PROJECT REPORT
ON

TESTING AND CHARACTERIZING WALSH SYSTEM FOR GMRT

Submitted By
Pranita Sante
Student Training Program -2016
(10th February 2016 – 09th August 2016)

Under the Guidance of

Mr. Ajith Kumar B.
Engineer-F
Mr. Sandeep C. Chaudhari
Engineer-D

GIANT METREWAVE RADIO TELESCOPE,

NCRA-TIFR

CERTIFICATE

This is to certify that Miss. Pranita Sante has successfully completed the project titled **"Testing and Characterization of Walsh system for GMRT"** as a part of Summer Training Program (STP) - 2016 at the Giant Metrewave Radio Telescope, Khodad. This report is submitted as fulfilment of STP-2016 program.

Place: GMRT, Khodad
Date:

Mr. Sandeep C. Chaudhari
(Guide)

ACKNOWLEDGEMENT

It has been a great honor and privilege to undergo training at Giant Metrewave Radio Telescope. I am very much thankful to my guides, Mr. Ajithkumar and Mr. Sandeep Chaudhari for providing me this great opportunity to work in such a prestigious place under his efficient guidance. I executed this project with his constant guidance and willingness to share his vast knowledge made me to understand this project and it's manifestation I great depth which helped me to complete the assigned task. I am also thankful to him for providing all facilities and support to meet the project requirements.

I would like to express my deep gratitude towards Mrs. N. S. Deshmukh, STP Co-ordinator, GMRT for offering me help to make my stay at GMRT more comfortable.

I would like to take this opportunity to express my humble gratitude to Mr. Shrihari Gadge for helping me at various points of time and solving my difficulties. I feel delighted to express my sincere thanks to MR. Aniket Hendre and Mrs. Sweta Gupta for providing me guidance and help through various stages of my project.

Finally, I would like to thank all those who helped me directly or indirectly during this project.

Pranita Sante

ABSTRACT

The purpose of this project was to test characterize and debug Walsh modulation/demodulation system for GMRT. In GMRT there are 30 antennas with dual polarization channels. There are strong possibilities of spurious coupling of signals from one to the other signal chains at various locations. This can cause a spurious correlation between the final signals from these different antennas. This cross talk can be reduced using Walsh scheme.

To implement Walsh scheme in antennas it is very important to characterize it thoroughly in the lab for cross talk reduction and correlation loss. Lab simulation of the scheme with noise source and then with antenna signals were carried out.

While doing the same we developed a standard operating procedure (SOP) for debugging of Walsh at antenna.

Contents

Chapter 1. Introduction 1-1
1.1 Introduction to GMRT 1-1
1.2 Signal Flow Chain of a Radio Receiver 1-1
1.3 Signal Flow Chain of GMRT:- 1-2
Chapter 2. Hardware and Software Requirement 2-4
2.1 Hardware requirements: 2-4
2.1.1 ROACH (Reconfigurable Open Architecture Computing Hardware) 2-4
2.1.2 CPLD Walsh Card 2-4
2.2 Software Requirement: 2-5
Chapter 3. Walsh Scheme 3-6
3.1 Introduction to Walsh Scheme 3-6
3.2 Walsh Generation Scheme 3-6
3.3 Walsh Switching at Front end 3-6
3.4 Walsh Modulator & Facilities 3-7
3.5 Walsh Demodulator & Facilities 3-8
3.6 Requirement of Walsh Synchronization 3-10
Chapter 4. Lab Simulation Tests 4-11
4.1 Initial requirements for the test 4-11
4.2 Testing Walsh modulation with CW signal 4-11
4.3 Delay Hunt Test: 4-12
4.3.1 Test Setup for Delay Hunting Test: 4-13
4.3.2 Test Results 4-13
4.4 Effect of Walsh on Cross Talk 4-14
4.4.1 Test Setup 4-14
4.4.2 Test Results 4-14
4.5 Correlation Loss Test due to Walsh with Noise 4-16
4.5.1 Test Setup 4-16
4.5.2 Test Results 4-16
4.6 Correlation Loss Test due to Walsh with Sine Wave 4-18
4.6.1 Test Setup 4-18
4.6.2 Test Results 4-18
Chapter 5. Design Modification for Basic Walsh Pattern 5-21
5.1 Basic Walsh and Its Need 5-21
5.2 Modifications from Modulator-Demodulator Side 5-21
5.3 Modified Delay Hunting Test 5-21
5.4 Debugging the System 5-22
5.4.1 Delay Hunting Failure 5-22

5.4.2 Debugging: Stability 5-23
5.4.3 Debugging C12 Antenna Walsh Channel Swap Test 5-24
5.5 Antenna Testing 5-25
5.5.1 Results of Antenna Testing 5-25
Chapter 6. Conclusion and Future Scope 6-1
Chapter 7. References 7-2
Chapter 8. Appendix-A: Development of Python Packaging 8-3
8.1 Introduction to Python 8-3
8.2 Scripts 8-3
8.3 Modules 8-5
8.4 Packages 8-6
8.5 Python Scripts for Walsh Testing 8-8
Chapter 9. Appendix – B: SOP for Lab Testing 9-14
9.1 Delay Hunt using Basic Walsh Pattern: 9-14
9.2 Correlation loss test due to Walsh Modulation/Demodulation: 9-17
Chapter 10. Appendix – C: SOP for Antenna Testing 10-18
10.1 Antenna geometric delay adjustment 10-18
10.2 Walsh Delay adjustment using Basic Walsh Pattern 10-19
10.3 Find Correlation Loss 10-21
10.4 SOP for Analysis of Data in Tax Format 10-22

Figure 1 : Block diagram of a single dish telescope _____ *1-2*
Figure 2: Schematic diagram of GMRT receiver chain. _____ *1-2*
Figure 3: ROACH Board. _____ *2-4*
Figure 4: CPLD based Walsh PIU (MCM-4) _____ *2-4*
Figure 5: Schematic diagram of existing Front-End system _____ *3-6*
Figure 6: Block diagram of CPLD based Walsh card _____ *3-7*
Figure 7: CPLD Walsh PIU (D49: MCM-4) _____ *3-7*
Figure 8: Pin diagram of antenna selection DIP switch _____ *3-8*
Figure 9: List of commands used to set Walsh frequency division _____ *3-8*
Figure 10: List of commands used for Walsh testing _____ *3-8*
Figure 11: Test setup for Walsh modulation on Pocket Correlator _____ *4-11*
Figure 12: Walsh pattern plot WP-1 and WP2 from top _____ *4-11*
Figure 13: Snap of CW signal without Walsh bit = 0 _____ *4-12*
Figure 14: Snap of CW signal with Walsh bit = 1 _____ *4-12*
Figure 15: Test setup for delay hunting _____ *4-13*
Figure 16: Walsh delay hunting program plot with zoom1 and zoom2 for WP-1 & WP-2 _____ *4-13*
Figure 17: Lab test setup to simulate cross talk in receiver chain _____ *4-14*
Figure 18: Lab Test - Cross talk reduction due to Walsh pattern-1 _____ *4-15*
Figure 19: Lab Test - Cross talk reduction due to Walsh pattern-2 _____ *4-15*
Figure 20: Lab Test - Cross talk reduction due to both Walsh patterns _____ *4-15*
Figure 21: Test setup to check correlation loss due to Walsh _____ *4-16*
Figure 22: Loss of correlation due to Walsh pattern-1 with NG _____ *4-17*
Figure 23: Loss of correlation due to Walsh pattern-2 with NG _____ *4-17*
Figure 24: Loss of correlation due to both Walsh pattern with NG _____ *4-17*
Figure 25: Test setup for correlation loss test using sine wave. _____ *4-18*
Figure 26: Correlation loss test due to Walsh pattern-1 on CW signal _____ *4-19*
Figure 27: Correlation loss test due to Walsh pattern-2 on CW signal _____ *4-19*
Figure 28: Correlation loss test due to both Walsh patterns on CW signal _____ *4-19*
Figure 29: Loss of correlation for No Walsh, Walsh pattern-1, Walsh pattern-2, Both Walsh patterns _____ *4-20*
Figure 30: Walsh delay hunting with basic Walsh pattern _____ *5-22*
Figure 31: Basic Walsh pattern hunting delay applicable to different antenna Walsh pattern _____ *5-22*
Figure 32: Inverted Walsh patterns from CPLD and FPGA causing delay hunting failure _____ *5-23*
Figure 33: Stability plot to ensure locking of clocks at modulator & demodulator side _____ *5-24*
Figure 34: Walsh channel swap test plot _____ *5-25*
Figure 35: Loss of correlation due to Basic Walsh pattern _____ *5-26*
Figure 36: Loss of correlation due to Walsh pattern-1 _____ *5-26*
Figure 37: Loss of correlation due to Walsh pattern-2 _____ *5-27*
Figure 38: Loss of correlation for No Walsh, Both Walsh Patterns, Walsh Pattern-1, Walsh Pattern-2 _____ *5-27*

Chapter 1. Introduction

1.1 Introduction to GMRT

NCRA (National Centre for Radio Astrophysics) has set up a unique facility for radio astronomical research using the metre-wavelengths range of the radio spectrum, known as the **Giant Metrewave Radio Telescope (GMRT)** located at a site about 80 km north of Pune. GMRT consists of 30 fully steerable gigantic parabolic dishes of 45m diameter each spread over distances of up to 25 km. GMRT is one of the most challenging experimental programmes in basic sciences undertaken by Indian scientists and engineers.

The meter wavelength part of the radio spectrum has been particularly chosen for study with GMRT because man-made radio interference is considerably lower in this part of the spectrum in India. The number and configuration of the dishes was optimized to meet the principal astrophysical objectives which require sensitivity at high angular resolution as well as ability to image radio emission from diffuse extended regions. Fourteen of the thirty dishes are located more or less randomly in a compact central array in a region of about 1 square km. The remaining sixteen dishes are spread out along the 3 arms of an approximately Y-shaped_configuration over a much larger region, with the longest interferometric baseline of about 25 km.

The multiplication or correlation of radio signals from all the 435 possible pairs of antennas or interferometers over several hours will thus enable radio images of celestial objects to be synthesized with a resolution equivalent to that obtainable with a single gigantic dish 25Km in diameter. The array will operate in six frequency bands centered around 50, 153, 233, 325, 610 and 1420MHz. All these feeds provide dual polarization outputs. In some configurations, dual-frequency observations are also possible.

1.2 Signal Flow Chain of a Radio Receiver

Radio waves from the distant cosmic source impinge on the antenna and create a fluctuating voltage at the antenna terminals. This voltage varies at the same frequency as the cosmic electromagnetic wave, referred to as the Radio Frequency (RF). The voltage is first amplified by the front-end (or Radio Frequency) amplifier. The signal is weakest here and hence it is very important that the amplifier introduce as little noise as possible.

Front end amplifiers hence usually use low noise solid state devices. After amplification, the signal is passed into a mixer. A mixer is a device that changes the frequency of the input signal. Mixers have two inputs, one for the signal whose frequency is to be changed (the RF signal in this case), and the other input is usually a pure sine wave generated by a tunable signal generator, the Local Oscillator (LO). The output of the mixer is at the beat frequency of the radio frequency and the local oscillator frequency. So after mixing, the signal is now at a different (and usually lower) frequency than the RF, this frequency is called the Intermediate Frequency (IF). The main reason for mixing (also called

heterodyning) is that though most radio telescopes operate at a wide range of radio frequencies, the processing required at each of these frequencies is identical.

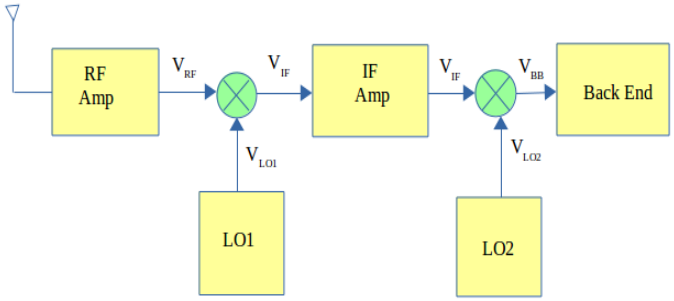

Figure 1 : Block diagram of a single dish telescope

The economical solution is to convert each of these incoming radio frequencies to a standard IF and then to use the same back-end equipment for all possible RF frequencies that the telescope accepts. In telescopes that use co-axial cables to transport the signal across long distances, the IF frequency is also usually chosen so as to minimize transmission loss in the cable. Sometimes there is more than one mixer along the signal path, creating a series of IF frequencies, one of which is optimum for signal transport, another which is optimum for amplification etc. This is called a 'super-heterodyne' system. After conversion to IF, the signal is once again amplified (by the IF amplifier), and then mixed to a frequency range near 0 Hz (the Base Band (BB)) and then fed into the backend for further specialized processing. What backend is used depends on the nature of the observations.

1.3 Signal Flow Chain of GMRT:-

The GMRT accepts radio waves in six bands from 50 MHz to 1.4 GHz and has IFs at 130MHz, 175MHz and 70MHz. The GMRT receiver chain is shown schematically in *Figure 2*.

The first block is the multi-frequency front-end. This is located in a rotating turret at the prime focus. All the feeds and low noise RF front-ends have been configured to receive dual polarization signals. Lower frequency bands (from 50 to 610 MHz) have dual circular polarization channels, i.e. left circular and right circular polarizations which have been

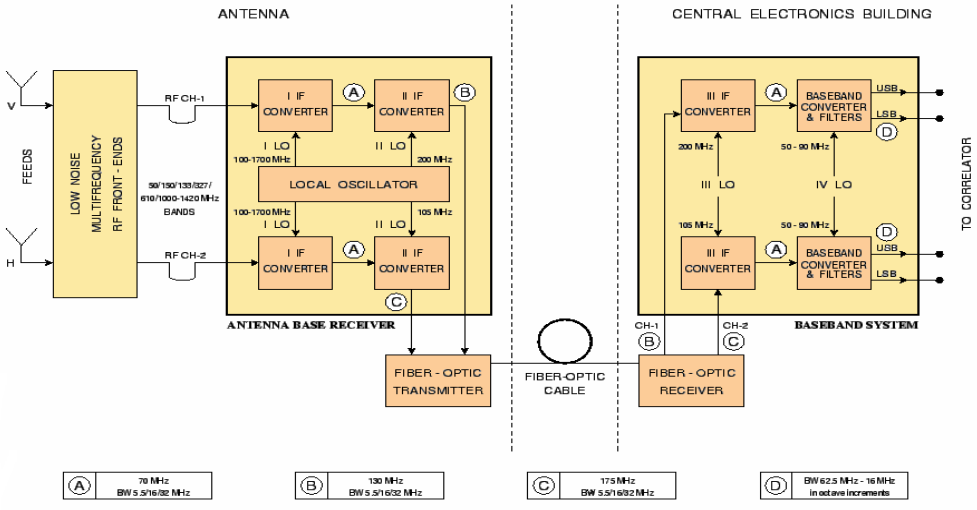

Figure 2: Schematic diagram of GMRT receiver chain.

labeled as CH1 and CH2 respectively. The L-band (1000-1450 MHz) system has dual linear

polarization channels, i.e. vertical and horizontal polarizations (also labeled CH1 and CH2 respectively).

The first local oscillator (I^{st} LO, situated at the base of the antenna, inside a shielded room) converts the RF band to an IF band centered at 70 MHz. After passing the signal through a bandpass filter of selectable bandwidth, the IF at 70 MHz is then translated (using II LO) to a second IF at 130 MHz and 175 MHz for CH1 and CH2, respectively.

The maximum bandwidth available at this stage is 32 MHz for each channel. This frequency translation is done so that signals for both polarizations can be frequency division multiplexed onto the same fiber for transmission to the Central Electronics Building (CEB). The IF signal at 130 MHz and 175 MHz along with telemetry and LO round trip phase carriers directly modulate a laser diode operating at 1300nm antennas and the CEB.

At the CEB, these signals are received at 'nm' wavelengths which are coupled to a single mode fiber-optic link by the Fiber-Optic Receiver and the 130 and 175 MHz signals are then separated out and sent for base band conversion. The baseband converter section converts the 130 and 175 MHz IF signals first to 70 MHz IF (using III LO), these are then converted to upper and lower sidebands (each at most 16 MHz wide) at 0 MHz using a tunable IV LO. There are also two Automatic Level Controllers (ALCs) in the receiver chain.

There are a variety of digital Back-Ends available at the GMRT. The principal backend used for interferometric observations is a 32 MHz wide FX correlator. The FX correlator produces a maximum of 512 spectral channels for each of two polarizations for each baseline.

All control and monitoring commands to the system are sent from the central computer through a telemetry system. The telemetry system provides a Measurement, Control and Monitor (MCM) circuit for sending digital control data to the circuits and for acquiring analog monitor data.

Chapter 2. Hardware and Software Requirement

2.1 Hardware requirements:

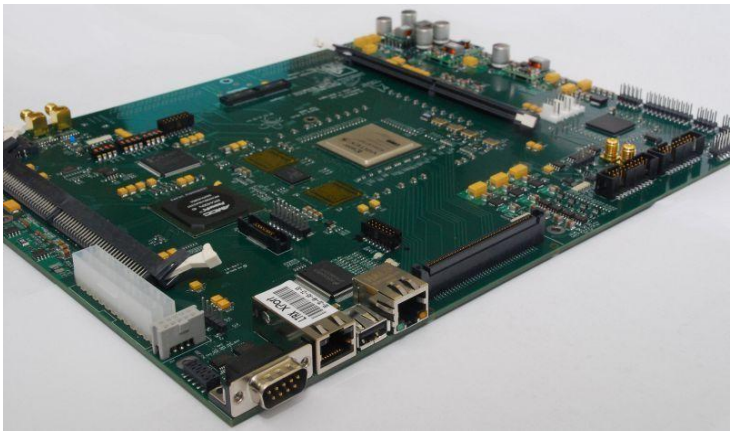

Figure 3: ROACH Board.

2.1.1 ROACH (Reconfigurable Open Architecture Computing Hardware)

ROACH is a standalone FPGA processing board built around Virtex-5 (SX95T) FPGA and developed by CASPER (Collaboration for Astronomical Signal Processing and Engineering Research) community. It is an upgrade to current CASPER hardware and merges aspects from the iBOB and BEE2 platforms into a single board. The ROACH board consists of various peripherals including PPC440EPx as embedded processor that acts as an interface between FPGA design and programmer. Virtex-5 FPGA of ROACH boards have sufficient BRAMs to cater GMRT's maximum geometric delay correction need. iADC board is a 1x Atmel/e2V AT84AD001B 8-bit dual 1Gsps, with clock 10MHz – 1GHz 50Ω 0dBm is used to sample analog signal from GMRT antennas.

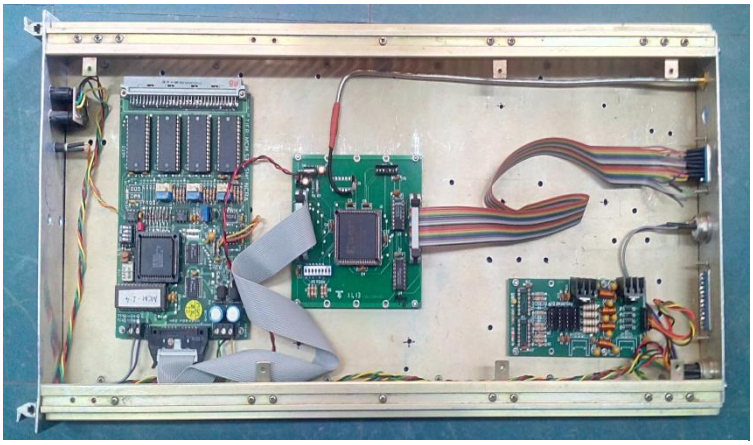

Figure 4: CPLD based Walsh PIU (MCM-4)

2.1.2 CPLD Walsh Card

CPLD based Walsh card is put in D-49 PIU (Plug-In-Unit) which is located at the ABR in the LO Synthesiser section, controls the selection of various patterns at the ABR which will be combined with the RF signals at the front-end and control of Front-End MCM (MCM-5), as per the online commands issued by the user.

CPLD chip in a Walsh card is Xilinx XC95108-15 PC84C high-performance CPLD providing advanced in-system programming and test capabilities for general purpose logic integration. It is comprised of 6 function blocks of 18 macrocells (total of 108 macrocells) providing 2,400 usable gates with propagation delays of 7.5 ns and is available in 84 pin PLCC package.

2.2 Software Requirement:

- MSSGE Toolflow – Matlab Simulink System Generator Toolflow
- Ubuntu Linux 15.10
- Python 2.7 or above & packages

Chapter 3. Walsh Scheme

3.1 Introduction to Walsh Scheme

The signal flow chain of radio telescope receiver consists of various sub-systems viz. Feeds, Front-End electronics, signal conditioning circuits, signal transportation to central station, Baseband conversion, and digital back-end receivers. There is a possibility of signal coupling at various locations especially at the central station where the signals from all antennas are processed together. This can cause a spurious correlation between baseband signals from these different antennas. This leakage is referred to as cross talk. Since a Radio Telescope is a very sensitive instrument, such cross talk can seriously affect its usefulness and capabilities.

A suitable method for reducing the effect of such cross talk is use of Walsh modulation-demodulation scheme. The suggested scheme utilises phase modulating the received signals using ortho-normal patterns directly at the antenna site and demodulate them just before combining the signals from antennas at the Digital Backend.

In GMRT there are 30 antennas with two polarization channels from each antenna providing RF signals. They are modulated using different Walsh patterns – we need 64 independent patterns. So 128 bit Walsh patterns are used. These patterns are generated using a CPLD based circuit at antenna base and send to the prime focus in differential mode. The pattern is used to phase modulate the signal between 0/180 degrees. At the Central station same pattern is used for demodulation with appropriate delay to match with the input modulating pattern.

3.2 Walsh Generation Scheme

IF signals from each GMRT antenna will be modulated with a Walsh function in order to reduce the effect of coupling between the different channels. They are modulated using a set of 64 independent, 128 bit Walsh patterns. For modulation a CPLD based Walsh card circuit is used to generate various Walsh patterns at ABR (64: Cal patterns and 64: Sal patterns). This Walsh modulated RF signals from all antennas are then transferred via. Broad-band fibre optics system to CEB(Central Electronics Building) where they gets converted into baseband signals using a GAB (GMRT Analog Backend) system and digitized in Digital Back-End system. FPGA based Walsh demodulation is done at correlator in ROACH board where digitization occurs. In this scheme the generation of Walsh pattern in FPGA is same as in case of CPLD based Walsh modulation.

3.3 Walsh Switching at Front end

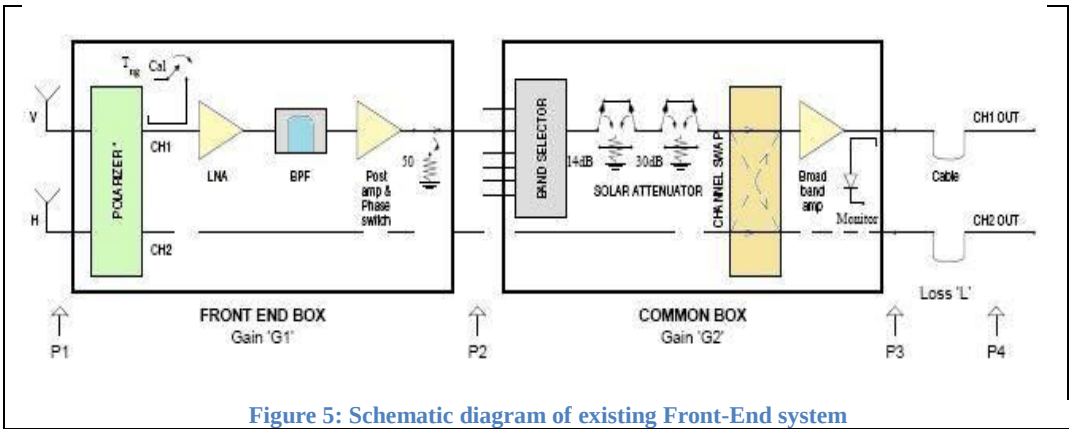

Figure 5: Schematic diagram of existing Front-End system

A phase switching facility using separate Walsh functions for each signal path is available at the RF section of the receiver. At the focus of each antenna, each feed has two low noise amplifiers (one for each polarization). These two signals go to a common box located on turret where the user can select which frequency signals appear at the output of the common box. The common box has facilities for the user to select solar attenuators (0, 14, 30 or 44 dB) and swap the two polarization channels. There is a bandpass filter after the LNA in the front end box which is followed by a post amplifier to have the required output power at the end of front end box. Thereafter the signal is modulated with Walsh function using phase switching to minimize the effect of crosstalk between different signals.

In correlator, exact reverse phase switching is done for each antenna so that the original phase is recovered just before the cross correlation is done. Such a scheme can greatly reduce the cross-talk at all points between the RF amplifier and the baseband.

3.4 Walsh Modulator & Facilities

The CPLD based Walsh generation card circuitry designed for GMRT consist of a programmable Walsh pattern generator that can generate the desired pattern dynamically as per user requirement instead of storing all the 128 Walsh and sequency pattern in advance thus reduce resources for operation.

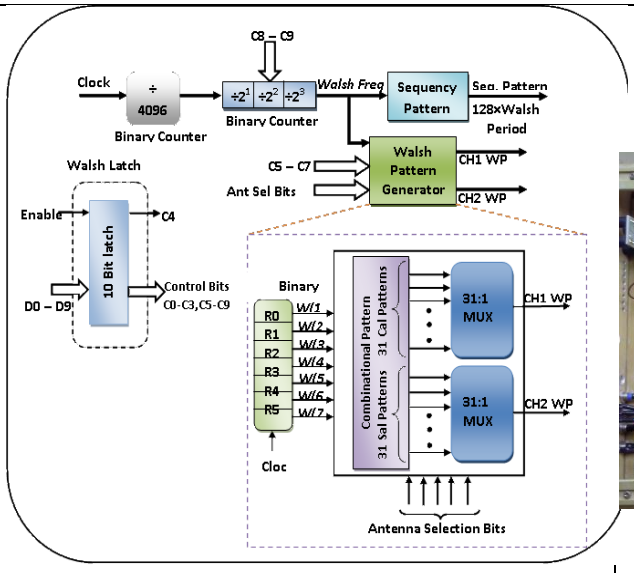

Figure 6: Block diagram of CPLD based Walsh card

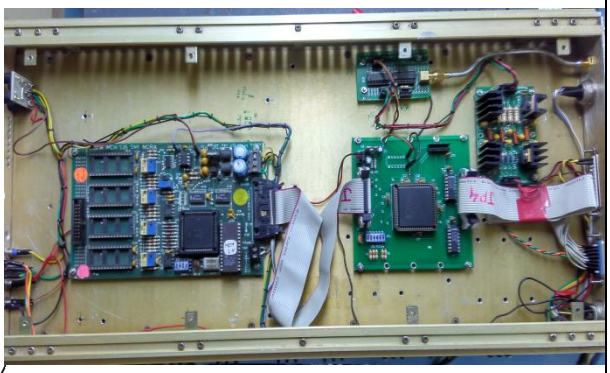

Figure 7: CPLD Walsh PIU (D49: MCM-4)

Walsh pattern on CPLD is generated using a 1MHz clock signal which is locked to GMRT Frequency and Time reference at each antenna. This clock is further divided by a 3-stage binary counter to fundamental Walsh frequency f_{walsh} = 1MHz/4096 = 244.140625Hz (4.096 mS) and this clock is used to generate Walsh patterns. Two 31:1 multiplexers are used for selecting Walsh patterns on individual channels. The select lines for these multiplexers are the antenna selection bits through Antenna selection switch.

- **Antenna selection facility:** A DIP-5 switch is used for selecting the antenna to be operated. From which we can select any antenna from 0 to 29, As follows:

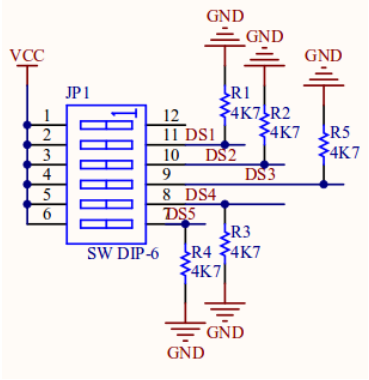

Figure 8: Pin diagram of antenna selection DIP switch

Antenna selection	NC	DS1 (MSB)	DS2	DS3	DS4	DS5 (LSB)
0	1	0	0	0	0	0
1	1	0	0	0	0	1
2	1	0	0	0	1	0
. .	. .	. .	. .	. .	. .	. .
29	1	1	1	1	0	1

- **Walsh Frequency Adjust Facility:** The complete 128 bits pattern for Walsh function and sequency may be needed for certain time duration depending on the need of user and system. Thus the circuit gives facility to select variable time period for each of the bits of these patterns. The user can select either a 4ms, 8ms, 16ms or 32ms duration time period each bit of every pattern as follows:

Functions	Hex Code
Enable Walsh frequency divide by 0 (4mS)	70 60 F0 60
Enable Walsh frequency divide by 2 (8mS)	71 60 F1 60
Enable Walsh frequency divide by 4 (16mS)	72 60 F2 60
Enable Walsh frequency divide by 8 (32mS)	73 60 F3 60

Figure 9: List of commands used to set Walsh frequency division

- **Independent channel selection facility:** The CPLD card has independent channel selection facility for taking the Walsh patterns output on any of the channels of an antenna where user can now have the provision to select independent channels of a particular antenna and add Walsh patterns on these channels as follows:

Functions	Hex Code
Enable wp-1 on ch-1 only	70 20 F0 20
Enable wp-2 on ch-2 only	70 40 F0 40
Enable wp-1 on ch-1 and wp-2 on ch-2	70 60 F0 60
Eanble wp-1 on ch-1 and ch-2	70 80 F0 80
Enable wp-2 on ch-1 and ch-2	70 A0 F0 A0
No Walsh	70 00 F0 00

Figure 10: List of commands used for Walsh testing

- **Sequency pattern generator:** Sequency pattern is generated after every 128 bits to indicate start of the Walsh pattern. It can be used to synchronized Walsh modulator and demodulator located at antenna and central station respectively.

3.5 Walsh Demodulator & Facilities

Walsh demodulation scheme is implemented in FPGA based ROACH-1 board where digitization occurs. In this scheme generation of Walsh pattern is done same way as in case of CPLD based Walsh modulation scheme. The scheme contains extra modules to implement Walsh demodulation scheme using Walsh delay hunting algorithm. The scheme consists of following modules:

1. Walsh Generation module: Generates 64 independent, 128 bit Walsh patterns.

2. Walsh Delay BRAM: Stores selected Walsh patterns as per antenna selection from user.
3. Integer Walsh delay module: Used by Walsh delay hunting algorithm for adjusting coarse Walsh delay so as to maximize normalized cross correlation.
4. Fractional Walsh delay module: It is used to adjust fine Walsh delay to further maximize normalized cross correlation.
5. Phase Switcher: Changes phase of digitized signals according to Walsh pattern to either 0 ◦ or180 ◦.

Each ROACH board F-engine (Frequency processing engine) contains Walsh demodulation module which can be independently configured for any given pattern using antenna selection bits and then stored 128-bits of Walsh into Walsh Delay BRAM during initialization. Sequence pulse is used to mark start of Walsh sequence and a marker for writing Walsh pattern into Delay BRAM only once. Once Walsh pattern gets stored in Walsh Delay BRAM then Walsh delay module (Integer & Fractional) is used to adjust delay so as to get maximum cross correlation.

Following facilities similar to CPLD Walsh modulator side are also implemented in FPGA modulator side as mentioned below:

- **Antenna selection facility:** User can select any antenna similar to that of CPLD Walsh card DIP switch. Here in design antenna Walsh pattern can be set through a software register inside FPGA design.
- **Walsh Selection facility from FPGA side :** Walsh pattern selection from FPGA can be done by providing multiplexer selection bit for each polarisation channel as follows:

MUX Selection	Functionality
0	Walsh Disabled
1	Polarisation-1 with Walsh Pattern -1 Sequency
2	Polarisation-1 with Walsh Pattern -1
3	Polarisation-1 with Walsh Pattern -2
4	Polarisation-2 with Walsh Pattern -1 Sequency
5	Polarisation-2 with Walsh Pattern -1
6	Polarisation-2 with Walsh Pattern -2
7	Basic Walsh Pattern with 50% duty cycle.
8	Walsh clock out

Table 1: Walsh selection on FPGA demodulator side

e.g. *> corr_inst.walsh_sel (walsh_sel= (2, 6))*

i.e Walsh pattern-1 on polarisation-1 & Walsh pattern-2 on polarisation-2.

Similarly we can give other combinations of Walsh Pattern to another polarization channel.

- **Frequency selection facility:** Walsh frequency division factor to be applied

Functions	MUX Selection
Enable Walsh frequency divide by 0 (4mS)	0
Enable Walsh frequency divide by 2 (8mS)	1
Enable Walsh frequency divide by 4 (16mS)	2
Enable Walsh frequency divide by 8 (32mS)	3

- **Additional facility of Walsh Delay Hunt:** This FPGA based Walsh demodulation scheme is also provides additional facility of Walsh delay adjustment that is required to match and align patterns at modulator and demodulator side.

3.6 Requirement of Walsh Synchronization

It is very important to ensure that the two independent Walsh function generator, one at antenna and one at CEB (central electronic building) are synchronized with each other. The time difference between the two Walsh functions when received at the CEB does not change by more than 10μs. Thus, if GMRT is prepared to tolerate a 1% loss in the sensitivity due to imprecise synchronization, the maximum synchronization error that can be tolerated in our case will be less than 0.0025T, where T is the highest Walsh frequency time period.

The synchronization between modulator and demodulator is achieved by disciplining their clocks with the GMRT Frequency & Time (F&T) standard. In case of modulator at antenna base the Walsh clock is derived from 1MHz reference frequency. At CEB the same clock has been generated from a clock source locked to 10MHz reference frequency from GMRT F&T standard.

Chapter 4. Lab Simulation Tests

4.1 Initial requirements for the test

Before starting of any test ensure the following:

a) Check signal generator status of ROACH based Pocket Correlator is locked to external 10 MHz reference of GMRT time and frequency standard.
b) Set the required power levels of clock to 0dBm and ADC inputs of Pocket Correlator as shown in below *Figure 11*.
c) All the instruments used for the tests should be highly stable.
d) Check whether 1 PPS signal is detected or not.

4.2 Testing Walsh modulation with CW signal

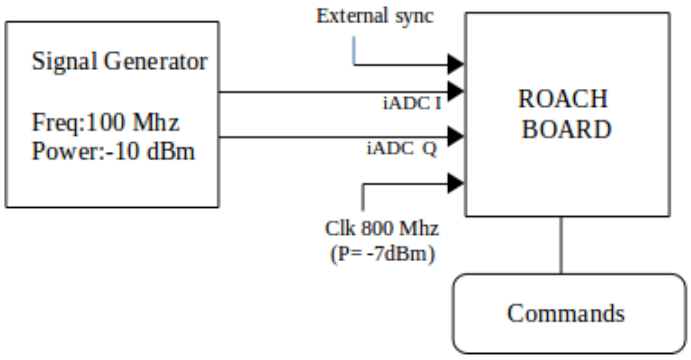

Figure 11: Test setup for Walsh modulation on Pocket Correlator

This test was intended to test the Walsh modulation block that changes phase of input signal to +/- 180° depending upon Walsh bits. The experiment was performed on ROACH based Pocket Correlator. An input of 100MHz sine wave with power level of -10dBm is given to Pocket Correlator. Below *Figure 12* shows a snap of Walsh patterns (wp1 & wp2) generated on FPGA based roach board applied to inputs of correlator.

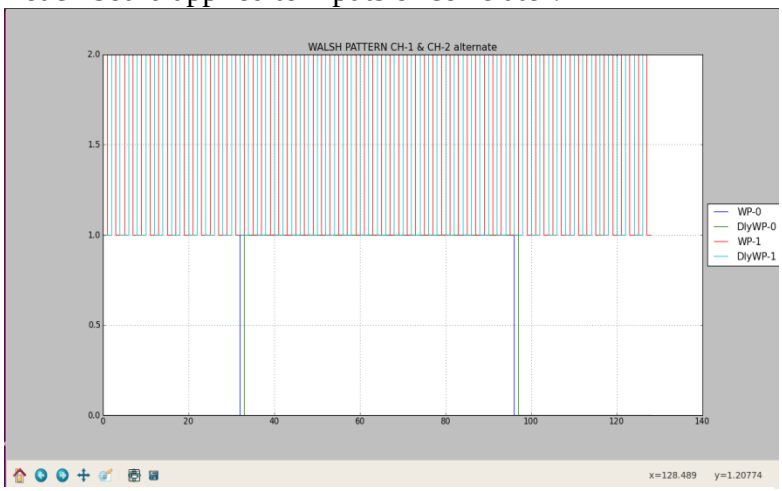

Figure 12: Walsh pattern plot WP-1 and WP2 from top

The plot in *Figure 13* is a digitized data snap which shows that there is no change in sine wave phase before (1st plot) and after (2nd plot) Walsh modulation when Walsh bit was 0 (3rd plot).

The plot in *Figure 14* is a digitized data snap which shows that there is 180° phase shift in sine wave before (1st plot) and after (2nd plot) Walsh modulation when Walsh bit was 1 (3rd plot).

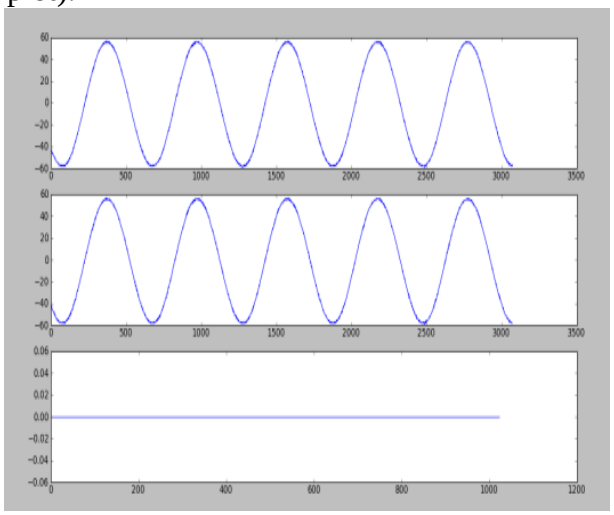

Figure 13: Snap of CW signal without Walsh bit = 0

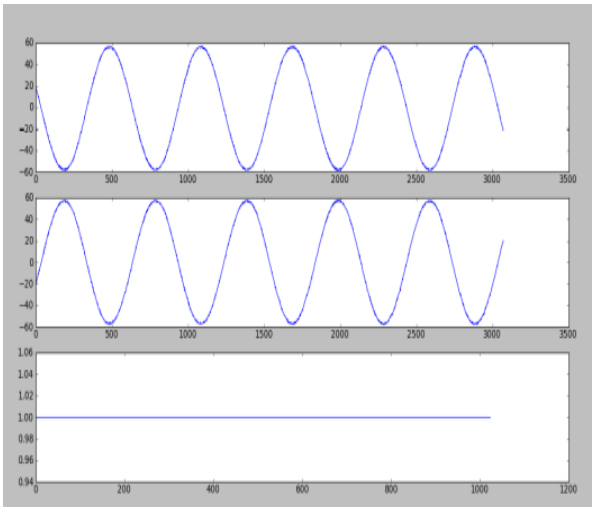

Figure 14: Snap of CW signal with Walsh bit = 1

4.3 Delay Hunt Test:

The aim of this test is to find Walsh delay between the modulator at antenna and demodulator at CEB in order to maximize cross correlation. For lab testing, give the common noise to both inputs of L-Band PA & PS and set the power level as shown in *Figure 15*. The output of the modulator is given to the input 'I' & 'Q' of Roach Board. Input power level of Roach Board should be between -15dBm to -17dBm. Give 1MHz clock with power level of +5dBm to Walsh PIU.

Initialise Pocket Correlator with delay hunting program. The algorithm first scans complete Walsh pattern period i.e. 524.288 mS with fastest Walsh frequency of 250Hz i.e. 4.096mS and then reduces hunting span & resolution accordingly in each iteration until normalized cross maxima reaches. It takes approximately ~7 minutes to find maxima. For the delay hunting algorithm to converge, the value of increment is restricted to below 8µS. The algorithm takes 7 minutes to converge and is well accurate up to 4th decimal place of normalized cross correlation value. For more detail about delay hunt algorithm go through reference [6].

4.3.1 Test Setup for Delay Hunting Test:

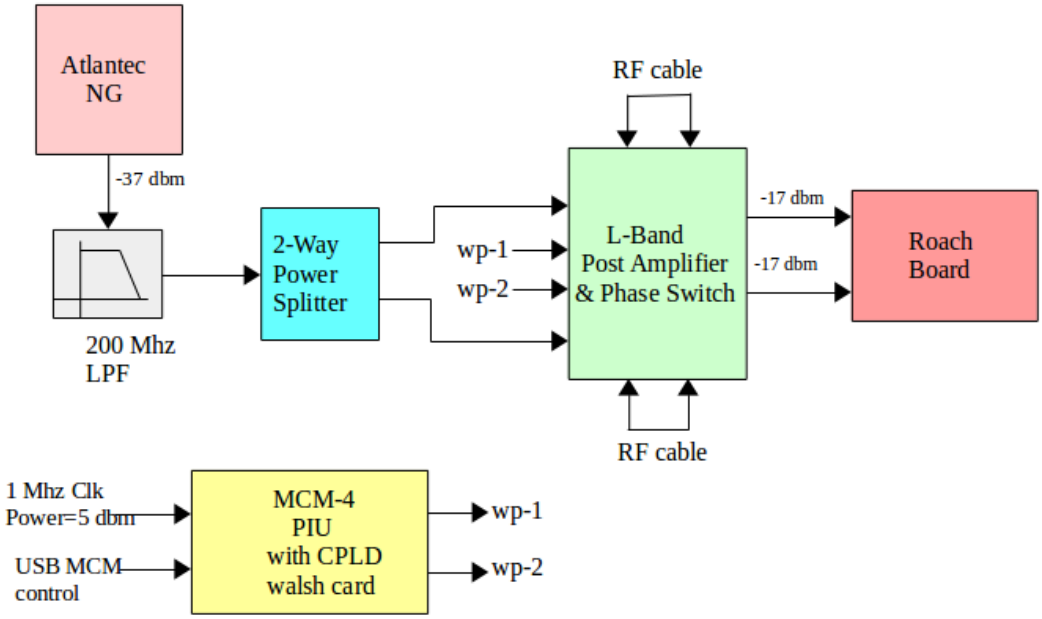

Figure 15: Test setup for delay hunting

4.3.2 Test Results

The Walsh delay hunting test was carried out using Walsh patterns of antenna number 7 where it scanned complete Walsh delay range in first round as shown in *Figure 16* Plot-1 for Walsh pattern-1 and Plot-2 below for Walsh pattern-2. Then it converged to Zoom1 section of Plot-1 & Plot-2 for Walsh pattern-1 & Walsh pattern-2 respectively with further reduced increment value and finally converged to Zoom2 section of Plot-1 & Plot-2 giving Walsh delay accurate ≤ 8µS.

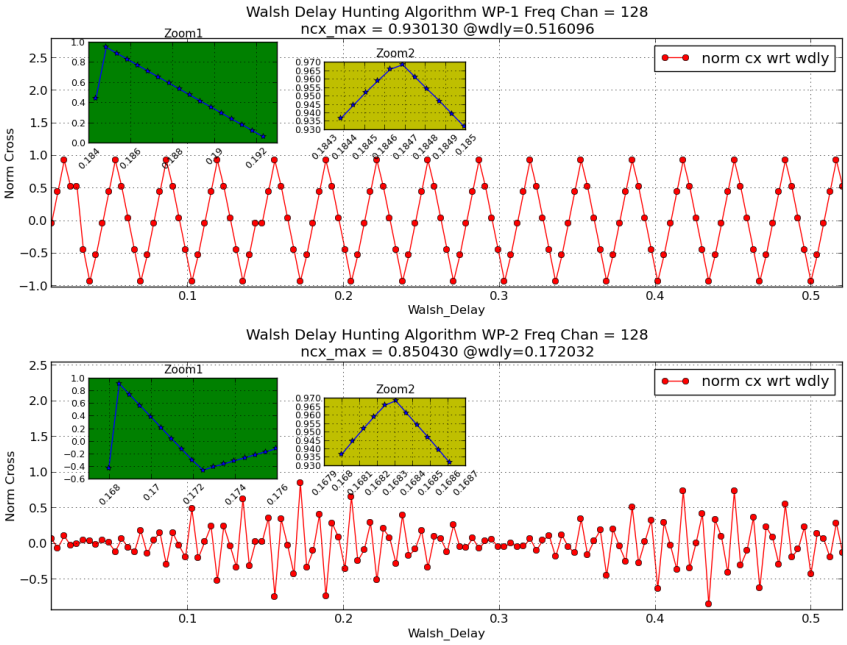

Figure 16: Walsh delay hunting program plot with zoom1 and zoom2 for WP-1 & WP-2

4.4 Effect of Walsh on Cross Talk

This test was intended to study the effect of Walsh on cross talk in the receiver chain of antenna to verify that it cancels out cross-talk in a receiver chain. The lab setup simulates the receiver chain behaviour where we used two different noise sources and a sine wave signal as shown in *Figure 17.* Sine wave is used to calculate precise rejection of cross talk in a receiver chain due to Walsh scheme. Two different noise sources are used to simulate two independent antenna polarizations at the input of L-Band post amplifier and phase switch (L-Band PA & PS) with Walsh control Walsh pattern-1 and Walsh pattern-2 from MCM-4 CPLD Walsh card. The configuration of L-Band PA & PS is also shown in *Figure 17*. The constant wave with equivalent power level of -10dbm of frequency (100 MHz) is injected after Walsh modulation done at L-Band PA & PS to simulate cross polar leakage. The inputs to the Pocket Correlator were maintained at a power level between -15 to -17 dbm.

4.4.1 Test Setup

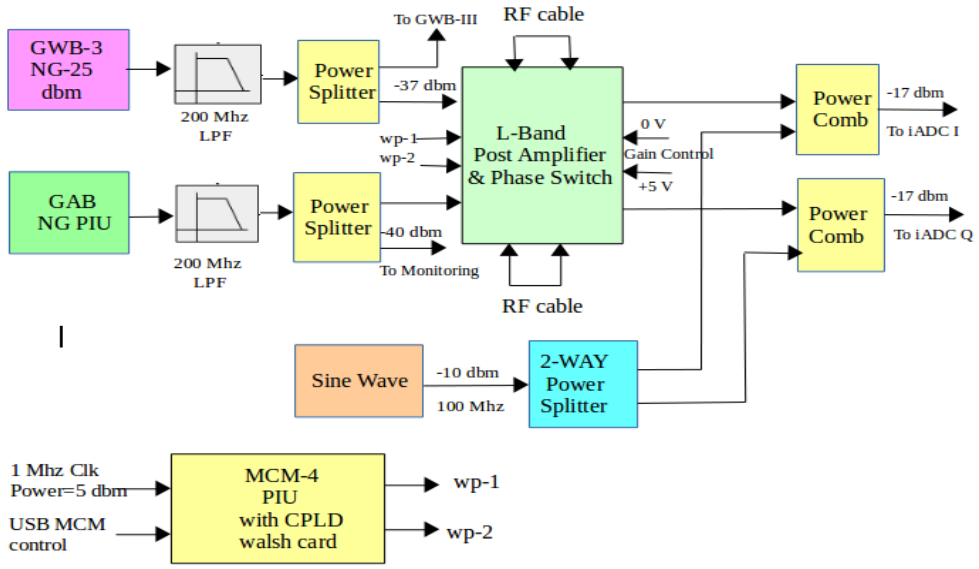

Figure 17: Lab test setup to simulate cross talk in receiver chain

4.4.2 Test Results

The test was carried out using Walsh patterns of antenna number 15, in which signal on pol-1 is modulated with Walsh pattern-1 and signal on pol-2 is modulated with Walsh pattern-2. Corresponding plots show correlation levels in first row are plots of frequency channel 128 w.r.t. Time plotted for the following conditions:

a) Without Walsh modulation
b) With Walsh modulation: Walsh Pattern – 1 (wp1)
c) With Walsh modulation: Walsh Pattern – 2 (wp2)
d) With Walsh modulation: Both Walsh Patterns (wp1 & wp2)

Following table shows rejection for various Walsh pattern conditions and it is above 99%.

Condition	No Walsh norm Cross	Norm cross with Walsh pattern	% Rejection
Walsh pattern-1 only	1.0	0.008117	99.1883
Walsh pattern-2 only	1.0	0.003677	99.6323
Both Walsh patterns	1.0	0.001329	99.8671

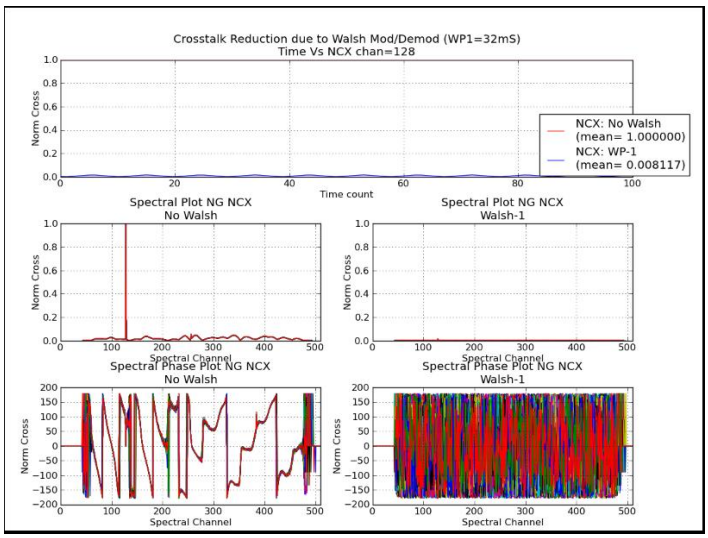

Figure 18: Lab Test - Cross talk reduction due to Walsh pattern-1

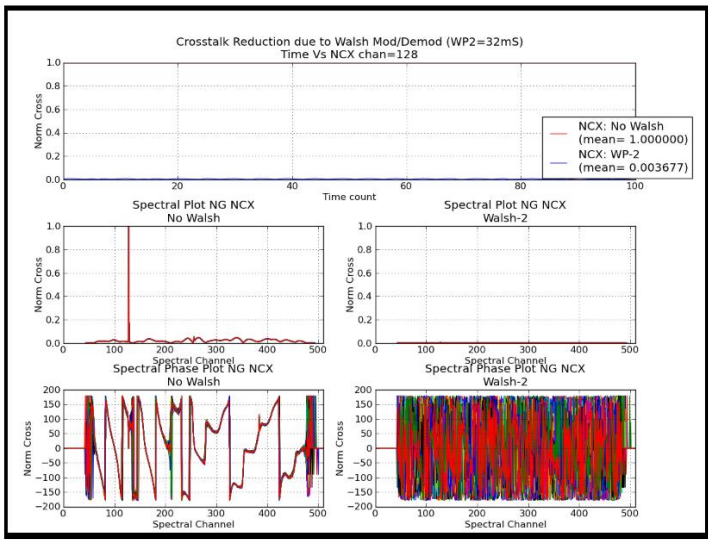

Figure 19: Lab Test - Cross talk reduction due to Walsh pattern-2

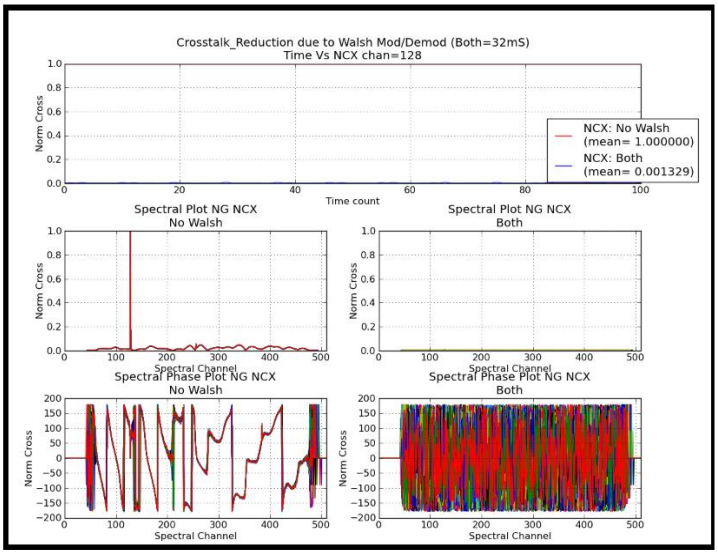

Figure 20: Lab Test - Cross talk reduction due to both Walsh patterns

4.5 Correlation Loss Test due to Walsh with Noise

This test is carried out to check the effect of Walsh modulation demodulation on cross correlation value of input signal. The test was done using three noise generators (NG) with NG-3(Atlantec NG) injected as a common noise before L-Band PA & PS along with NG-1(GWB-3 NG) & NG-2(GAB NG). After adjusting Walsh delay the input signal is modulated using Walsh patterns in L-Band PA & PS and this modulated signal is given to the input 'I' & 'Q' of Pocket correlator.

4.5.1 Test Setup

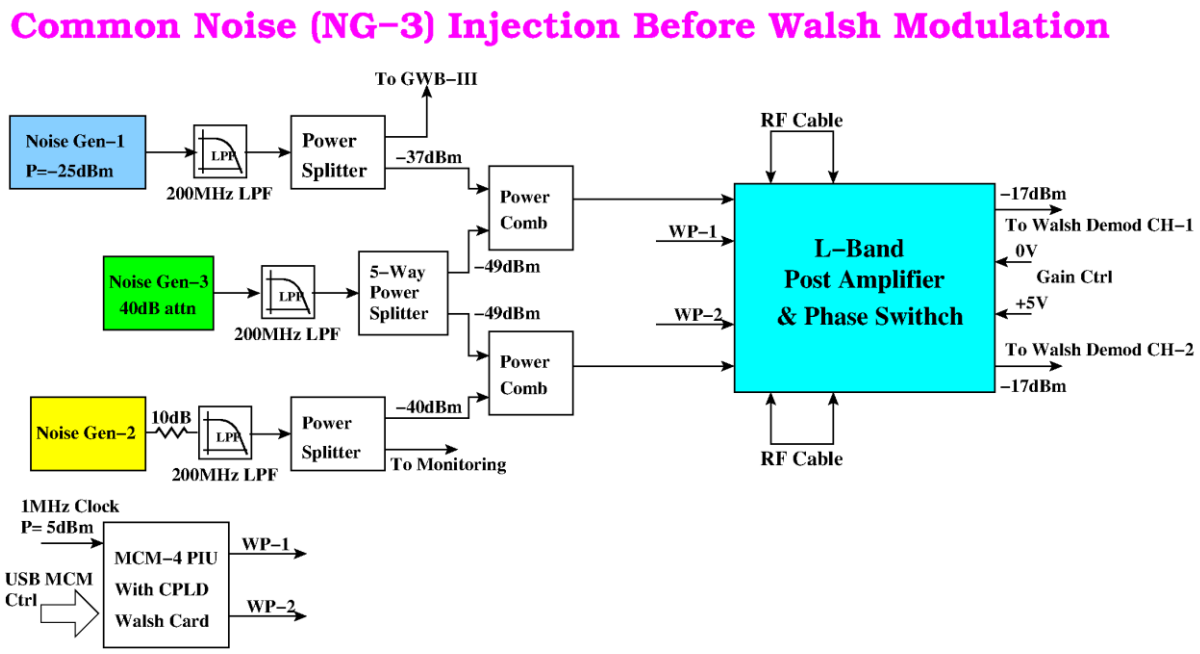

Figure 21: Test setup to check correlation loss due to Walsh

4.5.2 Test Results

The test was carried out using Walsh patterns of antenna number 15, in which signal on pol-1 is modulated with Walsh pattern-1 and signal on pol-2 is modulated with Walsh pattern-2 as per the testing conditions. The test was carried out for the following conditions:

a) Without Walsh modulation
b) With Walsh modulation: Walsh Pattern – 1 (wp1)
c) With Walsh modulation: Walsh Pattern – 2 (wp2)
d) With Walsh modulation: Both Walsh Patterns (wp1 & wp2)

Corresponding plots show correlation levels in first row are plots of frequency channel 100 w.r.t. time. 1st and 2nd column from 2nd row onwards are spectral plot of normalized cross correlation along with phase.

Figure 22 to 24 shows analysis of data using recorded file in tax format. In which x-axis is of time stamp and y-axis is normalised cross value. The recorded tax file used for analysis is (**Nowalsh_wp1_wp2_Both_corrloss_11June.dat)** and loss for each case is mentioned in bracket.

- No Walsh = (0%)
- walsh pattern-1 mod/demodulation on ch-1 = (~0%)
- Walsh pattern-2 mod/demodulation on ch-2 = (~0%)
- Both walsh pattern mod/demodulation (wp1 on ch-1 & wp2 on ch-2) = (~0%)

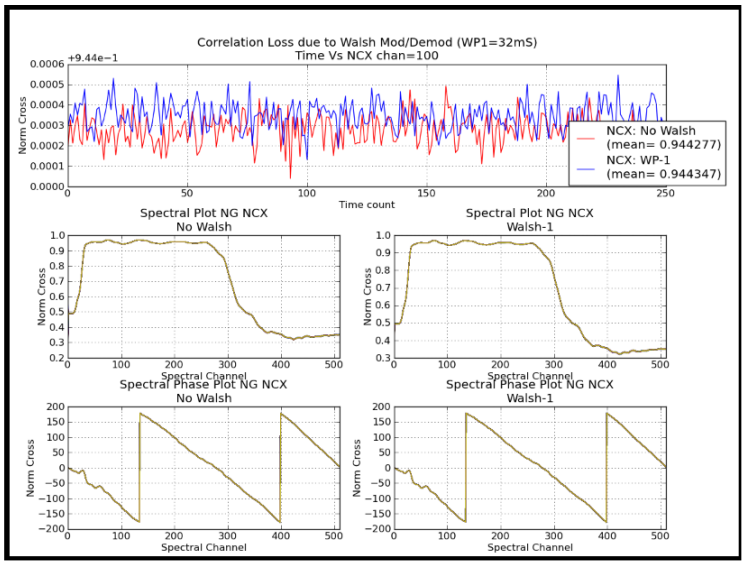

Figure 22: Loss of correlation due to Walsh pattern-1 with NG

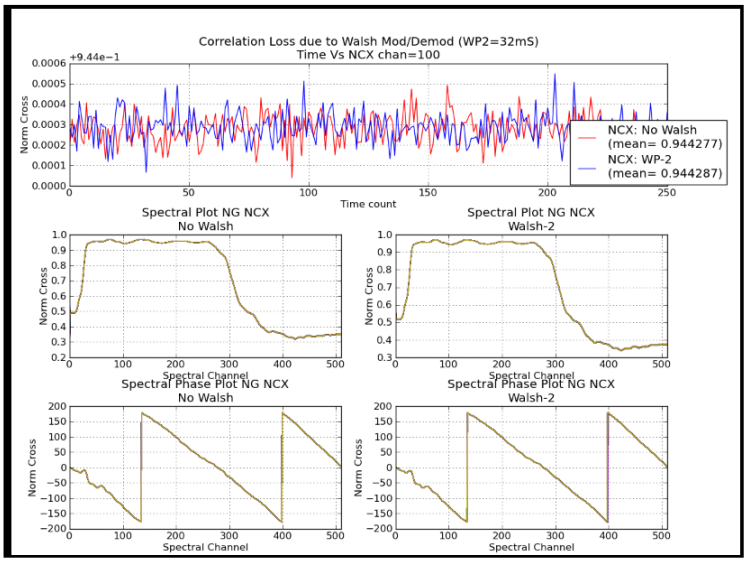

Figure 23: Loss of correlation due to Walsh pattern-2 with NG

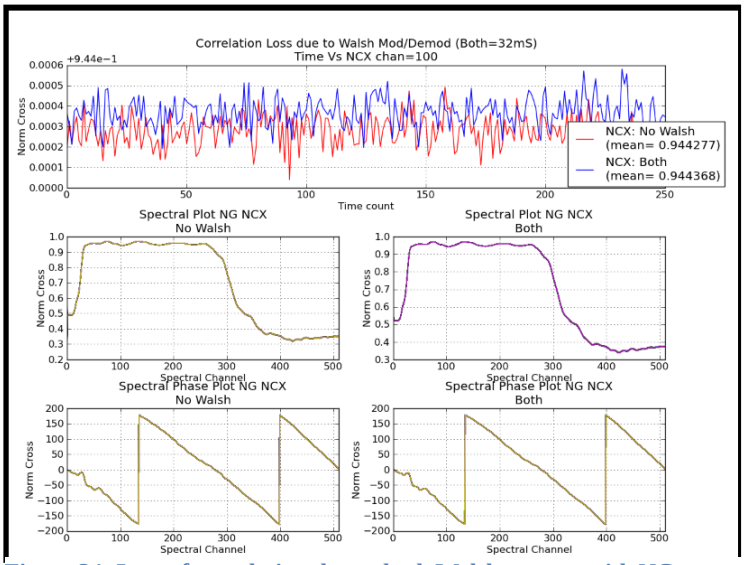

Figure 24: Loss of correlation due to both Walsh pattern with NG

4.6 Correlation Loss Test due to Walsh with Sine Wave

This test is carried out to check the effect of Walsh modulation demodulation on cross correlation value of input signal but unlike that in Section 4.5 with noise generator we used sine wave of 100MHz with -10dBm power level before Walsh modulation so as to measure correlation loss more accurately. The test was done using two noise generators (NGs) with sine wave generator injected as a common signal before L-Band PA & PS along with NG-1(GWB-3 NG) & NG-2(GAB NG). After adjusting Walsh delay the input signal is modulated using Walsh patterns in L-Band PA & PS and this modulated signal is given to the input 'I' & 'Q' of Pocket correlator.

4.6.1 Test Setup

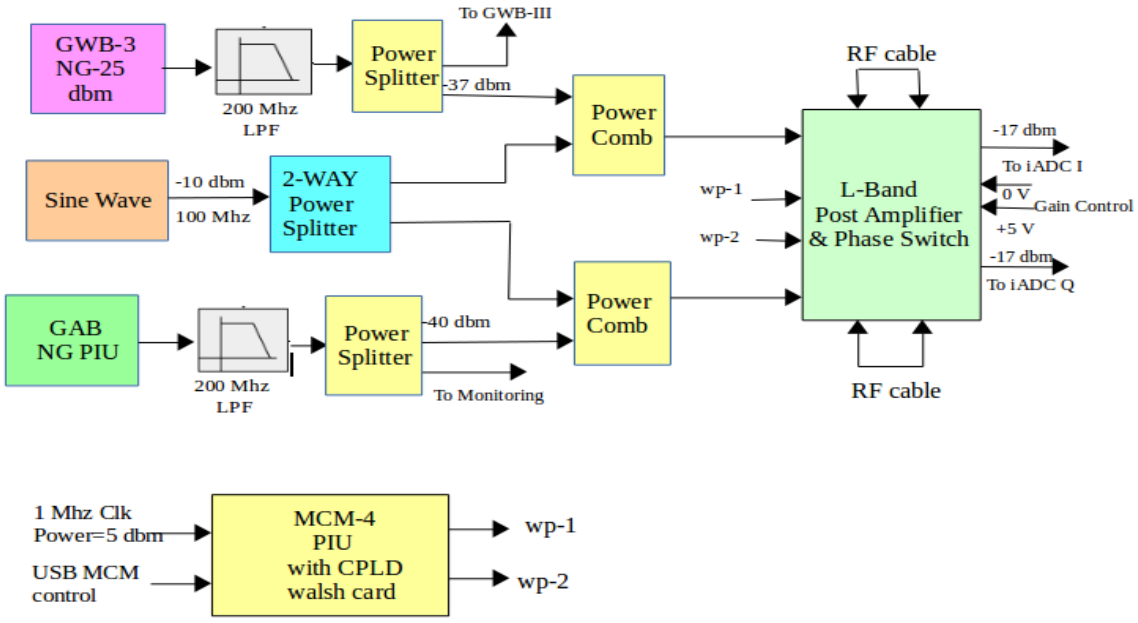

Figure 25: Test setup for correlation loss test using sine wave.

4.6.2 Test Results

The test was carried out using Walsh patterns of antenna number 15, in which signal on pol-1 is modulated with Walsh pattern-1 and signal on pol-2 is modulated with Walsh pattern-2 as per the testing conditions. Corresponding plots show correlation levels in first row are plots of frequency channel 100 w.r.t. times. 1st and 2nd column from 2nd row onwards are spectral plot of normalized cross correlation along with phase.

Figure 26 to 28 shows analysis of data using recorded file in tax format. In which x-axis is of time stamp and y-axis is normalised cross value. The recorded tax file used for analysis is (**14june_Nowp_wp1_wp2_both__sine_NG_clt.dat)** and loss for each case is mentioned in bracket.

- No Walsh = (0%)
- Walsh pattern-1 mod/demodulation = (~ 0.0468 %)
- Walsh pattern-2 mod/demodulation = (~ 0.0096 %)
- Both Walsh pattern mod/demodulation = (~ 0.0564 %)

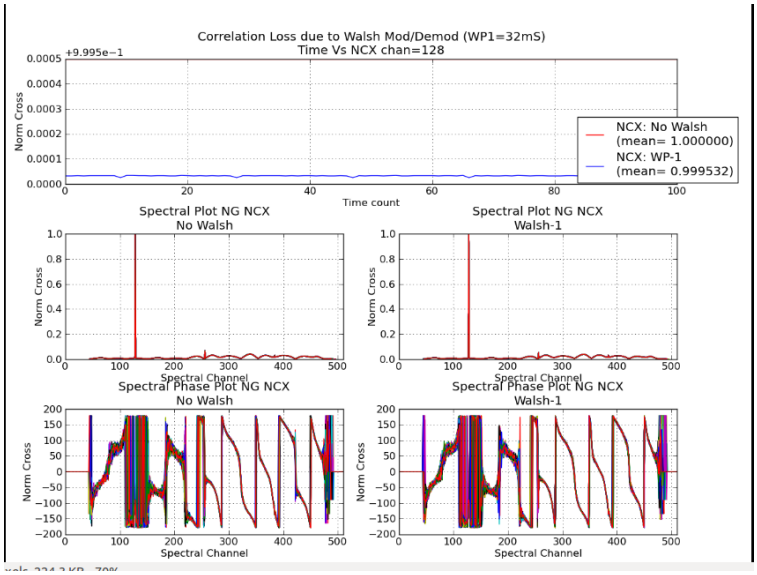

Figure 26: Correlation loss test due to Walsh pattern-1 on CW signal

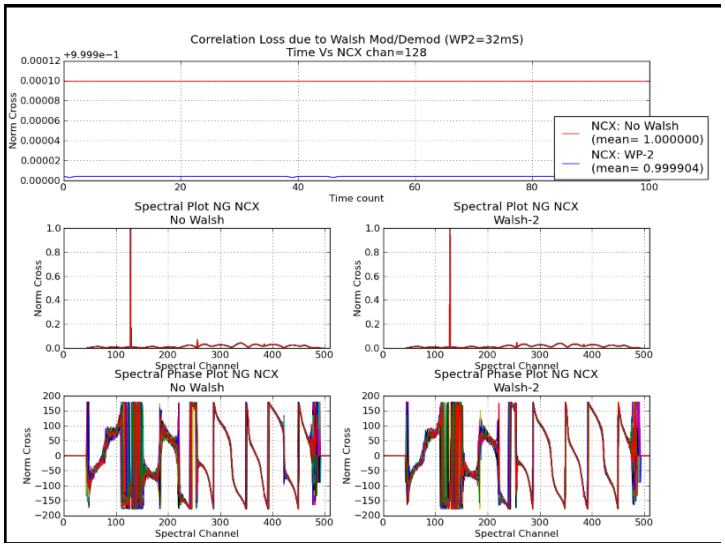

Figure 27: Correlation loss test due to Walsh pattern-2 on CW signal

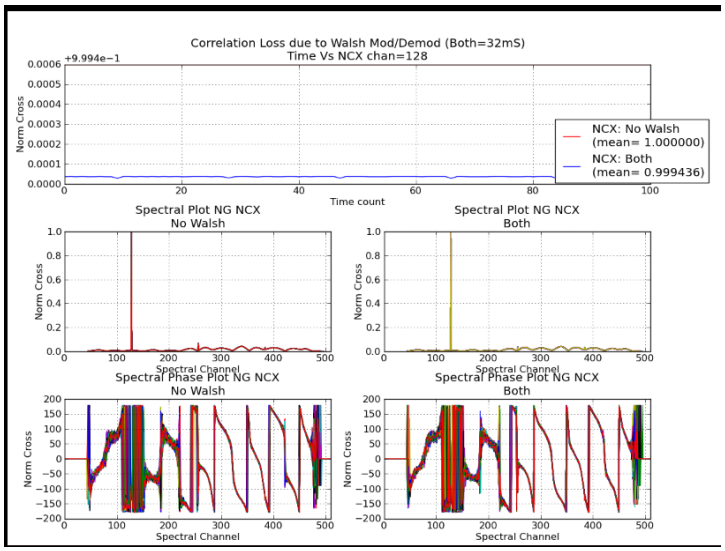

Figure 28: Correlation loss test due to both Walsh patterns on CW signal

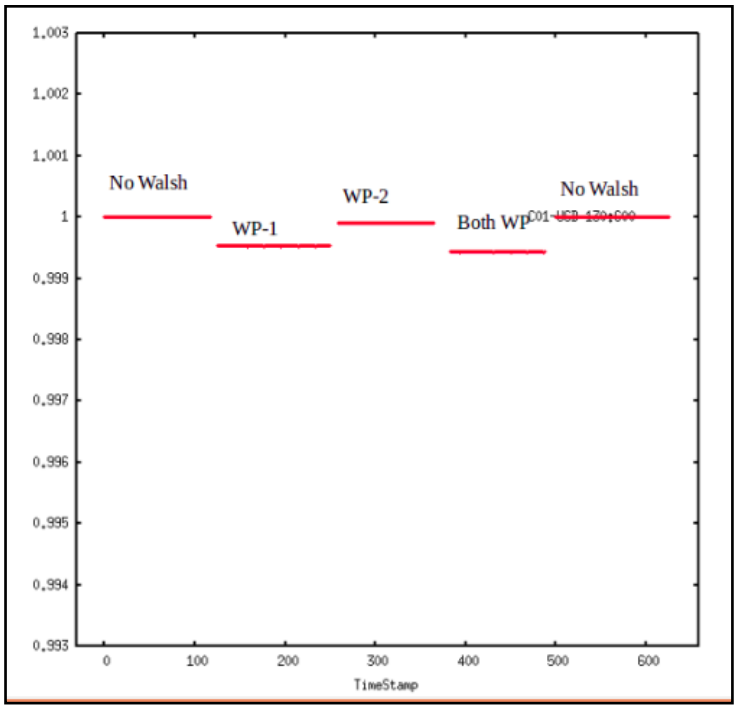

Figure 29: Loss of correlation for No Walsh, Walsh pattern-1, Walsh pattern-2, Both Walsh patterns

Chapter 5. Design Modification for Basic Walsh Pattern

5.1 Basic Walsh and Its Need

Basic Walsh pattern is a square wave with 50% duty cycle and having time period of a complete Walsh period i.e. 524.288 mS. The basic Walsh pattern was chosen for Walsh delay hunting as compared to other patterns as it has only one normalized cross correlation maxima in entire Walsh period of 524.288mS unlike other Walsh patterns thus making Walsh delay hunting simpler. The single maxima in case of basic Walsh coincides with rest of all Walsh pattern maxima and hence Walsh delay hunted with basic Walsh is equally applicable to rest of all Walsh patterns making hunting process simpler and reliable.
Using basic Walsh for delay hunting we can easily debug any swap in differential cable either at Front-End or at D49 PIU side.

5.2 Modifications from Modulator-Demodulator Side

There is new additional facility to generate basic Walsh pattern on pol-1 and pol-2 from both modulator and demodulator side. Corresponding modifications were done from CPLD side by ABR (Antenna Base Receiver) group to take out basic Walsh pattern out which can be controlled remotely. Following are the commands to generate basic Walsh patterns:

Functions	**Hex Code**
Basic Walsh pattern on Polarisation-1 only	70 C0 F0 C0
Basic Walsh pattern on Polarisation-2 only	70 E0 F0 E0

Similar modifications were done from FPGA demodulator side. Following are settings to set basic Walsh pattern on given polarisation:

Functions	**Hex Code**
Basic Walsh pattern on Polarisation-1 only	(7,0)
Basic Walsh pattern on Polarisation-2 only	(0,7)

5.3 Modified Delay Hunting Test

Previously the delay hunting for Walsh pattern-1 was carried out was not equally applicable to Walsh pattern-2 of the same antenna and hence the delay hunting has to be done again independently due to multiple maxima in different Walsh patterns. Therefore we went for basic Walsh pattern as there is no need to find delay every time we change Walsh pattern after delay hunt with basic pattern for given antenna settings and the same delay will be applicable to all remaining antenna patterns.

The basic Walsh delay hunting test was shown in *Figure 30* where main plot shows entire range scanned for cross maxima and it has only one maxima. Then it converged to Zoom1 section for basic Walsh pattern, with further reduction in increment value it finally converged to Zoom2 section giving Walsh delay accurate $\leq 8\mu S$.

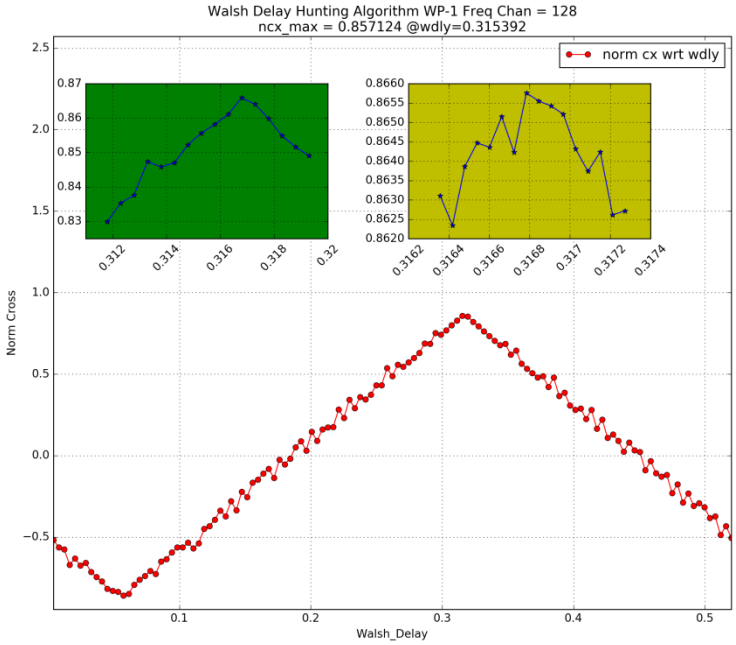

Figure 30: Walsh delay hunting with basic Walsh pattern

The maxima coincides with any antenna Walsh pattern and normalised cross value was very close to that of basic Walsh as shown in *Figure 31*.

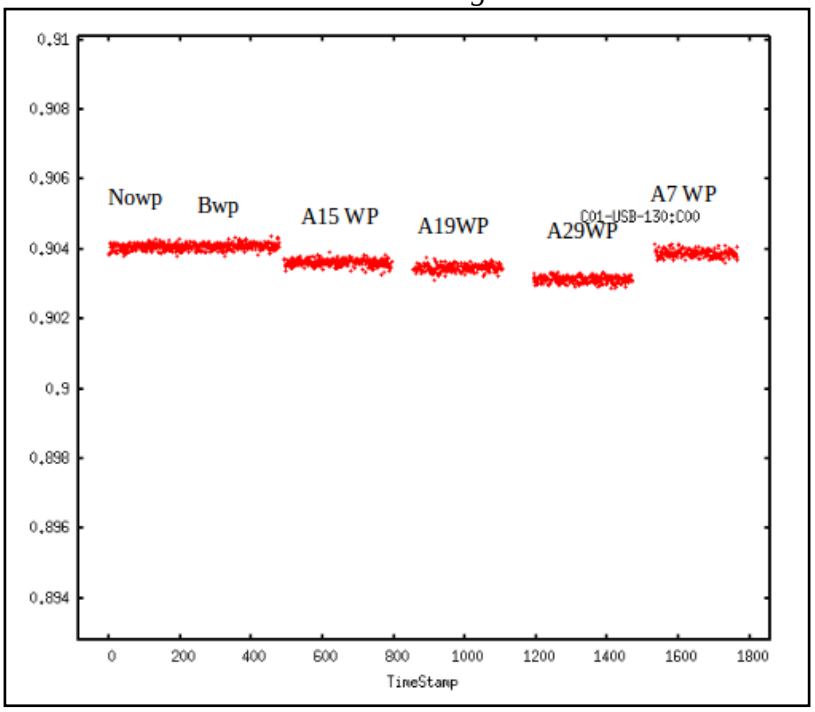

Figure 31: Basic Walsh pattern hunting delay applicable to different antenna Walsh pattern

5.4 Debugging the System

5.4.1 Delay Hunting Failure

The delay hunting reliability tests was performed by repeated testing for its conversion and we found that it was failing sometimes which we investigated. Following table shows random failure of delay hunt which is carried out for different attenuation levels.

Noise level	Walsh pattern	Polarization	Spectral Channel	No Walsh norm cross	After DH Ncx	DH Fails
10+1	wp-1	Pol-0	75	0.983677	0.9838	
	wp-2	Pol-1	75	0.983757	0.983	
10+2	wp-1	Pol-0	200	0.952746	0.9529	
	wp-2	Pol-1	200	0.952752	0.6378	X
20+2	wp-1	Pol-0	100	0.830036	0.8301	
	wp-2	Pol-1	100	0.82979	0.5785	X
20+5	wp-1	Pol-0	100	0.710728	0.7108	
	wp-2	Pol-1	100	0.710327	0.66	X
20+7	wp-1	Pol-0	100	0.607154	0.6074	
	wp-2	Pol-1	100	0.60739	0.6074	
30+1	wp-1	Pol-0	150	0.350362	0.3502	
	wp-2	Pol-1	150	0.3803	0.3802	
30+5	wp-1	Pol-0	150	0.1365	0.1364	
	wp-2	Pol-1	150	0.178205	0.178	
30+9	wp-1	Pol-0	100	0.090744	0.0896	X
	wp-2	Pol-1	100	0.191269	0.0579	X

Table 2: Normalised cross values before and after Walsh delay hunt

We found some possible reason of delay hunting failure was

a) At demodulator side Walsh patterns generated are delayed by 4.096 ms. This 4.096 ms value was constant for all remaining patterns.
b) The Walsh pattern generated from FPGA side is inverted ones with respect to CPLD side Walsh patterns.

These bugs were causing delay hunting to fail. Accordingly we corrected FPGA modulator design.

D1 is the CPLD side Walsh pattern.

D3 is the FPGA side Walsh pattern.

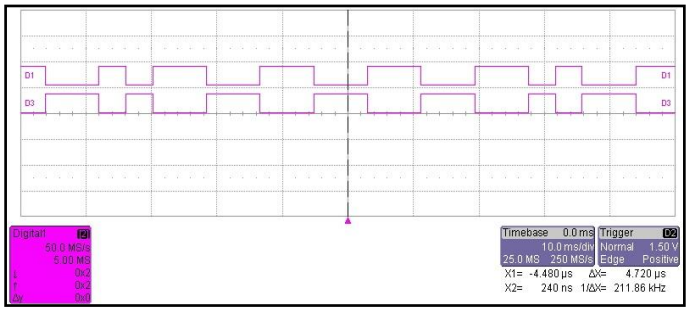

Figure 32: Inverted Walsh patterns from CPLD and FPGA causing delay hunting failure

5.4.2 Debugging: Stability

This test was carried out for checking the stability of normalised cross after delay hunt for 30 minutes. In this test, we get the maximum cross but after sometimes stability started decreasing gradually. This was due to the clock given to D-49 PIU was not locked to reference. After doing correction output results for the test is shown in below *Figure 33*:

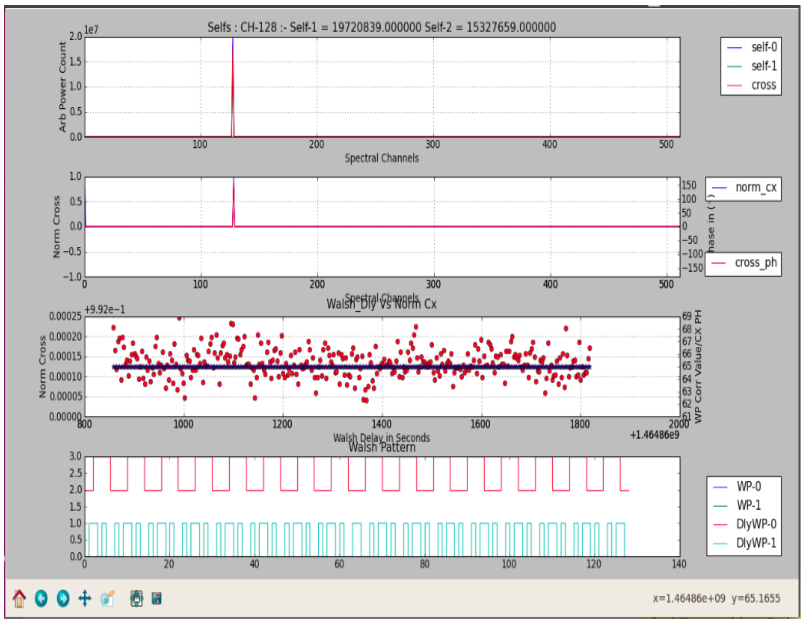

Figure 33: Stability plot to ensure locking of clocks at modulator & demodulator side

5.4.3 Debugging C12 Antenna Walsh Channel Swap Test

Before carrying out antenna testing there is a need to check Walsh channel swap. It is a simple test where applying a given Walsh pattern only to any one channel is getting applied or not. If it gets applied correctly then with antenna Extra-Hi Cal noise injection and after delay hunting we must get very low normalized cross correlation and if not then the cross value should not change.

The same test was carried out on C12 antenna. After applying Walsh pattern (wp1) on channel-1, ideally the cross should go to zero but we were getting cross confirming the said Walsh pattern was not getting applied to given channel. Then we applied both Walsh patterns and we got zero cross, confirming that there was Walsh channel swap. The same test was done by applying Walsh pattern-2 on channel-2 as shown in *Figure 34.* The recorded tax file used for analysis is:
(**C12W_C11_CH1_3C286_2aug2016_wdebug_nowp_wp1ch1_wp1bothch_wp2ch2.dat**)

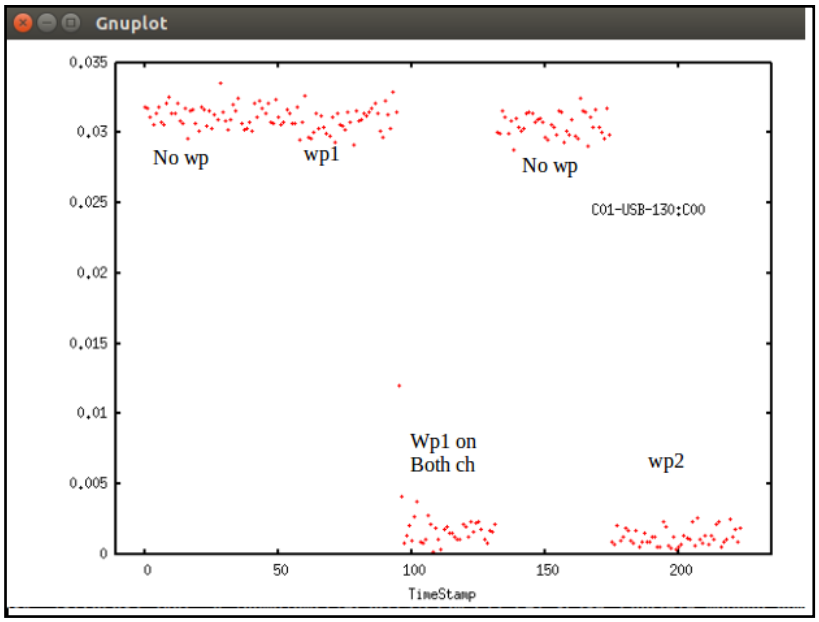

Figure 34: Walsh channel swap test plot

5.5 Antenna Testing

Actual implementation of Walsh scheme was carried out on central square antennas i.e.C11 & C12 to check the effect of Walsh modulation demodulation on cross correlation value of input signal. Only C12 antenna was having Walsh functionality, hence channel-1 of C11 was connected to input 'I' and channel-2 of C12 is connected to input 'Q' of ROACH. We adjusted Walsh delay for channel-1 of C12 antenna using basic Walsh pattern and then ran Pocket Correlator system with geometric delay correction. Finally loss of correlation by Walsh modulation-demodulation was then tested.

5.5.1 Results of Antenna Testing

Corresponding plots for above tests are shown in *Figure 35 to 37*, where first row plots are frequency channel 128 w.r.t. time. 1st and 2nd column from 2nd row onwards are spectral plot of normalized cross correlation along with phase.

Figure 38 shows analysis of data using recorded file in tax format. In which x-axis is timestamp and y-axis is normalised cross value. The recorded tax file used for analysis is **(C12w_C11_CH1_3C286_wswaptest_basicwp_wp1CH1.dat)** and loss for each case is mentioned in bracket.

- No Walsh = (0%)
- Walsh pattern-1 mod/demodulation = (~0.58%)
- Walsh pattern-2 mod/demodulation = (~1.12 %)
- Both Walsh pattern mod/demodulation = (~7.10 %)

The said test with antenna needs to be repeated so as to confirm loss of correlation due to Walsh.

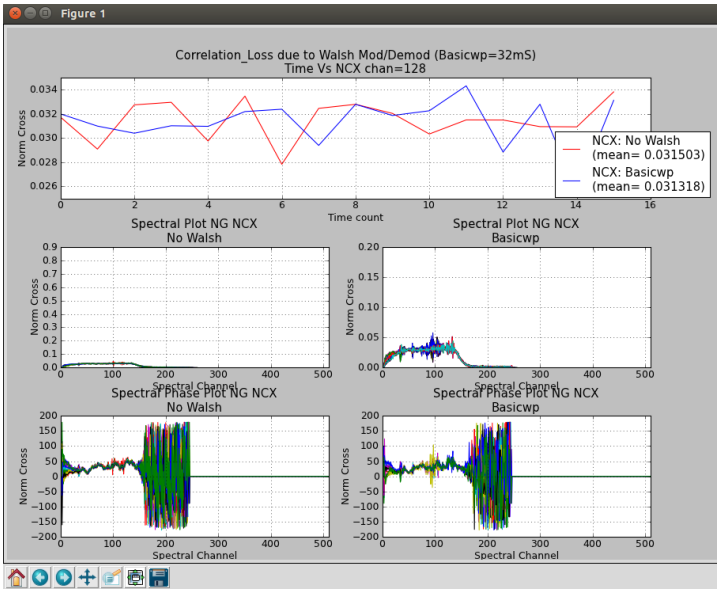

Figure 35: Loss of correlation due to Basic Walsh pattern

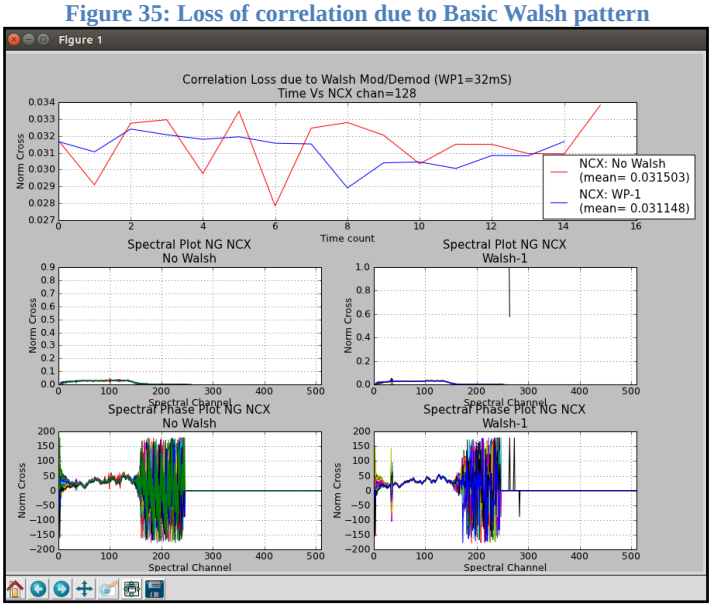

Figure 36: Loss of correlation due to Walsh pattern-1

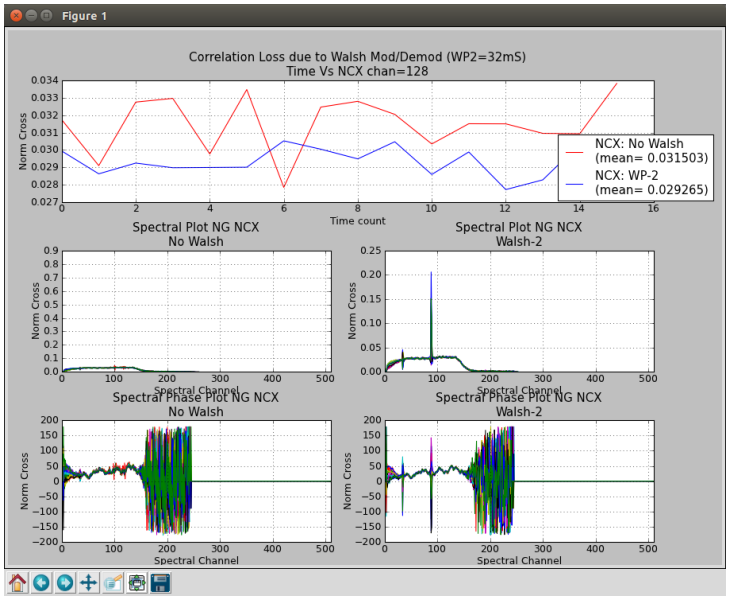

Figure 37: Loss of correlation due to Walsh pattern-2

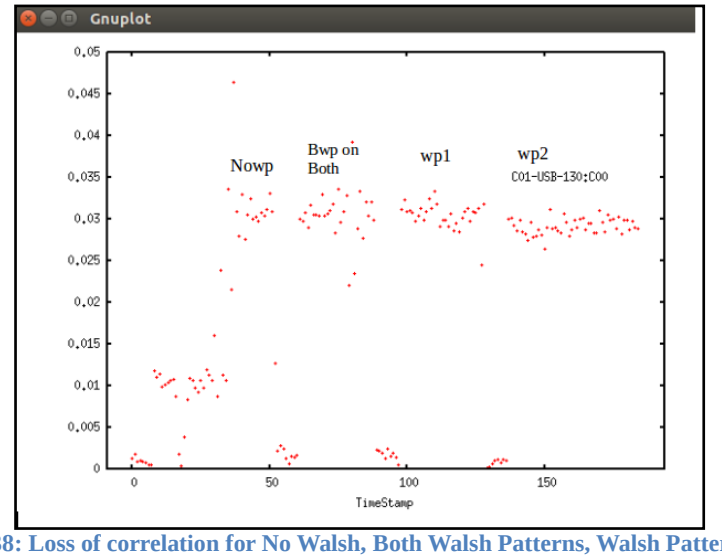

Figure 38: Loss of correlation for No Walsh, Both Walsh Patterns, Walsh Pattern-1, Walsh Pattern-2

Chapter 6. Conclusion and Future Scope

This project was aimed at understanding the Walsh Modulation-Demodulation system. The work carried out in this project was debugging of the system which served way for understanding the Walsh scheme.

Lab testing of the scheme shows encouraging results and required repeated testing with GMRT antennas. The scheme then may be expanded to 30 antennas of GMRT.

Walsh delay hunting algorithm need to be suitably modified to work with GWB system.

Chapter 7. References

[1] **www.gmrt.ncra.tifr.res.in**

[2] **http://www.tutorialspoint.com**

[3] "Learning Python 4th Edition by Mark Lutz"

[4] "Walsh switching -Required accuracy and possible Scheme" A Paramesh Rao & C.R.Subrahmanya 25 June '91.

[5] "Programmable walsh pattern generation using CPLD" by Meha Kainth.

[6] "Reducing effects of cross talk in a Radio Telescope using Walsh modulation" by Sandeep C. Chaudhari.

[7] "Fringe Tester for Parabolic dish" by Shrikant Chaudhari & Sanket Bhansali.

Chapter 8. Appendix-A: Development of Python Packaging

8.1 Introduction to Python

Python is a high-level, interactive and object oriented-scripting language. Python is interpreted it is processed at runtime by the interpreter and you do not need to compile your program before executing it. Python is Interactive**;** you can actually sit at a Python prompt and interact with the interpreter directly to write your programs. It supports Object-Oriented style or technique of programming that encapsulates code within objects. One of Python's greatest strengths is the bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

8.2 Scripts

A script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time. Script mode programming invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. All python files will have extension *.py*. The extension *.py*, just to help us know that the script contains python code. You can make different versions of the script by modifying the statements from one file to the next using a text editor. Then execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing. .

```
#! /usr/bin/python
# This is my first script
# Generating a config file
```

Before we can run this script, we must first make it executable. To do this we will use the unix chmod command.

```
chmod u+x config_gen.py
```

This command basically tells unix to set the x (executable) flag for the user level access of the file. Now we are able to run the file. You can execute your script by using the following command:

```
./config_gen.py
```

Configfile

The config files that ConfigObj will read and write are based on the 'INI' format. ConfigObj is a simple but powerful config file reader and writer: an ini file round tripper. Its main feature is that it is very easy to use, with a straightforward programmer’s interface and a simple syntax for config files.

- ## Writing Configuration File:

  Keywords and values are separated by an '=', and section markers are between square brackets. Keywords, values, and section names can be surrounded by single or double quotes. Indentation is not significant, but can be preserved. Subsections are indicated by repeating the square brackets in the section marker. You nest levels by using more brackets. You can have list values by separating items with a comma, and values spanning multiple lines by using triple quotes (single or double).

```
from configobj import ConfigObj
config = ConfigObj()
config.filename = filename

#
config['keyword1'] = value1
config['keyword2'] = value2
#

config['section1'] = {}
config['section1']['keyword3'] = value3
config['section1']['keyword4'] = value4

#
section2 = {
   'keyword5': value5,
   'keyword6': value6,
   'sub-section': {
      'keyword7': value7
      }
}
config['section2'] = section2

#
config['section3'] = {}
config['section3']['keyword 8'] = [value8, value9, value10]
config['section3']['keyword 9'] = [value11, value12, value13]
#
config.write()
```

- ## Reading Configuration File

  The normal way to read a config file is to give ConfigObj the filename. You can then access members of your config file as a dictionary. Subsections will also be dictionaries.

```
from configobj import ConfigObj
config = ConfigObj(filename)

#
value1 = config ['keyword1']
value2 = config ['keyword2']
```

```
#
section1 = config ['section1']
value3 = section1 ['keyword3']
value4 = section1 ['keyword4']
#

# you could also write
value3 = config ['section1']['keyword3']
value4 = config ['section1']['keyword4']
```

8.3 Modules

Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module. A module is a file containing Python definitions and statements. It is the highest-level program organization unit, which packages program code and data for reuse, and provides self-contained namespaces that minimize variable name clashes across programs. Modules typically correspond to Python program files. Each file is a module, and modules import other modules to use the names they define.

- **Module Creation**

To define a module, using text editor type some Python code into a text file, and save it with a ".py" extension; any such file is automatically considered a Python module. All the names assigned at the top level of the module become its attributes and are exported for clients to use.

Example

Write a def (function) into a file called module1.py and import it; create a module object with one attribute the name printer, which happens to be a reference to a function object:

```
def function(x):
Print(x)
```

- **The import Statement**

  The import statement simply lists one or more names of modules to load, separated by commas. Because it gives a name that refers to the whole module object, we must go through the module name to fetch its attributes

```
import module1 [, module2 [, moduleN]
```

- **The from Statement**

  By contrast, because from copies specific names from one file over to another scope, it allows us to use the copied names directly in the script without going through the module .Python's from statement import specific attributes from a module into the current namespace.

```
from modname import name1 [, name2 [, nameN]]
```

- **Reloading Modules**

Module's code can be reloaded and rerun explicitly by calling the reload built-in function. Imports (via both import and from statements) run a module code only the first time the module is imported in a process. Later imports use the already loaded module object without reloading or rerunning the file's code. The reload function forces an already loaded module's code to be reloaded and rerun. Assignments in the file's new code change the existing module object in place.

import module
...use module. Attributes...
.....
reload (module)
...use module. Attributes...

8.4 Packages

Python has a packaging system that allows people to distribute their programs and libraries in a standard format that makes it easy to install and use them. The Python Package Index (PyPI) will be provided which allows a developer to distribute a package to the greater community with little effort.

Packages are namespaces which contain multiple packages and modules themselves. Each package in Python is a directory which must contain a special file called __init__.py. This file can be empty, and it indicates that the directory it contains is a Python package, so it can be imported the same way a module can be imported. In addition to a module name, an import can name a directory path. A directory of Python code is said to be a package, so such imports are known as package imports. In effect, a package import turns a directory on your computer into another Python name- space, with attributes corresponding to the subdirectories and module files that the directory contains.

Package Initialization

Package __init__.py files are partly present to declare that a directory is a python package. The first time a Python program imports through a directory, it automatically runs all the code in the directory's __init__.py file. Because of that, these files are a natural place to put code to initialize the state required by files in a package. For instance, a package might use its initialization file to create required data files, open connections to databases, and so on. Typically, __init__.py files are not meant to be useful if executed directly, they are run automatically when a package is first accessed.

Steps to Create a Python Package

1] Create a directory and give it your package name.
This is the main directory of your project which contain setup file
Ex.mkdir Wcorr

2] Put your sub directories in your main directory.
cd wcorr
mkdir src
src :- It contain the config_gen.py ,config_read.py and __init__ .py

mkdir etc
etc :- etc contain the all data file of your project.

mkdir scripts
Scripts: - List of standalone script files to be built and installed.

3] Create a __init__.py file in the directory.

4] Write a Setup Script for your package.

The setup script is the centre of all activity in building, distributing, and installing modules using the Distutils. The main purpose of the setup script is to describe your module distribution to the Distutils, so that the various commands that operate on your modules do the right thing. The Distutils setup script is a Python script. The first line of every Distutils setup script is always the same.

- **Create the setup.py**

```
vim ~/Wcorr/setup.py
from distutils.core import setup
setup(
        # Application name:
                name="Wcorr",
        # Version number (initial):
                version="0.1.0",
        # Application author details:
                author="name surname",
                author_email="name@addr.ess",
        # Packages
                packages=["app"],
        # Include additional files into the package
                include_package_data=True,
        # Details
                url="http://pypi.python.org/pypi/MyApplication_v010/",
        # license="LICENSE.txt",
                description="Useful towel-related stuff.",
        # long_description=open("README.txt").read(),
        # Dependent packages (distributions)
                install_requires= [
                "flask",
        ],)
```

5] Build and Install the Application locally

Packages built and distributed using setuptools look to the user like ordinary Python packages based on the distutils.You would have to rebuild and reinstall your project every time you made a change to it during development.The recommended pip installer runs all setup.py scripts with setuptools.Now the application can be installed and used by others using the setup.py file created.

In order to built and install the application, run the following command:

```
python setup.py build
python setup.py install
```

8.5 Python Scripts for Walsh Testing

To configure Walsh functionality following python scripts are required.

1) Configuration File Generation Script Details: - (conf_file_gen.py)

Required Python Modules: ConfigObj from configobj.

- **SECTION** : **katcp**

  (Karoo Array Telescope Control Protocol) is a simple ASCII communication protocol for monitoring and control of hardware devices and software that control them.

  - [katcp_port]: Port to communicate with FPGA design through ROACH PowerPC.
  - servers: ROACH board IP/ name to connect.
  - bitstream: Borph file to program ROACH board.

- **SECTION: correlator**

  - [n_chans]: Number of frequency channels. (512)
  - [n_ants]: Number of antennas in design. (2)
  - [fft_shift]: fft shift to be applied to fft block. (1023)
  - [acc_len] : Intigration time in terms of FFT cycles (Time taken to complete 1 FFT in terms FPGA/ADC clock)
  - [adc_clk] : External ADC clock fed to the system (800MHz)
  - [sync_time] : The time since epoch (Boundry of 1pps on which correlator starts/trigger)
  - [Wfreq]: Walsh frequency to be set.
  - [fft_len]: Size of FFT inside design.
  - [fft_per_sync]: No. of FFT cycles within two sync pulses.(length vary depending upon PFB/FFT block).
  - [max_int_dly]: Maximum geometric delay in terms of ADC clock that can be compensated inside design.
  - [mcnt_bits]: A counter that is used to keep track of time on FPGA whose bit size is defined here.
  - [Scale]: To control resolution and dynamic range of correlator.
  - [Wfreq]: A clock used to genarate walsh pattern.
  - [i_offset]: Integer delay offset in terms of ADC clock.
  - [fr_offset]: Delay offset in terms of fraction of ADC clock.
  - [frn_offset]: fringe offset in terms of $\square^0$.

2) Configuration File Reading Script Details :- (poco_cn_config.py)

This script contains a class named that consists of all functions required to read config file of correlator and return the same to calling function.

- Required Python Modules: iniparse, exceptions, struct, numpy, os
- __init__: (Initialization function for Class to do initial settings)
  1) Check configuration file existence
  2) Parse the configuration file using iniparse module.
  3) Create an empty dictionary to store section wise parameter values.
  4) Read all config file items using function read_all () within the class
- __getitem__ (Return populated dictionary to calling function)
  1) Return populated dictionary to calling function.
- __setitem__ (Set the given item in dictionary by the calling function)
  1) Set the given item in dictionary by the calling function.
- Function file_exists():
  1) Try to open the configuration file.
  2) If exists close the file and return exists flag = True
  3) Else not exists return an IO error.
- Function: read_all() : (Reads all sections parameters within a configuration file)
  1) Read ROACH server.
  2) Read bitstream to program server.
  3) Read katcp section: katcp port.
  4) Read correlator section:
     a. n_chans
     b. n_ants
     c. fft_shift
     d. acc_len
     e. adc_clk
     f. sync_time
     g. fft_len
     h. fft_per_sync
     i. max_int_dly
     j. mcnt_bits.
  5) Read equalization section:
     a. scale0
     b. scale1
  6) Derive feng_clk from adc_clk.
  7) Derive mcnt_scale_factor from derived feng_clk.
- Function: read_int() :
  1) Read integer parameters of a variable inside a given sections.
- Function: read_str() :
  1) Read string parameters of a variable inside a given sections.
- Function : write()
  1) Write given parameter into dictionary.
  2) Write given parameter especially sync_time into configuration file.

3) Initialize pocket correlator: (poco_init.py)

This script is used to initialize pocket correlator with Walsh functionality, but run as correlator without Walsh enable.

- Required python modules : wpoco,time,sys
- use **optparse** to parse command line options for walsh intialization
  1) Search for the config file.
  2) If not found print "Error: Please specify configuration file!!!".
- **Initialise pocket correlator :**
  1) Initialize correlator instance.
  2) Read configuration file (Here it is example.cfg).
  3) Connect to ROACH board specified in configuration file.
  4) Deprogram ROACH board.
  5) Program ROACH board with bitstream specified in configuration file.
  6) Set register 'fft_shift' with config file fft_shift value.
  7) Set Accumulation Time.
  8) Set Equalisation Coefficients.
- **Antenna Specific Settings :**
  a) Arm the design i.e. trigger the design for accurate timing.
  b) Checks F-engine clk_frequency to confirm correct PPS operation.
  c) Disconnect the katcp link.

4) Initialises pocket correlator with Walsh functionality: (walsh_init.py)

- Required python modules : wpoco,time,sys

1) use **optparse** to parse command line options for walsh intialization

Options:

-a ANT, --ant=ANT	Set Antenna number from 0 to 29
-p	Skip Programming
-d WFREQ_DIV	Walsh frequency division factor to be applied
0 for divide by 1	
1 for divide by 2	
2 for divide by 4	
3 for divide by 8	

1) Search for the config file.
2) If not found print "Error: Please specify configuration file!!!".
3) Initialise pocket correlator with walsh function :
4) Connect to ROACH board specified in configuration file.
5) Deprogram ROACH board.
6) Program ROACH board with bitstream specified in configuration file.
7) Set register ‘fft_shift' with config file fft_shift value.
8) Sets the accumulation time for correlator.
9) Setting Equalisation Coefficients.
10) Configuring the Antenna with Walsh Functionality.
    a) With selected Walsh frequency and

b) wp-1 on ch-1 and wp-2 on ch-2 of selected Ant)

11) Adjust FPGA Walsh clock according to CPLD clock.
12) Arm the design i.e. trigger the design for accurate timing.
13) Checks F-engine clk_frequency to confirm correct PPS operation.
14) Set 0.0000 Walsh delay to both the polarization.
15) Disconnect the katcp link.

5) Walsh Pattern Delay Hunt Programme (delay_huntV2C01.py)

- Required python modules : wpoco,time,sys
- Options:

-h, --help	show this help message and exit
-m, --multiplot	Enable multiplot spectrum
-t TAX_DUMP,	--tax_dump=TAX_DUMP Tax native format dumping of POCO Design data
--chan=CHAN	Marker channel to indicate values on plot
--log=LOG	Logging file to dump Walsh delay & Normalized Cross Count
-d WFREQ_DIV	Walsh frequency division factor to be applied 0 for divide by 1 1 for divide by 2 2 for divide by 4 3 for divide by 8 (DEFAULT = 0)
-p POL	Polarisation selection for a antenna for doing delay hunting.
-a	which type of Testing: Antenna or Lab (DEFAULT: LAB TESTING)
-b	Basic Walsh pattern for delay hunt

1) Search for the config file.
2) If not found print "Error: Please specify configuration file!!!".
3) Create a pylab figure object for plotting purpose.
4) Define figure properties and subplot properties.
5) Set initial Walsh delay 0.
6) Increment Walsh delay for selected wfreq_div.
7) Print initial increment value.
8) Set maximum Walsh delay value for selected wfreq_div.
9) Adjust the Walsh delay between 0 to maximum delay value.
10) Reads Cross Correlation Value from BRAM.
11) Disable Walsh modulation and read accumulation without Walsh.
12) If -t option is enable then write the self and cross values in tax file.
13) If pol-0 is selected enable walsh demodulation on channel-1
14) If pol-1 is selected enable walsh demodulation on channel-2
15) Print value of ncx without Walsh.
16) Press enter to continue to delay hunting.

17) Function Updatefig():
(a) Read accumulation count as pre_read count.
(b) Increment Walsh delay upto maximum resolution of normalised cross.
(c) If wdly>max_dly Print maximum correlation value and fine delay value.
(d) For fine Walsh delay correction reduces increment from 4ms to user defines increment.
(e) Increment for wdly resolution upto 4^{th} decimal place.
(f) Continue until maximum cross reaches .
(g)After finishing nth iteration check for exit of program and write the Corresponding self value and cross value in tax file.

18) If command line multiplot option is enabled plot 4 subplots as :
(a) Plot self in subplot-1
(b) Plot Norm Cross Vs Spectral Channel in subplot-2
(c) Plot Walsh_Dly Vs Norm Cx in subplot-3
(d) Plot Walsh patterns in subplot-4

6) Checks Stability of Pocket Correlator with Walsh Functionality: (stability.py)

1) Required python modules: numpy, pylab, time, struct, gtk, gobject, sys, math, wpoco, matplotlib

- Options:

-h, --help show this help message and exit

-m, --multiplot Enable multiplot spectrum

-t TAX_DUMP, --tax_dump=TAX_DUMP
Tax native format dumping of POCO Design data

--chan=CHAN Marker channel to indicate values on plot

-p, --read_patt Read walsh pattern on CH-2 path BRAMs

2) Function Interleave (): (these functions take any number of lists as arguments and interleave/comb their contents.)
a. Iterate over the maximum contents inside lists and over total number of lists.
b. Put the results in a local variable x.
c. Return the interleaved/combed result x to calling function.

3) Initialise pocket correlator with Walsh function:
4) Search for the config file.
5) Connect to ROACH board specified in configuration file.
6) Adjust FPGA Walsh clock according to CPLD clock frequency variation.
7) Read cross correlation value.
8) Set the initial normalise cross an wdly value 0 and maximum wdly of 0.511.
9) Adjust figure properties and subplot properties to plot the Walsh delay, self normalise cross.
10) Create complex array of log_wdly, log_ncx, ant_data, selfs, cross, norm_cx.
11) Read accumulation number.

12) Function Updatefig ():
    a. Read accumulation count as pre_read count.
    b. Read cross values and Walsh patterns.
12) If tax dump option is enable write no Walsh data into tax file.
13) Append time stamp values in tax file.
14) If command line multiplot option is enabled plot 4 subplots as:
    a) Plot Selfs in subplot-1.
    b) Plot Cross amplitude and cross phase in subplot-2.
    c) Plot Normalised cross in subplot-3
    d) Plot Walsh patterns in subplot-4

Chapter 9. Appendix – B: SOP for Lab Testing

Python Scripts required for testing:

- **walsh_init.py**
- **walsh_delay_huntV2.py**
- **stability.py**

9.1 Delay Hunt using Basic Walsh Pattern:

Step 1:- Make the test setup and adjust power level as per the fig.15

Step 2:- Login into ppc using following command

ssh -X gmrt@192.168.4.71
Password: gmrttifr

Step 3:- For Delay hunting test on Pol 0 with Basic Walsh Pattern

a) Open new konsole window

sudo konsole

b) Go to following directory:

cd /Pranita/Hello/MCM/

c) Run mcmtest program for communication with MCM - 4

sudo ./mcmtest

- set_mcm_address = 0
- Enter Mcm Address = 4
- nul_cmd = 1
  (See the msg “MCM COMMAND SUCCESSFUL”)
- set_dmask_32b = 6
- Enter Dig Mask 32 = xx xx xx xx

Ex. WP1 = 70 00 F0 00

d) Initially set the 32 bit HEX code as 70 00 F0 00

Step 4:-

a. Open new konsole window

sudo konsole

b. Go to following directory:

cd /home/Pranita/

c. Initialize pocket correlator with Walsh functionality using following command (select any antenna no. for initialization Ex. a15):

./walsh_init.py -p -d0 -a15 example_basicwp.cfg

d. Run Walsh delay hunting programme to find Walsh Delay for Pol0.

./walsh_delay_huntV2.py -m --chan=128 -d0 -a -p0 example_basicwp.cfg

Note the following:

- no walsh normalized cross (nowalsh_ncx) =
- nowalsh_ph =
- Before pressing enter set basic walsh pattern on pol-0 from CPLD side, using following command :

  70 C0 F0 C0
- And set the basic walsh pattern on pol-0 from FPGA side in ipython :

  a) import wpoco
  b) Corr_inst=wpoco.poco_functions.Correlator (config_file=/home/ gmrt/Pranita/example_basicwp.cfg)
  c) Corr_inst.walsh_sel (walsh_sel= (7, 0))

- Now press enter to continue delay hunting until normalised
- Note the value of normalized cross maxima (ncx max) =
- And note walsh delay (wdly) =

e. Disable Walsh pattern from CPLD side & FPGA side

1. 70 00 F0 00 ..... (Cpld side)
2. from ipython ..... (Roach side)
   a. import wpoco
   b. corr_inst=wpoco.poco_functions.Correlator (config_file='/home/gmrt/Pranita/example_basicwp.cfg')

c. corr_inst.walsh_sel (walsh_sel= (0, 0))

Step 5:- Again continue delay hunting with basic Walsh pattern on Pol-1.

./walsh_delay_huntV2.py -m --chan=128 -d0 -a -p1 example_basicwp.cfg

Note the following:

- no walsh normalized cross (nowalsh_ncx) =
- nowalsh_ph =
    - Before pressing enter set the basic walsh pattern on pol-1,use following command from Hello Directory

      70 E0 F0 E0
- And to set the basic walsh pattern on pol-1 from Roach side :

a) import wpoco

b) Corr_inst=wpoco.poco_functions.Correlator (config_file=/home/ gmrt /Pranita/example.cfg)

c) Corr_inst.walsh_sel (walsh_sel=(0,7)

- Now press enter to continue delay hunting.
- Note the value of normalized cross maxima (ncx max) =
- walsh delay (wdly) =

Step 6: After adjusting the Walsh delay on both channels start modulation and demodulation process.

9.2 Correlation loss test due to Walsh Modulation/Demodulation:

1. Follow the steps of delay hunt algorithm as explain above.
2. For correlation loss test make the test setup as explain in fig 21 & follow the below steps:
   a) Run mcmtest program for communication with MCM - 4
      sudo ./mcmtes
      - set_mcm_address = 0
      - Enter Mcm Address = 4
      - nul_cmd = 1
      - (see the following msg "MCM COMMAND SUCCESSFUL ")
      - set_dmask_32b = 6
      - Enter Dig Mask 32 = xx xx xx xx
      - Ex. 70 00 F0 00

   b) from ipython .....(Roach side)
      - import wpoco
      - corr_inst=wpoco.poco_functions.Correlator (config_file='/home/gmrt
      - /Pranita/example_basicwp.cfg')
      - corr_inst.walsh_sel (walsh_sel= (0, 0))

3. Record the stability for Nowalsh using following command:
   ./stability.py -m --chan=128 example_basicwp.cfg -t STB_nowalsh.dat

4. Again with same stability plot record Normalise cross for

   a) Modulation and demodulation of wp-1 on channel-1 only.

      70 20 F0 20..... (CPLD side)

      (Wait for (10-15 sec) then give following command)

      corr_inst.walsh_sel (walsh_sel= (2, 0))..... (Roach side

   b) Modulation and demodulation of wp-2 on channel-2 only.

      70 40 F0 40..... (CPLD side)

      (Wait for (10-15 sec) then give following command)

      corr_inst.walsh_sel (walsh_sel= (0, 6))..... (Roach side)

   c) Modulation and demodulation of wp-1 on ch-1 and wp-2 on ch-2

      70 60 F0 60..... (CPLD side)

      (Wait for (10-15 sec) then give following command)

      corr_inst.walsh_sel (walsh_sel= (2, 6))..... (Roach side)

5. Compare all values of Ncx of Walsh patterns with Nowalsh Ncx for equal time stamp and find the loss in each case, using the procedure explain in part-D

Chapter 10. Appendix – C: SOP for Antenna Testing

10.1 Antenna geometric delay adjustment

1. Connect Ant-1 CH-1/CH-2 to I and Ant-2 CH-1/CH-2 to Q of roach board with power level (-15 to -18 dbm @ 400 Mhz)
2. Track any point source. (Ex.3C147, 3C286)
3. Set the following from online user.

   L- Band frequency

   GAB LO frequency

   GAB LPF

   GAB attenuation
4. Go to the following directory and follow the steps.

   cd/usr/local/etc/wpoco

   a. ssh observer@lenyadri

   b. Password : obs@gmrt

   c. > workm

   d. > ./user0.5

   e. >? 40

   f. gts ' src '
5. Modify source.hdr file using the values obtained from step 4.

   Ex.

   (Src name RA Dec LO val BW

   3C286 3.5425294950141 0.53109165984435 1450000000 400000000

   Band RF Val BB Freq

   1451000000 70000000)
6. Modify sampler.hdr

   {

   Sampler.def

SamplerId =	**Ant**	**Band**	**I/p to roach board**
SMP000 =	C11	USB-130	I
SMP001 =	C12	USB-130	Q

   }

7. Walsh off from antenna side (Walsh Modulation) using following command
   - **0x7000x 0xF000x**
     st32dig (4)

8. Walsh off in pocket correlator (Walsh Demodulation) from ipython
   - **import wpoco**
   - **corr_inst = wpoco.poco_functions.Correlator(config_file='/home/gmrt/Pranita/example_basicwp.cfg)**
   - **corr_inst.walsh_sel?**
   - **corr_inst.walsh_sel ((0,0))**

9. Initialise pocket correlator using following command.

   ./walsh_init.py -p -d0 -a12 example_basicwp.cfg

10. Run following delay_cal program.

    ./delay_cal_update_wpoco.py -f -t -d -r example_basicwp.cfg --dly_offset=0 0

11. See if phase is stable, clear and ncx as expected according to that adjust dly_offset value.

10.2 Walsh Delay adjustment using Basic Walsh Pattern

1] Delay Hunt for pol-0 / (CH-1)

1. Stop delay cal and initialise again.

2. Connect Ant-1/2 (single antenna) CH-1 to I and CH-2 to Q

3. Run ngon command from online user (control room).
   - ante 2 12 2 (Select C10 & C12 antennas)
   - cp 0;defs 0;suba 0
   - run ngon (Enable Noise Cal ON)
   - mpa 2 4 5
   - comm 29;dest 17; t3V

4. Ensure Walsh off at antenna and pocket correlator
   - **ana 0x7000x 0xF000x ..... Antenna side**
     st32dig (4)
   - **corr_inst.walsh_sel((0,0)) .....pocket correlator side**

5. Start Walsh delay hunting program for channel one only (pol-0) using basic Walsh pattern.

./walsh_delay_huntV2.py -m --chan=128 -d0 -a -p0 -b example_basicwp.cfg -t <tax file name>

Note :

nowalsh_ncx =
nowalsh_ph =

6. Ensure basic Walsh on from Antenna and pocket correlator.

- **0x70D3x 0xF0D3x .....Antenna side**
  st32dig (4)
- **corr_inst.walsh_sel((7,0)) .....pocket correlator side**

7. Press enter to continue walsh delay hunting program.(~7 min to delay hunt)

8. Note the following.

Maximum cross correlation =
fine delay value =

.....Do Not Initialise Between DH on pol-0 & DH on pol-1.....

2] Delay Hunt for pol-1 /(CH-2)

1. Ensure Walsh off at antenna and pocket correlator

- **0x7000x 0xF000x ..... Antenna side**
  st32dig (4)
- **corr_inst.walsh_sel((0,0)) .....pocket correlator side**

2. Start Walsh delay hunting program for channel two only (pol-1) using following cmd.

./walsh_delay_huntV2.py -m --chan=128 -d0 -a -p1 -b example_basicwp.cfg -t <tax file name>

Note:

nowalsh_ncx =
nowalsh_ph =

3. Ensure basic Walsh on from Antenna and pocket correlator.

- **0x70F3x 0xF0F3x .....Antenna side**
  st32dig (4)
- **corr_inst.walsh_sel((0,7)) .....pocket correlator side**

4. Press enter to continue walsh delay hunting program.(~7 min to delay hunt)

5. Note the following values obtained after delay hunt.

Maximum cross correlation =

fine delay value =

.....For Walsh Modulation on ch-1of C10 antenna only.....

6. From online user run following command:
   - **run ngof (To make Noise Cal OFF)**
   - **Connect C10 pol-1 (CH-1) to 'I' input of roach040238**
   - **Connect C12 pol-1 (CH-1) to 'Q' input of roach040238**
   - **ante 1 12 (for C10 antenna)**
   - **cp 0;defs 0;suba 0**
   - **ana 070a0x 0f0a0x (This is to enable WP-1 on both Chans of an antenna)**
   - **st32dig(4)**

10.3Find Correlation Loss

1. After delay hunt connect Ant-1 CH-1/2 to 'I' and Ant-2 CH-1/2 to 'Q'

2. Run ngof command from online user (control room).

   (Do not initialise)

3. Walsh off on both sides (Ant & pocket correlator side)

4. Run stability program / recording program in tax file for following conditions.

   ./stability.py -m --chan=128 example_basicwp.cfg -t <tax file name.dat>

   a. Nowalsh

   b. Basic Walsh

   c. Selected antenna pattern

10.4SOP for Analysis of Data in Tax Format

[1] Correlation loss test data analysis:

Recorded Tax Files for CLT test:
a)../CLT_10June/Nowalsh_wp1_wp2_Both_corrloss_11june.dat (For NG-3 noise source)
b).../ 14june_clt_crosstalk / 14june_Nowp_wp1_wp2_both__sine_NG_clt.dat (For Sine wave)
c) STB_NOWP_BWP_A15WP_13JULY.dat (Modified delay hunting test)
d) C12w_C11_CH1_3C286_wswaptest_basicwp_wp1CH1.dat (For Antenna Testing)

e) C12W_C11_CH1_3C286_2aug2016_wdebug_nowp_wp1ch1_wp1bothch_wp2ch2.dat

(Walsh channel swap test for antenna)

Ex.

Step: 1 Login into ppc using following command:

ssh -X gmrt@192.168.4.71

password: gmrttifr

Step: 2 Go to the following directory and follow the steps

cd Pranita

cd tax_prog_2Ant_V03

ls -ltr (see the program xtrgsb32_2Ant)

Step: 3 Run the following command:

./xtrgsb32_2Ant -v ../CLT_10June/Nowalsh_wp1_wp2_Both_corrloss_11june.dat

-c 128 -t 1,10000 -r C00 -n 1

Step: 4 GNU window will open then follow the steps:

Click on C00
Click on amplitude button
Click on lines button
Click on plot

Step: 5 Note the time stamp:

Ex.

No Walsh = 0 - 300
WP-1 = 350 - 600
WP-2 = 620 – 900
Both wp = 950 – 1200

Step:6 Open new Konsole window & login into Casper server using following command:

ssh pranita@casper -X
Password : pranita@2016

Step: 7 Go to the following directory:

cd Pranita
cd tax_prog_2Ant_V03

Step: 8 To find correlation loss:

1] For wp1 run the below command for equal time stamp:

./wcorr_loss_wp1.py -t.../CLT_10June/Nowalsh_wp1/wp2_Both_Corrloss_11june.dat --time__offset=350 600 50 300 --chan=128

2] For wp1 run the below command for equal time stamp:

./wcorr_loss_wp2.py -t.../CLT_10June/Nowalsh_wp1/wp2_Both_Corrloss_11june.dat --time__offset=650 900 50 300 --chan=128

3] For Bothwp run the below command for equal time stamp:

./wcorr_loss_both.py -t.../CLT_10June/Nowalsh_wp1/wp2_Both_Corrloss_11june.dat --time__offset=950 1200 50 300 --chan=128

Step: 9 For other files follow the same procedure.