4-Antenna Packetized IA TESTING PROCEDURE

Version 1 , Released Date : 11/01/2014.
Written By : Pranjali Chumbhale/Shreya Shetti/Swapnil/
Kaushal D. Buch/I. M. Halagali.

[CTRL_PC] : 192.168.4.71
[USER_CTRL_PC] : root
[CTRL_PC_PASS] : gmrttifr

[SOFT_DIR] : */home/gmrt/Packetized_BF*
[ACQ_DIR] : /home/gmrt/*Packetized_BF/Pranjali/Depack_prog/gulp/*
[DATA_DIR] : /home/gmrt/Packetized_BF/IA/

[CONFIG_FILE] : corr_init.py config_4ant
[INIT_FILE] : init_8new.py

I SETUP :

1. HARDWARE SETUP

- a. DESIGN SPECIFICATIONS
- b. ROACH BOARDS USED
- c. CONNECTIONS TO THE F-ENGINE
- d. 10 GbE PORT CONNECTIONS OF X-ENGINE
- e. CONNECTIONS BETWEEN F-ENGINE AND X-ENGINE
- f. CONNECTIONS FROM X-ENGINE TO CONTROL PC

A. for NOISE SOURCE TEST
B. for ANTENNA TEST

I SETUP :

1. HARDWARE SETUP

In this setup, four ROACH boards each with one iADC card is used. Each iADC card has a clock, sync and two inputs for signal . While the clock & Sync input is a must for all iADC cards. Each GPU PC (a total of 4 GPU PCs) is equipped with two 10 GbE cards apart from the GPU card. One 10 GbE card is directly connected to one of the two ROACH boards while the other 10GbE card is connected to neighbouring GPU PCs via a 10 GBE switch.

a. DESIGN SPECIFICATIONS :

Number of F-engines : 4

Number of X-engines : 8

b. ROACH BOARDS USED :

ROACH boards used as F-engine :

ROACH040241, ROACH040242, ROACH040237, ROACH040246.

ROACH boards used as X-engine :

ROACH030167, ROACH030116, ROACH030174, ROACH040235.

There are two X-engines per ROACH board. The X engines and the correspondin g ROACH boards are as mentioned in the following table:

ROACH BOARD NUMBER	CORRESPONDING X -ENGINES
030167	X1 and X5
030116	X2 and X6
030174	X3 and X7
040235	X2 and X8

Table (Y): ROACH board corresponding to the 8 X-engines.

(Note: Any other ROACH board can be used by providing the name of the desired ROACH board in the config_4ant script. Also make changes in the server_f and server_x accordingly.)

c. CONNECTIONS TO THE F-ENGINE :

F-engine is given four inputs:

1) Sync

2) I (Polarisation 0 input)

3) Clock

4) Q (Polarisation 1 input)

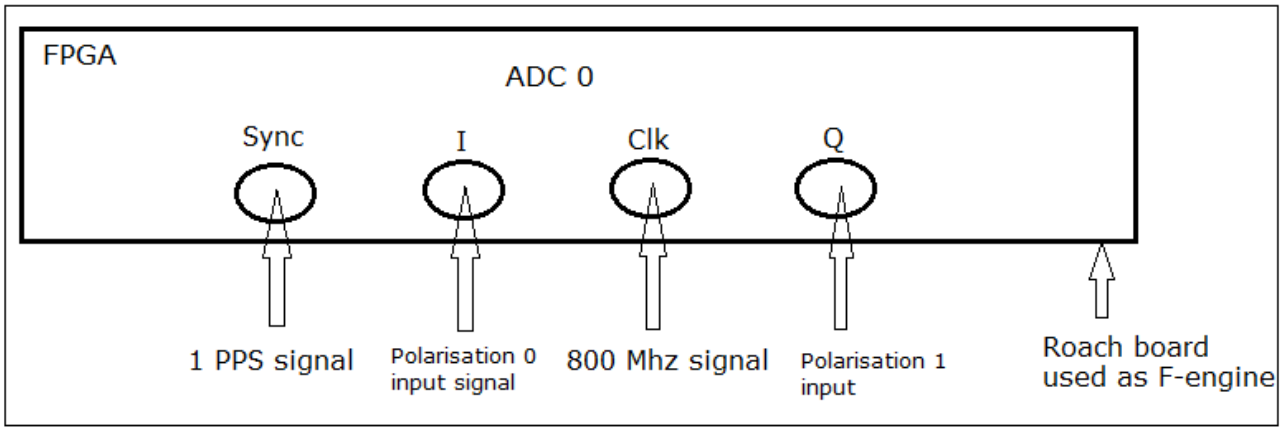


Figure 1 : Connections to F-engine.

d. 10 GbE PORT CONNECTIONS OF X-ENGINE :

Every ROACH board has 4 10 GbE ports. The connections to them are as shown in the figure :

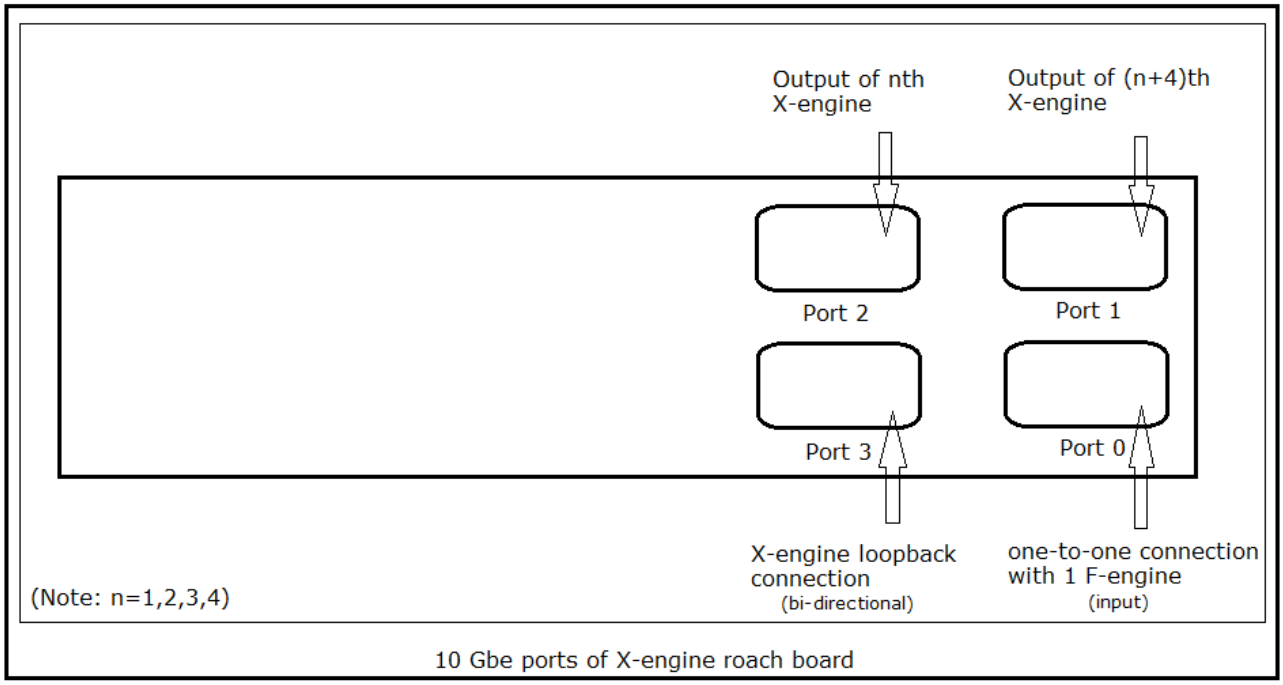


Figure 2 : 10GbE port connections of X-engine.

e. CONNECTIONS BETWEEN F-ENGINE AND X-ENGINE :

The connections between F-engine and X-engine are as shown in the following diagram:

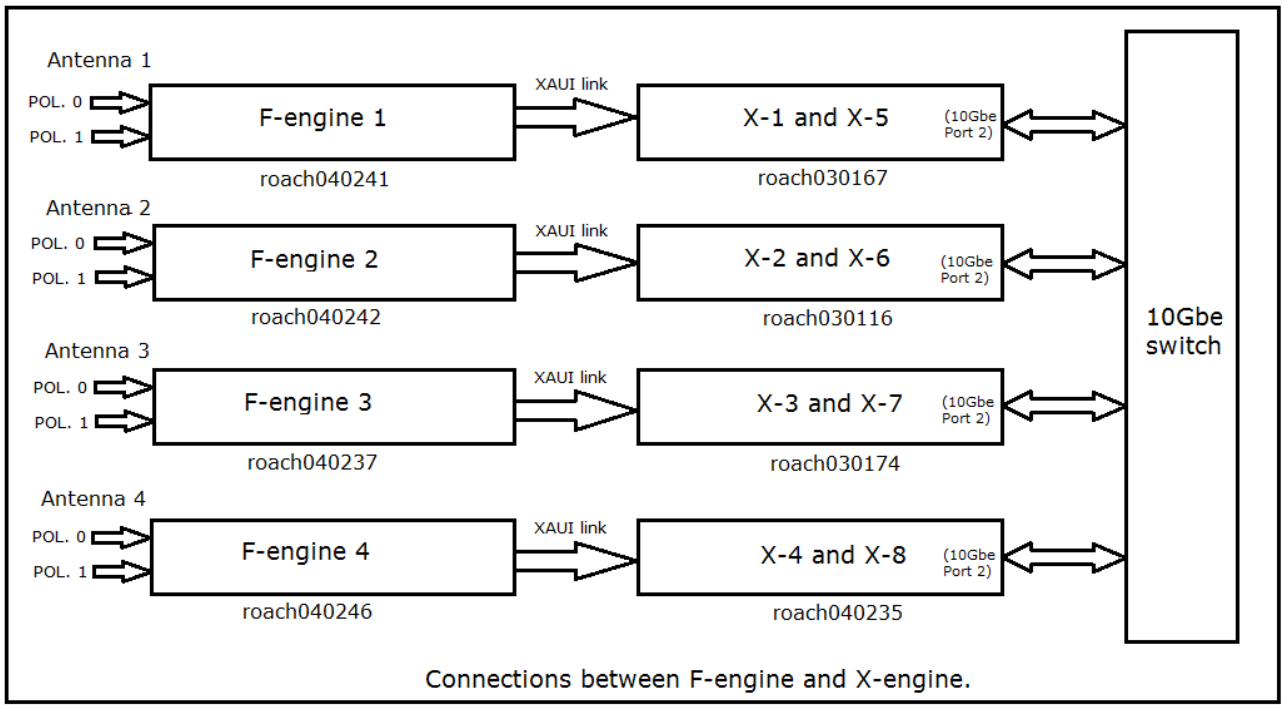


Figure 3 : Connections between F-engine and X-engine

f. CONNECTIONS FROM X-ENGINE TO CONTROL PC :

The connection from the X-engines to control PC is made via the 10GbE switch. The following diagram shows these connections.

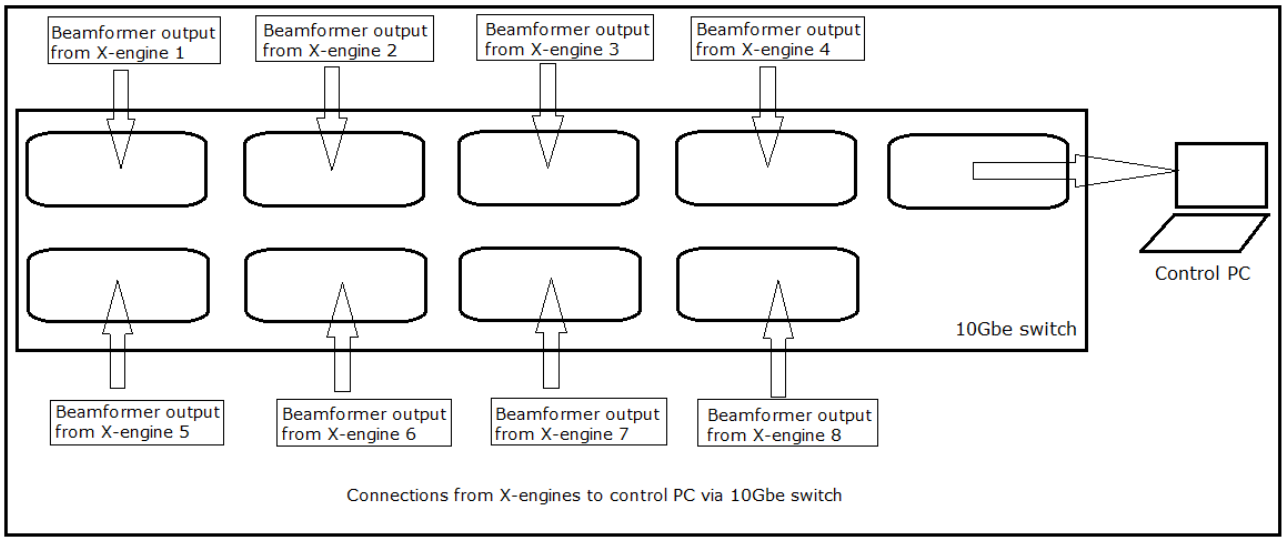


Figure 4 Connections from X-engines to control PC via 10 GbE switch.

A. NOISE SOURCE TEST

The following connections needs to be done for the noise source test.

1. Connect the ROACH boards to the controlling PC using 100MBPS ethernet.

2. Feed the 400MHz (8-bit/4-bit) or 800MHz (4-bit), 0dbm clock signal to the clk_i inputs of the iADC cards from the signal generator. Also feed the PPS signal to the Sync input of the iADC 0 cards of both ROACH boards.

3. Connect the input signals to I+ and Q+ of the iADCs from the noise generator source. The input signal should be of -12 to -17 dbm@200MHz/400MHz bandwidth at the iADC card input.

4. Connect the ROACH boards to the GPU PCs using 10gbE cables. There are four 10 GbE connectors on each ROACH board. The design has been made to allot one 10 GbE connector per iADC board, which means only two 10 GbE connectors are used. Also, each 10 GbE connector sends out data at the rate of 800 MiB/s(Million Bytes per second).

The mapping of 10 GbE connections is as follows:

10 GbE port on ROACH		GPU PC
ROACH 0 PORT 0	-------	Node 0
ROACH 0 PORT 1	-------	Node 1
ROACH 1 PORT 0	-------	Node 2
ROACH 1 PORT 1	-------	Node 3

5. Turn ON the ROACH boards.

B. ANTENNA TEST

The following connections needs to be done for antenna test.

1. Connect the ROACH boards to the controlling PC using 100MBPS ethernet.

2. Feed 400MHz, 0dbm clock signal to the clk_i inputs of the iADC cards from the signal generator. Also feed the PPS signal to the Sync input of the iADC 0 cards of both ROACH boards.

3. Connect the antenna signals to the iADCs and make note of the sequence as it is needed for “fstc” and “fringe” corrections. The sequence is as follows:

ROACH 0 iADC 0 (I+ & Q+)	--------	Antennas 0 and 1
ROACH 0 iADC 1 (I+ & Q+)	--------	Antennas 2 and 3
ROACH 1 iADC 0 (I+ & Q+)	--------	Antennas 4 a nd 5
ROACH 0 iADC 1 (I+ & Q+)	--------	Antennas 6 and 7

The integer delay correction is different from fringe and fstc corrections. For fstc and fringe corrections, every GPU node corrects corresponding values of all antennas for different time slices. While for integer delay corrections, every node corrects corresponding values of only two antennas. The mapping for integer delay correction is as follows.

Node 0 ---------- Antennas 0 and 1

Node 1 ---------- Antennas 2 and 3

Node 2 ---------- Antennas 4 and 5

Node 3 ---------- Antennas 6 and 7

4. Connect the ROACH boards to the GPU PCs using 10gbE cables. There are four 10 GbE connectors on each ROACH board. The design has been made to allot one 10 GbE connector per iADC board, which means only two 10 GbE connectors are used. Also, each 10 GbE connector sends out data at the rate of 800 MiB/s.

The mapping of 10 GbE connections is as follows:

10 GbE port on ROACH		GPU PC
ROACH 0 PORT 0	----------	Node 0
ROACH 0 PORT 1	----------	Node 1
ROACH 1 PORT 0	----------	Node 2
ROACH 1 PORT 1	----------	Node 3

5. Turn ON the ROACH boards.

2. SOFTWARE SETUP

1. open a terminal and log into $ ssh -X [USER_CTRL_PC]@[CTRL_PC]
2. Move into the DELAY_DIR. $ cd [SOFT_DIR]
3. Edit the file for bof file(if required) information
    a. $ vim config_4ant

In this check the bitstream_x=des2x_30oct_2013_Oct_30_1655.bof . This is the bof file to be burnt on the X-engines

    b. $ vim init_8new.pyt

In this check my_corr.write_int("no_cycle",1) from line 43 to 46 this means base integration

II CONFIGURING THE ROACH :

In this step, we will program the ROACH with the [BOF_FILE] and configure parameters like the IP address and port of the destination 10gbE. After running the commands given below, the ROACH starts sampling the input signals given to the iADC card and makes UDP packets that can be sent over 10GbE. The size of packet is 8k bytes with 4k bytes of data from each iADC input signal.

1. Open a terminal and log into CTRL_PC. $ ssh -X [USER_CTRL_PC]@[CTRL_PC]
2. move into the directory. $ cd [SOFT_DIR]
3. Run the configuration script. $ ./[CONFIG_PY]
4. Run the initializing script. $ ./[INIT_PY]

III ACQUIRING DATA PACKETS :

Gulp captures the packets along with the header.

1. Open a terminal and log into CTRL_PC. $ ssh -X [USER_CTRL_PC]@[CTRL_PC]
2. move into the directory $ cd [ACQ_DIR]
3. edit the file gulp.c for number of packets (if required) in the line number 295 and then run make command. -1 if infinite number of packets to be captured. Minimum number of packets = 8

   $ vim gulp.c # In line 295: num_packets=<enter the number of packets desired>

   $ make

4. Run the acquisition script. $ ./gulp -i eth0 > <name of dump file.dat>

In case of infinite packet capture, it will capture untill you press cntrl+C. Keep capturing the data for 10-15 minutes.

IV BREAK DATA IN CHUNKS OF 2GB PACKETS :

NOTE : THIS IS ONLY FOR DATA GREATER THAN 2GB (Else segmentation fault).

1. move into the directory $ cd [ACQ_DIR]
2. Run the below command by changing the OutputFile name and value in skip = 1, 2, 3, 4 ... Till you have sufficient data. Usually 6-8 GB for strong pulsar data is enough.

dd if=<name of dumped file.dat>.dat of= <name of dumped file.dat1>.dat bs=1999999832 count=1 skip=0

This command takes the first 2GB of the dumped InputFile and writes in the OutputFile with ByteSize 1999999832 without skipping the data equals to ByteSize.

dd if=<name of dumped file.dat>.dat of= <name of dumped file.dat2>.dat bs=1999999832 count=1 skip=1.

This command takes second 2GB of the dumped InputFile and writes in the OutputFile with ByteSize 1999999832 and skipping data equals to ByteSize*skip.

So on.....

V DEPACKETIZING THE DATA PACKETS :

For each of the above 2 GB files, the Depacketization codes have to be run separately.

Give different names to the binary files created in the end.

1. From binary to ascii:

./gulp_ascii_8.o <name of dumpedfile1.dat> <packet_size> <scaling factor>

#as per Swapnil's report.

for_pmon_pol0_pol1.c: this script processes both the polarizations and generates 16 text files received from different X-Engines. depack_both_pols.c

./pole12 <input file name.dat> <packet size> <scaling factor>

Example: ./pole12 example 554 4

packet size=554 and scaling: 4 taken as standard by us. The above command has also separated the packets into 8 different files corresponds to 8 X-engines data.

Now to convert them to single interleaved file with all 512 channels enter the following command. This is required because file 1 from X-engine 1 has channels 0,8,16,24... 504 processed and file 2 from X-engine 2 has channels 1,9,17,25,...505 and so on

2. ./intleave_signed.o > <name of ascii interleaved file1.txt>

At this satge interleaving is done.

#as per Swapnil's report.

Interleaving of Polarization-Q: This scripts generates a single text file coming from different X-engines.

File name:pol1.c

Syntax to run the the script: interleave_both_pols.c

./interpol1 > <outputfilename.txt>

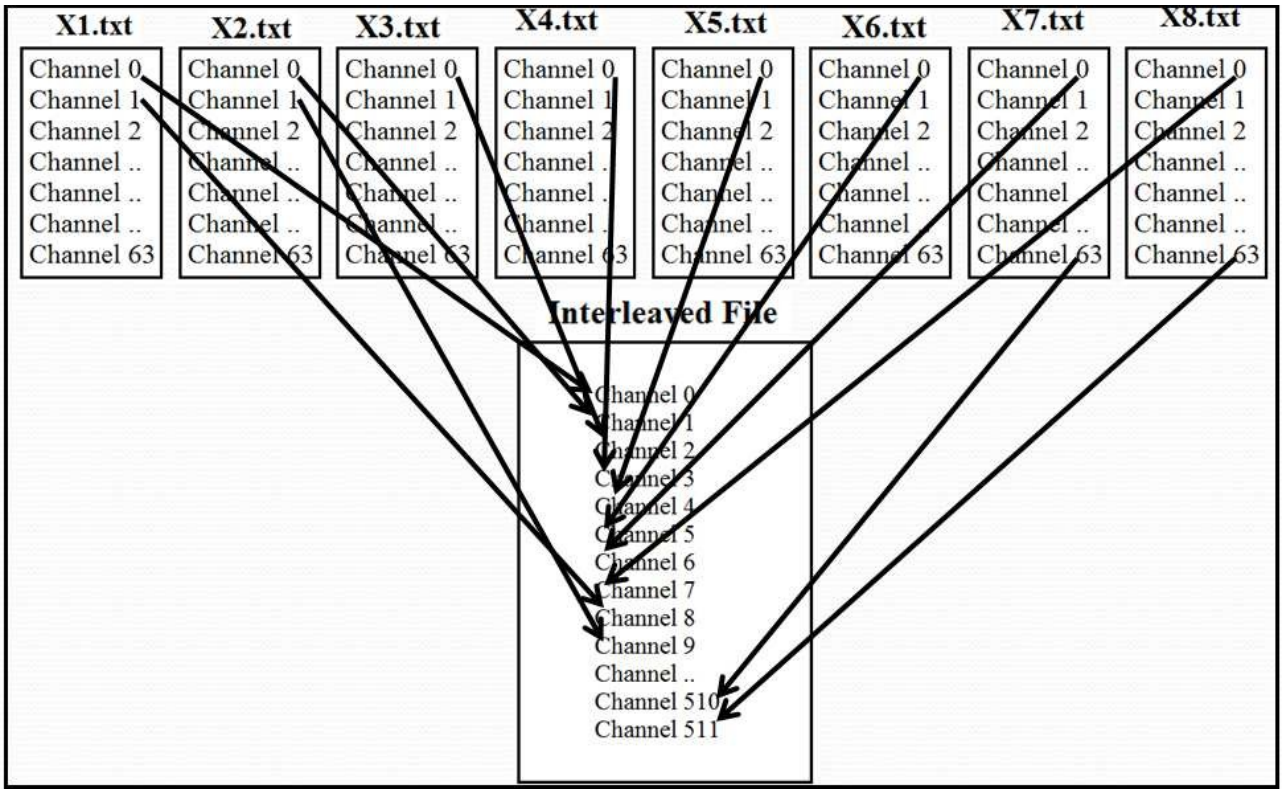
Example: ./interpol1 > pole1.txt

File name:pol0.c

Syntax to run the the script:

./interpol0 > <outputfilename.txt>

Example: ./interpol0 > pole0.txt



In the figure above , X1.txt contains data from X-engine 1, X2.txt contains data from X- engine 2 and so on. A “packet count” can be an add-on to the system. It will ensure interleaving of time synchronized packets.

VI CONCATENATING THE DATA :

NOTE : THIS IS ONLY FOR DATA GREATER THAN 2GB (Else segmentation fault).

Each of the above binary files will be of 441 MB.

We have to concatenate these files to see proper pulsar shape on pmon.

The command used is :

cat <name of binary interleaved file1.raw> <name of binary interleaved file2.raw> <name of binary interleaved file3.raw> <.....> > <name of final concatenated file.raw>

then, Keep concatenating till the last .raw file.

This final .raw file will contain data from all the binary files created earlier.

This final .raw file can be seen on pmon.

#as per Swapnil's report.

Integration of Cycles:

File name: intgeration.c: This scripts asks a number of cycles to be integrated from the user.

Syntax to run the the script: integration_both_pols.c

Note: this script is working only after head command.

./intgeration <input file name.txt> <outputfilename.txt>

Example: ./intgeration pole1.txt intgpole1.txt

4. Addition of both polarizations:

File name: addp.c: This scripts adds the two files which contains polarization I and polarization Q

Syntax to run the the script: add_both_pols.c

./adpolel1-2 <input file name.txt> <input file name.txt> <outputfilename.txt>

Example: ./adpolel1-2 pole0.txt pole1.txt addpole12.txt

VII ANALYSING THE RESULT USING GNU PROGRAM :

The correlator program writes the results to output file “out.txt” located in [DELAY_DIR] directory. The results are analysed using the software tax (built at GMRT). Follow the steps given below:

1. Open a new terminal and log into GPU_PC. $ssh -X [USER_GPU_PC]@[GPU_PC]

2. Move into the directory TAX_DIR. $ cd [TAX_DIR]

3. Run the command $ ./xtrgsb32_timestamp_8ant -v [DELAY_DIR]/out.txt -c 1,2047 -t 3
   This command will open a window, the screenshot of it is shown below:

V ANALYSING THE RESULT USING PMON PROGRAM :