CASPER Home Group Documentation Mail Archive About
COLLABORATION FOR ASTRONOMY SIGNAL PROCESSING AND ELECTRONICS RESEARCH

2 antenna correlator

From Casper

Jump to: navigation, search

Tutorial 9: 2 antenna GPU correlator

Author: Harshavardhan Reddy (Version 1).

Expected completion time: 1.5 hours

Contents

[hide]

The Hardware and software used for this tutorial

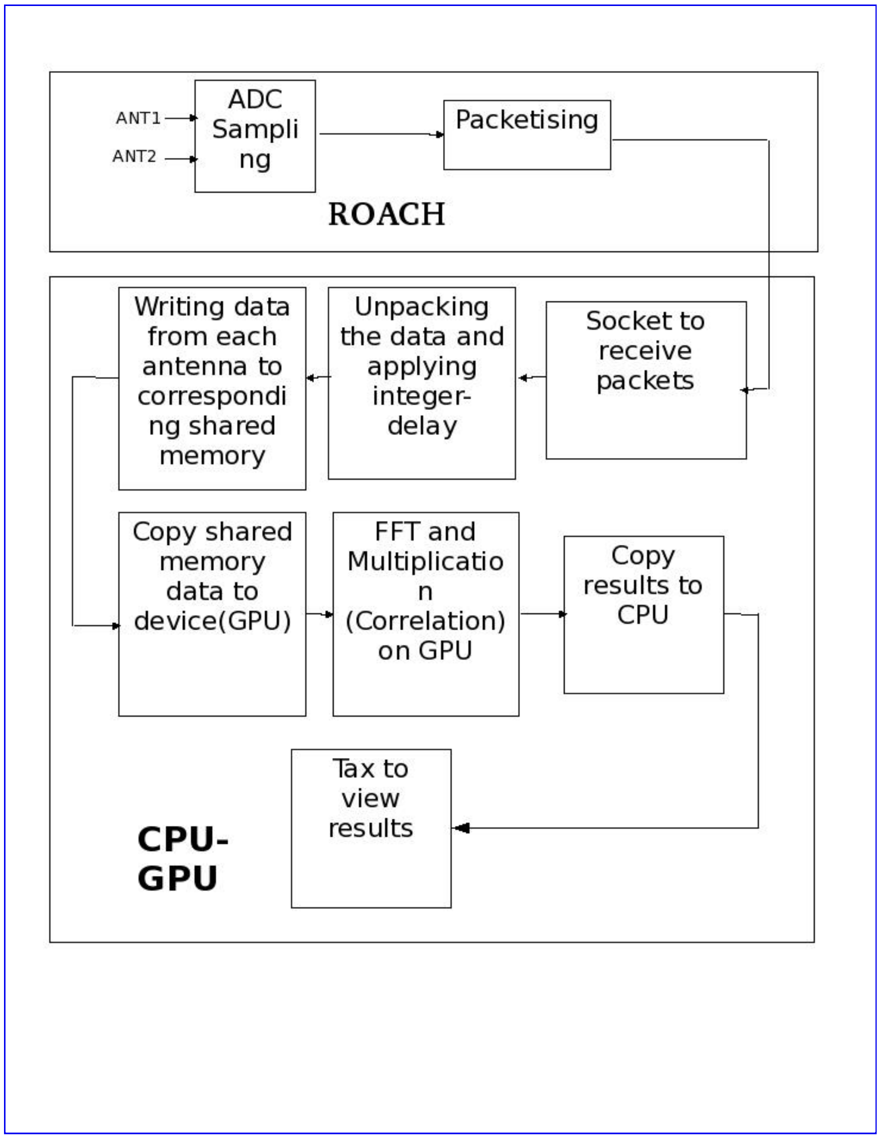
Hardware/Software	Description
PC	Dell Precision T7500 with Intel(R) Xeon(R) CPU X5550 @ 2.67GHz with 12GB RAM
OS	CentOS release 5.5 (Final)
GPU Card	Tesla C1060/C2050
GPU device driver version	270.41.19
CUDA Toolkit version	4.0
10GbE card	Myricom
Wireshark	Version 1.0.3

Cutecom	Version 0.22.0
Python	version 2.6
corr package	corr-0.6.5
Other softwares	PSRDADA and TAX
ROACH unit	version 1.0 Rev 3 2009, uboot : uboot-2010-07-15-r3231-dram, Linux Kernel Image : uImage-jiffy-20091110, iADC : BEE2 DUAL 1 GHz ADC BOARD version 1.1
Signal generator	Signal generator to feed clock of 400MHz, 0dbm to ROACH unit through iADC's clk_i input.
Waveform generator	Input signals from waveform generator should be of -13dbm@400 MHz BW (Total power over BW) to the I+ & Q+ inputs of iADC.

Introduction

In this tutorial, you will run the correlator using ROACH and GPU to find the correlation between two signals and also the effect of delay on correlation.

Dataflow


ANT1
ANT2
ADC Sampling
Packetising
ROACH
Writing data from each antenna to corresponding shared memory
Unpacking the data and applying integer-delay
Socket to receive packets
Copy shared memory data to device(GPU)
FFT and Multiplication (Correlation) on GPU
Copy results to CPU
Tax to view results
CPU-GPU

Step1:Setup

The lab at workshop is pre-configured with the necessary softwares. Please refer the file “LOCATIONSandFILES.pdf” in the home/Desktop area or LOCATIONSandFILES slides displayed , for the locations/directories and files information required in the tutorial. Note: All the following connections given below are already done

1. Connect the ROACH board to the controlling PC using 100MBPS ethernet and RS-232 serial cable.

2. Feed the clock of 400MHZ,0dbm to the clk_i input of the iADC card from the signal generator.

3. Connect the input signals to i and q of the iADC from the noise generator. The input signal should be of -13dbm(with 2 way power splitter)@200MHz bandwidth at the iADC card input.

4. Connect the ROACH board to the CPU_GPU PC using 10gbE cable. Connect one end of 10gbE cable's CX4 connector to the CX-4 port of ROACH and the other end to the 10gbE card on the CPU-GPU PC.

5. Boot the ROACH using mmcboot and copy the [TUT9_BOF_FILE] to the directory [TUT9_FPGA_PROG_BOF_DIR] on controlling PC. Design details of the bof file are not given. But, if Matlab is available, the MDL file can be opened and design can be viewed. The MDL file [TUT9_MDL_FILE] is located at [TUT9_MDL_DIR].To have a look at the MDL file click here

Step2.Configuring the ROACH

In this step, we will program the ROACH with the [TUT9_BOF_FILE] and configure parameters like the ip address and port of the ROACH's 10gbE. After running the commands given below, the ROACH starts sampling the input signals given to iADC card and sends the data in UDP packets over 10gbE cable to CPU-GPU PC. The size of packet is 8k bytes with 4k bytes of data from each iADC input signal.

1. Open a terminal.If we are in [TUT9_CTRL_PC] no need to log in. Or else log in to the [TUT9_CTRL_PC] by using ssh. $ssh -X [TUT9_USER]@[TUT9_CTRL_PC_IP] password:[TUT9_CTRL_PC_PASS]

2. [TUT9_USER]@[TUT9_CTRL_PC]$cd [TUT9_PYTHON_DIR]

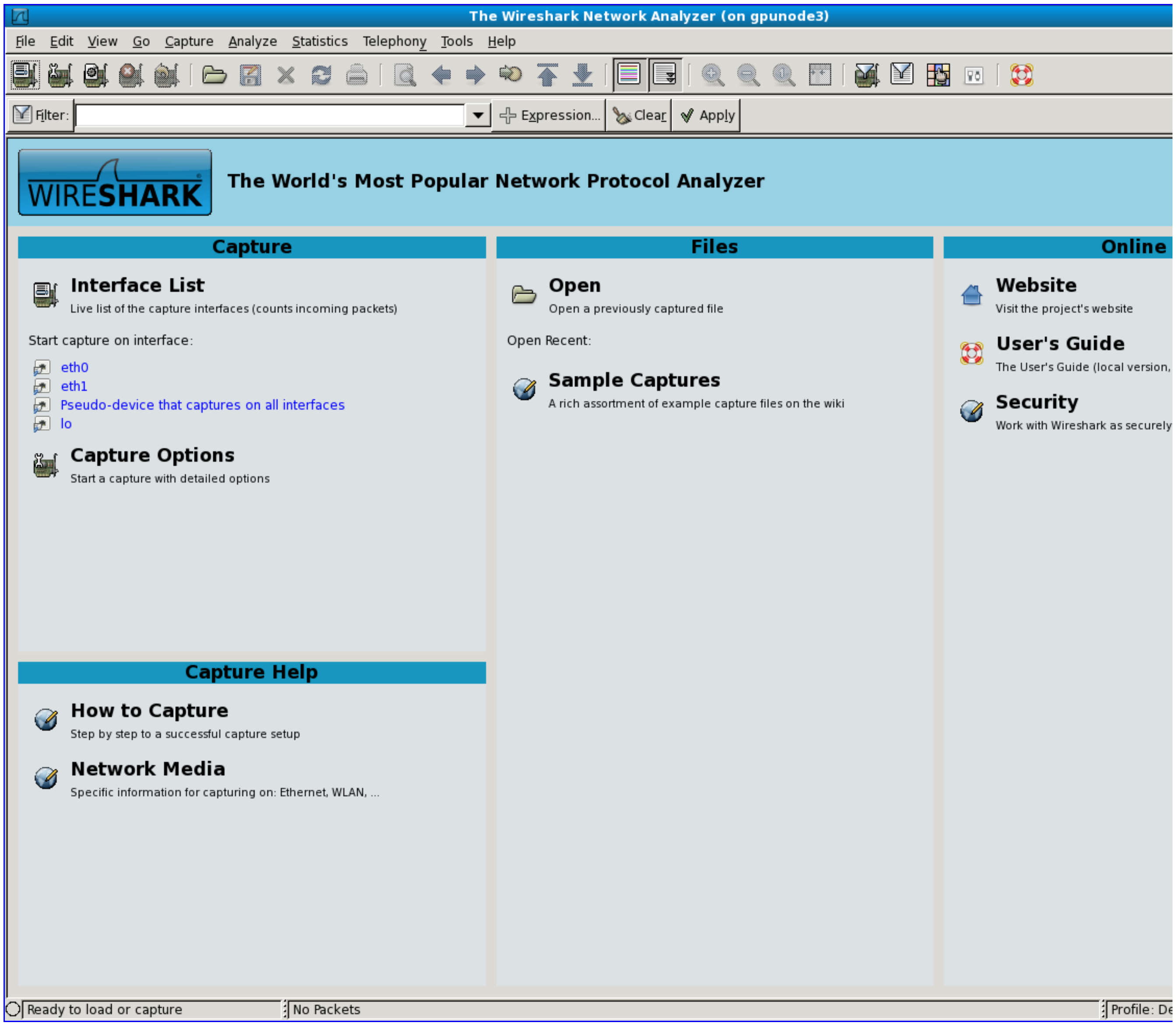
3. [TUT9_PYTHON_DIR]$./[TUT9_CONFIG_PY]

Step3.Checking for packets

After configuring the ROACH to send data to CPU-GPU PC, we need to ensure that data is being received by the CPU-GPU PC. We will do this by using wireshark software.

1. Open a terminal

2.[TUT9_USER]@[TUT9_GPU_PC]$sudo wireshark This command opens the wireshark window which appears as shown below.


The Wireshark Network Analyzer (on gpunode3)
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression...
Clear
Apply
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start capture on interface:
eth0
eth1
Pseudo-device that captures on all interfaces
lo
Capture Options
Start a capture with detailed options
Capture Help
How to Capture
Step by step to a successful capture setup
Network Media
Specific information for capturing on: Ethernet, WLAN, ...
Files
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files on the wiki
Online
Website
Visit the project's website
User's Guide
The User's Guide (local version,
Security
Work with Wireshark as securely
Ready to load or capture
No Packets
Profile: De

3. In the window opened click on 'Capture' and click on 'Interfaces' in the menu opened

4. Another window opens which shows information about all the interfaces available on the CPU_GPU PC. The large numbers for 'packets' and 'packets/s' for the device corresponding to [TUT9_TENGB_IP] confirms the packets being sent on 10gbE. The 'packets' denotes the total number of packets and 'packets/s' denotes the packets per seconds. Now, close the wireshark window.

Step4.Creating shared memory

After confirming that packets are being received, we need to create shared memory to write the data into. Here, we create two shared memories with 4 buffers each of 256 MB. One shared memory for each of the input signals. The shared memories are represented by the keys 'bada' and 'cada'.

1. Open a terminal

2. [TUT9_USER]@[TUT9_GPU_PC]$cd [TUT9_DELAY_DIR]

3. [TUT9_DELAY_DIR]$./ada.csh

This command actually destroys the shared memories, if any, created before and creates new shared memories. After running this command, there will be message printed which look as given below

Destroyed DADA data and header blocks

Destroyed DADA data and header blocks

Created DADA data block with nbufs=4 bufsz=268435456

Created DADA header block with nhdrs = 8, hdrsz = 4096 bytes

Created DADA data block with nbufs=4 bufsz=268435456

Created DADA header block with nhdrs = 8, hdrsz = 4096 bytes

Step5.Running the correlator program

After configuring the ROACH and creating the shared memory, we need to run the correlator. By running the commands given below, the CPU-GPU PC starts the correlator program, initialises the variables needed and waits for the shared memory data. After receiving the shared memory data, the data is copied into the GPU and correlation operation is done in GPU. After the correlation operation, the results are written back to CPU and finally written to a file named 'out.txt' in the [TUT9_DELAY_DIR] folder. Now follow the steps given below

1. Open a new terminal

2. [TUT9_USER]@[TUT9_GPU_PC]$cd [TUT9_DELAY_DIR]

3. [TUT9_DELAY_DIR]$gmrt_correlator -k ~/dada_key_list -g 1024 -q 4 268435456 2048 2 -w

After running this command, the program waits for data from the shared memories. In the command, the arguments give the following information to the process -k [filename] This argument provides the location and name of the file in which the shared memory keys are present

-g [number] This argumentt gives the number of FFTs that are to be computed parallelly on GPU.The optimised value is 1024

-q [count] This argument gives the number of streams to be run on GPU.The optimised value is 4. The -g and -q options combinedly gives that 1024 FFTs are parallelly computed using 4 streams on GPU.

The number 268435456 gives the integration time. This number gives the amount of data to be accumulated for one integration. The time to sample this amount of data at 400MHZ comes out to be 0.671088. So, our integration time is 0.671088. For every one integration the program writes the integrated output data to the file 'out.txt'.

The number 2048 gives the FFT size.

The number 2 gives the number of antennas or the number of input signals.

-w This argument informs the program to write the output to file 'out.txt' located in [TUT9_DELAY_DIR}

More information about the function can be known by command $gmrt_correlator -h.

Step6.Running the packet capture program

As the correlator program is running and waiting for the data from shared memory, we need to capture the data over 10gbE cable and write it to the shared memories. By running the following commands, the CPU-GPU PC starts capturing the packets over the 10gbE, unpacks the packets and writes the data from each signal to corresponding shared memories. As mentioned earlier, each packet is of 8k bytes with 4k bytes from each input signal.

1.Open a terminal

2. [TUT9_USER]@[TUT9_GPU_PC]$cd [TUT9_DELAY_DIR]

3. [TUT9_DELAY_DIR]$gmrt_udpdb_multi -i [TUT9_TENGB_IP] -p [TUT9_TENGB_PORT] -H ~/header 2 bada cada 0 0 -n 10

After running this command, the program starts capturing packets,unpacks the data and writes the data to corresponding shared memories. The arguments give the following information to the process

-i [ipaddress] ipaddress of the interface from which the data is being received

-p [port] port that is open for UDP packets to be received

-H [filename] location and name of the header file.This header file gives the information regarding the sampling frequency,number of bits per sample etc. The file contents can be viewed using the command $cat ~/header

The number 2 in the command arguments gives the no. of antennas or the input signals

bada and cada are the keys of shared memories to which the data is to be written

The numbers 0 0 are the delays for each input. We will come back to this later. For now just keep them 0 0

-n [secs] time for which the data needs to be received.After the amount of time given using this argument the packet capturing program and the correlator program stops.

[Note : If the packet capturing program does not stop after the amount of time given using -n option, then, first stop the packet capturing program by ctrl+c, and go to the terminal started in step 4 and type the command $./dda.csh This command actually destroys the shared memories and stops the correlator program.]

More information can be obtained by using the command $gmrt_udpdb_multi -h

While running this program, the following messages will be printed on the screen. [Note: The numbers may vary]

```
[2011-09-08-17:04:57] Using current time for UTC_START = 2011-09-08-11:34:57

[2011-09-08-17:04:57] B/sample=2.000000, TSAMP=0.002500, B/second=800000000.000000

[2011-09-08-17:04:57] header marked filled

start of udpdb_multi : 1315481697

timestamps 1315481697 0

[2011-09-08-17:04:57] stream[0] delay offset -0.0000366729982488 s -> -14669 bytes -> 0 bytes

[2011-09-08-17:04:57] stream[1] delay offset -0.0000008345460287 s -> -333 bytes -> 0 bytes

[2011-09-08-17:04:57] T=0.0, R=0.0, D=0.0 MiB/s bsleeps=0 dropped=0

[2011-09-08-17:04:57] START : received packet 477233637

timestamps 1315481697 671088

[2011-09-08-17:04:58] stream[0] delay offset -0.0000366720269085 s -> -14668 bytes -> 0 bytes

[2011-09-08-17:04:58] stream[1] delay offset -0.0000008345475635 s -> -333 bytes -> 0 bytes

[2011-09-08-17:04:58] T=537.0, R=537.0, D=0.0 MiB/s bsleeps=1106278 dropped=0

.......................................
```

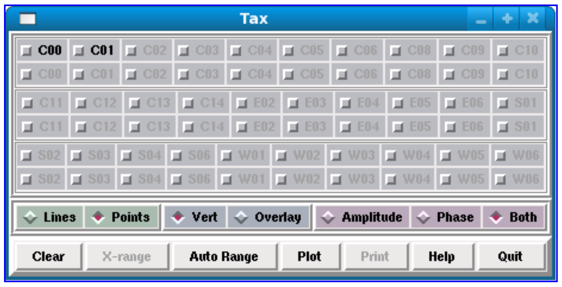
Step7:Analysing the result using tax program

The correlator program writes the results to output file “out.txt” located in [TUT9_DELAY_DIR] folder. The results are analysed using the software Tax(built in GMRT). Follow the steps given below

1. Open a new terminal

2. [TUT9_USER]@[TUT9_GPU_PC]$cd [TUT9_TAX_DIR]

3. [TUT9_TAX_DIR]$./xtrgsb32_rawsamp -v [TUT9_DELAY_DIR]/out.txt -c 1,2047 -t 0 This command will open a window which looks as below



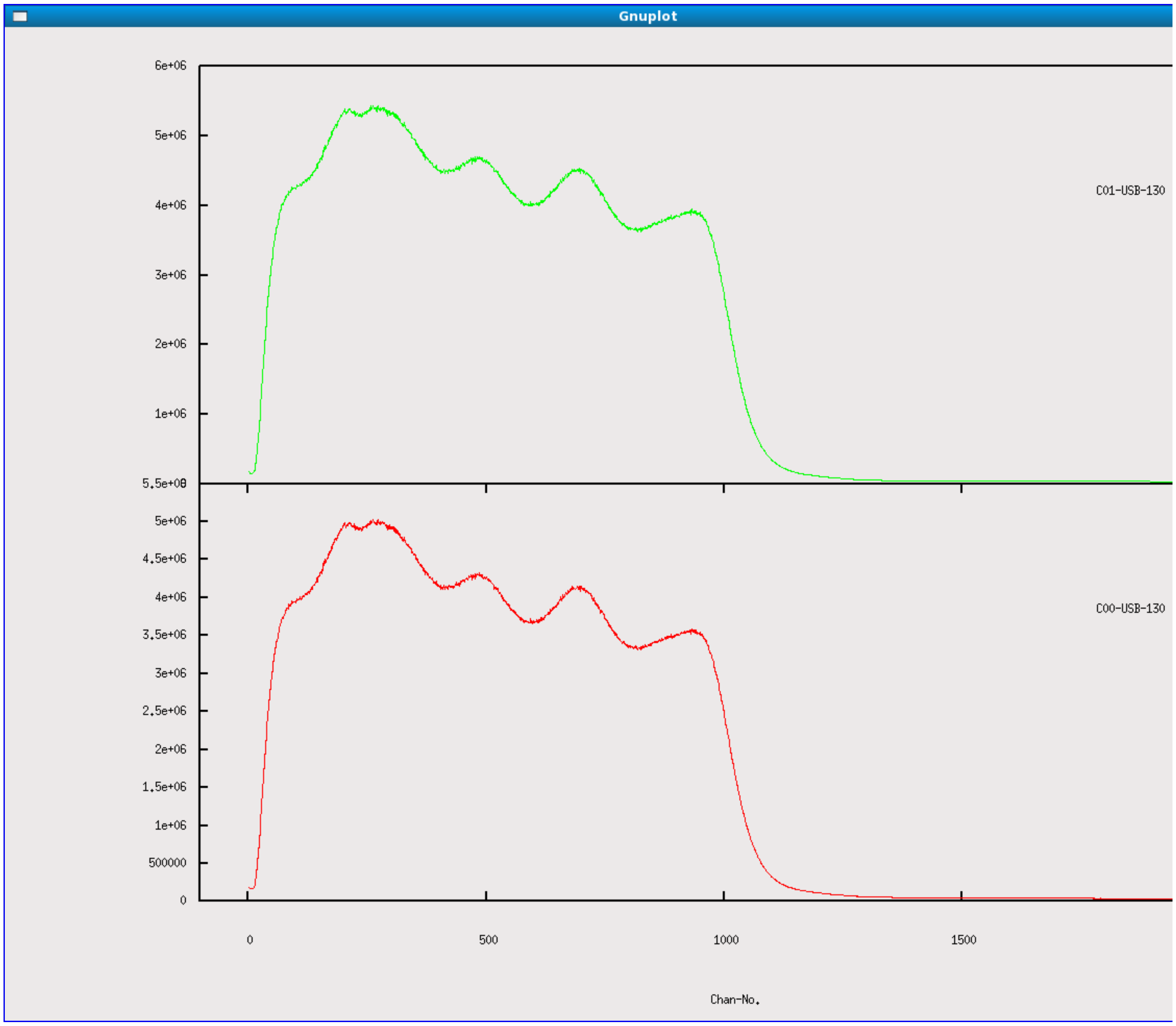
The arguments given to this command represents

-v [filename] the output filename to which the correlation output results are written

-c [start,final] this argumnet gives the channel range to be plotted.start and final values can be anywhere between 1 to 2047 as our FFT size is 2048. But, with final > start

-t [timestamp] this argumnet gives the output of which integration to be plotted. It can be any number in between 0 to t, where t is given by t = { [time given in packet capture program] / [integration time] } - 1, here it is 13

In the window opened click on 'C00','C01','Amplitude' and then click 'Plot'. A window opens which shows the self amplitudes of both the signals. The plot corresponding to 'C00-USB-130' is of ADC input 'i' and the plot corresponding to 'C01-USB-130' is of ADC input 'q'.



The x-axis is frequency channels and the y-axis is power. Bandwidth is 200MHZ as sampling frequency is 400MHZ.

4. Close the plot window and click 'quit' on tax window.

5. Now we will plot the cross amplitude and phase using the command given below

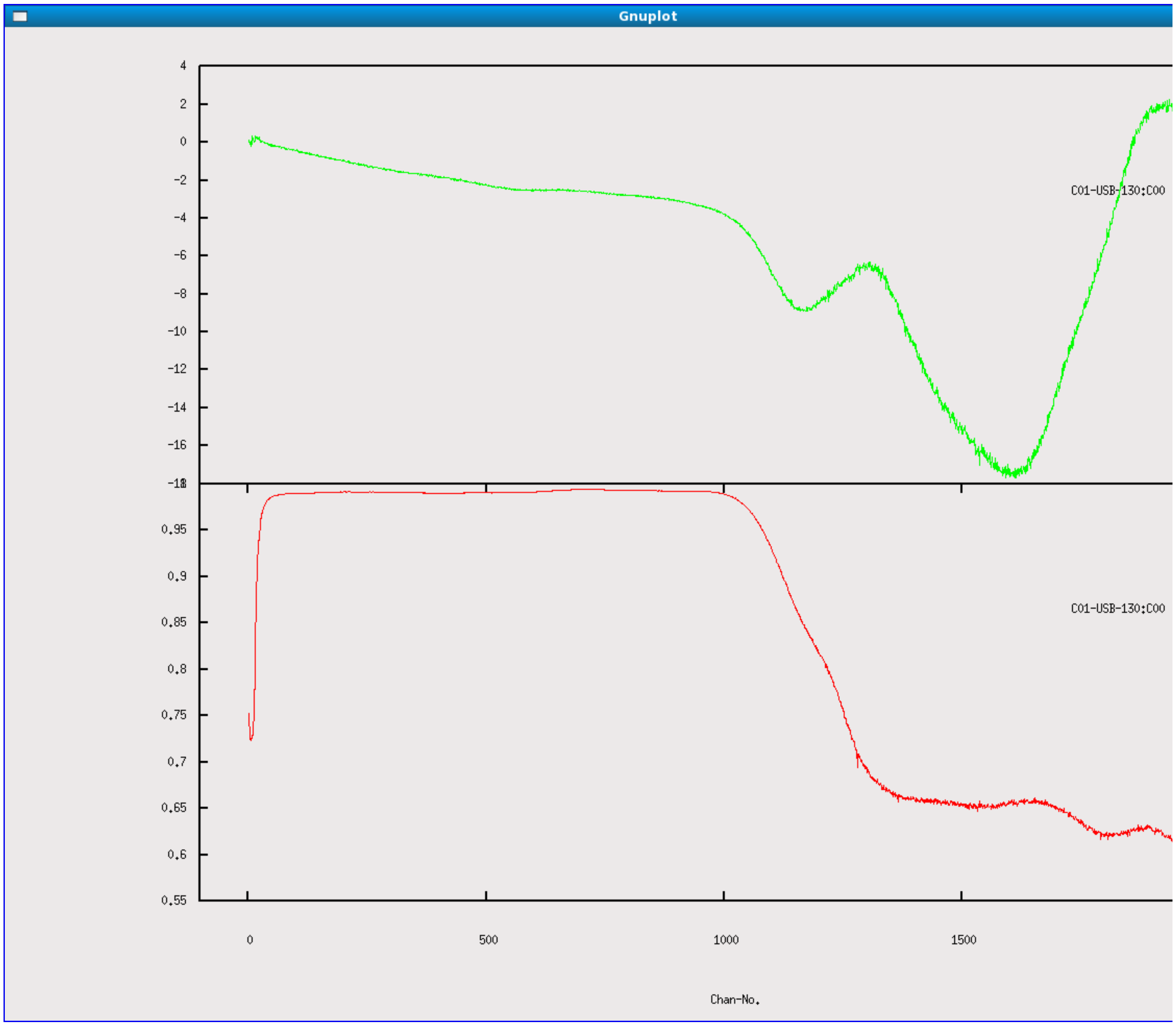
```
[TUT9_TAX_DIR]$./xtrgsb32_rawsamp -v [TUT9_DELAY_DIR]/out.txt -c 1,2047 -t 0 -r C00 -n 1
```

This command is similar to the previous command with extra arguments.

-r [antenna] this argument gives the reference antenna with which cross is to be plotted

-n 1 this agrument gives information that the cross amplitude to be normalised

After running this command a window similar to tax window opens with only 'C01' option.Click on 'C01' and click on 'Plot'. A window opens wich shows the cross amplitude(normalised) and cross phase.



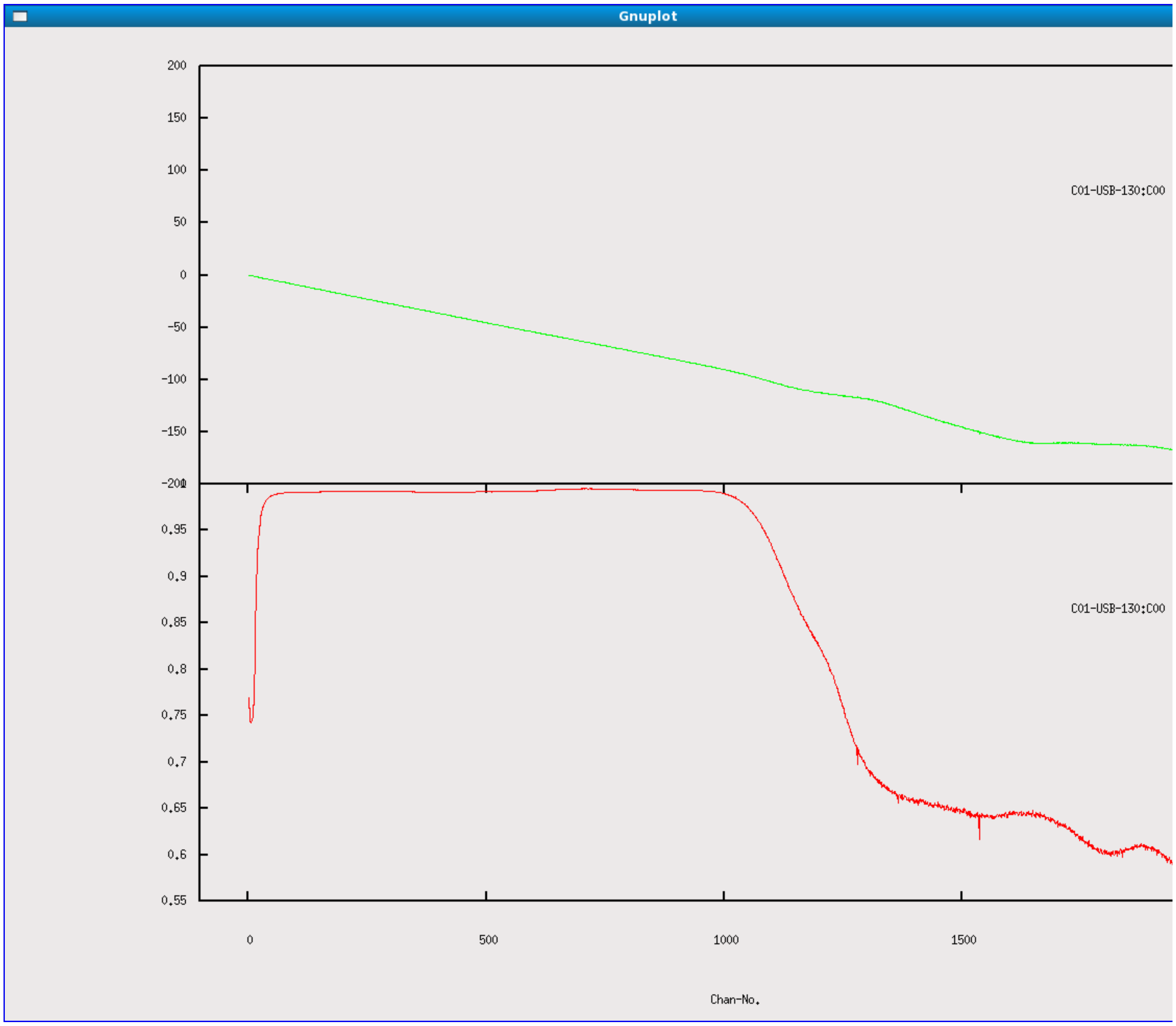
For the correlated signals, the cross amplitude has to be nearly 1, while phase has to be nearly 0.

Changing delays

Now we will come to the effect of delay on correlation. For smaller delays, the cross amplitude doesn't change.But for larger delays, the signals tend to decorrelate and the effect can be seen even on cross amplitude. For this, repeat the whole process from step 4 to step 7, with a small change in step 6 i.e; the packet capturing program.

As mentioned earlier, the 0 0 in the packet capturing command are delays corresponding to each input. If we give 1 0, then the first signal will be advanced by 1 sample and if we give 0 1, then the second input signal will be advamced by 1 sample.

So repeating the process from step4 to step7, with the change 0 1 instead 0 0 in the 3rd step of step6 will give you change in the cross phase. The change will be 180 degrees over the whole bandwidth. This is because, adding delay of 1, is similar to adding extra cable length equal to the one wavelength metres to the correspinding input, which results in the difference in phase of 360 degrees. But as we are seeing only half the spectrum, the overall change in the phase will be 180 degrees.



We can repeat this process with various delay values, by changing 0 0 in the 3rd step of step6 with 0 n (n should be a +ve number). As the n value is increased, we can see that the cross amplitude gets decreased and more ramps in the cross phase. For very high values of delay, the cross amplitude becomes 0, and cross phase becomes random.

Offline testing

Offline testing is for processing the raw data signals that are captured and are already present on the disk. It can be done in 3 steps

Creating a file with the names of raw data files

Enter the file names of raw data files on disk into a file named file.list. This can be done by vim editor.

1. Open a terminal

2. [TUT9_USER]@[TUT9_GPU_PC]$cd [TUT9_DELAY_DIR]

3. $vim file.list

Enter the names of [TUT9_RAW_DATA_FILES] in separate lines and save it. For example purpose, file_example.list exists which contains the names of raw data files from noise source.

Running the correlator program

[TUT9_DELAY_DIR]$gmrt_correlator -f file.list -g 1024 -q 4 419430400 2048 2 -w

In this command the -f option gives the file which contains the names of raw data files. Everything else is similar to the step5.

Analysing the result

Refer to step7:Analysing the result using tax program.But with -t option always 0.

Conclusion

In this tutorial, you have seen the output of a GPU based correlator and also the effect of delay on correlation phase and amplitude.