line if the core is in Verilog:
   qmfir_plb_v1_00_a BRAM verilog

   b.) Add the following line if the core is in VHDL:
   lib qmfir_plb_v1_00_a BRAM vhdl

5.) Go to /pcores/qmfir_plb_v1_00_a/data/ folder.

   a.) Create a file called qmfir_plb_v1_1_0.bbd. Note that this file name should be the same as the -.pao and -.mpd files in that folder.

   b.) Edit this file to include the following:
   FILES
   BRAM.ngc, blackbox1.ngc, blackbox2.ngc
   **(Note that blackbox1.ngc, blackbox2.ngc are additional netlists if there are more than one core generated from CoreGen. These are not required if only one core generated from CoreGen.)

6.) Open file system.mhs in the main EDK directory.

   a.) Look for the start of the instantiation of your core. In this example, the instantiation is as follows:
   BEGIN qmfir_plb.

   b.) Add the following line within that block:
   PARAMETER c_family = virtex4
   **(This depends on the FPGA family of device you are targetting: virtex2, virtex4, virtex5, etc..)

Chapter 3

PPC Software

3.1 Intro

The PowerPC embedded processor inside the Virtex-4 FPGA is the PPC440. This processor is used by the demo to control the ethernet data transfer, the commands from UART, and can be used to debug the Filter.

3.2 Load SW into PPC

In order to load the software into the PPC from the host machine, start with a command prompt. Locate the folder which holds the software elf file (qmfir_udp_demo.elf in this case). Type in the prompt:

$ xmd

This opens the Xilinx XMD program. After the program opens, type in the prompt:

$ connect ppc hw
$ dow qmfir_udp_demo.elf
$ start

In order to stop the processor, just type:

$ stop

To get into the XMD help menu, type:

$ help

from the prompt.

Chapter 4

MATLAB

The purpose of the GUI for the 2008 EoY Newton demo was to create an environment that was both stimulating and informative to the audience through use of graphs, pictures, sounds, etc. Note that the current files in the repository for the MATLAB GUI Demo system is written for a host machine running Ubuntu operation system. The transfer protocol can be used for a Windows machine, but is extremely unreliable and should not be used for official purposes. This also has not been tested on any other Linux operation systems.

4.1 Demo GUI Instructions

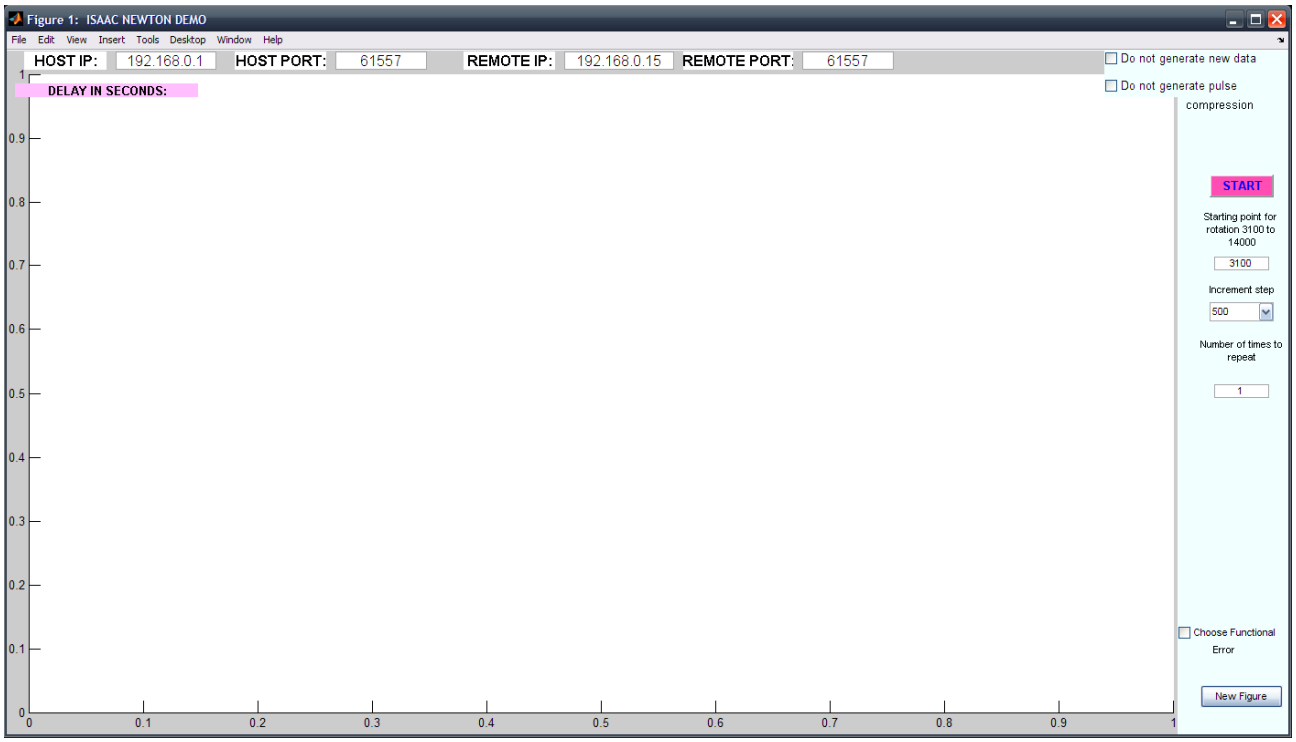

Figure 4.1: Newton Demo GUI

The GUI for the Newton demo is shown in figure (4.1). The GUI was created using the following MATLAB's functions:

figure	Used to create an initial figure for the GUI
uicontrol	Used to create buttons, pull-down menus, edit boxes, etc. for the GUI
set	Used to set certain parameters for the GUI figure and the buttons, edit boxes, etc.
movegui	Used to center the GUI on the display screen
function	Used to write callback functions for the GUI buttons, edit boxes, pull-down menus, etc.
get	Used to retrieve user information from the GUI buttons, edit boxes, pull-down menus, etc.

4.1.1 Usage Instructions

4.1.1.1 GUI Figure

To get the GUI running, open MATLAB and make sure that the current directory is the one with the file 'isaacdemogui.m'. In the command prompt, type:

> isaacdemogui

then press Enter. This should automatically pop up the GUI as shown in figure (4.1).

4.1.1.2 GUI Interface Commands

There are a variety of options to choose from in the GUI interface. Figure (4.2) shows the options that are available on the interface.

1. On the top on the GUI is the ethernet transfer information. Users can choose the Host IP, the Host Port number, the Remote IP, and the Remote Port number. The Host information is the information of the computer which is running the MATLAB GUI. The Remote information is the information of the board that is running the FPGA. To change the information of the Host and Remote, just simply click on white box and type in a new IP or port number. Make sure that this matches with the C script that is programmed into the Power PC running on the FPGA. Default values: Host IP is 192.168.0.1, Host Port is 61557, Remote IP is 192.168.0.15, Remote Port is 61557.

2. This is the 'Do not generate new data' option. If this option is clicked, input data to the FPGA will be loaded from a series of previously generated data. This option will provide a quicker demo time because loading data from a file is much faster than generate the chirp data from a MATLAB algorithm. Default value is UNselected.

3. This is the 'Do not generate pulse compression' option. If this option is clicked, the data that is transferred from the board will not be generated into pulse compression. This option will provide a quicker demo time because generating the pulse compression demands running the long MATLAB program. Default value is UNselected.

4. This is the button to start the demo run. This will first either generated data or load data from file, transfer to board, wait and receive the processed data from board, and plot the data and error, and could plot the pulse compression. This button should be the last to press after all of the options have been looked at and chosen.

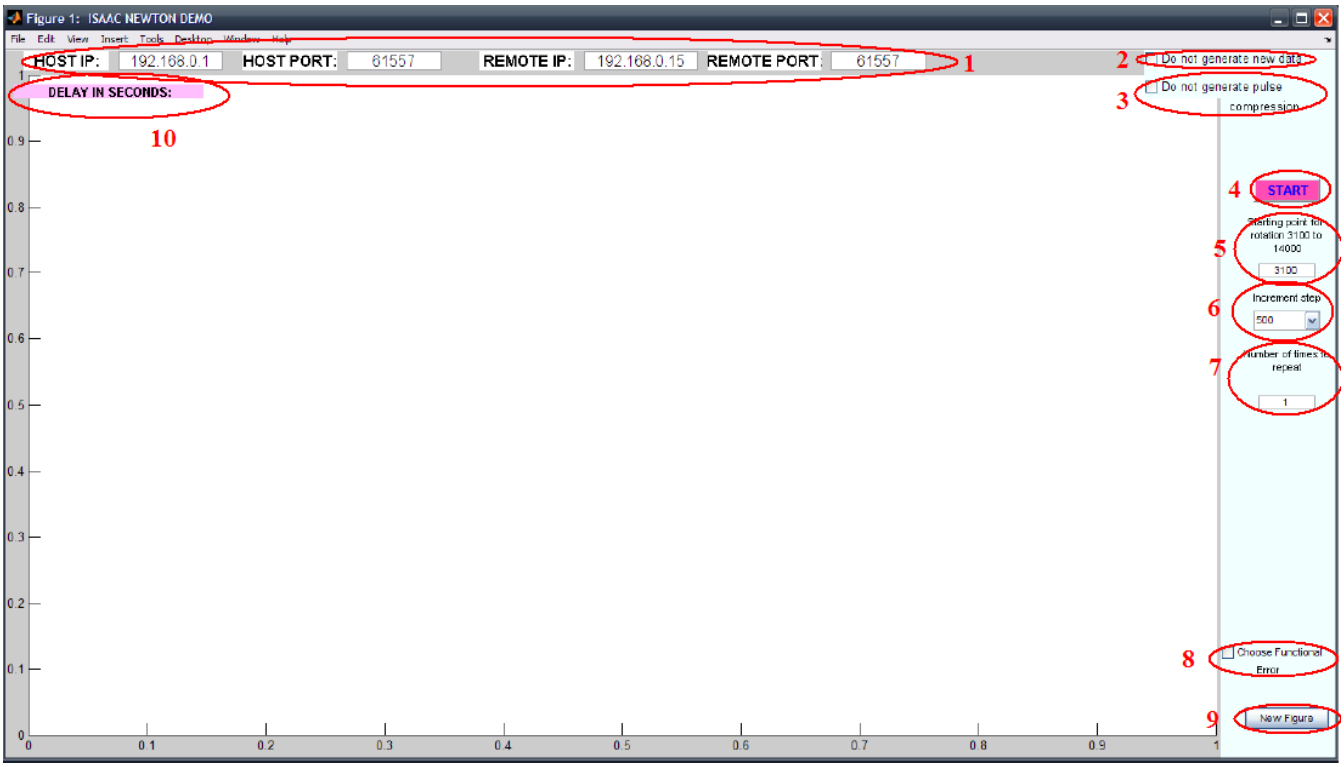

Figure 4.2: Newton Demo GUI Commands

5. This is the 'Starting point for rotation 3100 to 14000' option. This allows the user to choose the starting position of the rotation of the input data. This will dictate the first the first rotation point for the input data. From algorithm design, this number should be between 3100 and 14000. This is only valid if option 2 is UNclicked (meaning that new data is generated). Default value is 3100.

6. This option allows user to choose the increment step to be taken between rotation. Starting from the value in option 5, subsequent rotation will be incremented by the number chosen in this step. Default value for this is 500 points.

7. This option allows user to repeat the demo run (sending data and receiving data) more than one time. User than input the number of times to send data to be processed and receive the data to plot. There are currently 23 data points on file if option 2 is clicked. And the maximum number for this step if option 2 is UNclicked depends on option 5 and 6. If option 5 and 6 combined is larger than 14000, the demo system will stop. The default value is 1. The default value is UNselected.

8. This option allows user to see the function error between the FPGA implementation and MATLAB algorithm simulation. The default is UNclicked, which means that the error that is plotted is the quantization error from the MATLAB representation of the RTL code.

9. This option allows user to generate a new GUI figure to clean up current figure.

10. This is the real-time "how long does each step take" portion of the demo. This demonstration system is able to display in real-time the amount of time each step in the demonstration system takes.

4.1.1.3 Change in File Location

In the file qmfir_udp_demo.m, the parameters 'fileextension1' (Line 32) and 'fileextension2' (Line 33) allows user to change the file location of the folders InputFiles and InputFiles_C, respectively. This is where all of the pre-generated data are stored. InputFiles stores the data for the quantization error files, the InputFiles_C stores the data for the function error files. In order to load correctly from different computers, this parameter must be changed according to each user. For example, for windows system, the parameter 'fileextension' would be something like:

fileextension1 = 'C:\Documents and Settings\kaylangu\Desktop\matlab_gui_ubuntu\InputFiles\';
fileextension2 = 'C:\Documents and Settings\kaylangu\Desktop\matlab_gui_ubuntu\InputFiles_C\';

For a Linux system, it could be something like:

fileextension1 = '/home/kaylangu/Desktop/matlab_gui_ubuntu/InputFiles/';
fileextension2 = '/home/kaylangu/Desktop/matlab_gui_ubuntu/InputFiles_C/';

4.2 UDP Transfer Protocol using pnet

The UDP transfer protocol through ethernet that has been chosen for this demo system is through pnet. This protocol was taken from TCP/UDP/IP Toolbox 2.0.6 by Peter RydesAxter. It is available through:

http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=345&objectType=file

This is a C program that needs to be compiled inside of MATLAB using mex compile. First, this toolbox needs to be downloaded from the website given above. Make sure that the files are in the same directory as the demo system files.

For Linux system, to compile the file, type in the MATLAB command window:

> mex -O pnet.c

For windows, type in MATLAB command window:

> mex -O pnet.c {MATLAB_INSTALL_DIR}\sys\lcc\lib\wsock32.lib -DWIN32

This will create a .dll file which will be used by MATLAB to establish a UDP connection from MATLAB to the FPGA board.

4.3 GUI After Running Demo System

Figure (4.3) shows what the GUI is expected to look like after running the demo system. On the left side of the GUI, there is a list of Steps and time corresponding to each step. The steps are explained in figure (1.3). The corresponding time is the amount of time it takes for each steps to be processed. The top

figure shows the data that is sent to the board to be processed. This data is sampled at 60MHz. The 2nd line shows the real data from Band 1, Band 2, and noise band from left to right. The 3rd line shows the imaginary data from Band 1, Band 2, and noise band from left to right. Line 4 shows the error between the real data of the FPGA processed data versus the data from simulation (Band 1 and Band 2 from left to right), and line 5 shows the error between the imaginary data. The last plot of lines 4 and 5 shows the pulse compression plots of the FPGA processed data.

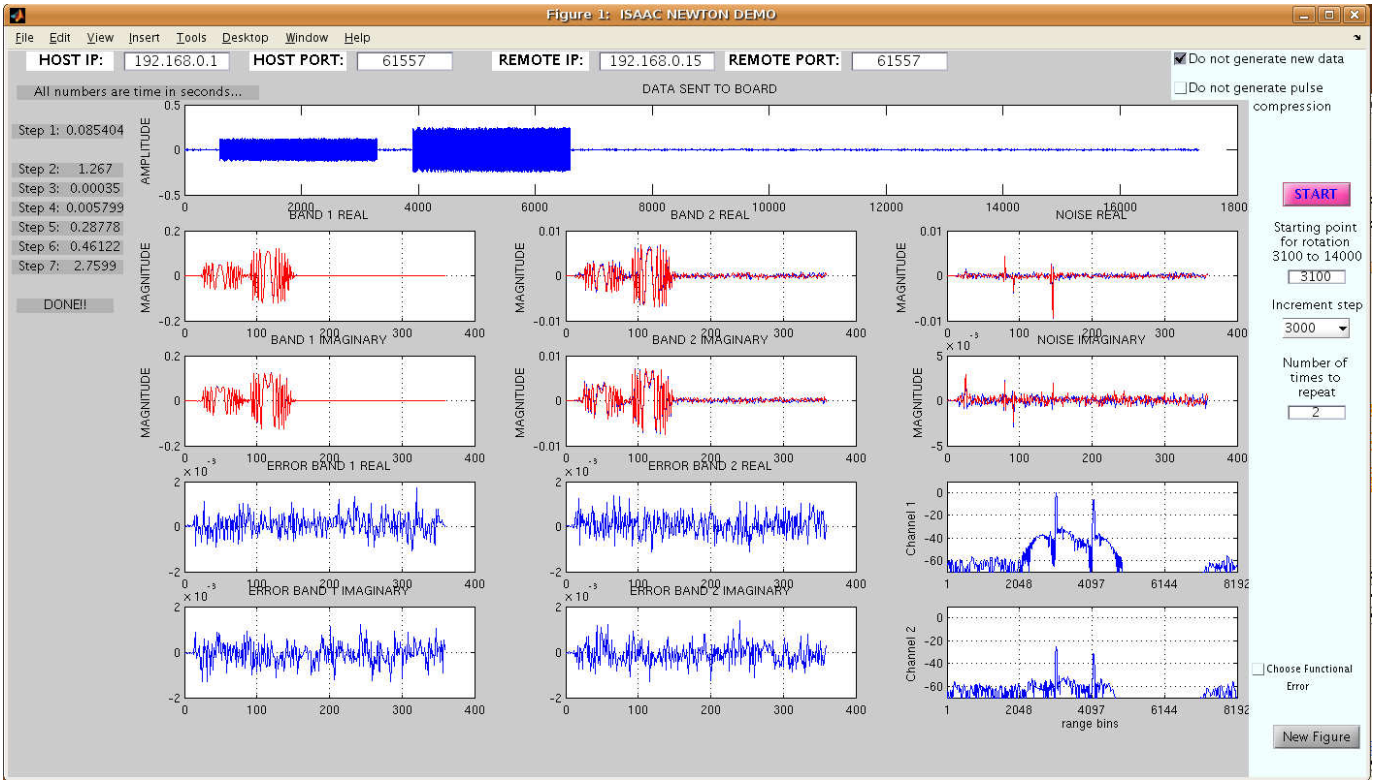

Figure 4.3: Newton Demo GUI After Demo

Chapter 5

Host Machine

5.1 Ethernet Connection

The demo system was demonstrated using a direct ethernet connection from the board to the host computer. It was found that through the router in the ISAAC lab (198-135B), the ethernet connection cannot be made. Therefore, the host computer must be directly connected to the board through ethernet and able to reliably connect to JPL wireless internet in order to run MATLAB wirelessly. Another option is to connect to the board using the ethernet to USB converter. This will allow the host machine to connect to JPL internet through the ethernet port of the computer.

Also note that a cross-over ethernet cable may be required for the ethernet connection to establish a 100Mb connection.

5.1.1 Linux System

5.1.1.1 Host IP

The host IP of the host computer must match the IP that is given in the MATLAB GUI interface. In order to do this, open a command prompt and connect the ethernet cable from the board to the computer (either straight to the ethernet port or through the ethernet to USB converter). If it is connected straight to the ethernet port, the ethernet name is 'eth0'. If it is connected through the ethernet to USB converter, the ethernet name may be either 'eth1' or eth2'. To check this, type in the prompt:

$ ifconfig

Scroll up to then see the name of the new ethernet connection. Usually, Avahi runs on ethernet and might conflict with the IP address change that is necessary for the demo. Therefore, Avahi must be stopped by typing in the prompt:

$ sudo etc/init.d/avahi-daemon stop

To change the IP address of this connection, type in the prompt:

$ sudo ifconfig eth1 192.168.0.1/24

The name eth1 will depend on the connection name that was found previously through ifconfig.

5.1.1.2 Connection Speed

Currently on the ML410 evaluation board and through EDK ethernet configurations, the ethernet can run only at 100Mb/s using full duplex. The default speed for an ethernet connection is 1Gb/s and must be changed to 100Mb/s. Note that if an ethernet to USB converter is used, this will automatically configure the ethernet connection to 100Mb/s. If the ethernet is connected directly through the ethernet port, the speed must be changed by hand. To check what the configuration for the ethernet connect is, type:

$ ethtool eth1

If the Speed is set to 1000Mb/s, it needs to be changed to 100Mb/s. To do this, in the prompt, type:

$ sudo ethtool -s speed 100 eth1

The name eth1 will depend on the connection name that was found previously through ifconfig. Check again using ethtool to make sure that the speed has been changed to 100Mb/s.

5.1.2 Windows System

As mentioned earlier, the demo system was originally written to be run in conjunction with a Linux host machine. The steps for the Windows operation system will be given here for completeness purposes since the system may work with Windows in the future.

5.1.2.1 Host IP

The host IP of the host computer must match the IP that is given in the MATLAB GUI interface. In order to change the IP address of the ethernet connection:

1. Navigate to Control Panel > Network Connections.
2. In Network Connections, find the connection that corresponds to the ethernet port which is used to connect the host machine to the FPGA board. Usually this is a Local Area Connection.
3. Right click on Local Area Connection and go to Properties.
4. Under 'This connection uses the following items:' unclick all except for 'Internet Protocol (TCP/IP)'.
5. Select 'Internet Protocol (TCP/IP)' and click on Properties.
6. Change the IP address to 192.168.0.1, Subnet mask to 255.255.255.0, Default gateway to 192.168.0.1 (This should match the IP given in the MATLAB GUI interface). Clear all the numbers under 'Preferred DNS server' and 'Alternate DNS server'. Click OK.
7. Click OK in the Local Area Connection Properties to accept the IP changes.

5.1.2.2 Connection Speed

In order to change the connection speed that is default to 1Gb/s to 100Mb/s, do the following:

1. Navigate to Control Panel > Network Connections.
2. In Network Connections, find the connection that corresponds to the ethernet port which is used to connect the host machine to the FPGA board. Usually this is a Local Area Connection.
3. Right click on Local Area Connection and go to Properties.
4. Click on Configure.
5. Go to the Advanced tab > Speed & Duplex.
6. Under 'Value:' change the value to 100 Mb full.
7. Click OK out to the Network Connections windows.

5.2 Serial Port Connection

The serial port is used as a debugging connection from the host computer to the board. This is where commands can be made to (depending on the PPC program) to reset the FPGA, to write a certain value in the internal register, to write input data into the QMFIR core, etc.

5.2.1 Linux System

In the Linux system, one program that can be used to establish a connection through serial port is kermit. In order for kermit to recognize the serial connection, certain parameters must first be initialized. Create a new text file, call it something like 'serialprompt.k'. Edit this text file to look like this:

```
set line /dev/ttyS0
set speed 9600
set parity none
set flow-control none
set carrier-watch off
connect
```

Save this file and exit. To find out what the device name for the serial connection is, search in the /dev/ folder. If the serial port is connect through USB, the dev name will start will ttyUSBn (n is a number given by the machine). If the serial connection is connected straight into the serial port, the name will start with ttySn (n is a number given by the machine).

To start kermit, on a command prompt, type:

```
$ kermit serialprompt.k
```

This should start a kermit session and is waiting for data to be transferred through serial port.

5.2.2 Windows System

In the Windows system, the program that can be used to view serial port data is HyperTerminal. This program can be found under Accessories in the Start menu.

1. Make a new connect and name it something like 'test'.
2. Under 'Connect using:' choose a COM connection (usually it's COM1). In order to check this, you must check which COM number the host machine is under. Press OK.
3. Change 'Bits per second:' to 9600. Change 'Flow control:' to None. Click OK

Now the HyperTerminal prompt is waiting to receive data from the serial port.

Part III

FILE LOCATION

Chapter 6

Repository

The ISAAC team is currently using Subversion as the repository for the soft copies of all documents and files created. To get instructions on how to use subversion, go to http://subversion.tigris.org/ for more information.

6.1 Link to Files

EDK Demo Project and Matlab GUI Files:

svn co svn+ssh://isaacdev/proj/isaac/repo/isaac_newton/trunk

The EDK project files are under /xilinx_10.1/. The software for the data transfer is under /sw/. The MATLAB GUI software is under /matlab_gui/.

pcores Directory for EDK Project:

svn co svn+ssh://isaacdev/proj/isaac/repo/edk_repository/trunk

The EDK project refers to a repository of pcores from an external source (not within the EDK project). This directory is where the EDK project looks for the user core. The file note.txt has instructions on how to link the EDK project to this library of pcores.

This location also includes the Verilog code for the core under user_logic.v.