National Centre for Radio Astrophysics

Internal Technical Report

GMRT/CRR/017-Dec2013

Packetized Correlator for GMRT

Sandeep C. Chaudhari

Mekhala V. Muley

Shelton G. J.

Irappa M. Halagali

Ajithkumar B.

Digital Back-End Group

Email: scc@gmrt.ncra.tifr.res.in

mekhala@gmrt.ncra.tifr.res.in

shelton@gmrt.ncra.tifr.res.in

irappa@gmrt.ncra.tifr.res.in

ajit@gmrt.ncra.tifr.res.in

Objective: To provide Technical Details, Test Results of Packetized correlator for GMRT.

Revision	Date	Modification/ Change
Ver. 1	10 September 2013	Initial Version

1. **Introduction :**
   As a part of the GMRT upgrade efforts, the Backend systems are undergoing major changes to achieve the upgrade system specifications, like increased bandwidth of 400MHz, direct processing of RF signals, increased dynamic range, improved channel resolution in the digital backends etc.
   The digital backends development took two approaches - with one using complete FPGA based hardware solution and second with GPU for the computations.
   At present an 8 antenna dual polarization Correlator based on FPGA using CASPER technology was built.

2. **Target performance specifications:-**
   System Specifications
   1. Digital System :
      1. ADC No. of Bits : 8 bits
      2. Inst Bandwidth : 400 MHz
      3. FFT No. of Channels : 8K
      4. Mode Available : Full polar
      5. Coarse and Fine Delay correction
      6. Fringe rotation
      7. Interferometer with dump times ~ 100 ms
      8. Incoherent and Phased array beam outputs : at least 2 beams for each; with full time resolution
      9. Pulsar back-ends attached to the beam outputs

3. **Design description :**

The CASPER Correlator (Collaboration for Astronomy Signal Processing and Electronics Research) is a packetized, highly scalable design using ROACH hardware developed in collaboration with MeerKAT, South Africa.

Hardware Components:-

1. ROACH board:- ROACH stands for **R**econfigurable **O**pen **A**rchitecture **C**omputing **H**ardware board is a standalone FPGA processing board build around Xilinx Virtex-5 (SX95T for DSP-slice-intensive applications).
   1. A seperate PowerPC runs Linux and is used to control the board viz. Program the FPGA and allow interfacing between the FPGA "software registers/BRAMs/FIFOs" and external devices using Ethernet).
   2. Two QDR SRAMs provide high-speed, medium-capacity memory (for corner-turns)
   3. DDR2 DIMMS – one for FPGA – provides slower-speed, high-capacity buffer memory and another for PowerPC in order to boot Linux/BORPH(**B**erkeley **O**S for **R**e-**P**rogrammable **H**ardware).
   4. 2 Z-DOK Connectors for iADC interfacing.
   5. Four CX-4 connectors provide a total 40Gbits/s bandwidth for interconnects to XAUI/10GbE-capable devices.

2. iADC board :- 1x Atmel/e2V AT84AD001B 8-bit dual 1Gsps, with clock 10MHz – 1GHz 50Ω 0dBm.
3. Fujitsu XG-700 10Gbe 12-port switch.
4. Control Computer

Software/Toolflow :-

1. CASPER MSSGE Toolflow -
    1. Matlab-Simulink
    2. System Generator
    3. Xilinx 11.5 - EDK
2. Ubuntu Linux 11.10 (Kernel 3.0.0-17-server on x86_64)
3. Python 2.7 or above & packeges – struct, time, sys, numpy, pylab, katsdisp, signals, h5py, corr

Implementation :-

Packetized correlator design is based on FX architecture, where channelization takes place first in F-engines and cross-multiply-accumulate operation occurs afterwards in X-Engines. Depending upon interconnections between F-engines and X-engines, two approaches were followed. (please refer CASPER Memo 017 – Packetized FX Correlator Architectures by P. McMahon, A. Langman et.al.)

1) 10Gbe design:- This architecture uses a single 10Gbe switch as it's interconnect mechanism between F-engines and X-engines and is most basic form of architecture. The switch needs 2N 10Gbe ports for N antenna correlator.

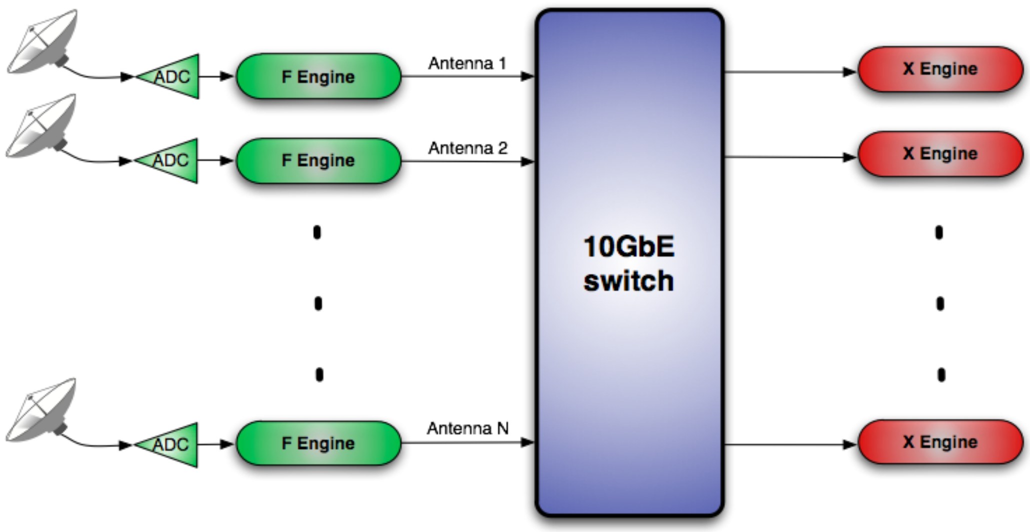


Fig 1: 10Gbe Design

2) XAUI design:- In this design, packets are not fed directly into the network switch from the F-engine as shown in *Fig. 1* that although ethernet provides full bi-directional links, connections are inefficiently used unidirectionally, with F-engines sourcing data but not sinking any. *Fig. 2* shows how packets can be sent from F-engine boards directly to X-engie boards over point-to-point XAUI links, X-engine can then loop this F-engine data out over a bi-directional exchange port. The data returning from the switch is the same as before the

optimization. The arrangement halves the number of required switch ports, as each switch port link is now used bi-directional.

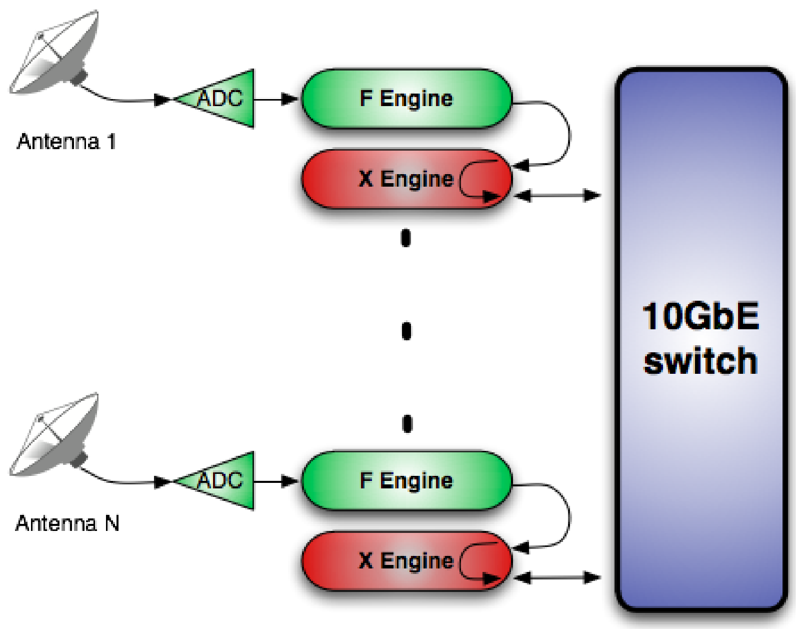


Fig 2: XAUI Design

The GMRT has adopted the later one approach of XAUI based design considering it's advantages.

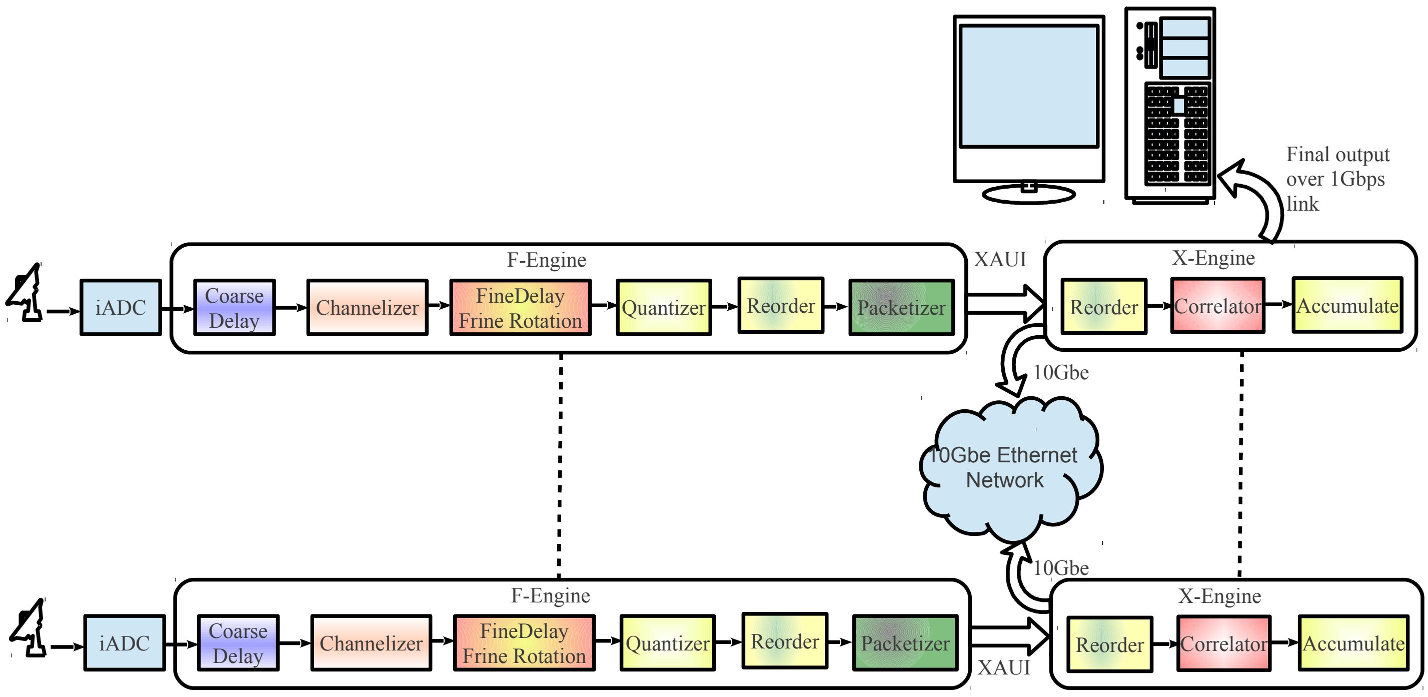


Fig 3: XAUI Based Packetized Design Setup

F-engine operation:-
As shown in Fig. 3, the F-Engines perform following main functions :

- Digitization of band limited RF signal (iADC)
- Channelization of the sampled signal (PFB+FFT)
- Apply delay correction and fringe rotation (Coarse Delay + Fringe Rotation)
- Apply gain correction (
- Re-quantization
- Reordering of data (packetization preparation)
- Packetization of data for transport over XAUI

Digitization :- iADC is used for digitization of analog broadband signals at a sampling rate of 800MSPS to allow a bandwidth of 400MHz. For synchronous correlation operation FPGA clock has been derived from ADC clock which is 1/4th of sampling clock rate.

Delay and phase compensation :- The delay compensation in FX correlator is constructed in two parts consisting of coarse delay (in units of ADC samples) and a fractional components of ADC samples.
Coarse delay correction has been implemented in time domain using an addressable memory configured as a programmable shift register and fractional delay correction has been implemented in frequency domain by multiplying the FFT output with an indexed sin/cos lookup table.
The control system for both coarse delay and fine delay has been a key that allows for timed loading of new values, to synchronize fine delay with coarse delay block.

Channelization :- It employs a polyphase filter bank (PFB). In this wideband design of correlator an N-point real CASPER butterfly FFT produces N/2 complex output channels per spectrum; only the real half is output (since the other half is a mirror image) in order to save logic resources. Bit growth is limited through the butterfly stages by an optional down shift at each stage. Care must be taken not to be too aggressive with the shifting schedule, especially with longer FFTs and risking degradation of SNR.

Quantization :- The PFB+FFT produces complex data at 36 bits per sample is re-quantized to a complex representation of 4 bits real + 4 bits imaginary. This saves interconnect bandwidth between boards. This quantization operation must be performed carefully.

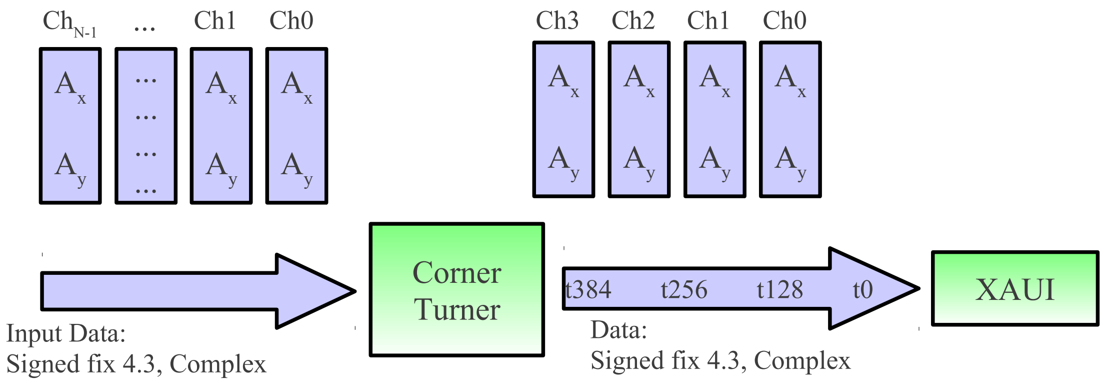


Fig 4: Pacektizer in F-engine over XAUI

Packetization :- The data stream is prepared for transport by the network prior to packetization for distribution to X-engines by a matrix transpose operation to group a selection of time series samples from single FFT channel into each packet.
The number of time samples in a packet is chosen to match the internal accumulation in the X-engine core, typically 128.
In contrast to the F-engines, the X-engines run asynchronously. They buffer and reorder packets received from multiple F-engines before processing can take place by labeling packet with a time stamp, source antenna and which frequency it represents.

X-engine operation :- Each X-engine in the system operates independently of all other engines in the system and processes a subset of the frequency band. All X-engines are identical and are differentiated only by their logical addresses. The X-engine's primary functions are to

- receive packets from network, reorder and account for missing packets,
- cross-multiply each antenna's data with every other antenna's and
- accumulate results

Systems optimization for lowest switch port counts was done by routing F-engine data through the X-engine boards, require some fraction of this data to be processed on the local X-engine board. The data must be retained locally and implemented with a small FIFO buffer on each X- engine which keeps copies of all packets destined for its own IP address and send rest to switch where all X-engines are connected to each other is called a loopback mux.

The X-engine core :-
The X-engine core is based on Lynn Urry's algorithm to calculate the number of products or baselines for N inputs as n_{bls} = N(N+1)/2. Fig. 5 illustrates it's operation.
The X-engine cores run at the native clock rate of the FPGA (210MHz), which is asynchronous to the rest of the system. The serialised X-engine cores are allowed to freewheel by purposefully clocking these cores faster than the incoming datarate to ensure that buffers will always underrun.
The concept of *windowed data* is used, where incoming packetized antenna data is buffered and assembled on the FPGA into blocks of data, or windows, which are aligned with the X-engine core's next window boundary before being processed. The X-engines remain free-running, processing windows back-to-back and process the window whether it's valid or not. Thus X-engines periodically process null, or invalid windows. The rate at which the windows are emitted from the buffer for processing is controlled by timestamps of arriving packets. As a packet with a new timestamp enters the ring buffer, a previous window is marked as ready to be shipped. Then

wait for the X-engine core to start a new window before emitting all the data for the oldest buffered timestamp to be processed. This process is outlined in *Fig.6*

The X-engines perform a small accumulation internally to reduce the output datarates so as to keep the output datarates less than or equal to the input datarates.

Fig. 5 : This table shows the output ordering for the 8-antenna core described in

0,0	7,0	6,0	5,0	4,0
1,1	0,1	7,1	6,1	5,1
2,2	1,2	0,2	7,2	6,1
3,3	2,3	1,3	0,3	7,3
4,4	3,4	2,4	1,4	0,4
5,5	4,5	3,5	2,5	1,5
6,6	5,6	4,6	3,6	2,6
7,7	6,7	5,7	4,7	3,7
0,0	7,0	6,0	5,0	4,0
1,1	0,1	7,1	6,1	5,1
2,2	1,2	0,2	7,2	6,1
3,3	2,3	1,3	0,3	7,3
4,4	3,4	2,4	1,4	0,4
5,5	4,5	3,5	2,5	1,5
6,6	5,6	4,6	3,6	2,6
7,7	6,7	5,7	4,7	3,7

The advantage of the two QDR parts with their narrower interfaces by accumulating for longer than necessary within the X-engine. This further reduces the datarates, allowing the output to be serialized into narrow, but longer, vectors based on the QDR memories. The vector accumulator is greatly simplified because the QDR interface operates synchronously with the FPGA fabric which results in a simpler vector accumulator design that reduces fabric logic requirements.

The four complex cross-pol terms can be demuxed by a factor of up to eight times if the real and imaginary components are also serialized. Some efficiency is lost for cores operating on an even number of antennas, where N_{ants} results from the last stage are redundant as shown in above table. These extra baselines are removed from the output products.

Buffering and Reordering of packets :- Below is the mechanism of buffering and reordering mechanism is explained in *fig.5*

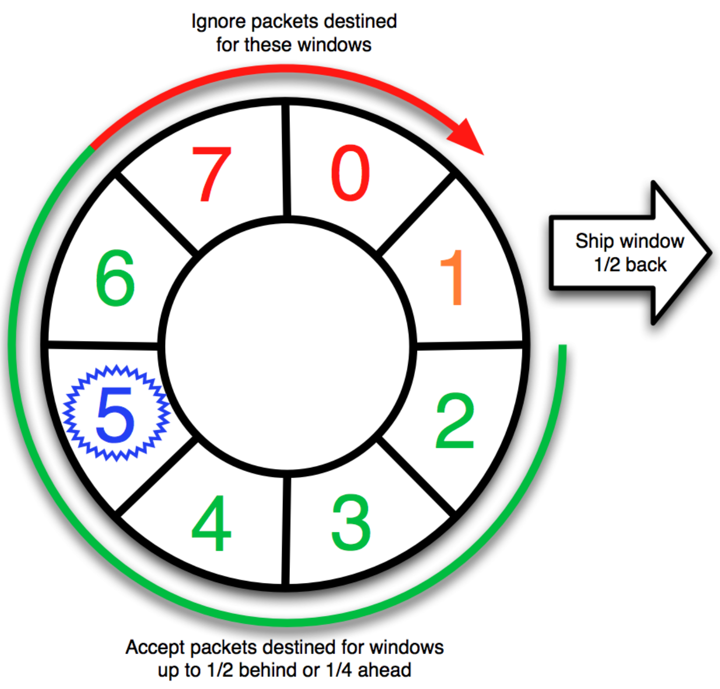
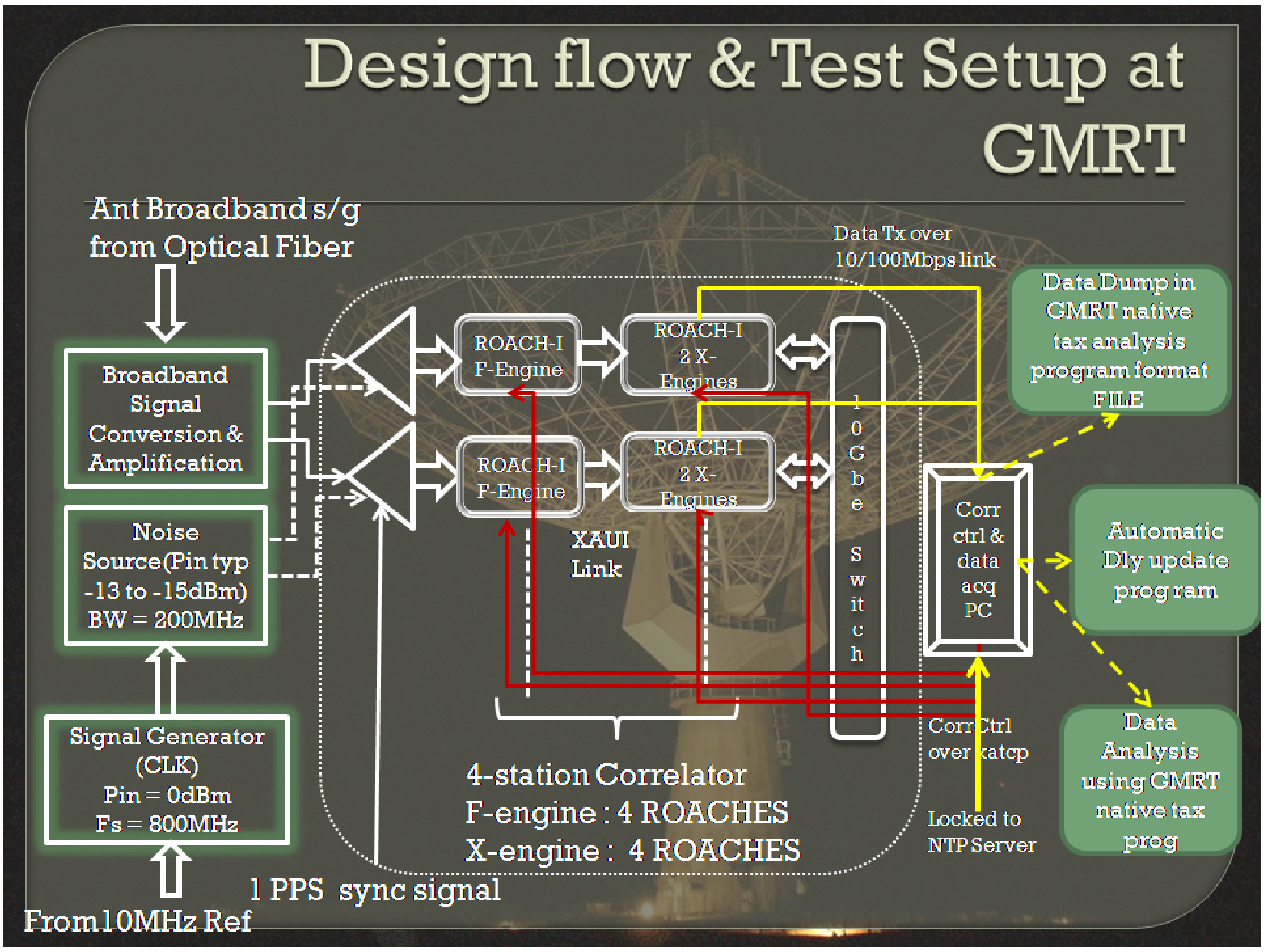

Ignore packets destined for these windows
7
0
1
2
3
4
5
6
Ship window 1/2 back
Accept packets destined for windows up to 1/2 behind or 1/4 ahead

Fig 5 Packetized reorder mechanism :- Eight windows are buffered. When the first packet destined for window five arrives, window one begins readout and the circular buffer advances, or rotates clockwise, by one window to accept packets destined for windows two, three, four, five and six(half the buffer space back and a quarter of the buffer space ahead, including the current window. Arriving packets destined for other windows are dropped.

Long term accumulation :- The accumulation that takes place within the X-engine cores (and hence M minimum output period for the correlator system) is given by t_{min} = PB where P is the packet size in words, M is the number of PFB channels and B is the system bandwidth in Hz. With a jumbo frame size of 4096 bytes and 512 channels over 1GHz bandwidth, it is possible to achieve accumulation times of 1.048576ms, which is typically too fast for imaging applications . Each X-engine core is assigned a separately addressable VACC to store its output. This is simply done on ROACHes (where multiple QDR SRAMs are available) .

Final accumulation read out :- The designs transmit their accumulated output products by reading shared memory from the PPC using BORPH, packetising them on the PPC using a non-documented protocol(SPEAD-Streaming Protocol for Exchange of Astronomical Data) and transmitting these through the boards' control and monitoring Ethernet ports.

Block Diagram of Packetized Correlator :-


Design flow & Test Setup at GMRT
Ant Broadband s/g from Optical Fiber
Broadband Signal Conversion & Amplification
Noise Source (Pin typ -13 to -15dBm) BW = 200MHz
Signal Generator (CLK) Pin = 0dBm Fs = 800MHz
From10MHz Ref
1 PPS sync signal
ROACH-I F-Engine
ROACH-I F-Engine
ROACH-I 2 X-Engines
ROACH-I 2 X-Engines
XAUI Link
10Gbe Switch
Data Tx over 10/100Mbps link
Corr ctrl & data acq PC
Corr Ctrl over katcp
Locked to NTP Server
4-station Correlator
F-engine : 4 ROACHES
X-engine : 4 ROACHES
Data Dump in GMRT native tax analysis program format FILE
Automatic Dly update prog ram
Data Analysis using GMRT native tax prog

The above block diagram is for 4-Antenna packetized correlator test setup with inputs to it and outputs from correlator.
ROACH PowerPC boots from control PC which has debian based operating system and corresponding file system over a private network of 192.168.100.xxx connected to 1Gbps switch.
The data from X-engines are also dumped through this private network for XAUI design. Control PC does automatic delay update, dumps final data from correlator and analyze the same.

➔ **Design Debugging :-**

1. Delay and phase jump issue
2. Timestamp mismatch issue
3. PPS signal offset issue
4. Slow phase wrap issue

➔ **System Characterization :-**

1. Long term stability test
2. ADC characterization

➔ **Design modification and upgrade :-**

1. 2 x F-engines per ROACH board design modification to 1 x F-engine per ROACH
2. Coarse delay depth increased from 2048 to 256k for 1k point FFT
3. 4-Antenna dual polarization packetized correlator expansion to 8-Antenna dual polarization
4. F-engine resource estimation to accommodate GMRT coarse delay requirement and maximum FFT size
5. Corr-0.6 package modification for online data recording in GMRT tax native format
6. Merging of packetized correlator F-engine and GPU packetizer part on F-engine ROACH board

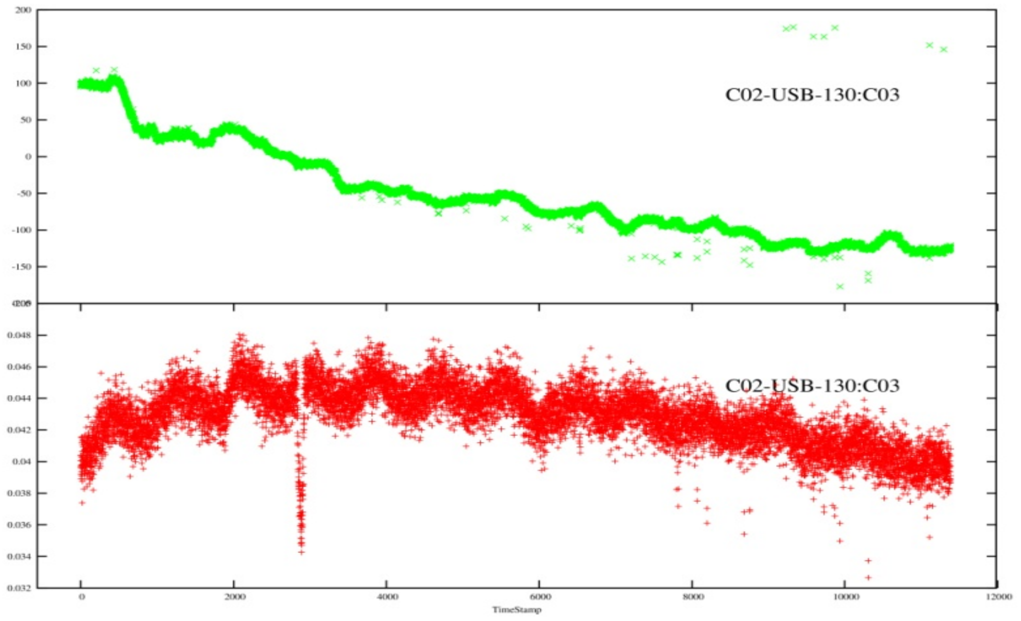
➔ **Antenna Testing :-**

1. Broad-band Tests Part – I
2. Broad-band Tests Part – II
3. 3C286 point source observation on 17th October 2012
4. 3C285 extended source observation on 5th December 2012
5. Packetized correlator with 4k point FFT test done on 3C48 dated 28th January 2013
6. Packetized, GSB and GPU correlator comparison test on 3C48 dated 14th February 2013
7. Combine packetized correlator and GPU packetizer design on F-engin ROACH board

➔ **Design Debugging :-**

1. **Delay and phase jump issue :-**

During initial stage of Packetized correlator testing, manually checked delay and phase and it was working fine. But when tested with antennas, the design was showing following artifacts as shown in *plot. 1*



Plot 1: Packetized Corr Delay & Phase Jump problem seen during antenna test (C06-S02 cross phase & amplitude)

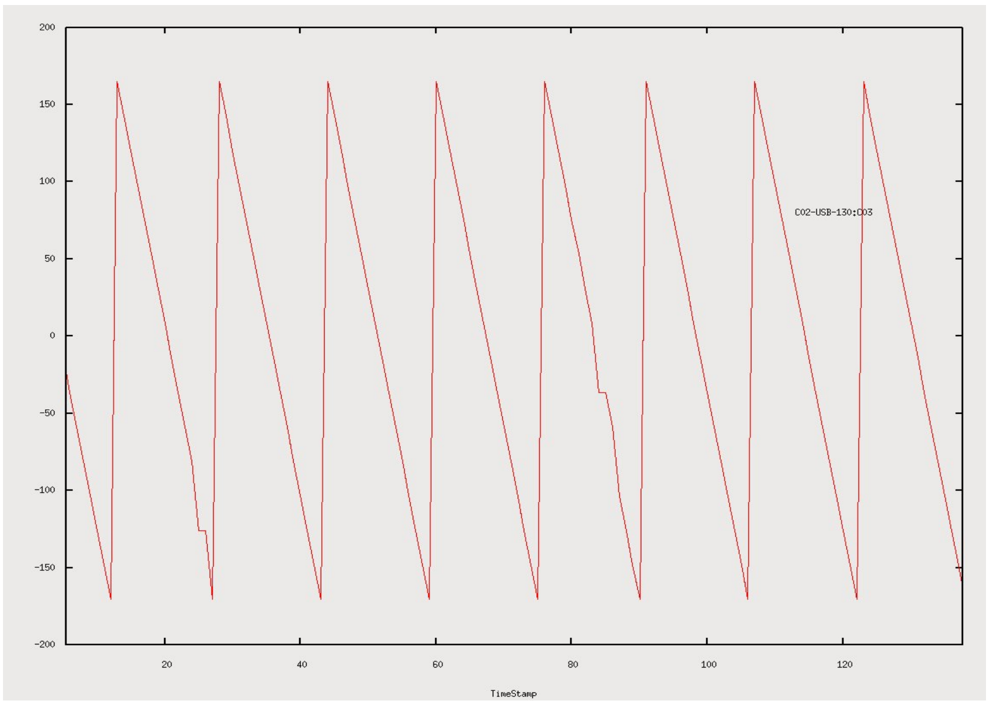
Here normalized amplitude seems to be fine but phase has random phase jumps as seen from C06 & C02 antenna cross phase plot.

The design then tested in the lab for phase jump problem using auto increment of integer delay, fractional delay, initial theta and then delay rate as shown in plots below :

1.1. Auto increment of integer delay only :-

In this experiment only integer delay was auto incremented by 1 clock/accumulation and the plotted for 64th channel. For 64th channel out of 512, the increment of phase is 180*64/512 = 22.5°. This shows sometimes retention of old integer delay value for next subsequent integration and then a jump of double the phase value indicating missed delay loading. The artifacts are not periodic but rather random. Hence it was suspected on delay_gen block which is missing the delay loading.

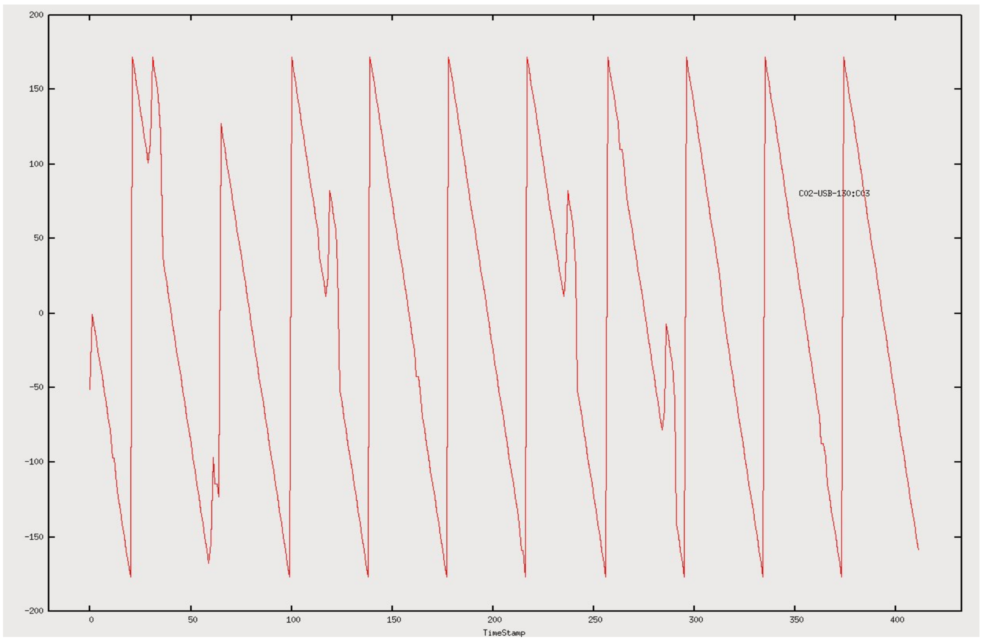
Still further testing was done with fractional delay auto increment, delay rate of 0.1 and extrapolated delay loading after 10 accumulation cycle.



Plot 2: Channel No. vs Time plot. Only integer delay auto increment by 1 clock plotted for channel 64

1.2. Delay rate of 0.1 clock + delay loaded after 10 accumulation cycle:-

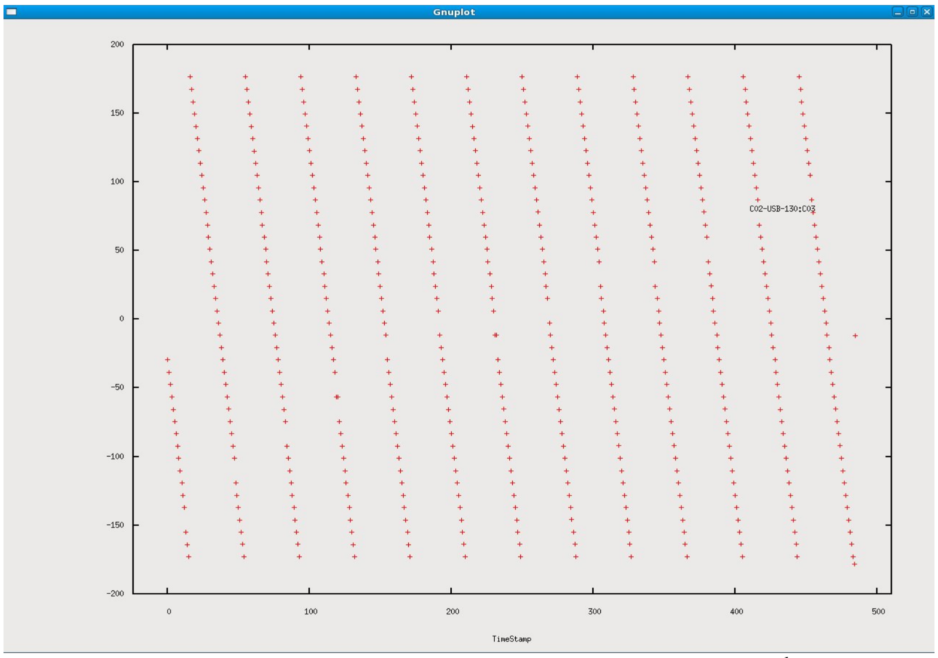
In this experiment delay rate of 0.1 clock/acc was applied and in advance calculation after 10 accumulations extrapolate integer delay value and load the same.



Plot 3: Channel no. Vs Time plot. Delay rate of 0.1 clock and delay loaded after 10 accumulation cycle plotted for channel 256

This experiment shows that there is something wrong in loading integer delay value.

1.3. Delay auto increment by 0.1 clock:-

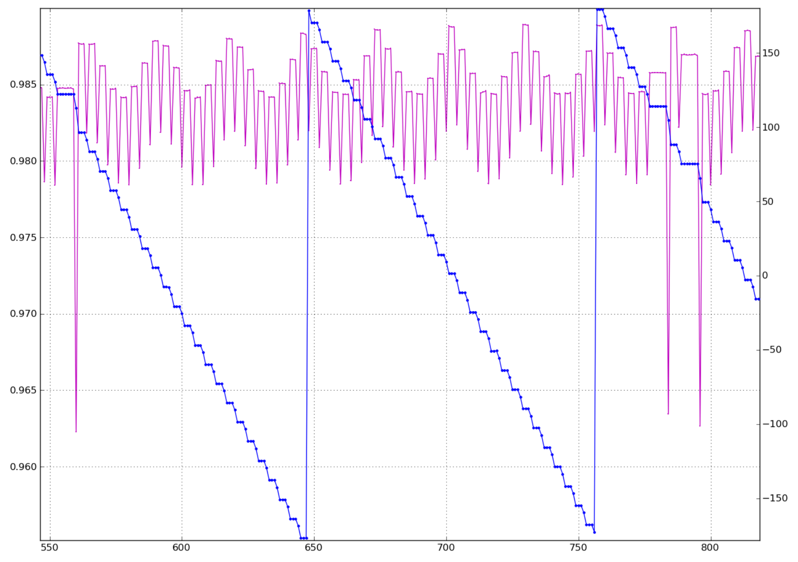


Plot 4: Delay auto increment by 0.1 clock and plotted for 64th channel

In this experiment delay was incremented by 0.1 clock continuously.
This again shows that delay loading has some problem which is inconsistent in behavior.

1.4. Initial theta update after 4 accumulations :-

The experiment was intended to check if initial theta loading behaves the same way as that of delay loading part which has random artifacts as show below :



Plot 5: Initial theta increment by 13 degrees after 4 accumulation cycles

Revision	Date	Modification/ Change
Ver. 1	10 September 2013	Initial Version

1.5. The bug and it's fix :-

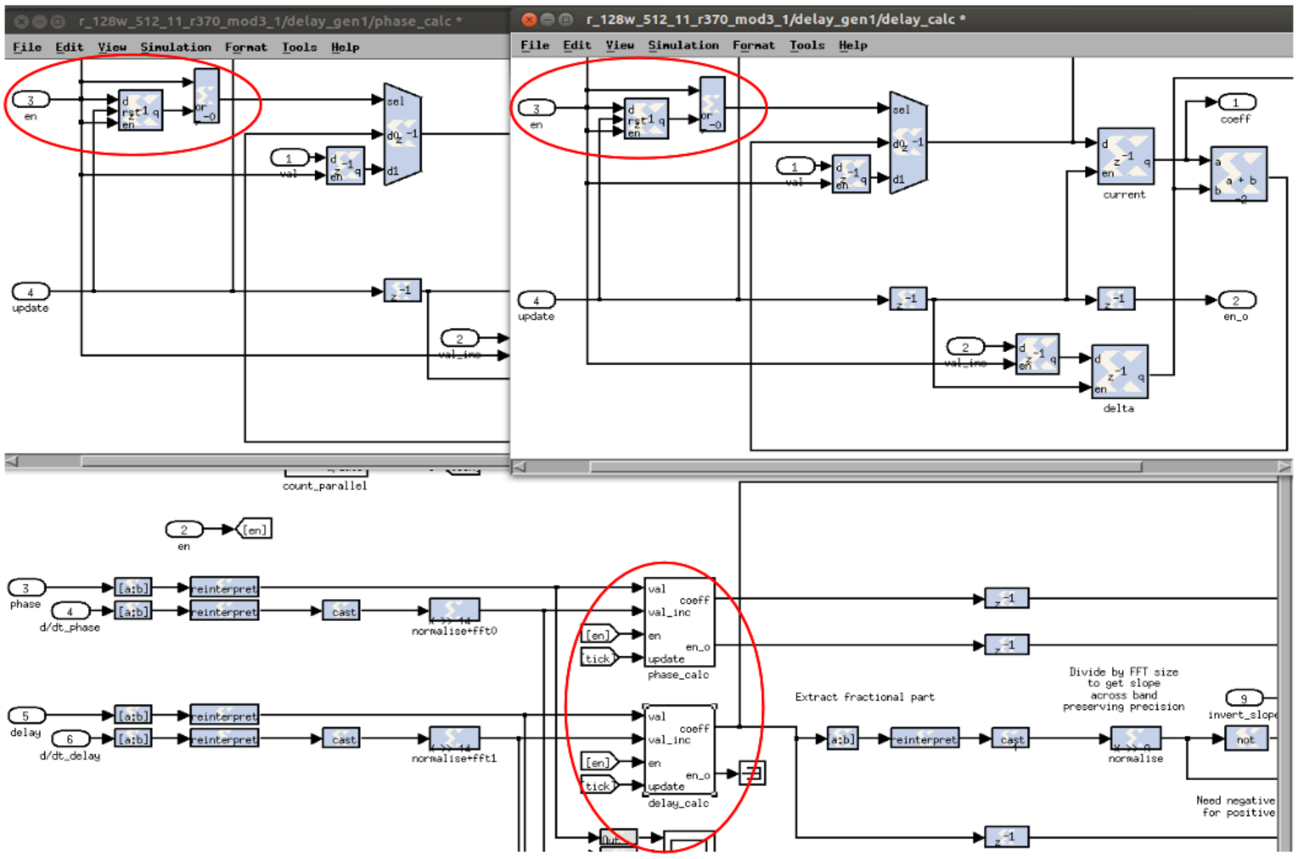
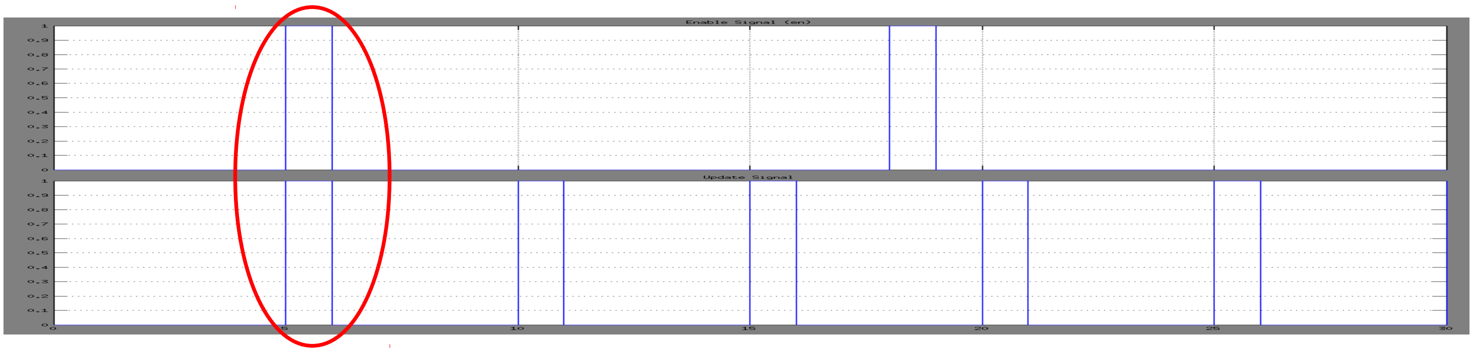

r_128w_512_11_r370_mod3_1/delay_gen1/phase_calc *
r_128w_512_11_r370_mod3_1/delay_gen1/delay_calc *
File Edit View Simulation Format Tools Help
en
update
val
val_inc
sel
current
coeff
delta
en_o
count_parallel
phase
d/dt_phase
delay
d/dt_delay
reinterpret
cast
normalise+fft0
normalise+fft1
phase_calc
delay_calc
Extract fractional part
Divide by FFT size to get slope across band preserving precision
invert_slope
normalise
not
Need negative for positive

Fig. 1: BUG: "delay_gen" block bug that was causing delay and initial theta loading problem

The bug was then found to be in “delay_gen” block located before “delay_wideband_prog” block and residing inside “fd_fs” block after “fft_wideband_real” . Following figure shows the bug location encircled. The *“update”* signal is generated at the end of every fft cycle and *“en”* is an enable signal which is generated when the loading time has reached. As shown in waveform below when *“update”* and *“en”* signals are not overlapped mux select line value is correct otherwise it was not updating mux's 2nd input. Also 2nd input of mux i.e. register, it's enable signal must be tied to select line of mux with positive edge detector. Below *plot 6* is the



Plot 6: Cause of delay and phase jump where enable signal and update signal aligns randomly

waveform showing enable signal (with some jitter) and update signal aligned at certain time causing delay and phase jump problem.

The whole mechanism was preventing only on special occasion as mentioned above from updating fresh initial theta (θ_0) and delay value when load time reaches. Below *fig.2* shows modified *"delay_gen"* block.

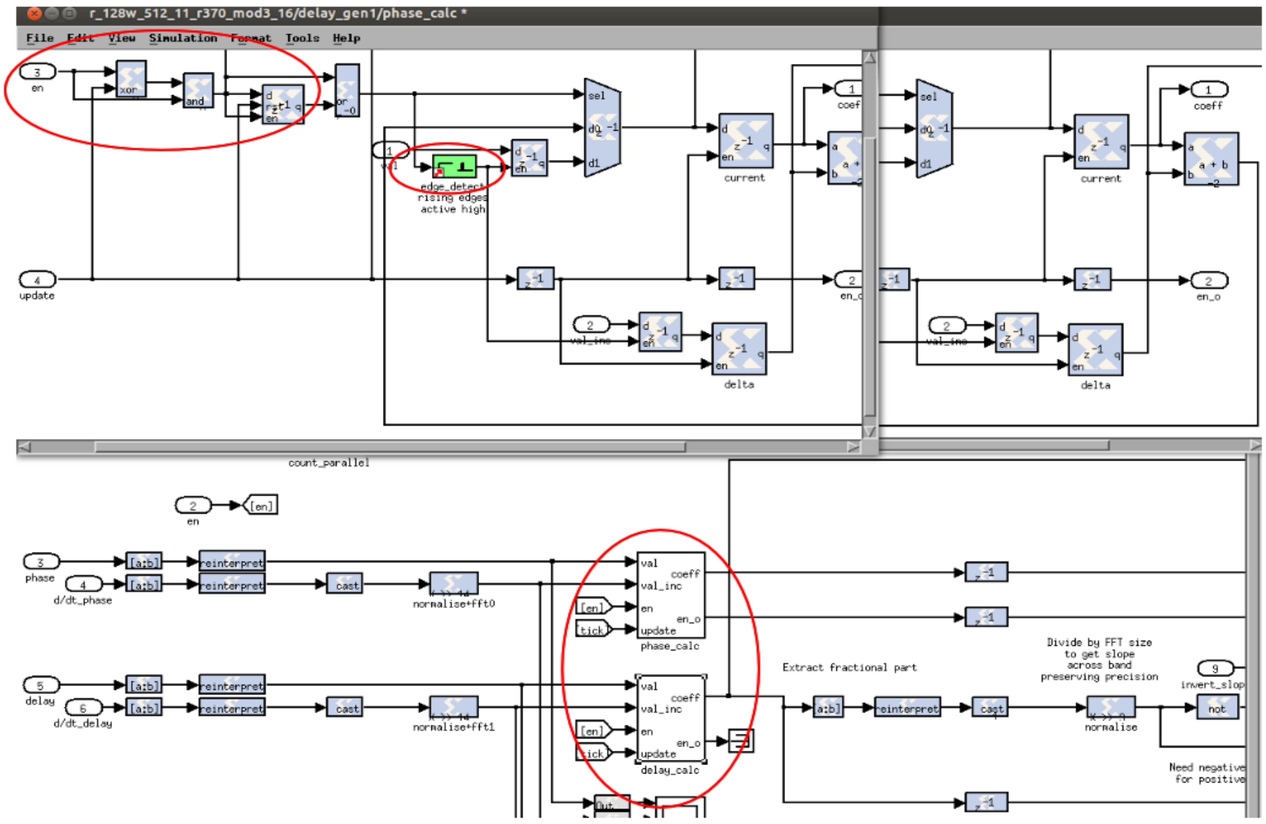


Fig. 2: BUG-FIX :"delay_gen" block bug fixed that was causing delay and initial theta loading problem

The design then tested in the lab and with antenna signals. Corresponding phase and normalized amplitude and phase plot taken on 2nd August 2013 with C09-S02 antenna on 3C468 for more than 3 hours and no random phase jumps were observed.
This way the phase jump problem was fixed.

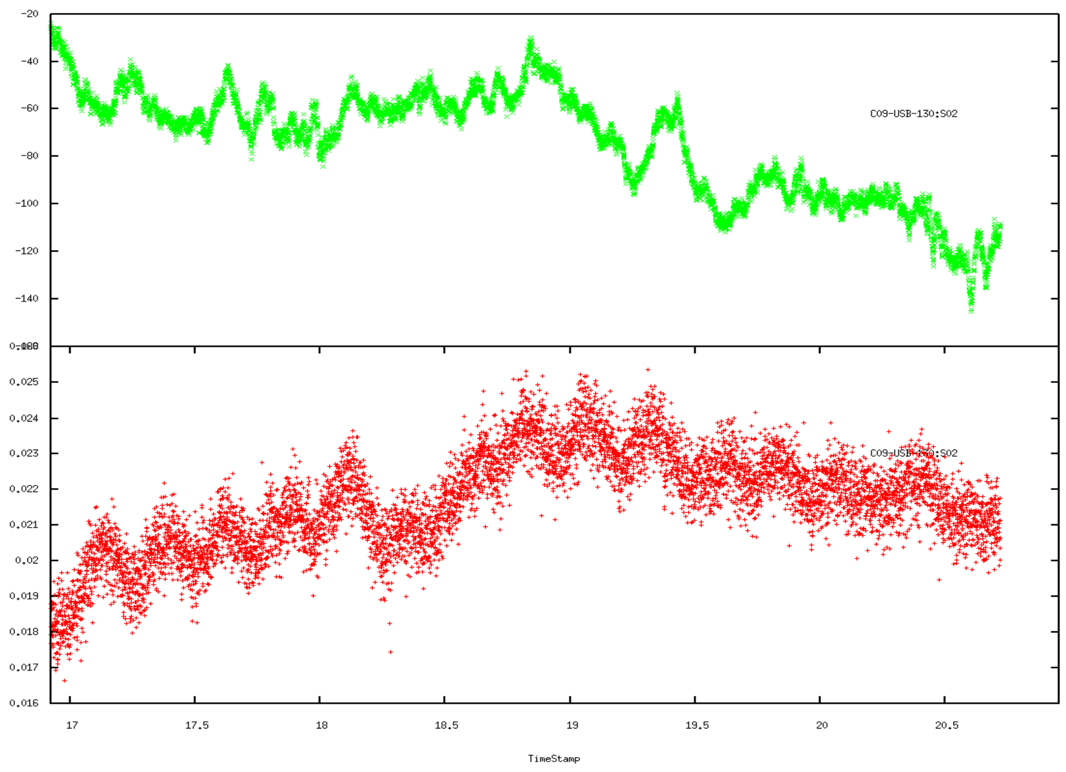


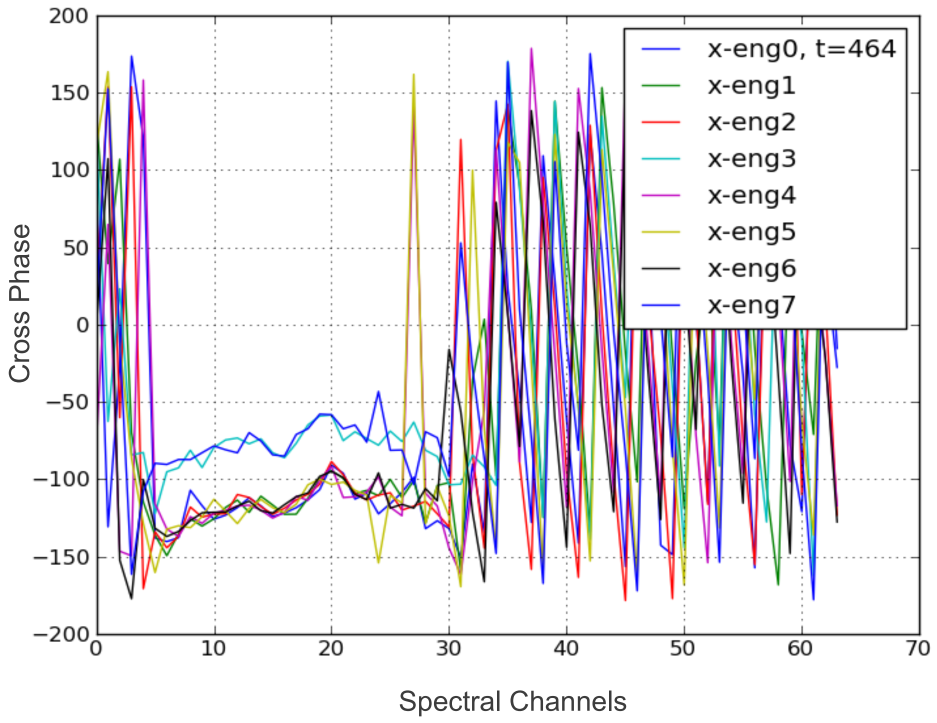
Fig. 3: After fixing Packetized correlator delay and phase Jump problem (C09-S02 cross phase and amplitude on 2nd Aug 2013 on 3C468)

2. Timestamp mismatch issue:-

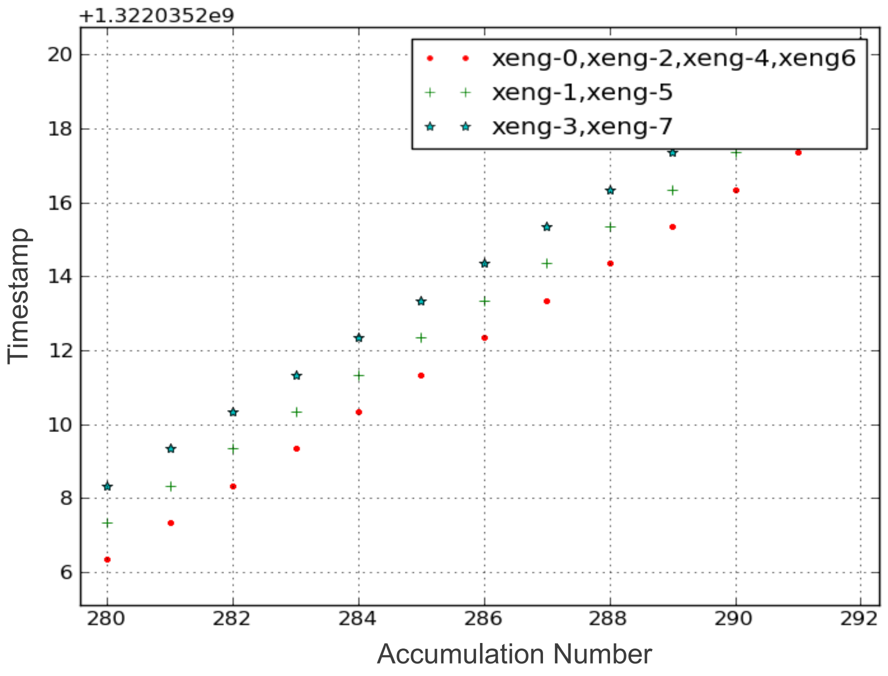
While debugging delay and phase jump issues, we found another problem while acquiring accumulated data from all X-engines. The accumulation timestamp difference in certain group of X-engine data was of previous accumulation time i.e. t_{acc} -1 and others with t_{acc} . It was observed while doing automatic delay update and acquiring individually processed (512 channels / 8 X-engines) 64 channels from each X-engine separately and plotting for the same timestamp. Below plots explain the same.

The *plot 6* is for accumulation counter i.e. 464 plotting phase of a given cross correlation for individual X-engine over channels. Here it shows X-engine 0 & 3 has different phase compared to rest of the X-engine group.

The plot 7 is a timestamp (in actual time from epoch) plotted against accumulation counter that shows certain group of X-engines are having different timestamp for given accumulation counter.

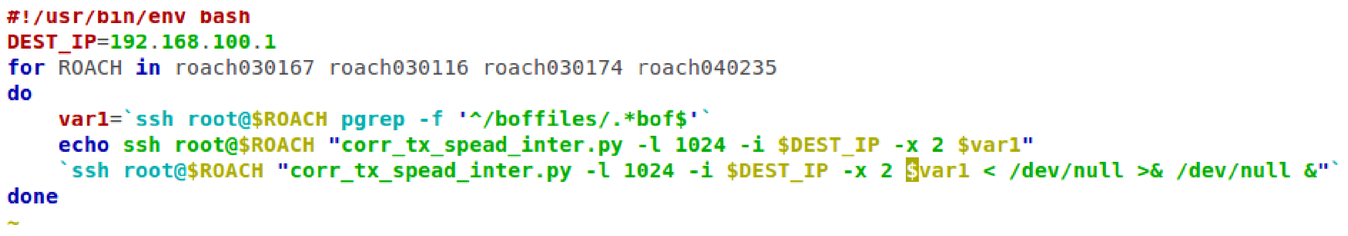


Plot 7: Phase plot over spectrum showing timestamp mismatch problem for a given accumulation number = 464



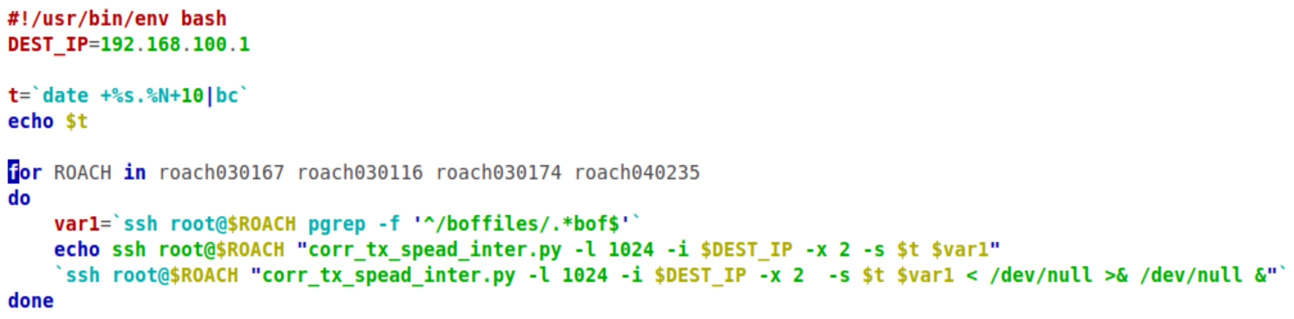
Plot 8: Timestamp from each X-engine plotted against accumulation counter

Timestamp mismatch problem solution :-
It was suspected that may be caused due to the python script *"corr_tx_spead_inter.py"* which runs on X-engine ROACH board PowerPC that acquires accumulated data may not be starting simultaneously due to latency in logging into each X-engine ROACH board and starting the script. Even though it's automated, the process to start the script on each X-engine ROACH occurs serially.
The original script is as follows :

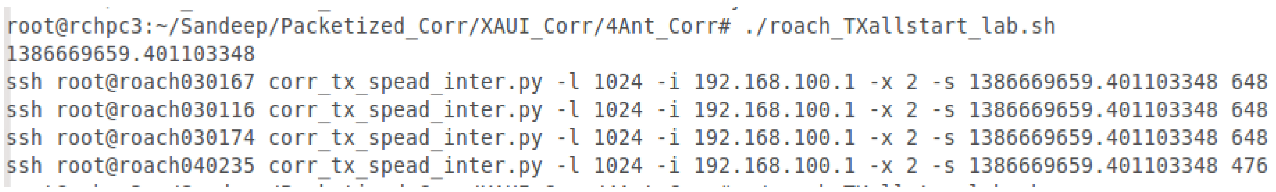


```
#!/usr/bin/env bash
DEST_IP=192.168.100.1
for ROACH in roach030167 roach030116 roach030174 roach040235
do
    var1=`ssh root@$ROACH pgrep -f '^/boffiles/.*bof$'`
    echo ssh root@$ROACH "corr_tx_spead_inter.py -l 1024 -i $DEST_IP -x 2 $var1"
    `ssh root@$ROACH "corr_tx_spead_inter.py -l 1024 -i $DEST_IP -x 2 $var1 < /dev/null >& /dev/null &"`
done
~
```

The modified shell script to start accumulated data reading and transfer from X-engine ROACH boards and it's output is as shown below :

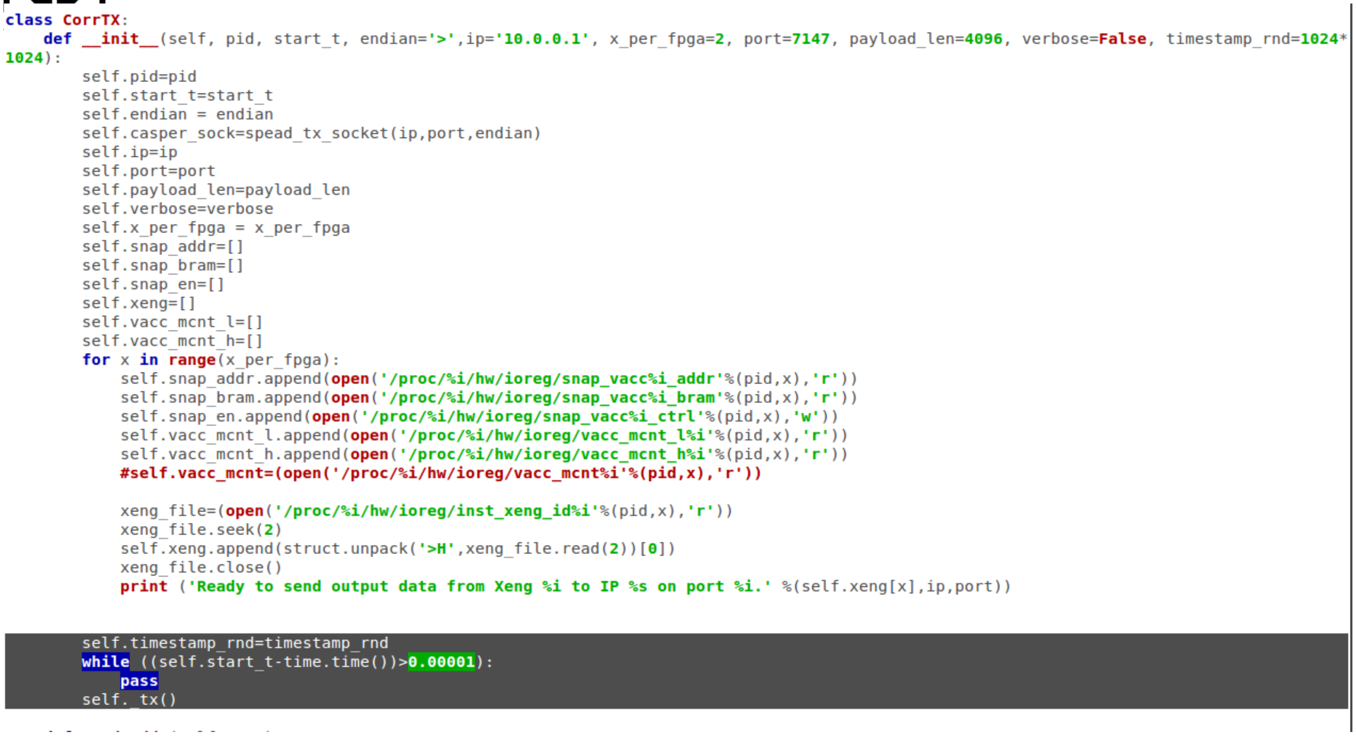


```
#!/usr/bin/env bash
DEST_IP=192.168.100.1

t=`date +%s.%N+10|bc`
echo $t

for ROACH in roach030167 roach030116 roach030174 roach040235
do
    var1=`ssh root@$ROACH pgrep -f '^/boffiles/.*bof$'`
    echo ssh root@$ROACH "corr_tx_spead_inter.py -l 1024 -i $DEST_IP -x 2 -s $t $var1"
    `ssh root@$ROACH "corr_tx_spead_inter.py -l 1024 -i $DEST_IP -x 2  -s $t $var1 < /dev/null >& /dev/null &"`
done
```



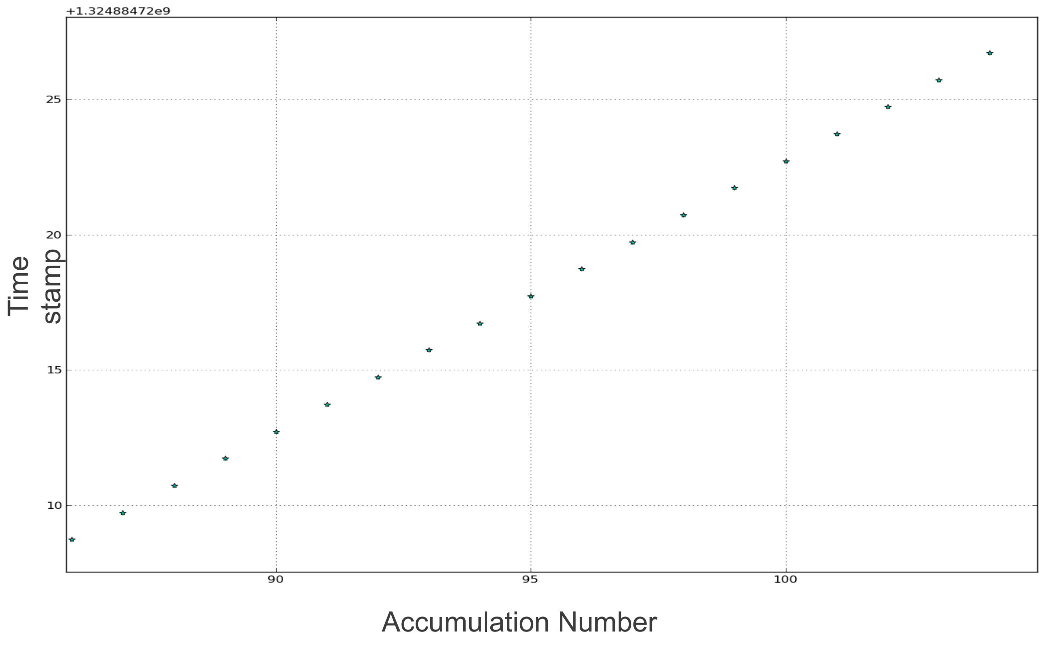
```
root@rchpc3:~/Sandeep/Packetized_Corr/XAUI_Corr/4Ant_Corr# ./roach_TXallstart_lab.sh
1386669659.401103348
ssh root@roach030167 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s 1386669659.401103348 648
ssh root@roach030116 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s 1386669659.401103348 648
ssh root@roach030174 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s 1386669659.401103348 648
ssh root@roach040235 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s 1386669659.401103348 476
```

In this shell script the modification is by adding a parameter '*t*' which provides advance start time. In this case the advance start time is (t+10) which is input to *'corr_tx_spead_inter.py'* script and ensures that on each X-engine ROACH board this script will start at exact time. The PowerPC on ROACH board running debian flavour of linux is locked to reference time standard machine i.e. *'samay1'* machine in the GMRT. The corresponding modification in script is as shown below with highlighted area :



```
class CorrTX:
    def __init__(self, pid, start_t, endian='>',ip='10.0.0.1', x_per_fpga=2, port=7147, payload_len=4096, verbose=False, timestamp_rnd=1024*
1024):
        self.pid=pid
        self.start_t=start_t
        self.endian = endian
        self.casper_sock=spead_tx_socket(ip,port,endian)
        self.ip=ip
        self.port=port
        self.payload_len=payload_len
        self.verbose=verbose
        self.x_per_fpga = x_per_fpga
        self.snap_addr=[]
        self.snap_bram=[]
        self.snap_en=[]
        self.xeng=[]
        self.vacc_mcnt_l=[]
        self.vacc_mcnt_h=[]
        for x in range(x_per_fpga):
            self.snap_addr.append(open('/proc/%i/hw/ioreg/snap_vacc%i_addr'%(pid,x),'r'))
            self.snap_bram.append(open('/proc/%i/hw/ioreg/snap_vacc%i_bram'%(pid,x),'r'))
            self.snap_en.append(open('/proc/%i/hw/ioreg/snap_vacc%i_ctrl'%(pid,x),'w'))
            self.vacc_mcnt_l.append(open('/proc/%i/hw/ioreg/vacc_mcnt_l%i'%(pid,x),'r'))
            self.vacc_mcnt_h.append(open('/proc/%i/hw/ioreg/vacc_mcnt_h%i'%(pid,x),'r'))
            #self.vacc_mcnt=(open('/proc/%i/hw/ioreg/vacc_mcnt%i'%(pid,x),'r'))

            xeng_file=(open('/proc/%i/hw/ioreg/inst_xeng_id%i'%(pid,x),'r'))
            xeng_file.seek(2)
            self.xeng.append(struct.unpack('>H',xeng_file.read(2))[0])
            xeng_file.close()
            print ('Ready to send output data from Xeng %i to IP %s on port %i.' %(self.xeng[x],ip,port))


        self.timestamp_rnd=timestamp_rnd
        while ((self.start_t-time.time())>0.00001):
            pass
        self._tx()
```

After these modification again a set of reading taken and timestamp (in actual time from epoch) plotted against accumulation counter that shows now all X-engines are having identical timestamp for given accumulation counter as shown below :



Plot 9: After Timestamp mismatch bug fixed and plotted accumulation number Vs timestamp

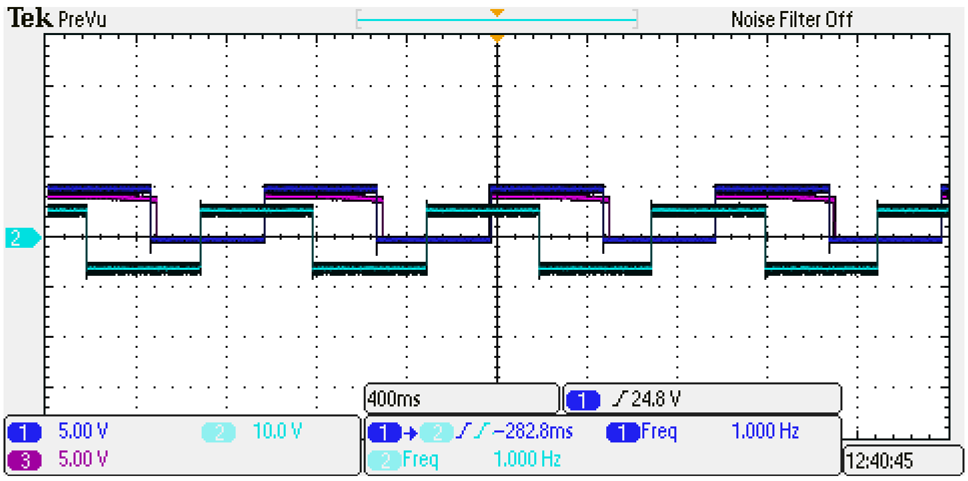
3. PPS signal offset issue :-

This problem was notice when packetized correlator was not initializing and was exiting with an error of *"No PPS or clock".* Packetized correlator F-engine starts after a PPS trigger signal from *'gps1'* and earlier it was *'TM-4'* receiver. The control PC of packetized correlator was locked to *'samay1'* machine.

The first suspect was unlocked status of control PC to *'samay1'* machine. But this was then verified by

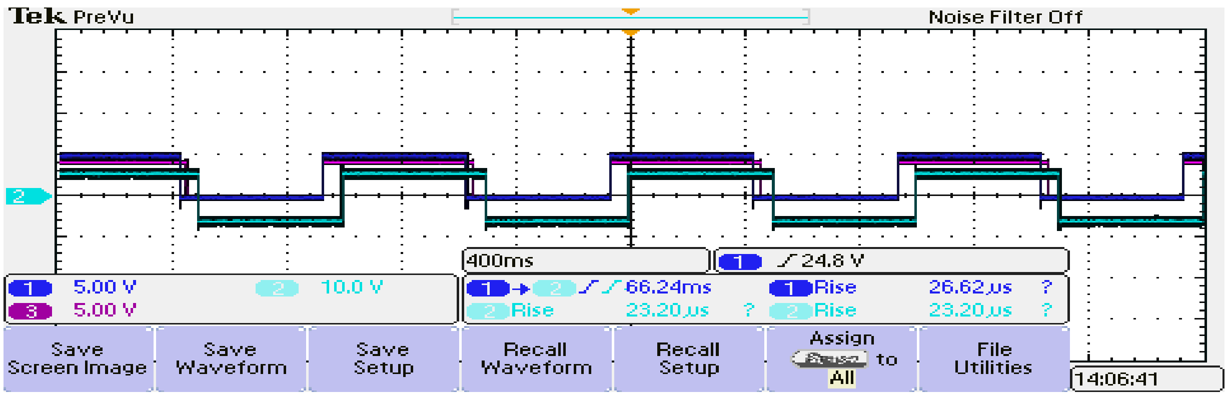
```
root@rchpc3:~/Packetized_Corr/XAUI_Corr/Basic_Test/Phase_Jumps/delay_cal
# ntpq -p
     remote           refid      st t when poll reach   delay   offset   jitter
==============================================================================
+samay1.gmrt.ncra. 77.67.82.0  3 u   84  128  377    2.258   12.120   9.064
*gps2.gmrt.ncra.   77.67.82.0  3 u   87  128  377    2.234   10.234   2.170
```

The issue then still persisted and it's decided to check the hardware PPS pulse to packetized correlator F-engine and control PC time. A PPS signal on every second was generated on control PC serial port and compared with hardware PPS pulse from *'TM-4'* receiver. Below waveform shows clear offset between both these signals. In the waveform blue waveform is from *'TM-4'* receiver hardware pulse and green waveform is from NTP locked control PC PPS pulse. The difference is -282.8mS from waveform, which is actually (1-0.2828) ~700mS.



Plot 10: Time offset between hardware pulse from TM-4 and NTP locked control PC PPS pulse

The same was then corrected by providing hardware PPS signal from *'TM-4'* receiver on serial port to *samay1* machine to make NEMA code more accurate

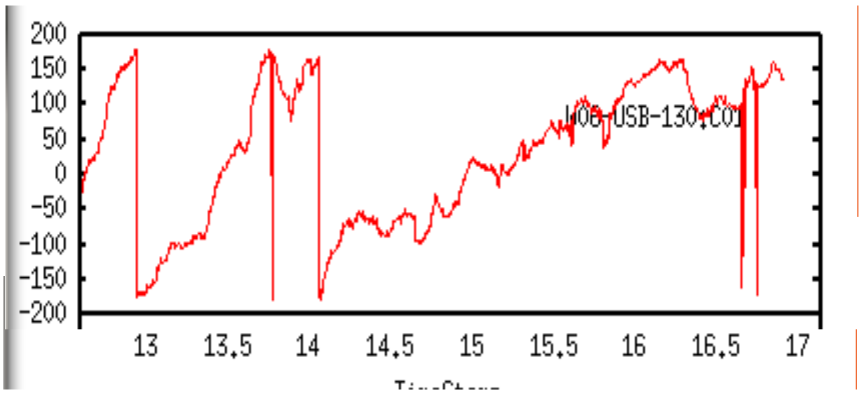


Plot 11: Time offset issue fixed and NTP locked control PC software time is correct within boundary

and hence PC timing. At present the *gps1* – a standalone PPS receiver is being used to lock *samay1* machine. Below waveform is the corrected one which shows hardware PPS and software PPS time difference ~ 60mS.

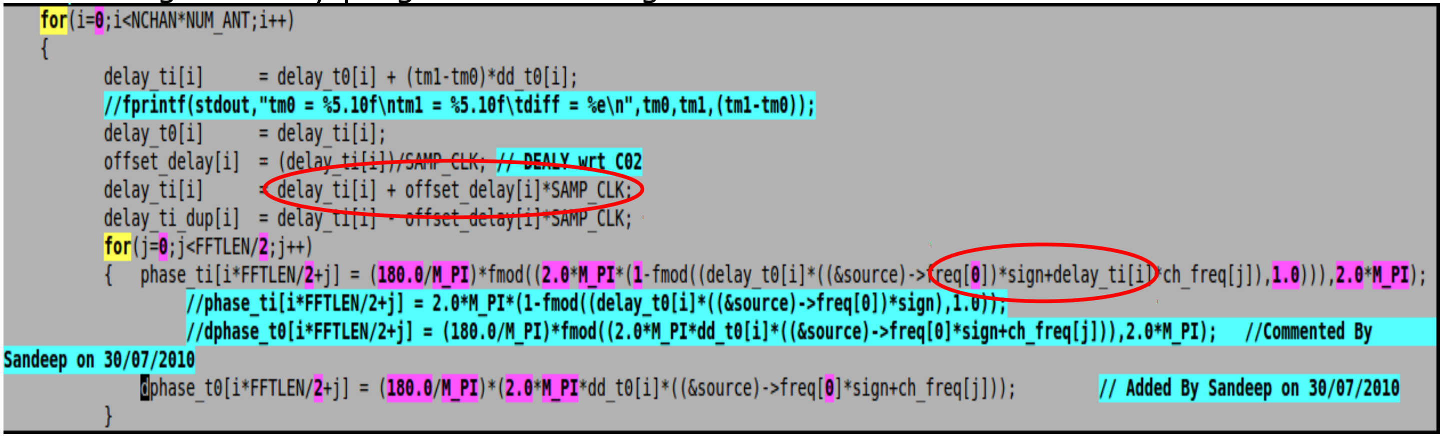
4. Slow phase wrap issue :-

After fixing all above bugs the design when tested with antenna signals for a considerable time duration shows slow phase wrap. As shown in *plot 12*



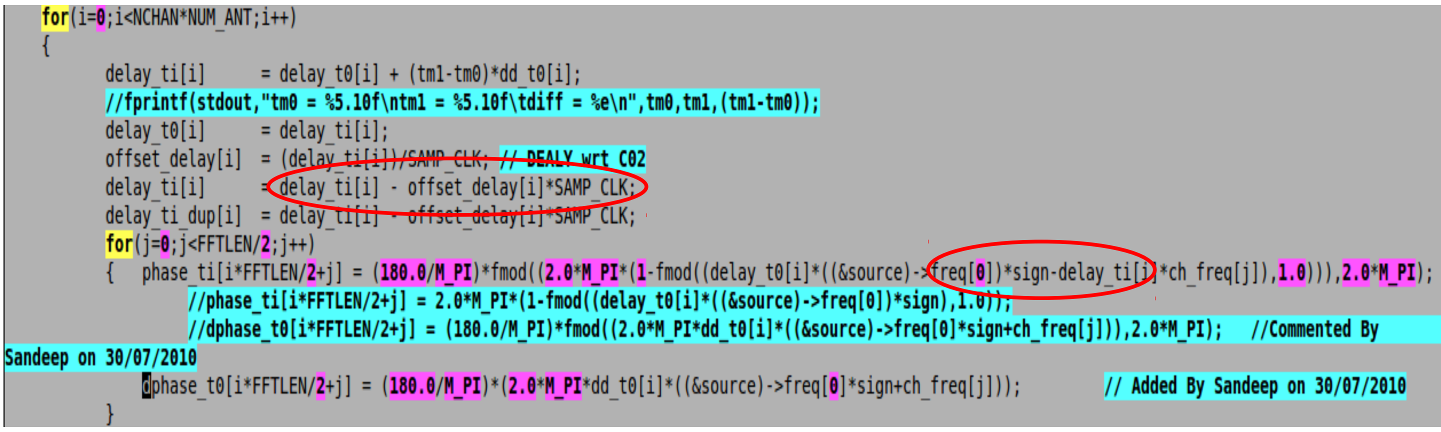
Plot 12: Phase wrap issues due to small bug in delay_cal program.

The above plot is a cross-phase plot of W06-C01 antenna where delay program bug introduced a slow phase wrap.
After some careful examination and trial basis found that the way fractional delay and fringe stop was implemented was different as compared to GSB. It implemented negative slope for positive fractional delay.
The original delay program with a bug was as below :



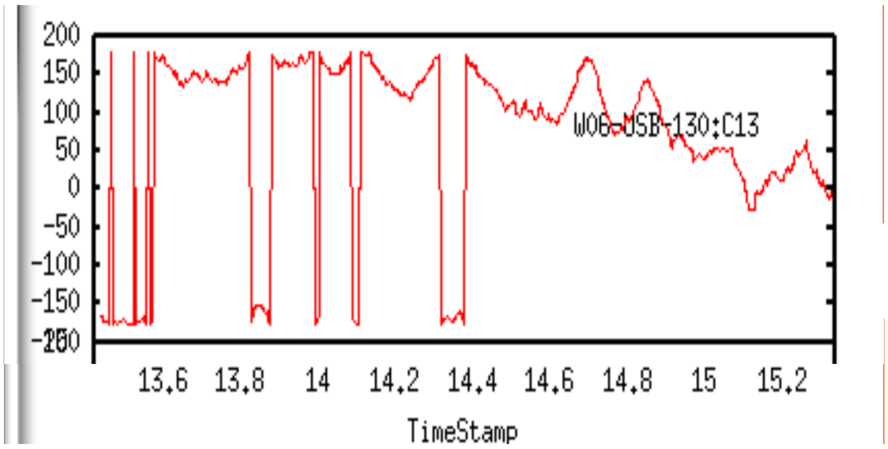
```
for(i=0;i<NCHAN*NUM_ANT;i++)
{
        delay_ti[i]      = delay_t0[i] + (tm1-tm0)*dd_t0[i];
        //fprintf(stdout,"tm0 = %5.10f\ntm1 = %5.10f\tdiff = %e\n",tm0,tm1,(tm1-tm0));
        delay_t0[i]      = delay_ti[i];
        offset_delay[i]  = (delay_ti[i])/SAMP_CLK; // DEALY wrt C02
        delay_ti[i]      = delay_ti[i] + offset_delay[i]*SAMP_CLK;
        delay_ti_dup[i]  = delay_ti[i] - offset_delay[i]*SAMP_CLK;
        for(j=0;j<FFTLEN/2;j++)
        {   phase_ti[i*FFTLEN/2+j] = (180.0/M_PI)*fmod((2.0*M_PI*(1-fmod((delay_t0[i]*((&source)->freq[0])*sign+delay_ti[i]*ch_freq[j]),1.0))),2.0*M_PI);
                //phase_ti[i*FFTLEN/2+j] = 2.0*M_PI*(1-fmod((delay_t0[i]*((&source)->freq[0])*sign),1.0));
                //dphase_t0[i*FFTLEN/2+j] = (180.0/M_PI)*fmod((2.0*M_PI*dd_t0[i]*((&source)->freq[0]*sign+ch_freq[j])),2.0*M_PI);   //Commented By
Sandeep on 30/07/2010
            dphase_t0[i*FFTLEN/2+j] = (180.0/M_PI)*(2.0*M_PI*dd_t0[i]*((&source)->freq[0]*sign+ch_freq[j]));       // Added By Sandeep on 30/07/2010
        }
```

The program then modified appropriately as shown below :



```
    for(i=0;i<NCHAN*NUM_ANT;i++)
    {
            delay_ti[i]      = delay_t0[i] + (tm1-tm0)*dd_t0[i];
            //fprintf(stdout,"tm0 = %5.10f\ntm1 = %5.10f\tdiff = %e\n",tm0,tm1,(tm1-tm0));
            delay_t0[i]      = delay_ti[i];
            offset_delay[i]  = (delay_ti[i])/SAMP_CLK; // DEALY wrt C02
            delay_ti[i]      = delay_ti[i] - offset_delay[i]*SAMP_CLK;
            delay_ti_dup[i]  = delay_ti[i] - offset_delay[i]*SAMP_CLK;
            for(j=0;j<FFTLEN/2;j++)
            {   phase_ti[i*FFTLEN/2+j] = (180.0/M_PI)*fmod((2.0*M_PI*(1-fmod((delay_t0[i]*((&source)->freq[0])*sign-delay_ti[i]*ch_freq[j]),1.0))),2.0*M_PI);
                      //phase_ti[i*FFTLEN/2+j] = 2.0*M_PI*(1-fmod((delay_t0[i]*((&source)->freq[0])*sign),1.0));
                      //dphase_t0[i*FFTLEN/2+j] = (180.0/M_PI)*fmod((2.0*M_PI*dd_t0[i]*((&source)->freq[0]*sign+ch_freq[j])),2.0*M_PI);    //Commented By
Sandeep on 30/07/2010
                  dphase_t0[i*FFTLEN/2+j] = (180.0/M_PI)*(2.0*M_PI*dd_t0[i]*((&source)->freq[0]*sign+ch_freq[j]));        // Added By Sandeep on 30/07/2010
            }
```

After the same modification phase wrap has marginally reduced as there may be another issues. As shown in below *plot 13* which was taken near about same time next day.

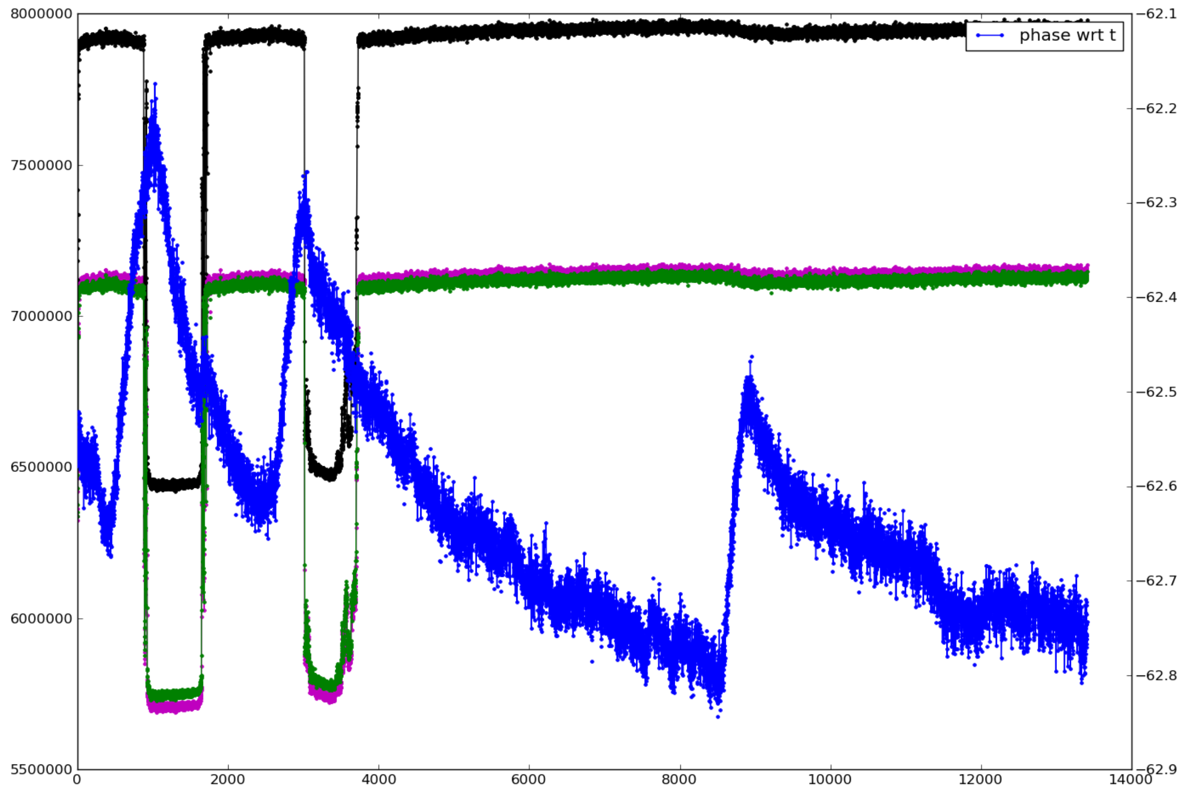


Plot 13: Phase wrap issue solved after fixing bug in delay cal program

➔ System Characterization :-

1. **Long term stability test :-**
   The test was carried out with noise source connected to '1x' and 2x' input of packetized correlator for a time duration of ~ 4 hours. The test shows for certain time duration a drop in cross correlation as there is a drop in selfs as well at the same time. But the phase is fairly stable. The test was carried out in correlator room where temperature is maintained fairly constant.

   Below plot is an overlay plot of selfs, cross amplitude on left hand Y-axis and cross phase on phase right hand Y-axis. '1x' self was indicated by black colour, '2x' self by green colour, cross amplitude '1x-2x' in pink colour and cross phase '1x-2x' in blue colour.



Plot 14: Packetized corr long term stability test noise source test of ~4 hours.

2. **ADC characterization :-**
   The iADC sampler card characterization is important in order to decide the stable operating range of FPGA back-end. This will also be useful in deciding the headroom for accommodating RFI. For more detail please *Refer "iADC*

Characterization Report" by Sandeep C. Chaudhari, Kaushal D. Buch,S. Harshvardhan Reddy in **Appendix-I**
The characterization was done with noise source that resembles in behavior to that of antenna signals. The spread of the distribution (in terms of standard deviation) decides the number of bits exercised in an adc.
With this experiment noise as input the stable operating range of digital back-end was calculated by determining iADC operating range. After providing 1-bit headroom for RFI, the optimal noise input power over the analog bandwidth of iADC comes out in the range of -16.7dBm to -22.5dBm

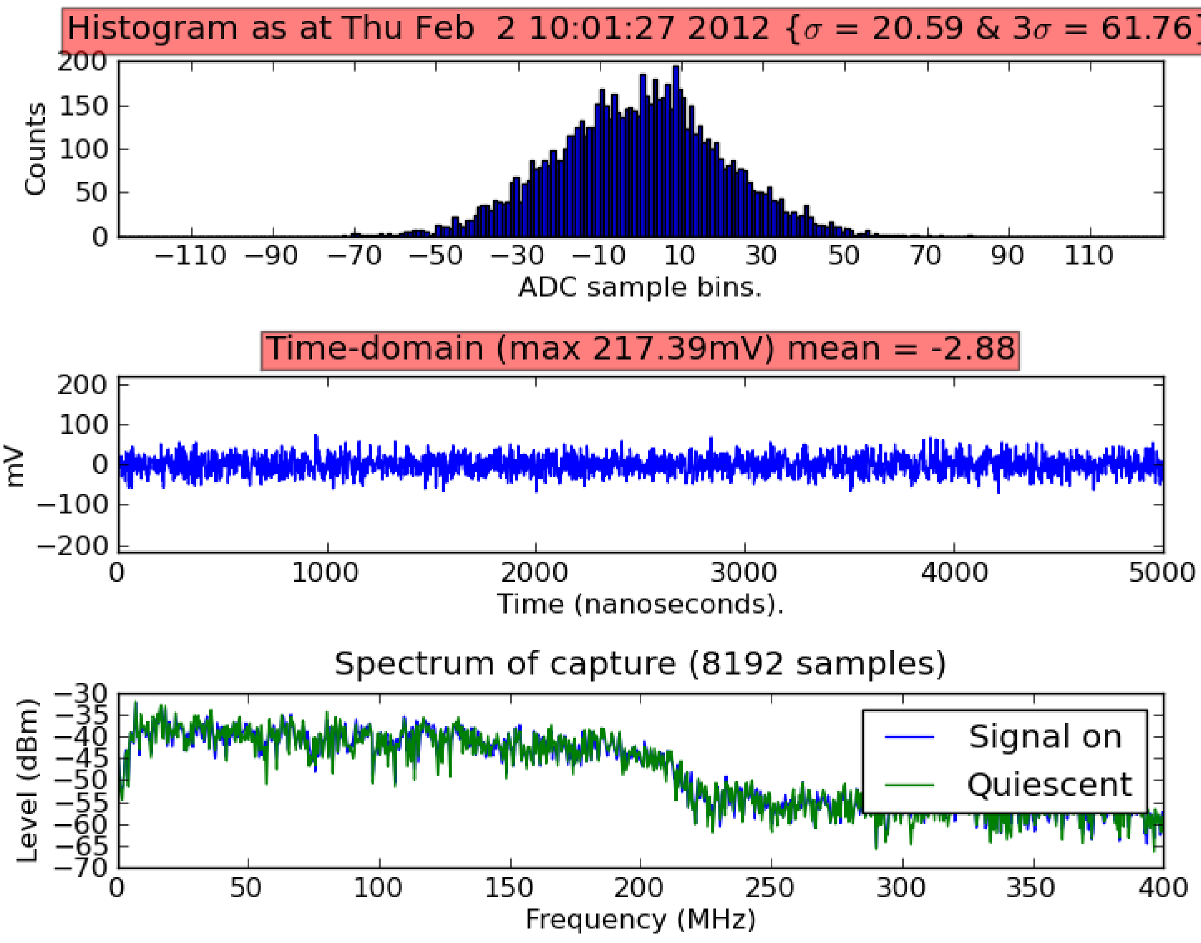


Fig 6: iADC characterization histogram showing 3sigma for 5-6 bit utilization

The above *fig.7* shows typical histogram of a noise source fed to iADC with P_{in} = -16dBm
Below table shows the bit utilization, theoretical 3σ, measured 3σ and input power. Yellow highlighted is the optimal operating range of iADC.

Bit Utilization	3σ (Theoretical)	3σ (Meas)	Pin@1.5GHz in dBm
6 to 7 bits	64 to 127	115 to 112	-11.09
5 to 6 bits	32 to 63	61 to 64	-16.62
4 to 5 bits	16 to 31	30 to 32	-22.54
3 to 4 bits	8 to 15	14 to 15	-28.54
2 to 3 bits	4 to 7	7.21 to 7.5	-34.91
1 to 2 bits	2 to 3	3.29 to 3.45	-42.06
0 to 1 bits	0 to 1	1.65	-50.30

➔ **Design modification and upgrade :-**

1. **2 x F-engines per ROACH board design modification to 1 x F-engine per ROACH :-**
   The generic packetized correlator design developed in collaboration with MeerKAT, South Africa was a basic design which needed further modifications so as to meet the GMRT digital back-end requirements, which are as follows :-
   1. Number of stations : 32
   2. Maximum instantaneous bandwidth : 400MHz
   3. Number of input polarization : 2
   4. Full stokes capability
   5. Dump time : 1 sec or better
   6. Coarse and fine delay tracking : +/- 128µS
   7. Fringe rotation : up to 5Hz

   The generic packetized correlator specifications were as follows :
   1. 4-Antenna dual polarization
   2. 2 x F-engines per ROACH board
   3. Processing bandwidth = 400MHz
   4. FFT channels = 1k point real
   5. Integer delay compensation : 2048 clock cycles
   6. Fringe compensation

   These requirements were met in step by step manner. First to accommodate +/-128µS delay, at least the delay depth needed was 256k clock cycle at a given sampling clock of 800MHz i.e. (256k x 8-bit) ~ 2Mb/channel and existing ROACH board resources has 8784Kb BRAMs. Hence the generic packetized correlator which has 2 F-engines per ROACH i.e. 4 inputs need at least 8Mb BRAM. This over maps the design on Virtex-5 sx95 device. Hence the design was modified to 1 x F-engine per ROACH board.

2. **Coarse delay depth increased from 2048 to 256k for 1k point FFT:-**
   To meet the delay of longest baseline for the GMRT comes out to be +/-128µS, which corresponds to a delay BRAM depth of 256k clocks at a given sampling clock of 800MHz. Accordingly the design was compiled and tested for the same. Corresponding delay update script was modified so as to compensate +/- delay. The above design was then tested with antenna signals. The detailed antenna tests are described in **"Antenna Testing"** section below.

3. **4-Antenna dual polarization packetized correlator expansion to 8-Antenna dual polarization:-**
   Further task was to expand 4-Antenna dual polarization packetized correlator to 8-Antenna design. Corresponding F-engine and X-engine design was modified along with corresponding configuration, testing and analysis softwares. A running test setup is also available for users/students to enable independent work on Packetized correlator without disturbing ongoing upgrade work.

4. F-engine resource estimation to accommodate GMRT coarse delay requirement and maximum FFT size:-

The aim of F-engine resource estimation is how much maximum FFT size that can be fitted in the design while fulfilling the GMRT coarse delay requirement. The modification involves F-engine and X-engine hardware design modification along with delay update program modification.

There are two approaches :-

1. **Scheme-I :** Each F-engine processes 1 antenna with dual polarization.

   In scheme-I maximum delay in both channels/polarizations is +/-128k clocks. This scheme utilizes maximum BRAMs in coarse delay compensation and restricts size of memory intensive blocks like FFT and PFB. But the advantage is that full polar mode can be possible with existing setup.

   The design was compiled for various FFT length and coarse delay to check the resource utilization. Following is the table and chart which gives clear idea of BRAM usage for given combination of spectral channels and coarse delay depth.

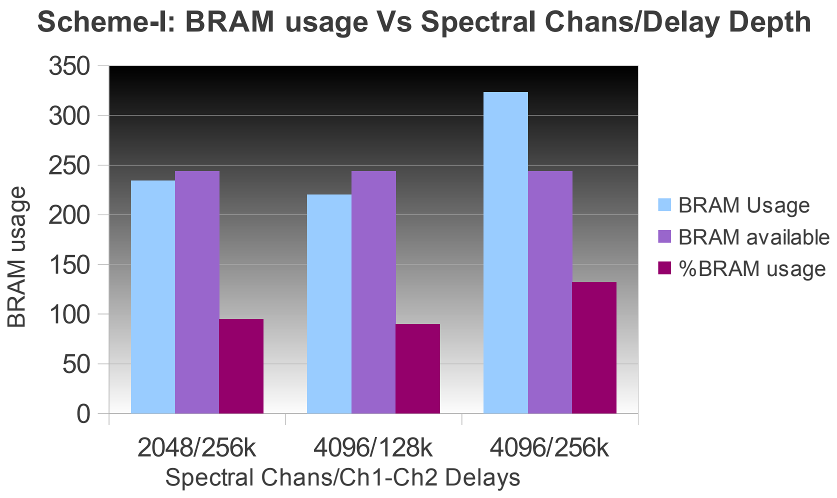
This is the maximum possible FFT that can be accommodated along with full GMRT delay of 256k clock cycles @ 800MHz on a F-engine ROACH-1 board with FPGA resource utilization of ~95%.

Spectral Chans	BRAM usage	Coarse delay

2048	234/244 (95%)	Chan-1 = 256K
		Chan-2 = 256K

4096	220/244 (90%)	Chan-1 = 128K
		Chan-2 = 128K

4096	323/244 (132%)
	Over-mapped
		Chan-1 = 256K
		Chan-2 = 256K



2. **Scheme-II :** Each F-engine processes 2 antenna single polarization in special combination.

   As coarse delay correction directly impacts BRAM utilization on an FPGA, considering the worst case delay it becomes a bottle neck. But not all antenna fully exercise maximum delay especially if we consider central square antennas whose requirement in our case comes out to be ~ 8k and rest arm antennas up to 256k.

   Hence in scheme-II unlike scheme-I, instead of using two polarizations from same antenna it combines a combination of central square antennas or to be precise those antennas whose worst case delay do not exceed 8k clock cycles and feed it to F-engine ROACH board.

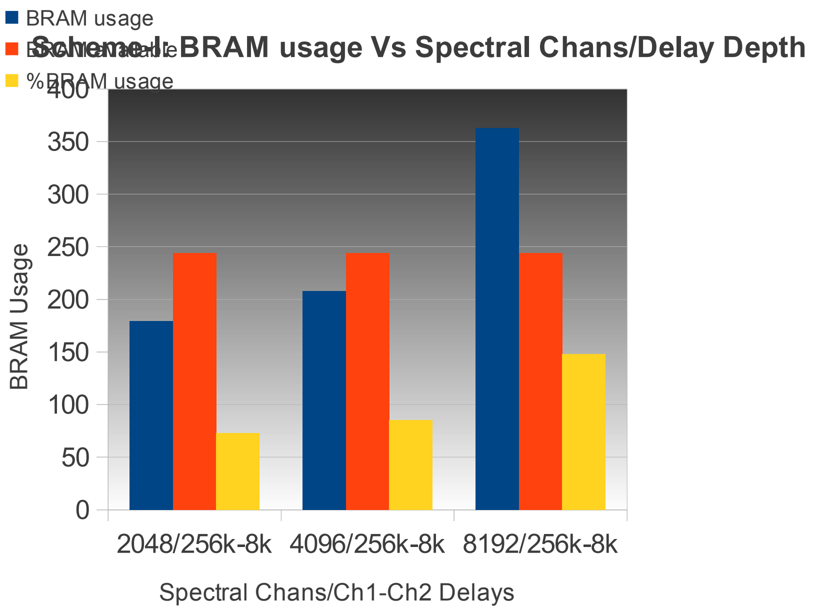
This scheme allows us to accommodate more number of spectral channels and at the same time meets GMRT worst case delay requirement. But the flip side is that it can not support full polar mode.

Spectral Chans	BRAM usage	Coarse delay

2048	179/244 (73%)	Chan-1 = 256K
		Chan-2 = 8K

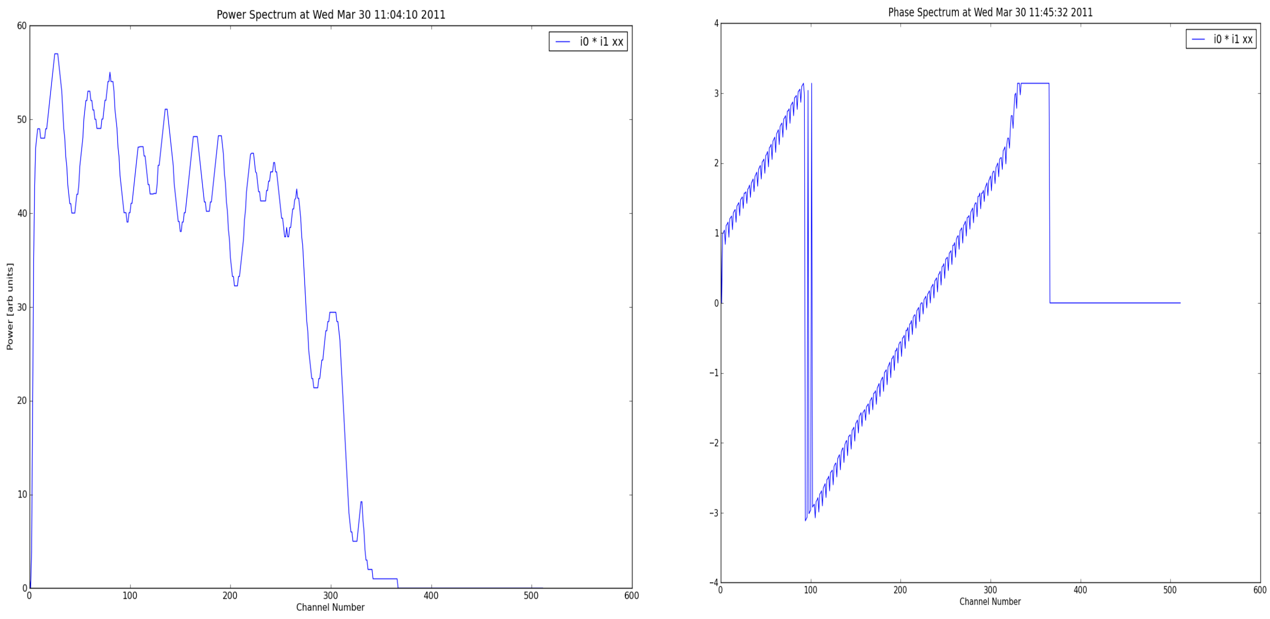
4096	208/244 (85%)	Chan-1 = 256K
		Chan-2 = 8K

4096	363/244 (148%)
	Over-mapped
		Chan-1 = 256K
		Chan-2 = 8K



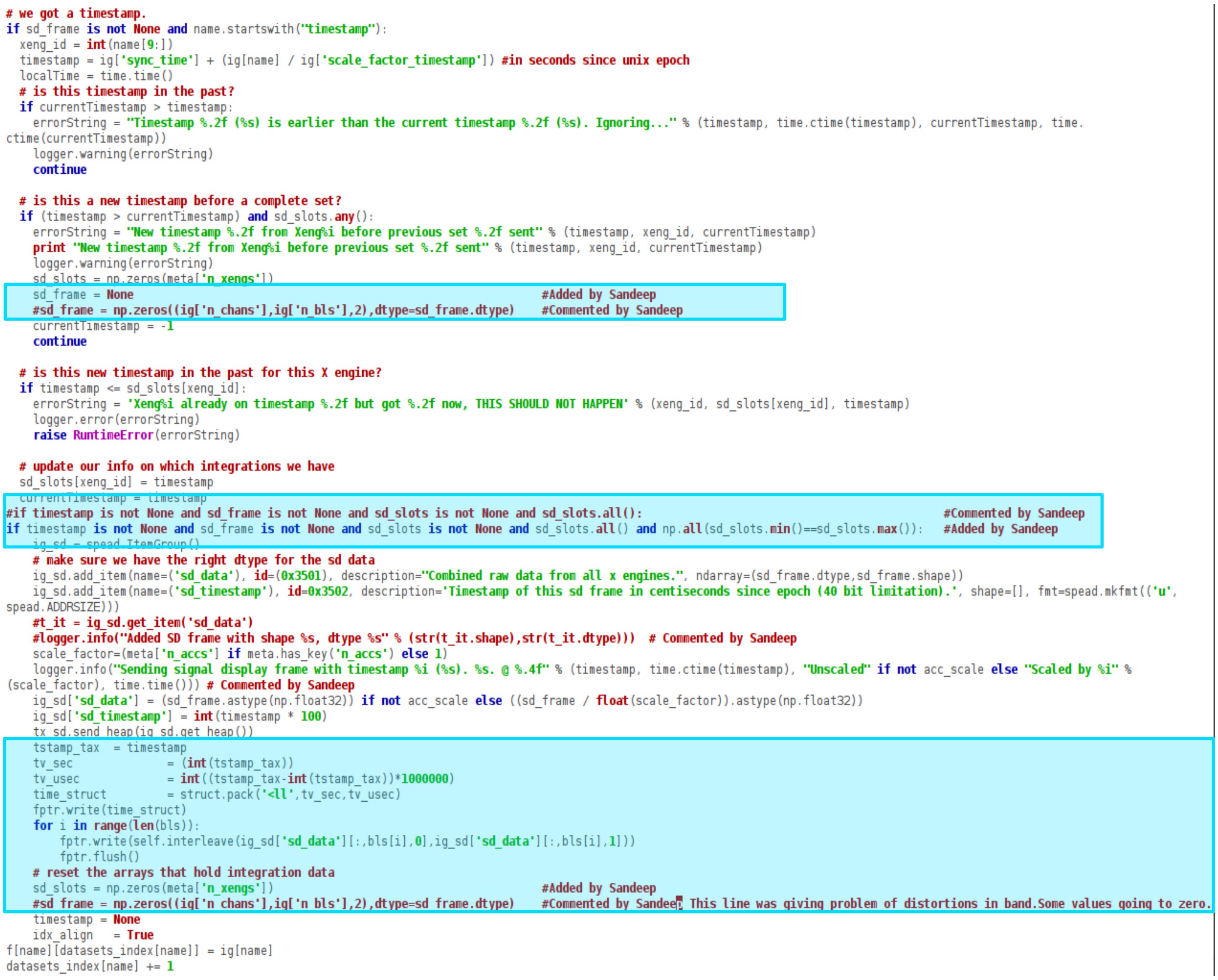
5. Corr-0.6 package modification for online data recording in GMRT tax native format :-

The packetized correlator data was dumped into HDF (Hierarchical Data Format) with extension *.h5*. The corr-0.6 package has a minute bug which was distorting the spectral waveform as shown below



Plot 15: Corr-0.6 package artifacts which distorts cross amplitude and phase intermittently

This problem was resolved while trying to dump the live date in tax native format.
In *"rx_inter"* function of *"rx.py"* script in corr-0.6 package modifications are highlighted with turquoise colour boxes along with comments made in front of each command. Also the tax dumping part added to the script as shown below :-

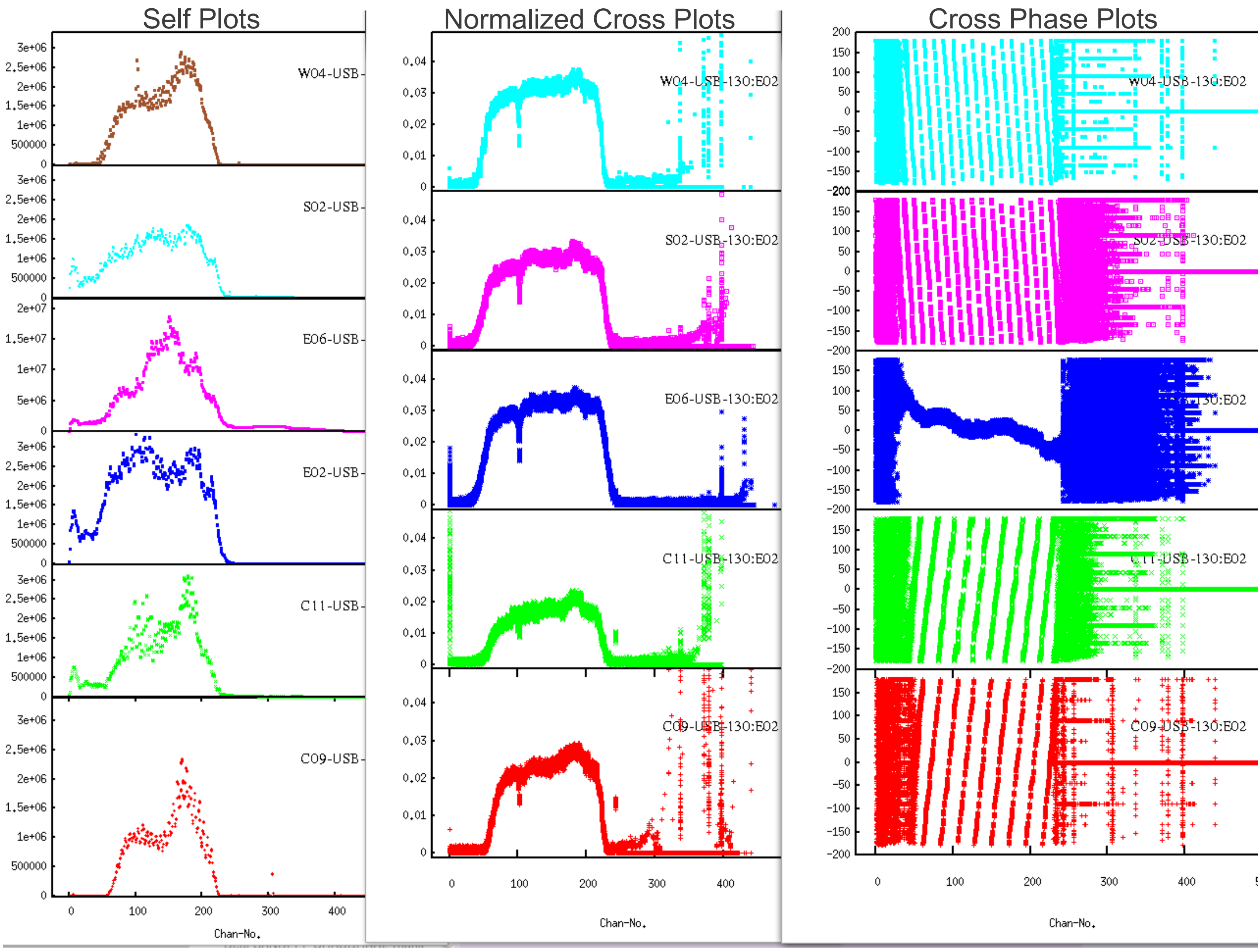


```
# we got a timestamp.
if sd_frame is not None and name.startswith("timestamp"):
  xeng_id = int(name[9:])
  timestamp = ig['sync_time'] + (ig[name] / ig['scale_factor_timestamp']) #in seconds since unix epoch
  localTime = time.time()
  # is this timestamp in the past?
  if currentTimestamp > timestamp:
    errorString = "Timestamp %.2f (%s) is earlier than the current timestamp %.2f (%s). Ignoring..." % (timestamp, time.ctime(timestamp), currentTimestamp, time.
ctime(currentTimestamp))
    logger.warning(errorString)
    continue

  # is this a new timestamp before a complete set?
  if (timestamp > currentTimestamp) and sd_slots.any():
    errorString = "New timestamp %.2f from Xeng%i before previous set %.2f sent" % (timestamp, xeng_id, currentTimestamp)
    print "New timestamp %.2f from Xeng%i before previous set %.2f sent" % (timestamp, xeng_id, currentTimestamp)
    logger.warning(errorString)
    sd_slots = np.zeros(meta['n_xengs'])
    sd_frame = None                                                         #Added by Sandeep
    #sd_frame = np.zeros((ig['n_chans'],ig['n_bls'],2),dtype=sd_frame.dtype)    #Commented by Sandeep
    currentTimestamp = -1
    continue

  # is this new timestamp in the past for this X engine?
  if timestamp <= sd_slots[xeng_id]:
    errorString = 'Xeng%i already on timestamp %.2f but got %.2f now, THIS SHOULD NOT HAPPEN' % (xeng_id, sd_slots[xeng_id], timestamp)
    logger.error(errorString)
    raise RuntimeError(errorString)

  # update our info on which integrations we have
  sd_slots[xeng_id] = timestamp
  currentTimestamp = timestamp
#if timestamp is not None and sd_frame is not None and sd_slots is not None and sd_slots.all():                      #Commented by Sandeep
if timestamp is not None and sd_frame is not None and sd_slots is not None and sd_slots.all() and np.all(sd_slots.min()==sd_slots.max()):   #Added by Sandeep
    ig_sd = spead.ItemGroup()
    # make sure we have the right dtype for the sd data
    ig_sd.add_item(name=('sd_data'), id=(0x3501), description="Combined raw data from all x engines.", ndarray=(sd_frame.dtype,sd_frame.shape))
    ig_sd.add_item(name=('sd_timestamp'), id=0x3502, description='Timestamp of this sd frame in centiseconds since epoch (40 bit limitation).', shape=[], fmt=spead.mkfmt(('u',
spead.ADDRSIZE)))
    #t_it = ig_sd.get_item('sd_data')
    #logger.info("Added SD frame with shape %s, dtype %s" % (str(t_it.shape),str(t_it.dtype)))  # Commented by Sandeep
    scale_factor=(meta['n_accs'] if meta.has_key('n_accs') else 1)
    logger.info("Sending signal display frame with timestamp %i (%s). %s. @ %.4f" % (timestamp, time.ctime(timestamp), "Unscaled" if not acc_scale else "Scaled by %i" %
(scale_factor), time.time())) # Commented by Sandeep
    ig_sd['sd_data'] = (sd_frame.astype(np.float32)) if not acc_scale else ((sd_frame / float(scale_factor)).astype(np.float32))
    ig_sd['sd_timestamp'] = int(timestamp * 100)
    tx_sd.send_heap(ig_sd.get_heap())
    tstamp_tax  = timestamp
    tv_sec              = (int(tstamp_tax))
    tv_usec             = int((tstamp_tax-int(tstamp_tax))*1000000)
    time_struct         = struct.pack('<ll',tv_sec,tv_usec)
    fptr.write(time_struct)
    for i in range(len(bls)):
        fptr.write(self.interleave(ig_sd['sd_data'][:,bls[i],0],ig_sd['sd_data'][:,bls[i],1]))
        fptr.flush()
    # reset the arrays that hold integration data
    sd_slots = np.zeros(meta['n_xengs'])                                    #Added by Sandeep
    #sd_frame = np.zeros((ig['n_chans'],ig['n_bls'],2),dtype=sd_frame.dtype)    #Commented by Sandeep This line was giving problem of distortions in band.Some values going to zero.
    timestamp = None
    idx_align   = True
f[name][datasets_index[name]] = ig[name]
datasets_index[name] += 1
```

6. Merging of packetized correlator F-engine and GPU packetizer part on F-engine ROACH board :-

The aim of combining packetized correlator F-engine with GPU packetizer was to be able to do simultaneous observation with same front-end i.e. iADC board and to be able to save connections at the time of observation. The design was tested along with GSB and a comparison was done. The details are mentioned below in **"Antenna Testing"** part.

➔ Antenna Testing :-
1. Broad-band Tests Part – I:-



Plot 16: Broad-band signal test with 4-Antenna packetized correlator - spectrum plot of self, normalized cross and cross phase

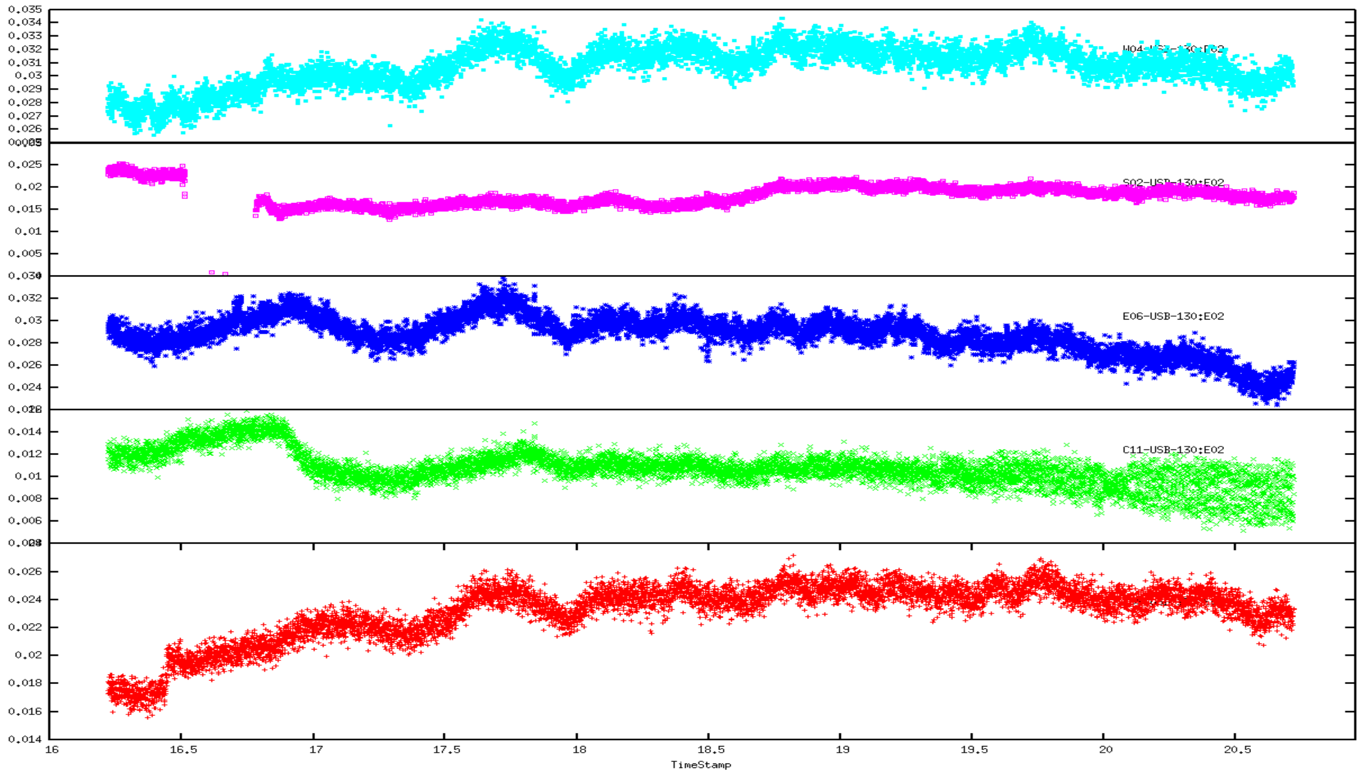
Direct RF from antenna at L-band processed by GMRT Analog Back-end (GAB) is available. Using this signal and packetized correlator tests were done on 9th August 2012. GSB was also ran simultaneously and comparative case study was done. The test details along with plots are as follows :

The test details are as follows :-

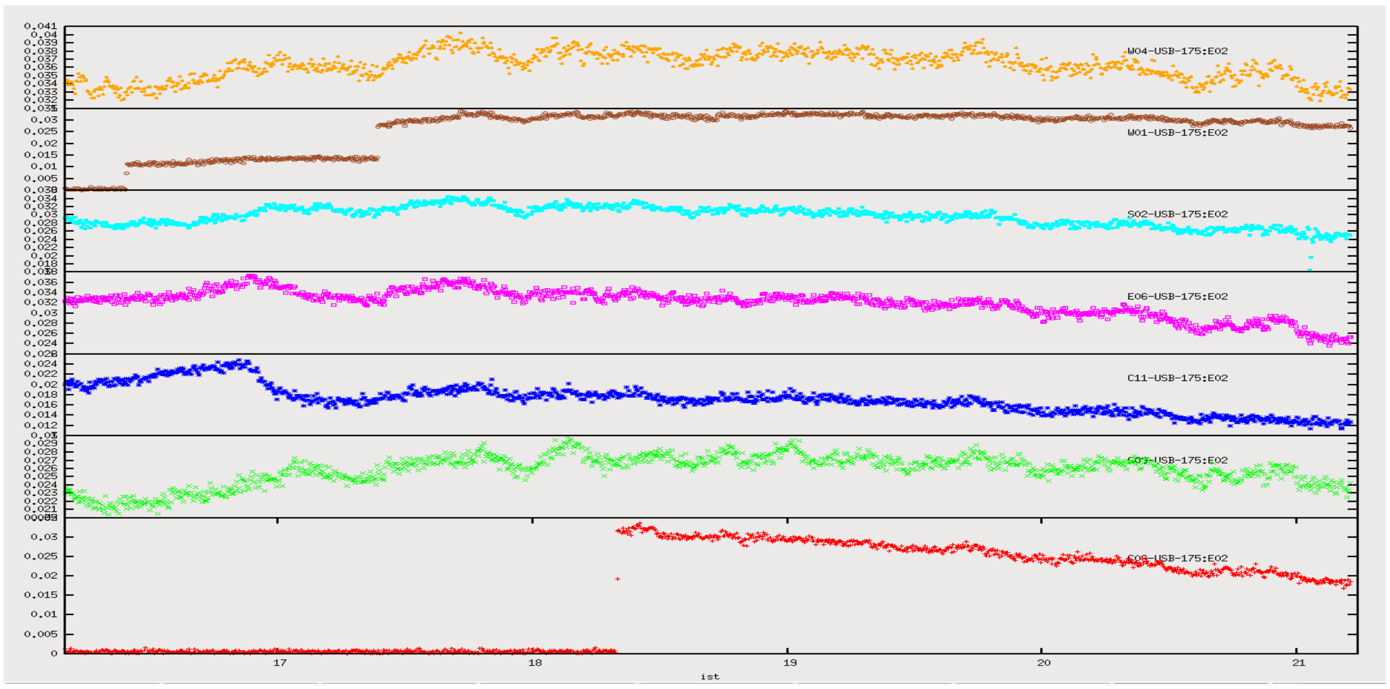
1. Date : 9th August 2012
2. Source : 3C286 (RA = 3.5418081960705 & DEC = 0.53140482780795)
3. Observation time : 16:15hrs to 21:10hrs (~5hrs)
    1. Packetized Correlator:
        1. 4-Antenna dual polarization design with spectral channels : 512 & BW = 400MHz
        2. Band : 1280MHz broad-band signal through GAB
        3. GAB LO : 1350MHz

Revision	Date	Modification/ Change
Ver. 1	10 September 2013	Initial Version

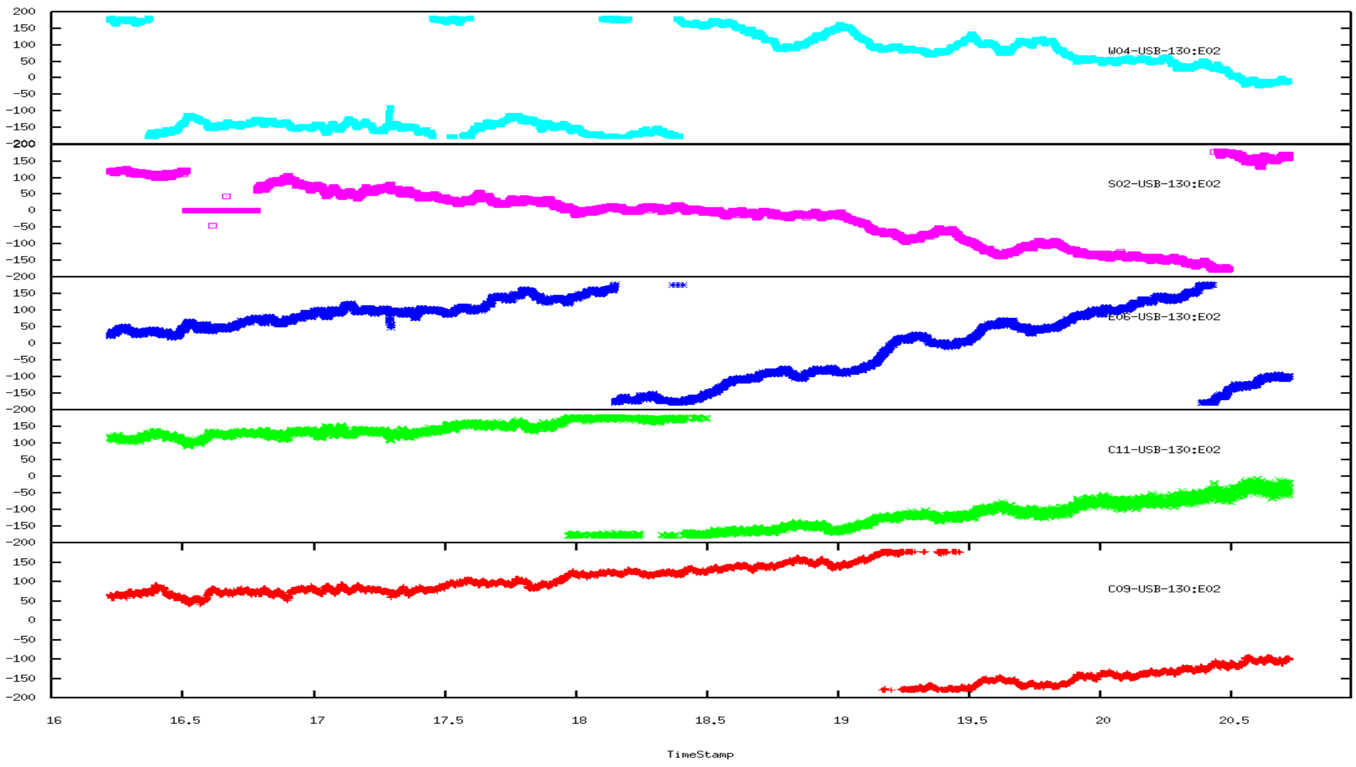
    4. Antenna used :
       0x => C06, 0y => W01, 1x => C11, 1y => C09
       2x => E02, 2y =>E06, 3x => W04, 3y => S02
    5. Integration time : ~2 seconds
2. GSB default settings viz.
    1. 32MHz default GMRT signal
    2. Spectral channels : 256
    3. Ist LO : 1210MHz & IVth LO : 51MHz
    4. Integration time : 16seconds



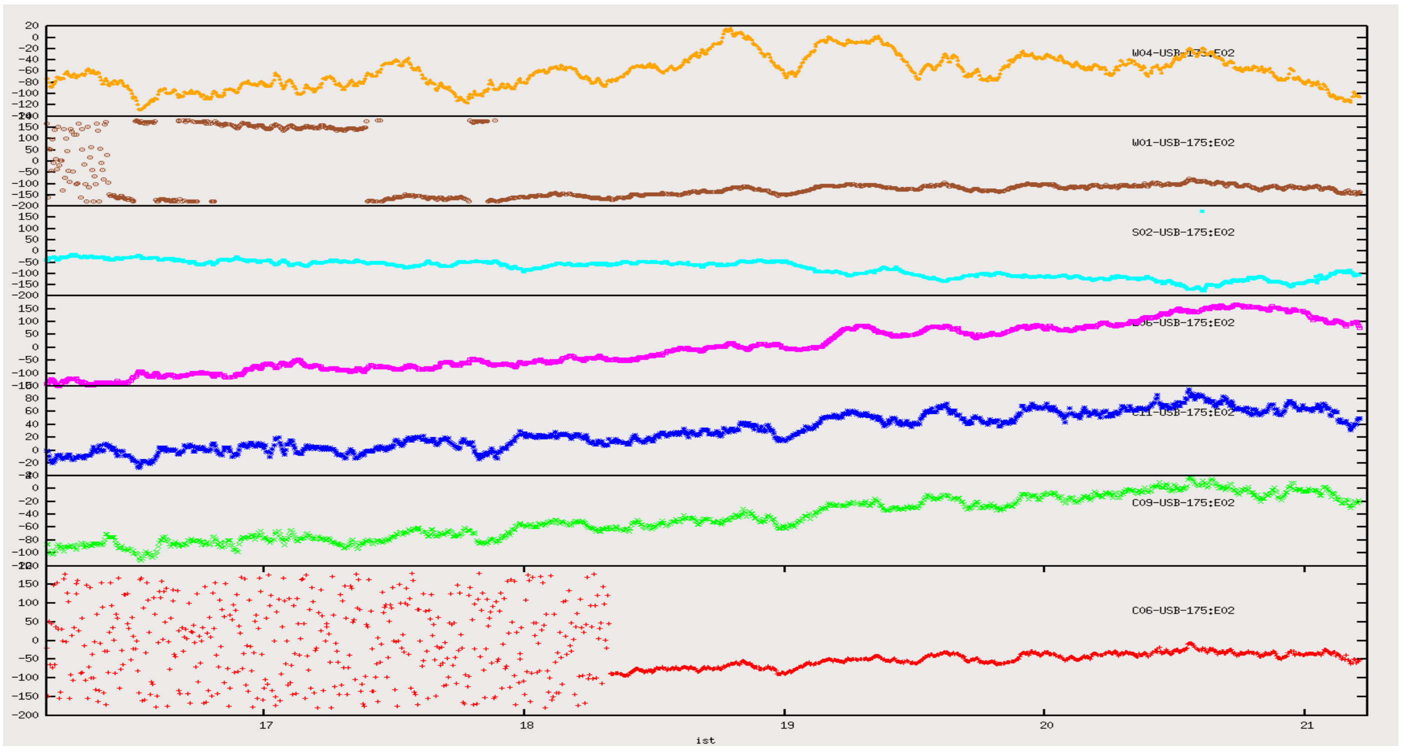
Plot 17: 4-Antenna packetized correlator broad-band signal normalized cross amplitude plot w.r.t. E02 for chan = 76 equivalent to GSB 125th chan plotted over time on9th August 2013



Plot 18: GSB normalized cross amplitude plot for chan = 76 equivalent to GSB 125th chan plotted over time on9th August 2013



Plot 19: 4-Antenna packetized correlator broad-band signal cross phase plot w.r.t. E02 for chan = 76 equivalent to GSB 125th chan plotted over time on 9th August 2013



Plot 20: GSB normalized cross phase plot for chan = 76 equivalent to GSB 125th chan plotted over time on 9th August 2013

Inferences :-

From above comparison plots we have following inferences :-

- The normalized cross amplitude of 4-antenna packetized correlator is comparatively less by ~ 0.005 to that of GSB. (Packetized corr chan ~ 76th & GSB chan 125th) over time.
- The cross phase of 4-antenna packetized correlator is having very small residual phase which is not visible in GSB when plotted over time.

2. Broad-band Tests Part – II:-

In the above test a minor change was instead of 4-antenna design a 8-antenna packetized correlator design was used. Rest of the test setup was same.

The test details are as follows :-

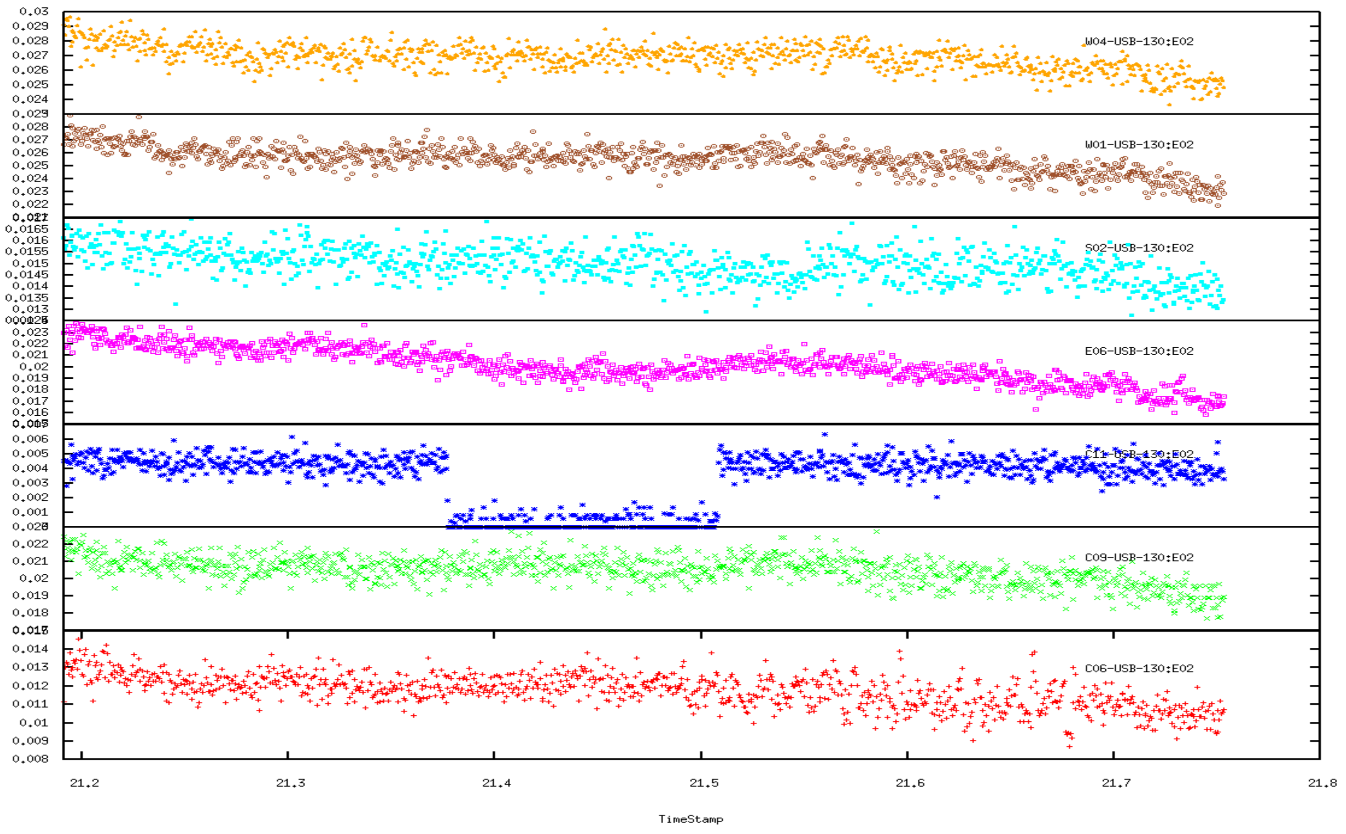
1. Date : 9th August 2012
2. Source : 3C286 (RA = 3.5418081960705 & DEC = 0.53140482780795)
3. Observation time : 21:20hrsto 22:05hrs (~45 minutes)
    1. Packetized Correlator:
        1. 8-Antenna dual polarization design with spectral channels : 512 & BW = 400MHz
        2. Band : 1280MHz broad-band signal through GAB
        3. GAB LO : 1350MHz
        4. Antenna used :

           0x => C06 , 0y => W01
           1x => C11 , 1y => C09
           2x => E02 , 2y => E06
           3x => W04 , 3y => S02
           4x to 7y => No connection
        5. Integration time : ~2 seconds
    2. GSB default settings viz.
        1. 32MHz normal GMRT chain
        2. Spectral channels : 256
        3. Ist LO : 1210MHz & IVth LO : 51MHz
        4. Integration time : 16 seconds

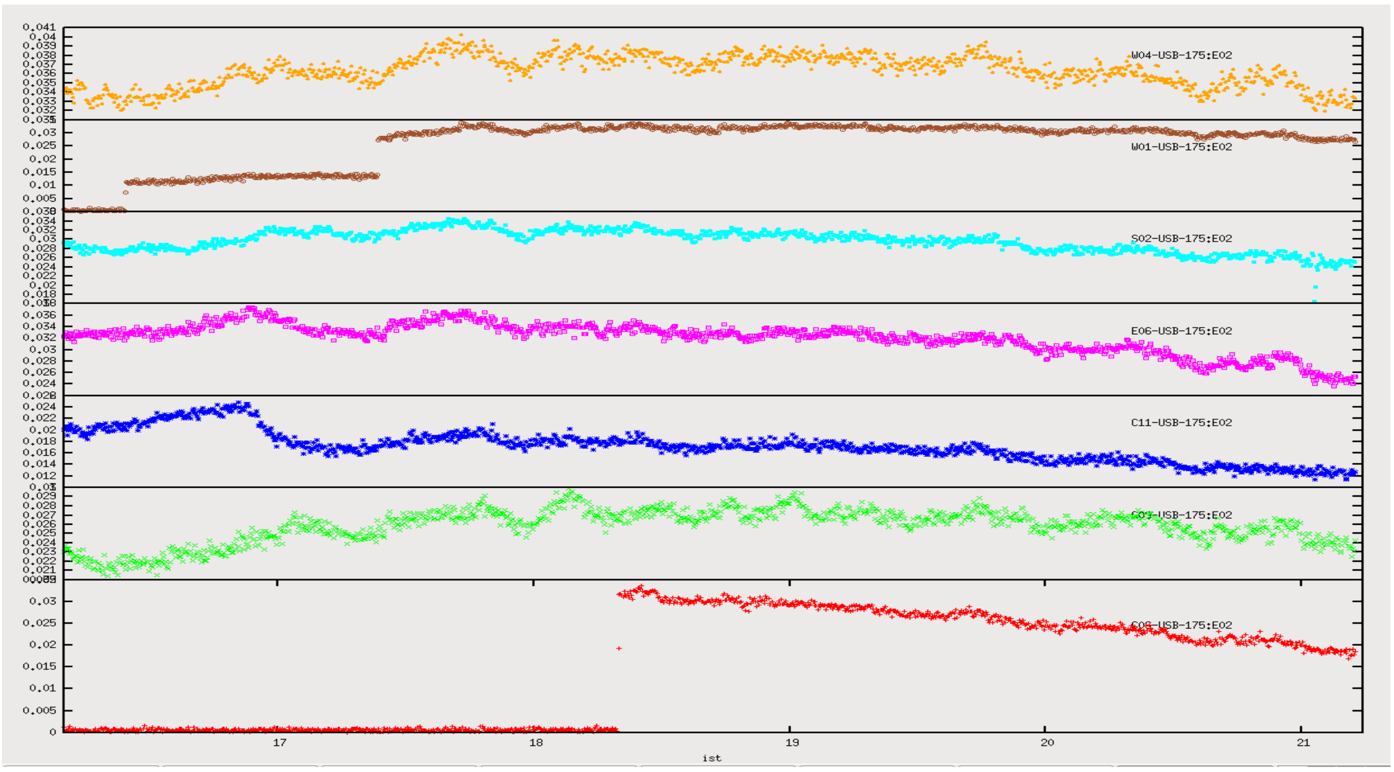
Inferences :-

From above comparison plots we have following inferences :-

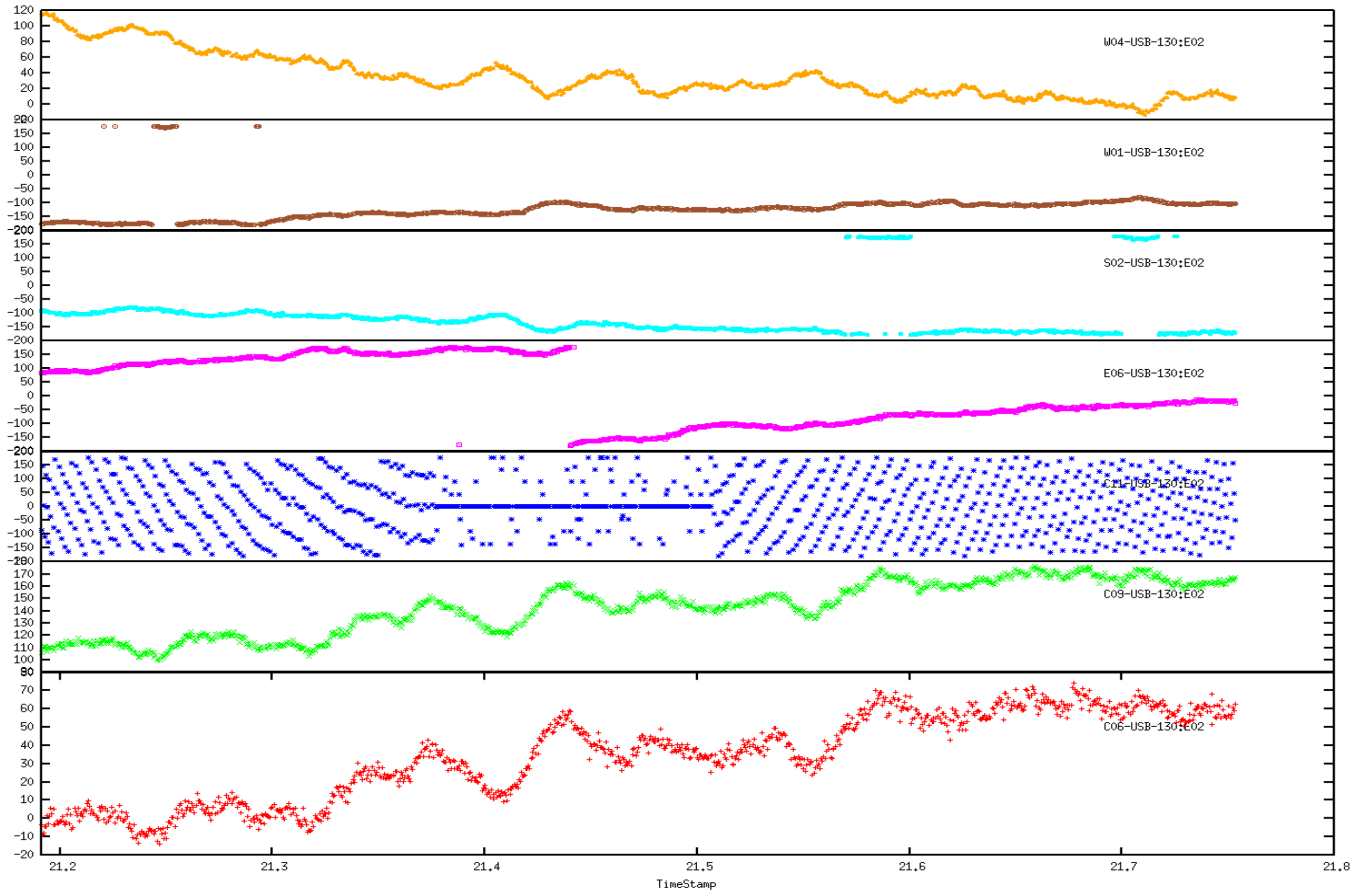
- The normalized cross amplitude of 8-antenna packetized correlator is also comparatively less by ~ 0.005 to that of GSB. (Packetized corr chan ~ 76th & GSB chan 125th) over time.
- The cross phase of 8-antenna packetized correlator is having very small residual phase which is not visible in GSB when plotted over time, but due to very short observation it is not visible. But we have to note that GSB is running on 32MHz narrow band signals to it's counterpart - Packetized correlator, which is running on Broad-Band antenna signals.



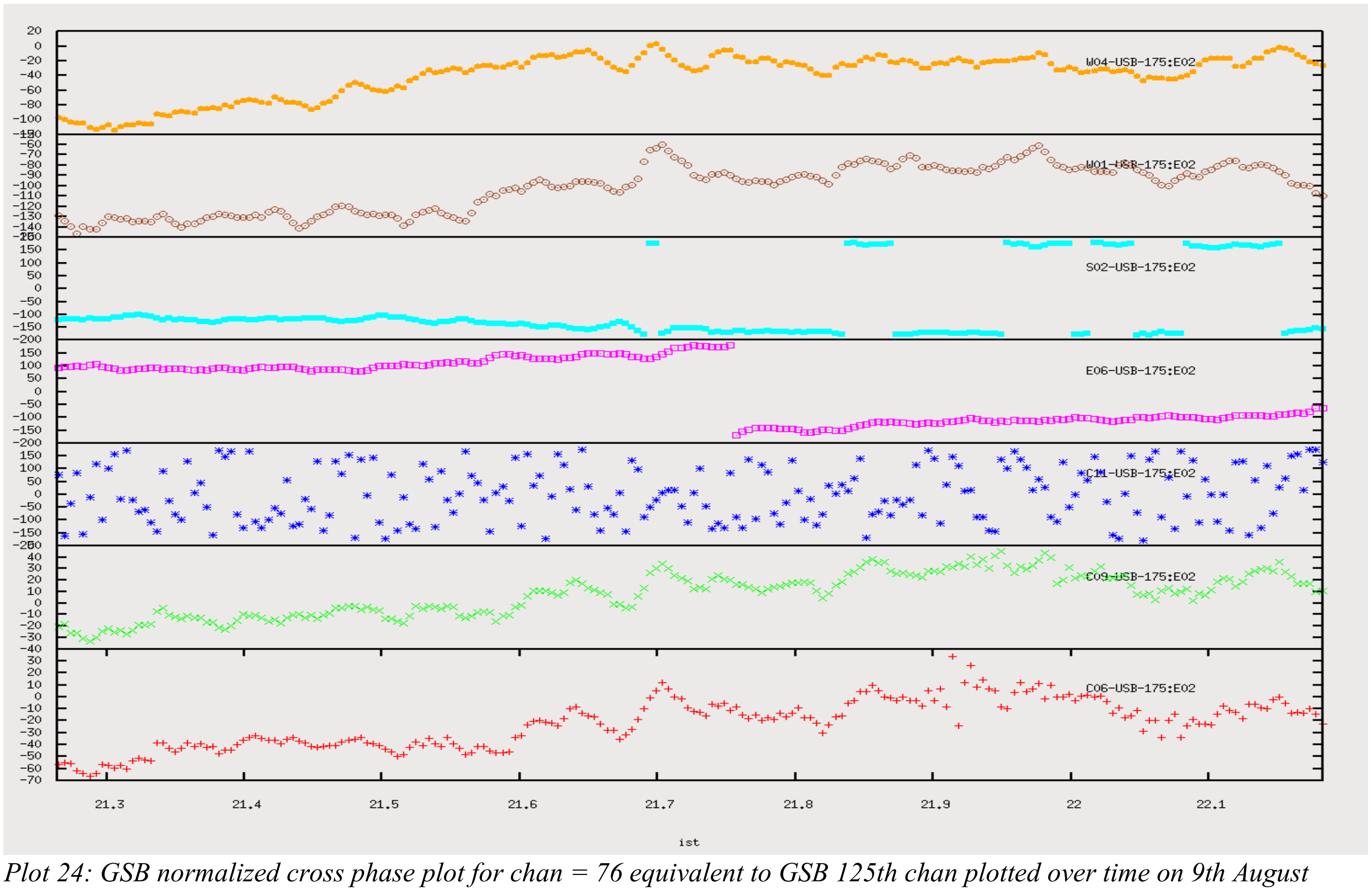
Plot 21: 8-Antenna packetized correlator broad-band signal normalized cross amplitude plot w.r.t. E02 for chan = 76 equivalent to GSB 125th chan plotted over time on9th August 2013



Plot 22: 8-Antenna-GSB normalized cross amplitude plot for chan = 76 equivalent to GSB 125th chan plotted over time on9th August 2013



Plot 23: 8-Antenna packetized correlator broad-band signal cross phase plot w.r.t. E02 for chan = 76 equivalent to GSB 125th chan plotted over time on 9th August 2013



Plot 24: GSB normalized cross phase plot for chan = 76 equivalent to GSB 125th chan plotted over time on 9th August 2013

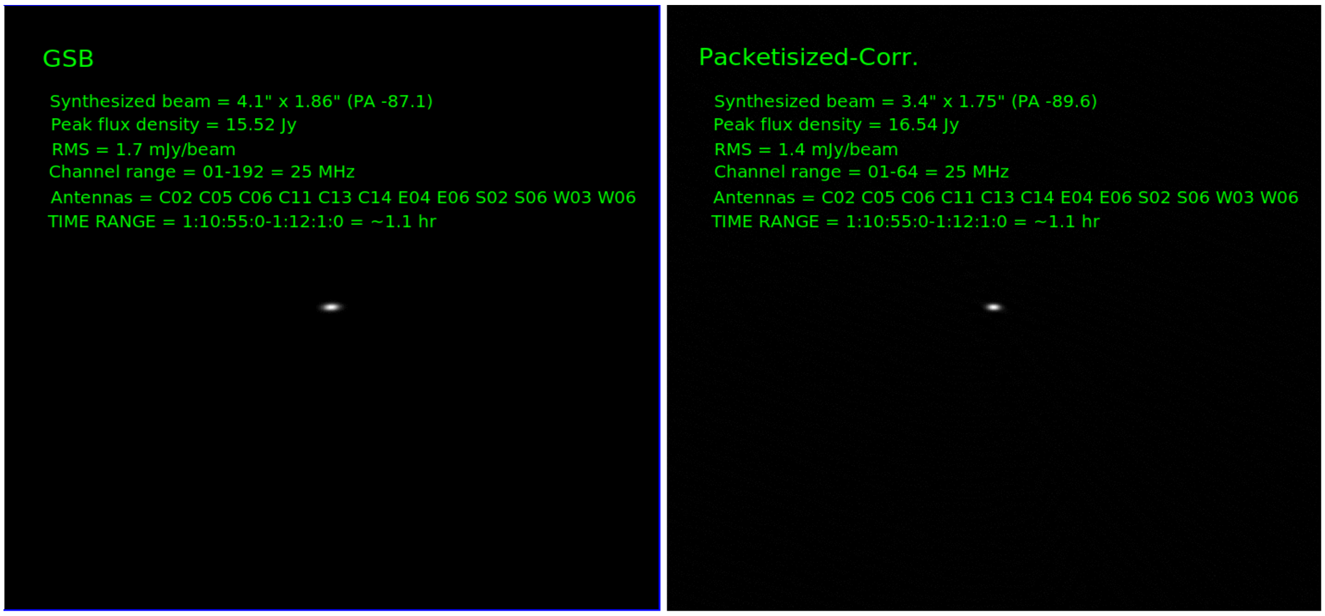
3. 3C286 point source observation on 17th October 2012:-

First image from packetized correlator data was made for point source 3C286 and the observation was done on 17th October 2012. The details of observations are as follows:

1. Source : 3C48
2. Observation time : 14:30hrs to 16:30hrs (~2hrs)
3. Packetized Correlator:
   1. 8-Antenna dual polarization design with spectral channels : 2048 (Usable channels : 10 to 370) & BW = 200MHz
   2. Band : 1280MHz with 32MHz default GMRT signal path (130 polarization connected)
   3. GAB LO : 1350MHz
   4. Antenna used : (Working antennas : 15)

0x	=>	C02 ,	0y	=>	C05
1x	=>	C06 ,	1y	=>	C11
2x	=>	C12 ,	2y	=>	C13
3x	=>	C14 ,	3y	=>	W01
4x	=>	W03 ,	4y	=>	W04
5x	=>	W06 ,	5y	=>	S02
6x	=>	S06 ,	6y	=>	E04
7x	=>	E05 ,	7y	=>	E06

   5. Integration time : 3.99998976 seconds
4. GSB default settings viz.
   1. Band : 1280MHz with 32MHz normal GMRT chain with 175 polarization connected.
   2. Spectral channels : 512
   3. Ist LO : 1350MHz & IVth LO : 51MHz
   4. Integration time : 16 seconds default



GSB

Synthesized beam = 4.1" x 1.86" (PA -87.1)
Peak flux density = 15.52 Jy
RMS = 1.7 mJy/beam
Channel range = 01-192 = 25 MHz
Antennas = C02 C05 C06 C11 C13 C14 E04 E06 S02 S06 W03 W06
TIME RANGE = 1:10:55:0-1:12:1:0 = ~1.1 hr

Packetisized-Corr.

Synthesized beam = 3.4" x 1.75" (PA -89.6)
Peak flux density = 16.54 Jy
RMS = 1.4 mJy/beam
Channel range = 01-64 = 25 MHz
Antennas = C02 C05 C06 C11 C13 C14 E04 E06 S02 S06 W03 W06
TIME RANGE = 1:10:55:0-1:12:1:0 = ~1.1 hr

Image 1: Point source image comparison made by using GSB and packetized correlator data

Revision	Date	Modification/ Change
Ver. 1	10 September 2013	Initial Version

4. **3C285 extended source observation on 5th December 2012:-**

The details of the observation of the extended source 3C285 was done on 5th December 2012 and 3C286 as a calibrator. The details of observations are as follows :

1. Source : 3C285 as an extended source & 3C286 as a calibrator
2. Observation time : 12:44hrs to 14:23hrs (~1hrs 45mins)
3. Packetized Correlator:
   1. 8-Antenna dual polarization design with spectral channels : 512 (Usable channels : 0 to 77) & BW = 200MHz
   2. Band : 1280MHz with 32MHz default GMRT signal path (130 polarization connected)
   3. GAB LO : 1350MHz
   4. Antenna used : (Working antennas : 15)

0x	=> C01 ,	0y	=> C03	
1x	=> C04 ,	1y	=> C05	
2x	=> C06 ,	2y	=> C10	
3x	=> C11 ,	3y	=> C12	
4x	=> C13 ,	4y	=> C14	(C14 – Servo time out)
5x	=> E03 ,	5y	=> E06	
6x	=> S02 ,	6y	=> S04	
7x	=> W01 ,	7y	=> W04	

   5. Integration time : 3.99998976 seconds
4. GSB default settings viz.
   1. Band: 1280MHz with 32MHz normal GMRT chain with 175 polarization connected
   2. Spectral channels : 256
   3. Ist LO : 1350MHz & IVth LO : 51MHz
   4. Integration time : 16.106 seconds

The scans were taken in following manner :

1. Scan-1 (Calibrator)	: 3C286 (time ~12:44 to ~13:01) ~17mins
2. Scan-2 (Extended Source)	: 3C285 (time ~13:02 to 13:30) ~30mins
3. Scan-3 (Calibrator)	: 3C286 (time ~13:32 to 13:42) ~ 10mins
4. Scan-4 (Extended Source)	: 3C285 (time ~13:43 to 14:10) ~ 27mins
5. Scan-5 (Calibrator)	: 3C286 (time ~14:12 to 14:23) ~ 11mins

Then packetized correlator data converted into raw then to LTA and finally to FITS file format. Corresponding comparative images with comments from astronomers viz. *Dr. Poonam Chandra* & *Dr. Dharam Vir Lal* who analyzed the data are as follow :

1. Packetized correlator shows very high visibilities and geometric delays need to be corrected.
2. In contour and B&W images more of the extended emission features are reproduced in case of packetized correlator data.
3. In B&W image, the GSB image being dull and packetized correlator image being brighter.
4. In contour map, the flux levels are same.
5. These two images are without self calibration because packetized correlator data were having many failed solutions in AIPS.

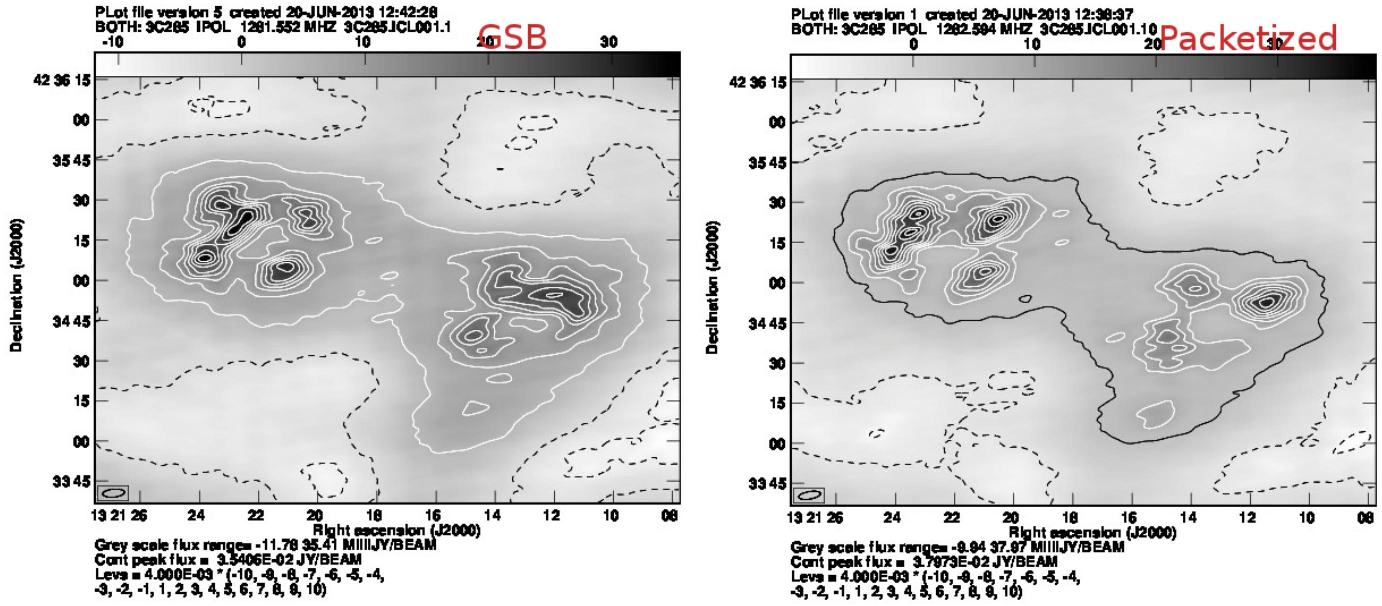


Image 2: Contour image of extended source 3C285 comparing GSB and packetized correlator data taken on 5th Dec 2012

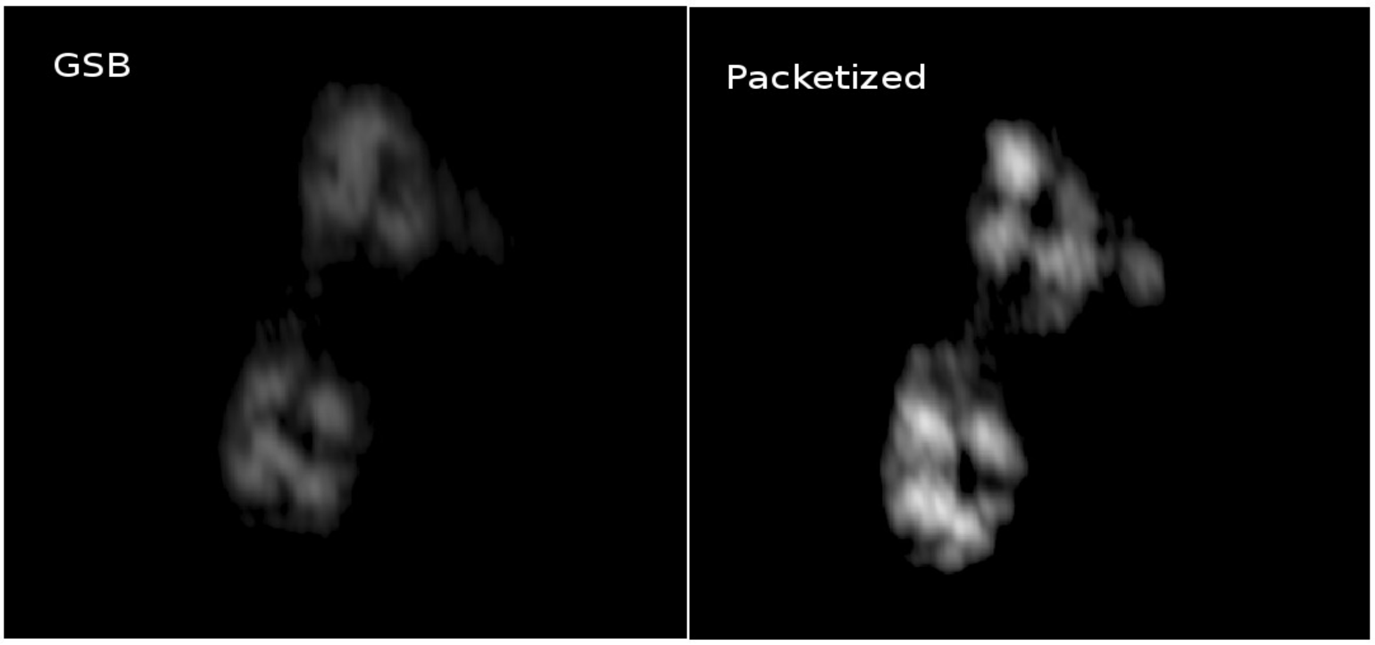


Image 3: B&W image of extended source 3C285 comparing GSB and packetized correlator data taken on 5th Dec 2012

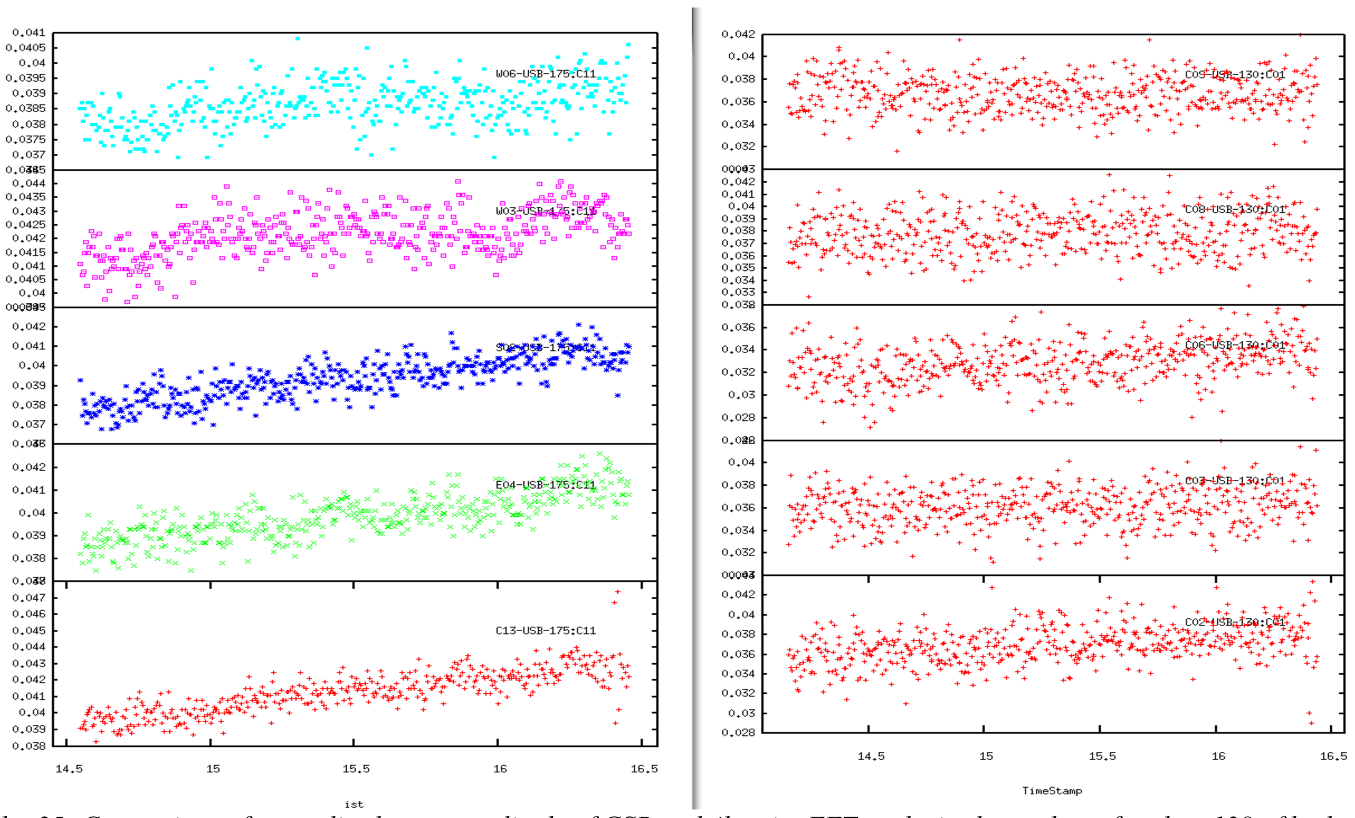
5. Packetized correlator with 4k point FFT test done on 3C48 dated 28th January 2013 :

Modified packetized design with 4k point FFT i.e. 2048 spectral channels was tested on 28th January 2013. The details of observations are as follows :

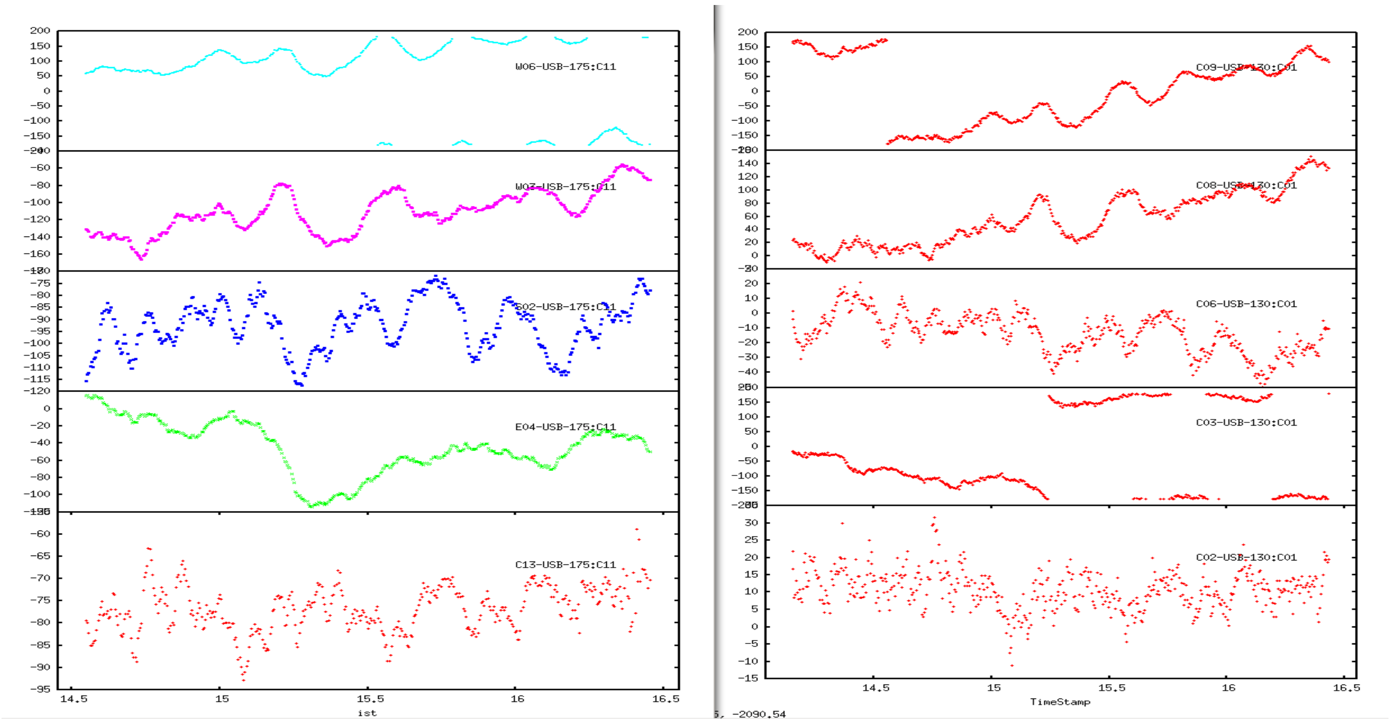
1. Source : 3C48
2. Observation time : 14:30hrs to 16:30hrs (~2hrs)
3. Packetized Correlator:
    1. 8-Antenna dual polarization design with spectral channels : 2048 (Usable channels : 10 to 370) & BW = 200MHz

   2. Band : 1280MHz with 32MHz default GMRT signal path (130 polarization connected)
   3. GAB LO : 1350MHz
   4. Antenna used : (Working antennas : 15)
      0x => C02 , 0y => C11
      1x => C13 , 1y => E04
      2x => NC , 2y => NC
      3x => S02 , 3y => W03
      4x => W06 , 4y => NC
      5x => NC , 5y => NC
      6x => NC , 6y => NC
      7x => NC , 7y => NC
   5. Integration time : 2.00409088 seconds
4. GSB default settings viz.
   1. Band: 1280 MHz 32MHz normal GMRT chain with 175 polarization connected
   2. Spectral channels : 256
      Ist LO : 1350MHz & IVth LO : 51MHz
   3. Integration time : 16 seconds default

   Following are the comparative plots of GSB and 4k point FFT packetized correlator design:



Plot 25: Comparison of normalized cross amplitude of GSB and 4k point FFT packetized correlator for chan 128 of both



Plot 26: Comparison of cross phase of GSB and 4k point FFT packetized correlator for chan 128 of both

6. Packetized, GSB and GPU correlator comparison test on 3C48 dated 14th February 2013:-

This was a very important comparative study of packetized correlator, GSB and GPU correlator on 3C48 done on 14th February 2013. The study shows behavior of the three correlators in case of same source. The test details are as follow:

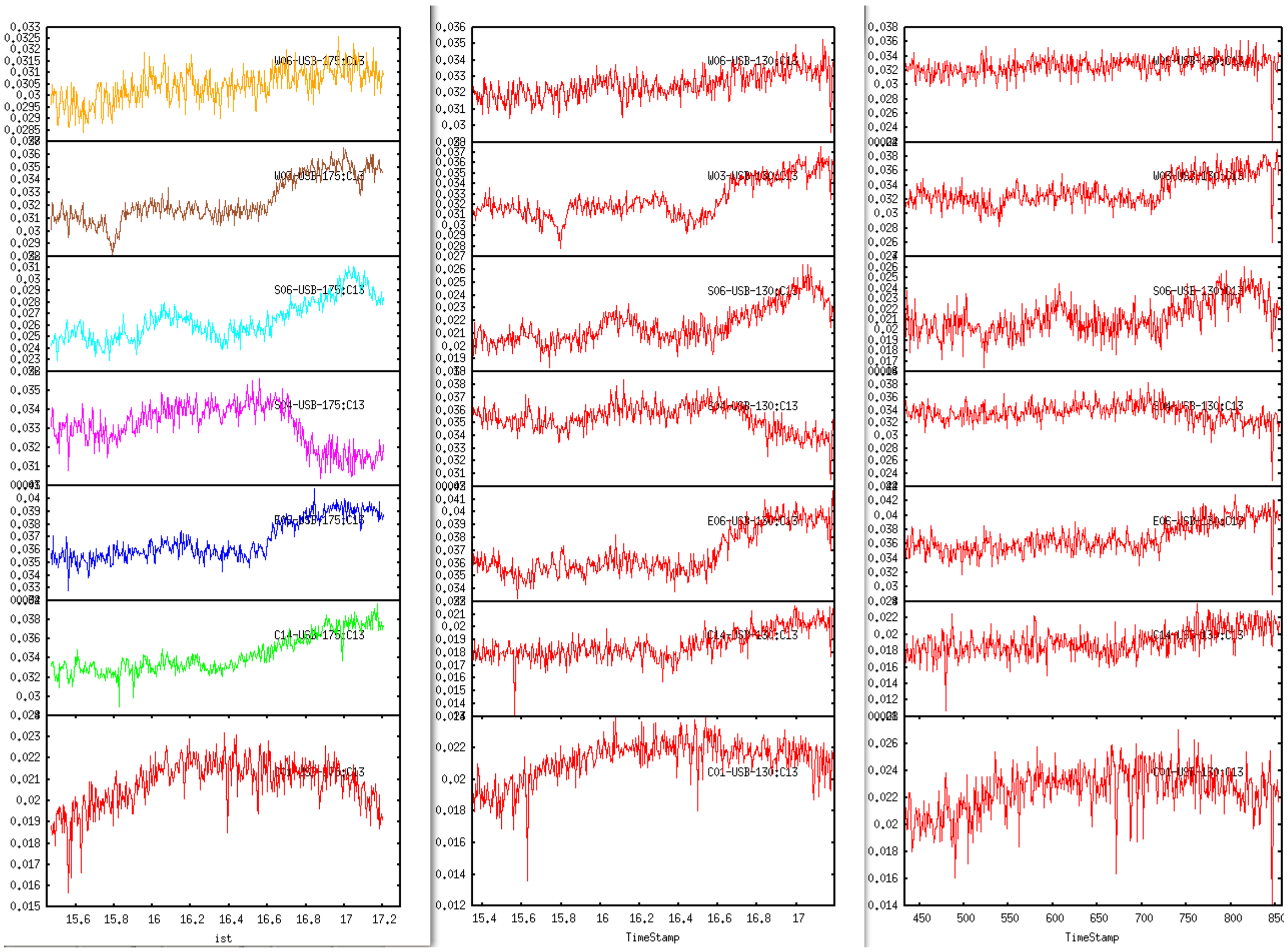
1. Source : 3C48
2. Observation time : 13:30hrs to 17:20hrs (~4hrs)
3. Packetized Correlator:
    1. 8-Antenna dual polarization design with spectral channels : 512 (Usable channels : 10 to 370) & BW = 200MHz
    2. Band : 1280MHz with 32MHz default GMRT signal path (130 polarization connected)
    3. GAB LO : 1350MHz
    4. Antenna used : (Working antennas : 15)
       0x => C01 , 0y => C13
       1x => C14 , 1y => E06
       2x => NC , 2y => NC
       3x => S04 , 3y => S06
       4x => W03 , 4y => W06
       5x => NC , 5y => NC
       6x => NC , 6y => NC
       7x => NC , 7y => NC
    5. Integration time : 3.99998976 seconds
4. GSB default settings viz.

Revision	Date	Modification/ Change
Ver. 1	10 September 2013	Initial Version

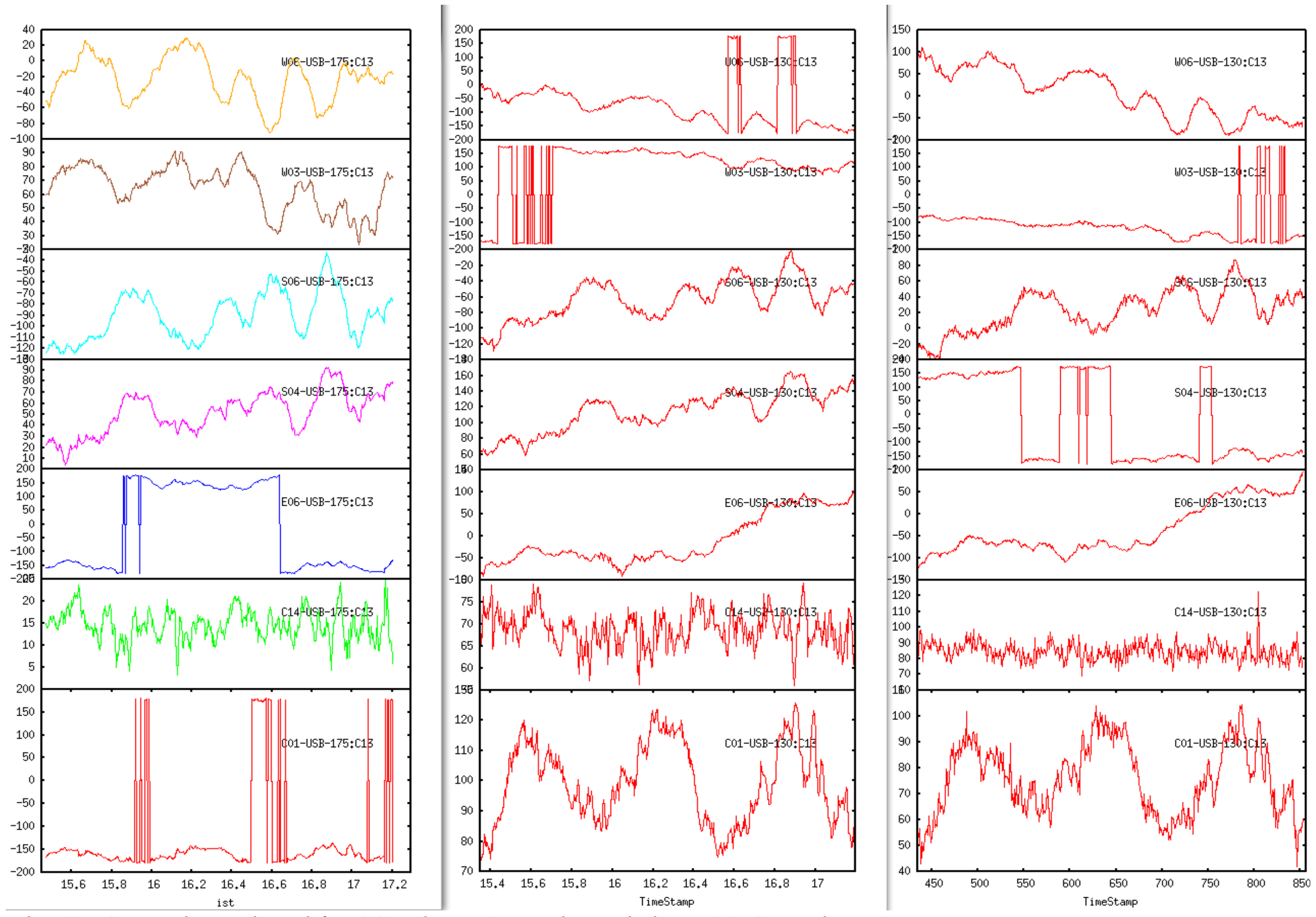
1. Band: 1280MHz with 32MHz normal GMRT chain with 175 polarization connected
2. Spectral channels : 256
   Ist LO : 1350MHz & IVth LO : 51MHz
3. Integration time : 16 seconds default

5. GPU correlator :
   1. 4-antenna dual polarization design with spectral channels :2048 & BW = 200MHz
   2. Band : 1280MHz with 32MHz default GMRT signal path (130 polarization connected)
   3. GAB LO : 1350MHz
   4. Integration time : 4.026528 seconds

   Following are the comparative plots of packetized correlator, GSB and GPU correlator designs:



Plot 27: Normalized cross plotted for GSB chan=64, Packetized chan=21, GPU chan=85 w.r.t. time



Plot 28: Cross phase plotted for GSB chan=64, Packetized chan=21, GPU chan=85 w.r.t. time

7. Combine packetized correlator and GPU packetizer design on F-engin ROACH board :-

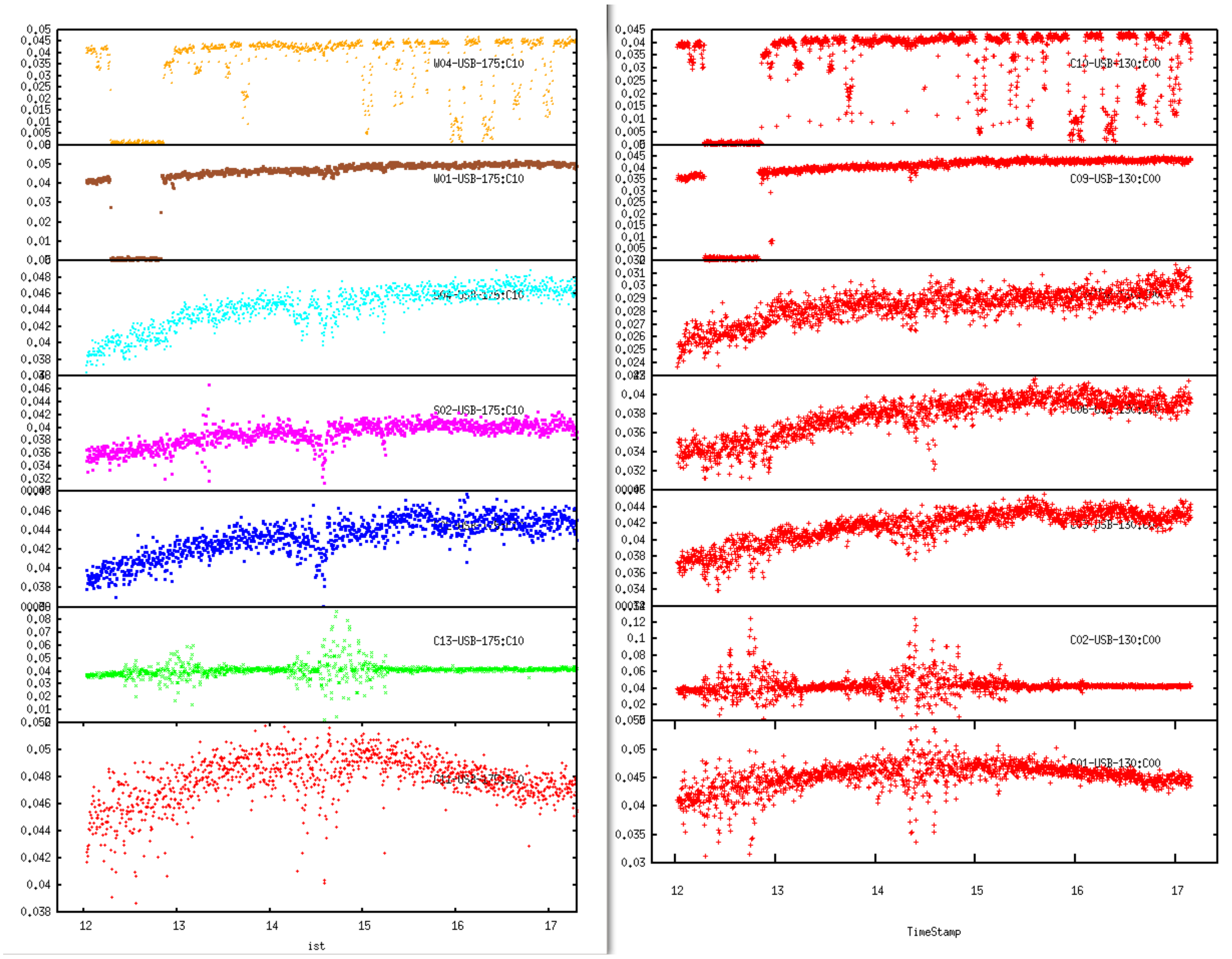
The design of packetized correlator F-engine and GPU packetizer design was combined on a single F-engine ROACH board to save ROACH boards, reduce 2-way power divider requirement by using same iADC for combined design and do simultaneous testing. To accommodate both design simultaneous, packetized correlator coarse delay depth was decreased from 256k to 128k clock cycles. Though design was compiled for 800MHz adc clock, it can be operated at 400MHz. (At 800MHz, all central square antennas + few arm antennas can be used and if used 400MHz clock frequency then complete GMRT range of delay can be compensated).

Here only packetized correlator part with GSB was tested for it's functionality. The testing of combined design was done on 3C48 done on 2nd February 2013. The study shows behavior of the three correlators in case of same source. The test details are as follow:

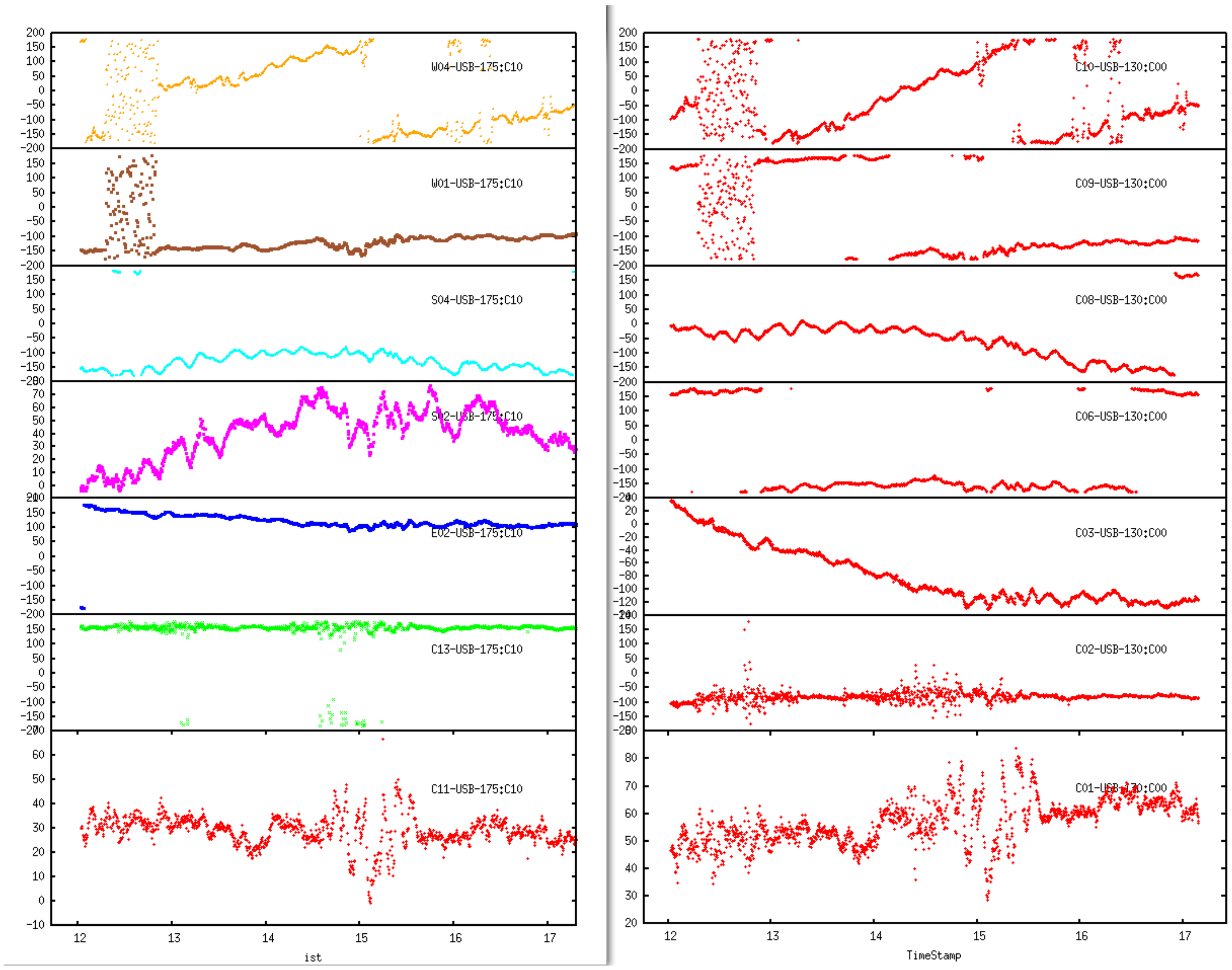
1. Source : 3C48
2. Observation time : 12:00hrs to 17:10hrs (~5hrs)
3. Packetized Correlator:

   1. 8-Antenna dual polarization design with spectral channels : 512 (Usable channels : 10 to 370) & BW = 200MHz
   2. Band : 1280MHz with 32MHz default GMRT signal path (130 polarization connected)
   3. GAB LO : 1350MHz
   4. Antenna used : (Working antennas : 15)
      0x => C10 , 0y => C11
      1x => C13 , 1y => E02
      2x => NC , 2y => NC
      3x => S02 , 3y => S04
      4x => W01 , 4y => W04
      5x => NC , 5y => NC
      6x => NC , 6y => NC
      7x => NC , 7y => NC
   5. Integration time : 3.99998976 seconds
4. GSB default settings viz.
   1. Band: 1280MHz with 32MHz normal GMRT chain with 175 polarization connected
   2. Spectral channels : 256
      Ist LO : 1350MHz & IVth LO : 51MHz
   3. Integration time : 16 seconds default

   Following are the comparative plots of packetized correlator+GPU packetizer on F-engine and GSB:



Plot 29: GSB and Pakcetized+GPU packetizer combined design with packetized corr comparison only, normalized cross.



Plot 30: GSB and Pakcetized+GPU packetizer combined design with packetized corr comparison only, cross phase w.r.t. C10.

APPENDIX-I
SOP for testing 4/8 antenna PACKETIZED CORRELATOR.

By : Sandeep C. Chaudhari & Irappa M. Halagali .
VERSION : 2 Dated : 21/03/2013

Packetized correlator is a general purpose re-configurable digital backend for radio astronomy applications on CASPER hardware ROACH-Vertex5 boards using XAUI or 10Gbe designs. This SOP is for XAUI designs.

I. Power ON and instrument settings :

(a) Power on Distribution boards , AC distribution boards in the RACK
(verify : PC , ethernet swithces & ROACH Units are powered ON).
(b) Now Switch ON the instruments and do the following settings -
(i) Signal generator settings as CLOCK to F engine ROACH boards : Freq = 800MHz , Power = 0dbm, Set "Mod to OFF and RF to ON"
(ii) Connect the PPS signal available in the rack to all Sync input of F engine ROACH boards.
(c) Power "ON" the ROACH boards by pressing the switch at front panel. All Roach boards will boot through NFS (Network File System) on control PC {192.168.4.71}.

II. Interconnections :

1. Connect the clock & pps signals available at top of the rack to all ROACH units clk_i and Sync inputs of iADC respectively.
2. Either connect 32MHz or Broadband analog signals or Noise Source through 200 MHz LPF, Total Power over BW between -14dbm to -17dbm to iADC's inputs.

III. XAUI BASED 4/8 Antenna packetized correlator setup :

GMRT Digital Backend specifications :

Number of stations	:	4/8 dual pol.
Maximum instantaneous Bandwidth	:	400 MHz
Number of spectral channels	:	512
Number of input polarizations	:	2
Full stokes capability	:	yes
Dump time	:	minimum 128 ms
Coarse and Fine delay correction range	:	+/- 128 us.
Base integration	:	1.00007936 seconds.

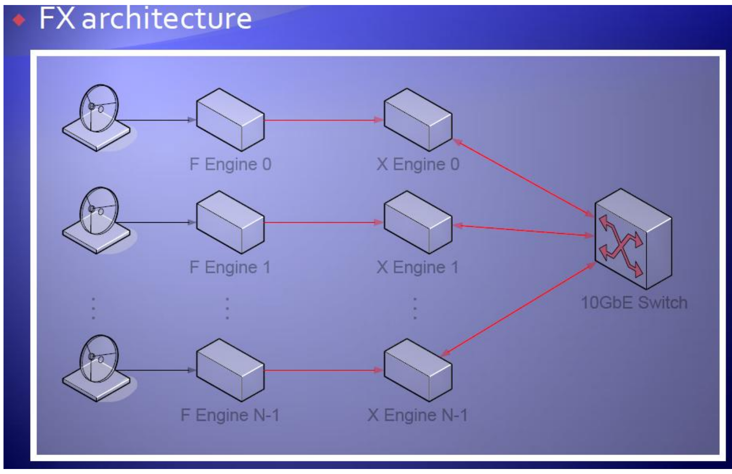
	4 antenna	8 antenna
Total ROACH boards required	8	16
F engine Roach boards	4	8
X engine Roach boards	4	8

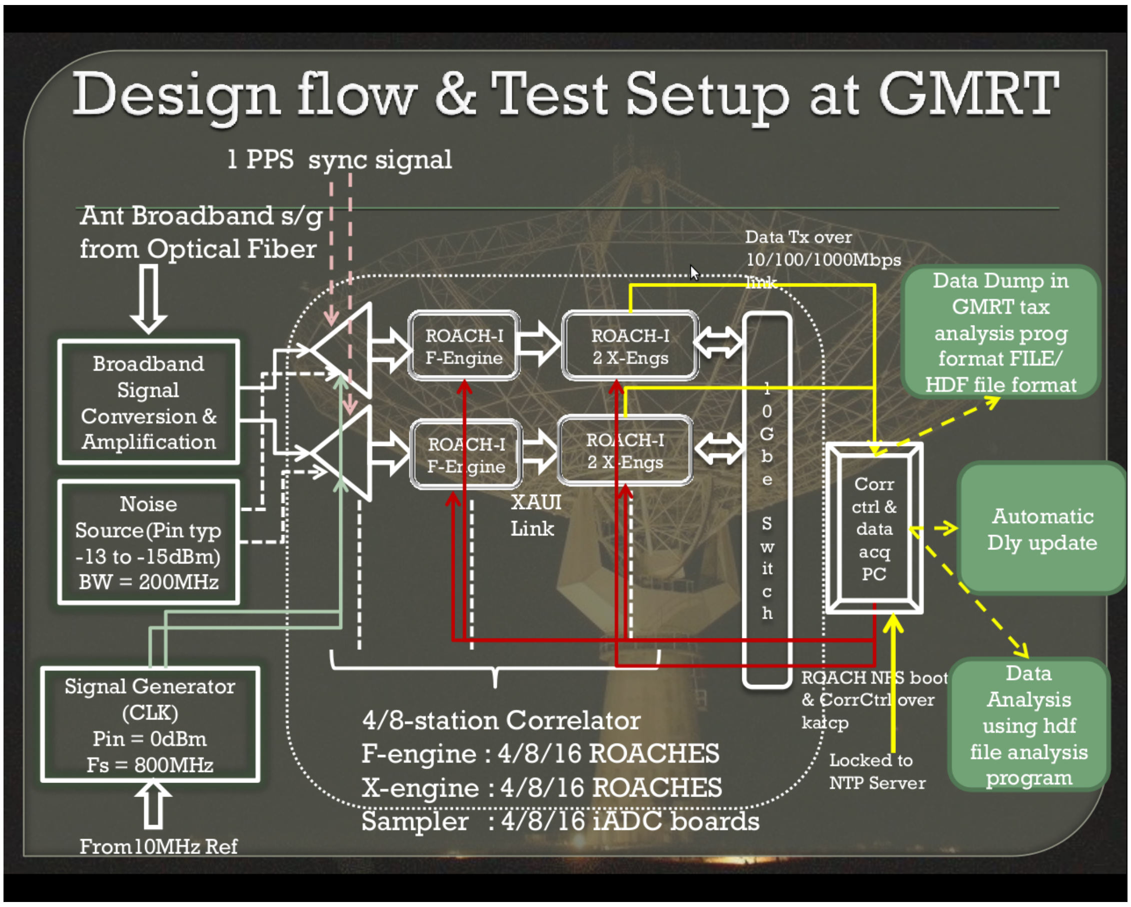
Note : For complete packetized correlator setup refer the following diagrams.

XAUI Packetized Correlator Connections :-

1. nth F-eng CX-4 port-0 => nth X-eng CX-4 port-0.
2. All X-eng CX-4 port-3 => 10Gbe Switch.
3. All ROACH board's eth-0 port connected to 1Gbps switch.
4. 1Gbps to the control PC's eth-1 port in a private network.

Control PC eth-0 IP : 192.168.4.71
Control PC eth-1 IP : 192.168.100.1 (Private Network)
ROACH Board eth-0 IP : 192.168.100.<Last 2 MAC No.>




Design flow & Test Setup at GMRT
1 PPS sync signal
Ant Broadband s/g from Optical Fiber
Broadband Signal Conversion & Amplification
Noise Source(Pin typ -13 to -15dBm) BW = 200MHz
Signal Generator (CLK) Pin = 0dBm Fs = 800MHz
From10MHz Ref
ROACH-I F-Engine
ROACH-I 2 X-Engs
ROACH-I F-Engine
ROACH-I 2 X-Engs
XAUI Link
Data Tx over 10/100/1000Mbps link
10Gbe Switch
Data Dump in GMRT tax analysis prog format FILE/ HDF file format
Corr ctrl & data acq PC
Automatic Dly update
ROACH NFS boot & CorrCtrl over katcp
Locked to NTP Server
Data Analysis using hdf file analysis program
4/8-station Correlator
F-engine : 4/8/16 ROACHES
X-engine : 4/8/16 ROACHES
Sampler : 4/8/16 iADC boards

Revision	Date	Modification/ Change
Ver. 1	10 September 2013	Initial Version

IV. Initialization & Data acquisition :

Login to control PC : ssh -X gmrt@192.168.4.71
Password : gmrttifr
> sudo konsole

This opens new console. Now start working in newly opened console. Press shift+ctrl+N to open more konsoles.

Note : 1. For all scripts -h commandline option will show options associated with that script.
2. Go to working Directory in Each KONSOLE TAB
Working Directories :

a. for 4 antenna
/home/gmrt/Packetized_Corr/XAUI_Corr/4Ant_Corr/Ant_Tests/ or
/home/gmrt/Packetized_Corr/XAUI_Corr/4Ant_Corr/Basic_Test/ (for Noise Source tests)

b. for 8 antenna
/home/gmrt/Packetized_Corr/XAUI_Corr/8Ant_Corr/Ant_Tests/ or
/home/gmrt/Packetized_Corr/XAUI_Corr/8Ant_Corr/Basic_Test/ (for Noise Source tests)

3. Determining integration number in configure file.

For base integration of 1.00007936 following integration number should be entered in the configure file in multiple of below mentioned number.
1K point FFT ===> n = 6104.
4K point FFT ===> n = 1526.
Multiple of n should be entered in configure file so as to get integration in multiple of base integration in acc_len field as acc_len = <desired n>.

KONSOLE TAB-1 : { To issue some basic commands from ipython }

> ipython
```
In [1]: import
corr,struct,time,sys,spead,numpy,pylab,katsdisp,h5py,signal
In [2]: dh=katsdisp.KATData()
In [3]: dh.start_spead_receiver()
In [4]: c=corr.corr_functions.Correlator(connect=True,
config_file='config_8ant' or 'config_4ant', log_handler=None,
log_level=20)
In [5]: c.deprog_all()
```
(This Step is required if we want to test 4-antenna correlator in a 8-antenna setup)

KONSOLE TAB-2 : { For Configuring the system }

```
> corr_rx.py -a config_4ant          #for 4 antenna
> corr_rx.py -a config_8ant          #for 8 antenna
```

```
Parsing config file...INFO:corrsys:Configuration file config_4ant parsed ok.
 done.
Initalising SPEAD transports for inter data...
Data reception on port 7148
Sending Signal Display data to 127.0.0.1:7149.
Storing to file 1346318311.91.corr.h5
starting target with kwargs  {'acc_scale': False, 'sd_port': 7149, 'sd_ip':
'127.0.0.1', 'filename': '1346318311.91.corr.h5'}
WARNING: This function is not yet tested. YMMV.
INFO:rx:Data reception on port 7148.
INFO:rx:Sending Signal Display data to 127.0.0.1:7149.
INFO:rx:Starting file 1346318311.91.corr.h5.
```

Note : Remove the latest h5 file generated by corr_rx.py in the same area without deleting corresponding file with .raw extension.

KONSOLE TAB-3 : **{ For initializing the system }**

> corr_init.py config_4ant **#for 4 antenna**
> corr_init.py config_8ant **#for 8 antenna**

```
Connecting... done                        # messages for 4 antenna initialization.
======================
Initial configuration:
======================
 Clearing the FPGAs... done.
 Programming the Fengines with r_128w_512_11_r370_mod3_16_2012_Jul_04_2019.bof and the
Xengines with r_1f_2x_4a_r340c_2011_Feb_11_1454.bof... done.

 Pausing 10GbE data exchange... Pausing Xengs... done.
 Syncing the F engines... Armed. Expect trigg at 14:55:40 local (09:25:40 UTC). SPEAD
packet sent.
 Checking F engine clocks... ok
 Setting the board indices... done
 Setting the FFT shift schedule to 0x7FF... done
 Configuring EQ... done
 Configuring the 10GbE cores... done
 Waiting 13.6 seconds for ARP to complete... done
 Starting 10GbE data exchange... X engines re-enabled.
 Flushing loopback muxs... done.
==================================
Verifying correct data exchange...
==================================
 Wait 2 seconds for system to stabalise... done
 Resetting error counters... done
 Checking that all XAUI links are working... ok
 Checking that the same timestamp F engine data is arriving at all X boards within a
sync period... ok
 Checking that FPGAs are sending 10GbE packets... ok
 Checking that all X engine FPGAs are receiving 10GbE packets... ok
 Waiting for loopback muxes to sync... ok
 Checking that all X engines are receiving all their packets... ok
 Setting the number of accumulations to 1562368 (2.000 seconds) and syncing VACCs...
done
```

```
 Checking vector accumulators... Waiting for an integration to finish... done.
Checking... ok
 Sending SPEAD metatdata and data descriptors to 127.0.0.1:7148... done
 Configuring output to 192.168.100.1:7148... skipped.
 Starting transmission of data... done
 Resetting error counters... done
 Enabling KITT... done
```

KONSOLE TAB-4 : { For running the data/packet transmitting script on X-engines to dump data in h5 format in the current directory }

A spead packet transmitting script invoked on all x-eng ROACH boards by this shell script.
Note : Dump file will be in the current directory with extention " .raw "
<corr_rx_script_start_time_since_epoch.h5.raw>

#for 4 antenna

> cd /home/gmrt/Packetized_Corr/XAUI_Corr/4Ant_Corr/

> ./roach_TXallstart_lab.sh

```
1346319332.159112223
ssh root@roach030167 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s
1346319332.159112223 315
ssh root@roach030116 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s
1346319332.159112223 315
ssh root@roach030174 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s
1346319332.159112223 325
ssh root@roach040235 corr_tx_spead_inter.py -l 1024 -i 192.168.100.1 -x 2 -s
1346319332.159112223 315
```

#for 8 antenna

> cd /home/gmrt/Packetized_Corr/XAUI_Corr/8Ant_Corr/

> ./roach_TXallstart_lab.sh

For Stopping the data transmission at any instant of time then run following shell script, which will kill spead packet tx script on all x-eng ROACH boards.

#for 4 antenna

>/home/gmrt/Packetized_Corr/XAUI_Corr/4Ant_Corr/roach_TXallstop_lab.sh

```
ssh root@roach030167 pkill -f corr_tx_spead_inter.py
ssh root@roach030116 pkill -f corr_tx_spead_inter.py
ssh root@roach030174 pkill -f corr_tx_spead_inter.py
ssh root@roach040235 pkill -f corr_tx_spead_inter.py
```

#for 8 antenna

>/home/gmrt/Packetized_Corr/XAUI_Corr/8Ant_Corr/roach_TXallstop_lab.sh

V. Delay UPDATE :

For doing delay update, go to delay_cal_brdbnd/ directory in the existing location. Following changes to be done during antenna testing :

1. Update source.hdr file with correspoinding RA & DEC information from one of the online machine (lenyadri/shivneri) as follows :

   > ssh observer@lenyadri

   > Password: obs@gmrt

   > work

   > ./user0.5

   ```
   GMRT 1: Enter user ID number
   ```

   >?40

   GMRT 1: Recovered POPS environment from last exit

> gts '3C286' **<or any other source name>**

```
 3C286       13h31m08.28s    +30d30'32.9"      12  12
GMRT 1: 3C286       RA DE(2000.) 13h31m08.28s   30d30'32.9"
 Epoch
     51544.000000000    56371.000000000    2000.0027379070    2013.2162946536
 Precess out
     3.5392571970105   0.53248492555936    2000.0000000000
3.5420542254091
     0.53123517233948    3.5419185663391   0.53130182972364
2013.2162946536
GMRT 1: 3C286          precessed 13h31m46.74s   30d26'15.1"
GMRT 1: 3C286        rises and s- 20h52m00.33s -  7h34m47.11s
 ONLINE NOT STARTED,ISHMSTAT=  0 ABORTING
GMRT 1: UNAVAILABLE!
```

Put red underlined RA & DEC values in source.hdr as follows

<SRC NAME-3C147> **3.5420542254091** **0.53123517233948** <I-LO = default 1400000000> <BW=400000000> <(Decides LO > RF)1401000000> <Redundent number 70000000>

2. In sampler.hdr enter antenna with connections as follows :
   0x=> <C01>, 0y=> <C06>
   1x=> <C09>, 1y=> <C12>
   2x=> <S01>, 2y=> <S04>
   3x=> <E06>, 3y=> <W06> For 4-Antennas (8-inputs)

   4x=> <E02>, 4y=> <E04>
   5x=> <S02>, 4y=> <S03>
   6x=> <W02>, 4y=> <W04>
   7x=> <W03>, 4y=> <E05> For 8-Antennas (16-inputs)

For 4 antenna : Run following command

```
>./delay_update_pktizd.py -f -t -d -r -p <config_file> --dly_offset=0 0 0 0 0 0 0 0
```

For 8 antenna :

```
>./delay_update_pktizd.py -f -t -d -r -p <config_file> --dly_offset=0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

VI. Offline data analysis :

For 4 antenna , ".raw" data file can be analysed using native tax file program in the directory ~/Packetized_Corr/XAUI_Corr/4Ant_Corr/Ant_Tests/tax_prog_4Ant_V03 is used.

For plotting spectrum :

```
>./xtrgsb32_4ant_tstamp -v <.raw file> -c 1,511 -t 1,20 -r C00
```

For plotting single channel normalized value w.r.t. Time :

```
>./xtrgsb32_4ant_tstamp -v <.raw file> -c 128 -t 1,1000000 -r C00 -n 1
```

For 8 antenna , ".raw" data file can be analysed using native tax file program in the directory ~/Packetized_Corr/XAUI_Corr/8Ant_Corr/Ant_Tests/tax_prog_8Ant_V03 is used.

For plotting spectrum :

```
>./xtrgsb32_8ant_tstamp -v <.raw file> -c 1,511 -t 1,20 -r C00
```

For plotting single channel normalized value w.r.t. Time :

```
>./xtrgsb32_8ant_tstamp -v <.raw file> -c 128 -t 1,1000000 -r C00 -n 1
```

VI. Raw to FITS file conversion for 8 Antenna dual pol (16-inputs):

NOTE: Code is kept in astro4:/temp30/ksanjay/SCLTA_PKT/
scp this directory to 32 bit m/c, e.g. astro0 is 32 bit m/c.
$ROOT_dir is any directory where you want to keep code.

```
> scp /temp30/ksanjay/SCLTA_PKT/ astro0:$ROOT_DIR/SCLTA_PKT/
```

# take care, $ROOT_DIR/data is data data dir, so do not scp it, it's huge file.
# It is kept as sample file, instead scp your own pkt_corr data.

Compile code by following command.

```
> cd $ROOT_DIR/sclta_pkt_master/
> make clean;make
> cd $ROOT_DIR/gvfits_pkt.working/
> make clean;make
```

working directory for analysis:

```
> cd $ROOT_DIR/sclta_pkt_master/
```

1. Edit sampler.hdr for the connection made to pkt corr.
# 16 antennas to be entered, no antenna name to be duplicate.
# although it is 8 antenna, dual pole, it is used in 16 antenna
# single pol mode. Working antennna connections has to be accurate.
# others, leftover antenna names can be arbitrary.

2. edit corrsel.hdr for

```
  a.    CHAN_NUM= 0:511:1
  b.  LTA     = 1
```

# leave others untouched.

3. edit scan.hdr for

```
    OBJECT  = 3C48
    RA-APP  = 24.609356
    DEC-APP = 33.225975
    RA-MEAN = 24.609356
    DEC-MEAN= 33.225975
    OBSERVER= PKT CORR
    PROJECT = PKT_CORR
    Code    = PKT_CORR
    ANTMASK = 3fffffff
    BANDMASK= 3
```

```
        SEQ      = 0
        MJD_SRC = 56320.000000
        DRA      = 0
        DDEC     = 0
        RF       = 1299000000   1299000000
        FIRST_LO= 1350000000   1350000000
        BB_LO    = 51000000  51000000
```

# RA-APP, Dec-APP, RA-MEAN, DEC-MEAN, MJD_SRC, RF, FIRST_LO, BB_LO, can be found out from parallel recording on GSB. by 'more lta' file.

4. edit gsb.hdr for

```
        GSB_ACQ_BW   = 400.0      /* 16.666 or 33.333              */
        GSB_CHAN_MAX = 512          /* 256/512
*/
```

# other parameters to be left as default in gsb.hdr.

4. execute raw-lta conversion by following command. In a given directory.

> ./scIta -i $DATA_DIR/3c48.raw -l $DATA_DIR/3c48.lta -a antsys.hdr -s sampler.hdr -c corrsel.hdr -S scan.hdr -g gsb.hdr -b 0

$DATA_DIR is a place where raw/lta data is kept.

# -b 0 option leave as it is. (data recorded on 32 bit m/c and analysis to be on 32 bit m/c).

5. $ROOT_DIR/gvfits_pkt.working/listscan 3c48.lta
# as per standard procedure.
6. $ROOT_DIR/gvfits_pkt.working/gvfits 3c48.log
# as per standard procedure.

APPENDIX-II

iADC Characterization Report

Sandeep C. Chaudhari,
Kaushal D. Buch
S. Harshvardhan Reddy

iADC sampler card is a part of ROACH-I based upgrade correlator which is being used to digitize the broadband analog signals. The characterization of iADC is important in order to decide the stable operating range of FPGA back-end. This will be useful in deciding the headroom for accommodating RFI.

iADC Specifications :-

AT84AD001C Dual 8-bit 1 Gsps ADC

Vpp	= 500mV	
Bits	= 8-bits	
lsb	= Vpp/2^8	= 1.953mV
Full power Input Bandwidth	= 1.5 GHz (–3 dB)	

For Each bit the power level required is as follows :-
Formula = [(Vpp x 2^n) / 2^8] + LSB/2

Based on above specifications, the theoretical input power required for setting each individual bit is as shown in the table below :-

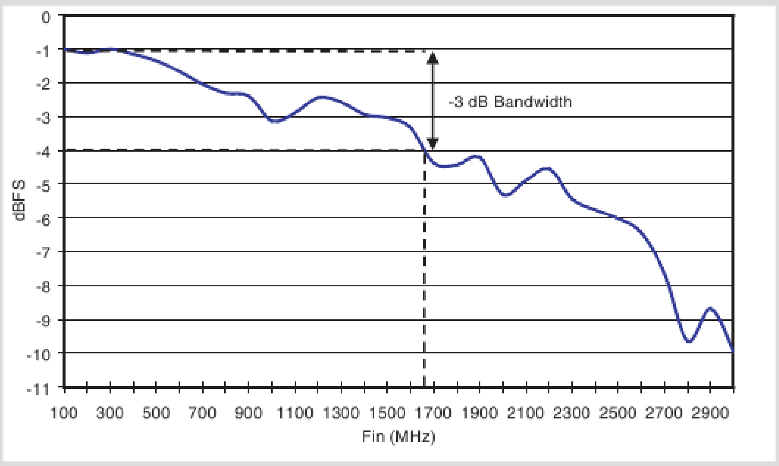
Bits	**Voltage (in mV)**	**Power in (dBm) (Bitwise power)**
0000 0010	2.93	-46.684174
0000 0100	6.836	-39.324638
0000 1000	14.648	-32.704774
0001 0000	30.273	-26.399365
0010 0000	61.523	-20.239788
0100 0000	124.023	-14.150525
1000 0000	249.023	-8.095795
[1]0000 0000	499.023	-2.058181

iADC Board Losses :-

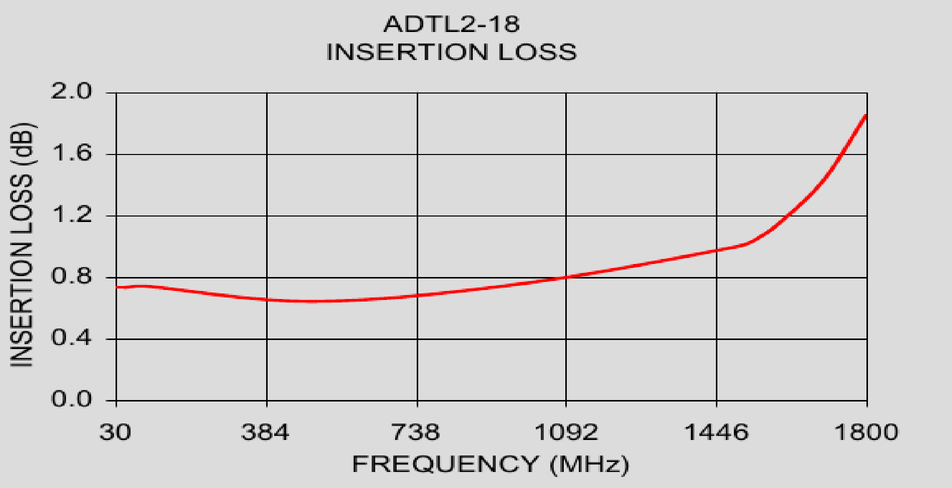
The iADC board losses are because of two components as mentioned below :-

1. ADC analog input bandwidth response is an important factor (Typical ~ 1.5 dB)

Figure 9-1. Full Power Input Bandwidth



2. RF Transformer :-
   ADLT2-18 surface mount RF Transformer 50Ω 30 to 1800MHz typical characteristics :-



Incurred insertion loss ~ 0.8 dB

iADC Characterization Test Setup :-

The characterization has been carried out using lab noise source and sine wave generator.

Instruments :-

1. Micronetics Wireless Noise Generator
2. Sine wave Generator – Agilent N9310A
3. Spectrum Analyzer – Rhode & Schwarz FSP (9kHz to 7GHz)
4. 4-Antenna Packetized Correlator F-engine & Corresponding python script.

iADC Board test with noise source & sine wave :-

Characterization of iADC with sine wave can gives us a good perspective of behavior for the fundamental component of fourier transform i.e. sine wave. But characterization with noise source resembles in behavior to that of antenna signals – random in nature. Hence power of a noise signal is nothing but it’s standard deviation. The spread of the distribution (in terms of standard deviation) decides the number of bits exercised in an ADC.

1. Sine wave test :- Saturated at Pin = 0.5 dBm@1.5GHz analog bandwidth of ADC
2. Noise source test :- Saturated at Pin = -16 dBm@1.5GHz analog bandwidth of ADC

Standard Deviation Test Using Noise Source:-

Design Test Setup :-

1. Design Tested :-
   Packetized Correlator–F-engine (r_128w_512_11_r370_mod3_1_2011_Nov_29_1610.bof)
2. Testing & plotting script : “corr_adc_time.py”
   The design has been modified to make it compatible with “corr_adc_time.py” script.
   This test is carried out to measure +/-3σ spread of input power level.

gmrt@rchpc3:~$ corr_adc_time.py -t 100000 -a 0x

Bits Utilized	3σ (Therotical)	3σ (Measured/adjusted)	Pin @1.5GHz (in dBm) with losses
6 to 7 bits	(64 to 127)	115 to 122	-11.09
5 to 6 bits	(32 to 63)	61 to 64	-16.62
4 to 5 bits	(16 to 31)	30 to 32	-22.54
3 to 4 bits	(8 to 15)	14.76 to 15.40	-28.54
2 to 3 bits	(4 to 7)	7.21 to 7.50	-34.95
1 to 2 bits	(2 to 3)	3.29 to 3.45	-42.06
0 to 1 bits	(0 to 1)	1.65	-50.30

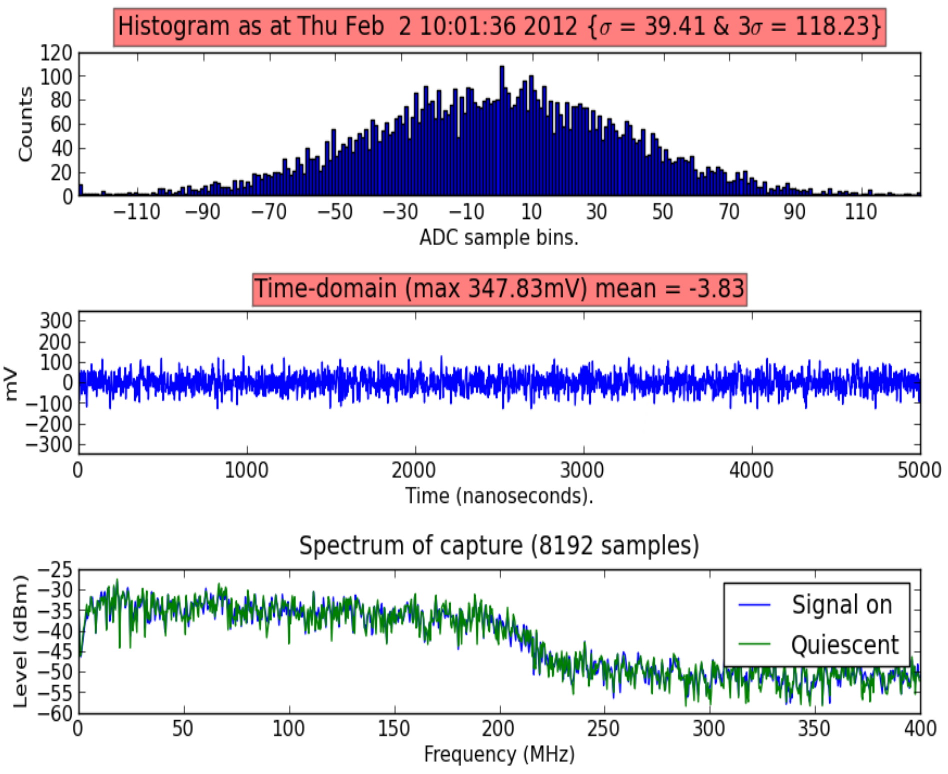


Fig 7: iADC 6 to 7-bits utilization Histogram

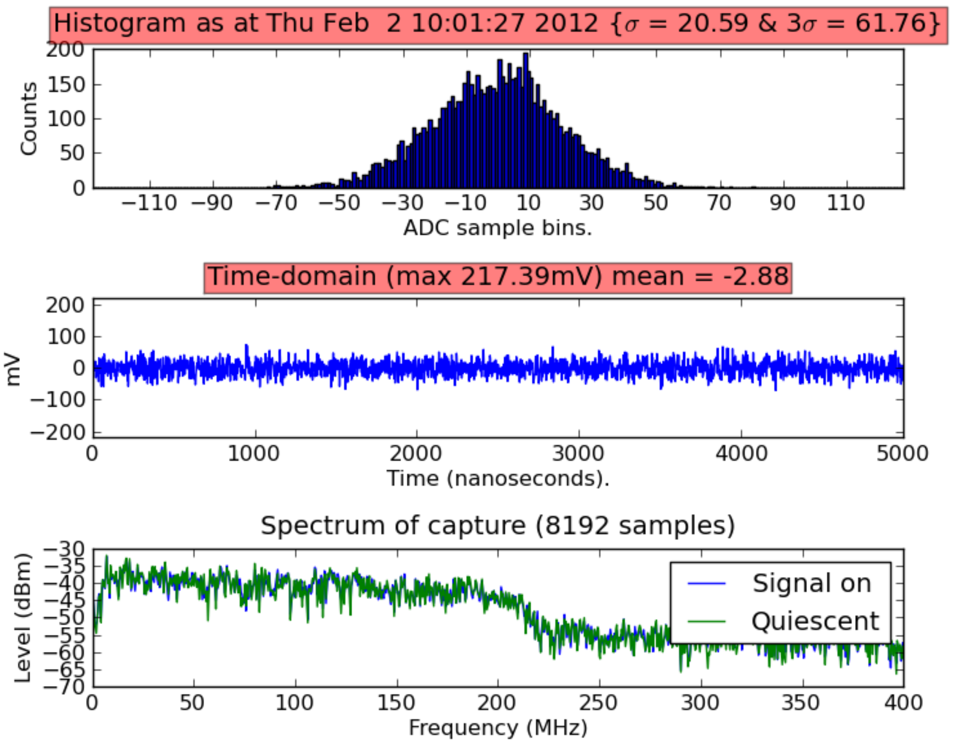


Fig 8: iADC 5 to 6-bits utilization Histogram

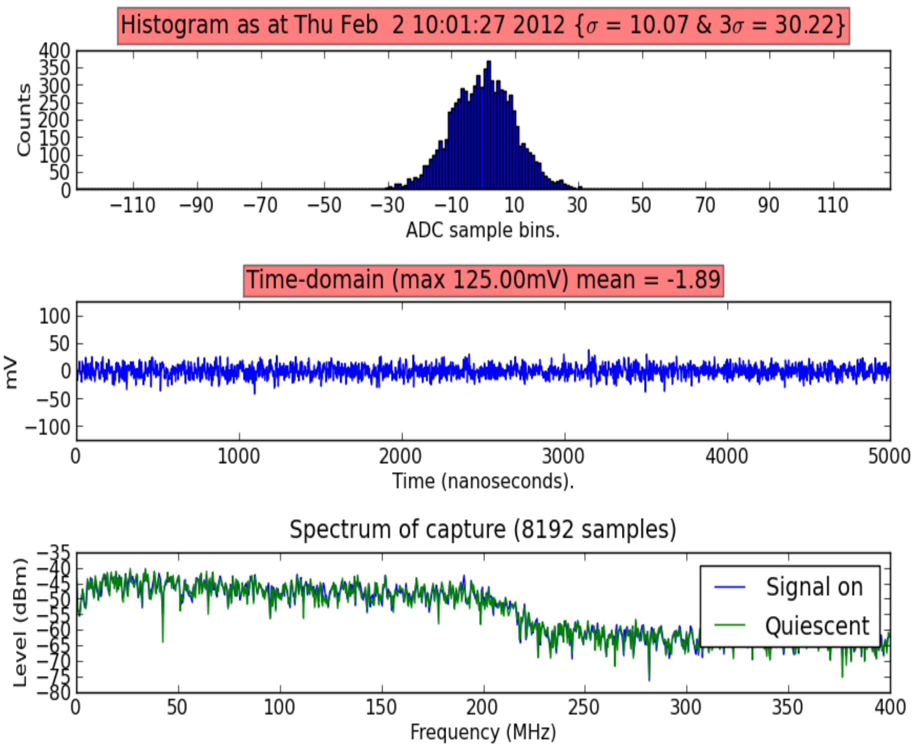


Fig 9: iADC 4 to 5-bits utilization Histogram

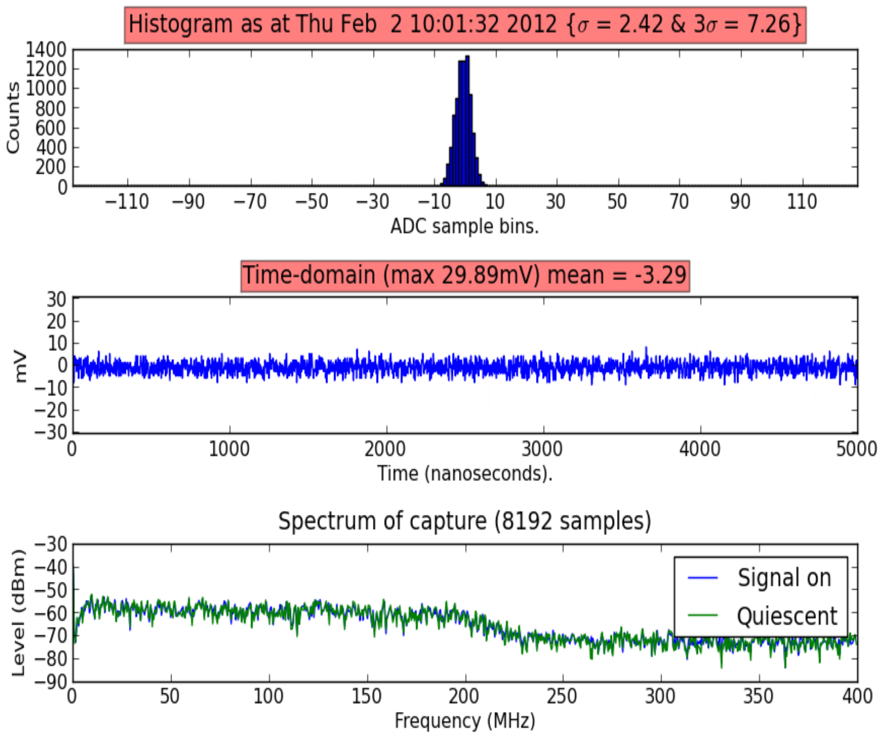


Fig 10: iADC 4 to 3-bits utilization Histogram

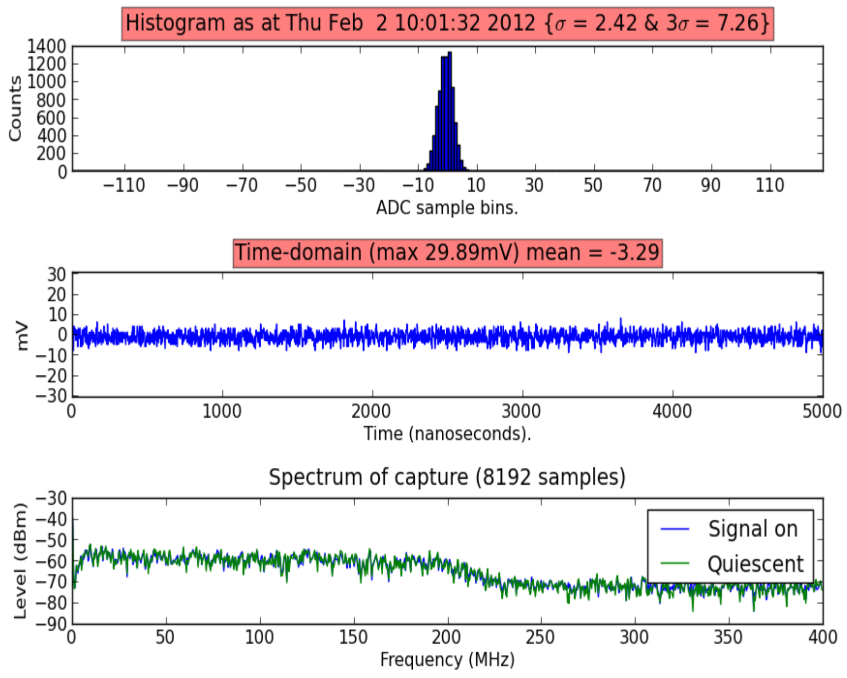


Fig 11: iADC 3 to 2-bits utilization Histogram

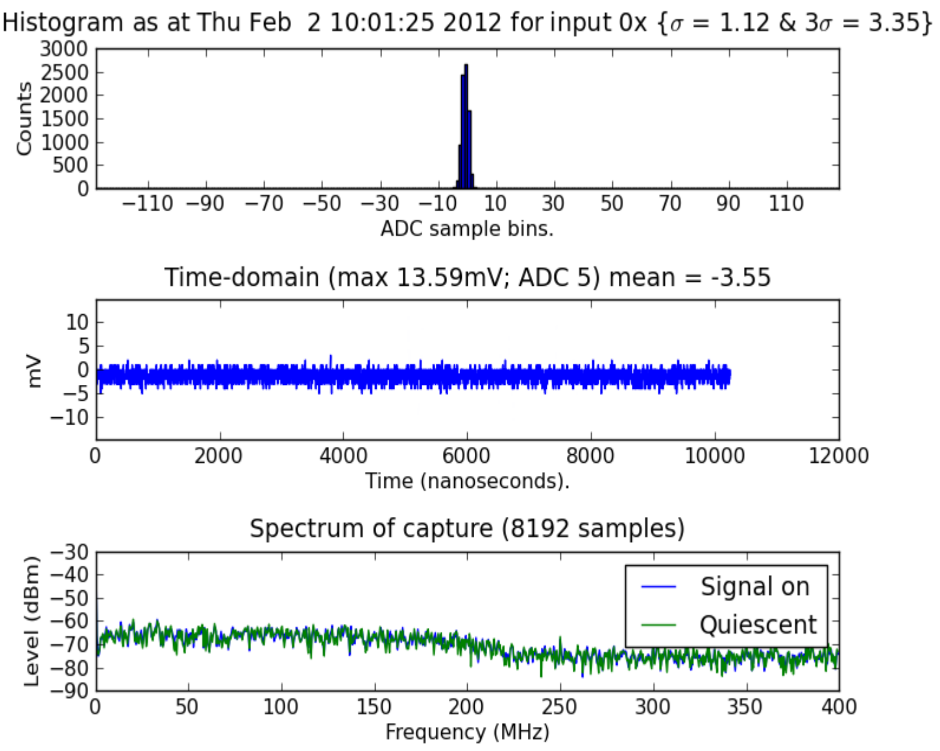


Fig 12: iADC 2 to 1-bits utilization Histogram

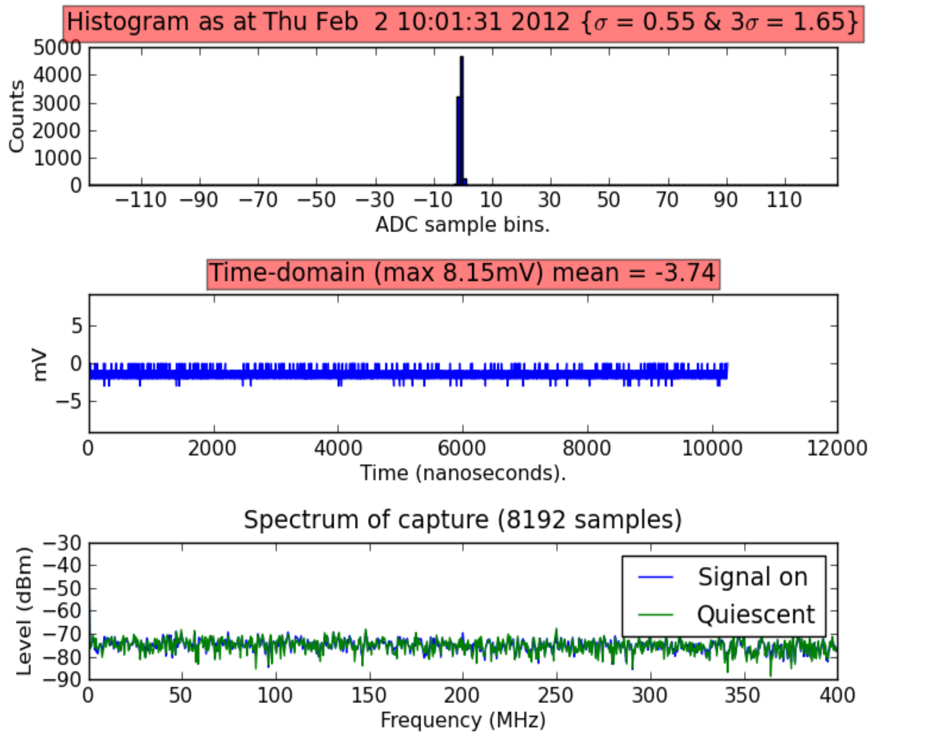


Fig 13: iADC 0 to 1-bits utilization Histogram

Conclusion :-

iADC characterization test were carried out for sine wave and noise inputs to determine range of operation of ADC for stable operation of digital back-end. After providing 1-bit headroom for RFI, the optimal noise input power over the analog bandwidth of ADC should be in the range of (-16.7dBm to -22.5dBm).

The ADC saturates at -11.2dBm power level over analog bandwidth of ADC.

References :-

1. AT84AD001C ADC Datasheet
2. ADLT2-18 RF transformer Datasheet

Version History :-

1. Version 1 : Initial Version 2nd February 2012
2. Version 2 : Details about test setup and conclusion 30th March 2012

References :-

1. Jason Manley Ph.D. Thesis selected output, “Packetized correlator”
2. Sandeep C. Chaudhari, Kaushal D. Buch, Harshvardhan Reddy, “iADC characterization”
3. Kaushal D. Buch, Sandeep Chaudhuri and Sanjay Kudale , “Optimization of on-chip memory for coarse delay correction ”
4. Dhiraj Tambade “PPS Generation on Serial Port using Python and Processing It ”