CONVERSION FROM LINEAR TO CIRCULAR POLARIZATION IN FPGA IN REAL TIME

Inaugural-Dissertation

zur
Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Koyel Das

aus Kolkata, West Bengal, Indien
Köln 2013

Berichterstatter: Prof. Dr. Andreas Eckart
Prof. Dr. Anton Zensus

Tag der mündlichen Prüfung: 28.06.2013

CONTENT

ACKNOWLEDGEMENT......ix

CONTRIBUTIONS IN WORK......xi

ZUSAMMENFASSUNG xiii

ABSTRACT......1

1. INTRODUCTION......2

1.1 Motivation to form circular polarization......2
1.2 Realization of circular polarization in analogue systems......3
1.3 The digital circular polarizer project......6
1.4 Aims of the project......8

2. THEORETICAL DEVELOPMENT......9

2.1 Significance of theoretical analysis......9
2.2 Method overview......9
2.3 Instrumental phase and gain calibration......10
2.3.1 Phase equalization......10
2.3.2 Gain equalization......13
2.4 Windowing......15
2.5 Forming circular polarization......16
2.6 Performance limitations......16
2.7 Phase stability of the analogue receiver chain......17
2.8 Expected polarization purity......19

3. PRELIMINARY TEST OF ALGORITHM......20

3.1 Phase and gain equalization and windowing......20
3.1.1 Experiment to collect test data......20
3.1.2 Description of spectral characteristics obtained by processing data in MATLAB......21
3.1.3 Phase equalization......24
3.1.4 Gain equalization......25
3.1.5 Windowing......27
3.2 Preliminary questions to be answered......28
3.3 Conclusion...... ...29

4. IMPLEMENTATION DETAILS......30

4.1 Overview of main logic blocks of the digital circular polarizer......30
4.2 Conclusion......32

5. EXPERIMENTS AND RESULTS......33

5.1 Design logic verification and simulation results....33
5.1.1 Lab setup for collecting test data....33
5.1.2 Time series obtained from the digital oscilloscope in the experiment....34
5.1.3 Power spectra from the two channels in the experiment....34
5.1.4 Simulation results from the design logic....35
5.2 Verification of polarization purity....41
5.2.1 Signal flow through the setup....42
5.2.2 Measurement details....42
5.2.3 Results....45
5.2.4 Discussion....46
5.3 Conclusion....48

6. APPLICATIONS IN RADIO ASTRONOMY....49

6.1 Significance of the digital circular polarizer in radio astronomy....49
6.2 Different astronomical phenomena generating polarization....49
6.2.1 Cyclotron and synchrotron emission....49
6.2.2 Plasma frequency and plasma oscillations....50
6.2.3 Zeeman effect....50
6.2.4 Thompson scattering....50
6.2.5 Brewster angle effects....51
6.3 Depolarization effects....51
6.3.1 Depolarization due to Faraday rotation....51
6.3.2 Bandwidth depolarization....51
6.3.3 Beam depolarization....51
6.4 Explorations and underlying techniques requiring polarimetric observations in VLBI....52
6.4.1 Magnetic field studies by the SKA....52
6.4.2 Studies of Sgr A*....56
6.4.3 Studies of circular polarization in AGN....59
6.4.4 Studies of the Perseus cluster....61
6.5 Conclusions....66

7. REMAINING TOPICS OF DISCUSSION....68

7.1 Brief overview of the antenna system in radio telescopes....68
7.2 Polarization ellipse: response of radio telescopes in ideal cases....68
7.2.1 Retrieving linear polarization in the sky from elliptical response of antenna....71
7.3 Polarization ellipse: response from imperfectly oriented dipole elements....73
7.4 Analysis of the ellipse in the experiment described in section 5.2....74
7.5 Effects of D-term in the received voltage by the crossed dipole.... 75
7.5.1 Discussion on relation between circular D-terms and ellipticity....76
7.6 Conclusion.... 76

FUTURE WORK....77

APPENDICES:

APPENDIX A: BASIC POLARIZATION DERIVATIONS....78

APPENDIX B: DESIGN AND CODE DESCRIPTION OF LOGIC BLOCKS OF CHAPTER 4........85
APPENDIX C: VHDL CODES OF SELECTED MODULES FROM APPENDIX B........127

REFERENCES........153

ERKLÄRUNG........155
DECLARATION........156

LEBENSLAUF........157

ACKNOWLEDGEMENT

The successful completion of this PhD thesis has entailed a lot of support and cooperation from the people who are involved directly or indirectly in my PhD project and also sometimes from the people who are not involved in the project at all. First of all, I would like to thank my supervisor, Dr. Alan Roy of MPIFR, without whose cooperation, the project wouldn't have been completed. The project's basic algorithm, which came from him, seemed to be simple in the beginning but later I and my supervisor realized the length of the project taking all technical challenges into account. Alan solved all technical problems that were causing road block in the commencement of the project. He has provided the best supervision along with being an excellent discussing partner. He has organized regular thesis committee meetings to keep everyone, involved in the thesis, informed about the work progress and also to plan timing of the remaining work. Since there are too many contributions, in work, on his part, I will include a separate paragraph on contributions in work towards my PhD thesis. I would also want to thank my official supervisors, Prof. Dr. Andreas Eckart of University of Cologne and Prof. Dr. Anton Zensus of MPIFR for being there in my PhD related matters. I simultaneously thank IMPRS for funding my PhD projects.

There were many difficult times during the commencement of my PhD and all three of the above mentioned people have supported me in one way or the other so that the project was completed successfully. Andreas Eckart has attended all my thesis committee meetings and his presence in all those meetings enhanced my confidence about my performance in the project. I am thankful to Andreas Eckart for taking the effort of traveling from Cologne to Bonn to attend my thesis committee meetings. I was able to be fully optimistic towards completion of my thesis due to the presence of Andreas Eckart in my PhD related matters.

Anton Zensus has supported my project by being there towards completion of the project. He has also attended thesis committee meetings in the beginning. I thank Anton Zensus and other members of IMPRS for selecting me as a PhD student. I am thankful to Anton Zensus for providing me an interrupt in work after my mother's demise as it was really necessary when I lost my most beloved one suddenly in the mean time of my PhD; my mother's death was the most disastrous happening for me that could happen. Anton Zensus provided financial support for my stay in Bonn during the interruption period, which was extremely kind.

Due to the support from Alan Roy, Andreas Eckart and Anton Zensus, I could proceed with my thesis writing with ease. My PhD work has helped me to keep my mind away from the feeling of the loss of my mother and also helped me to cope with the situation; thanks to Alan for discussing work with me continuously throughout my PhD, which kept my mind in the work only.

After the above mentioned three people who are involved in my thesis, I would thank Dr. Reinhard Keller, also involved in my project, for organizing the experiment in my project through members of the RF lab and attending my thesis committee meetings sometimes; I would thank Thomas Berenz of RF lab for doing the setup in the experiment of my project as per the guidance from Alan. I am extremely thankful to Thomas Berenz for translating the abstract of this thesis to German language as required by the university of Cologne. Then I thank Prof. Dr. Brian Corey for taking interest in my

thesis and also for going through the first draft of this thesis. Then I thank Dr. Dorothea Samtleben who reviewed my paper before publication as my internal referee. Next I would thank Dr. Gino Tuccari, also involved in my project, for discussing his DBBC project and this project with me. Next I would thank Dr. Stefan Hochgürtel of the digital lab for answering a few of my questions in the project. I would thank Prof. Bernd Klein for attending my thesis committee meetings at crucial times. I would also thank Prof. Dr. Rajaram Nityananda for answering three questions and Prof. S. Ananthakrishnan. Also thanks to Prof. Dr. John Black and Dr. Gary Smith Jonforsen for their encouragement. Also thanks to my father and sister for their support. Also thanks to Simon Pott. Finally, I dedicate my PhD thesis to Dr. Alan Roy, Prof. Dr. Andreas Eckart and Prof. Dr. Anton Zensus for keeping up their support towards completion of my PhD and to my mother who had given her whole life for my education.

CONTRIBUTIONS IN WORK

It took a lot of effort on my part and on my supervisor's, Dr. Alan Roy's part to finish this project successfully. My supervisor was always there to solve whenever there was a problem in my project. He has collaborated with me in the best possible manner towards completion of this PhD thesis. The effort started with the verification and theoretical justification of the basic algorithm he laid out to me regarding the digital circular polarizer. I discussed thoroughly with Alan each and every stage of signal processing to explore the exact steps of signal processing towards formation of pure circular polarization from two orthogonal linear polarization inputs. I laid out a theoretical justification of his algorithm and he went through that and we discussed thoroughly each and every step of the theory. We collected data from an experiment described in section 3.1.1 and 5.1 of this thesis for me to check the steps of the algorithm. I also played the main role in defining the data collection method in this experiment. I as per Alan's suggestion simulated the logic and checked the correctness of the algorithm at each stage and also showed the results from each stage to Alan.

After the theoretical justification was complete, I as per Alan's suggestion proceeded towards firmware development where first I had to design logic blocks to convert from two orthogonal linear polarization to two hands of circular polarization in real time. After design of each block, corresponding VHDL code was written by me. Alan discussed with me during block design suggesting methods sometimes, which I implemented. Stefan Hochgürtel of the digital lab also discussed with me sometimes about the block design, which was also useful. After writing the codes for different logic blocks and verifying those in simulation, I needed to connect all the blocks together to have the digital circular polarizer ready for testing. I also tested the the implementation of the individual blocks in Xilinx ISE with timing constraints met so that there is no problem later on in implementing the connected logic. In the design I used many logic blocks developed by Xilinx and while connecting them I faced difficulties since I was not getting the correct simulation results for one of the Xilinx logic blocks. Alan solved this problem by corresponding with Xilinx and correcting the technical problems, which were causing the road block. Alan also solved other many more technical problems that were snagging the project.

After I connected all the logic blocks together, I tested the connected logic in simulation by using self generated data streams. After I saw the simulation results were correct, I reported to Alan and we decided to test first the complete design using the same data obtained from the experiment described in section 3.1.1 and 5.1 of this thesis where I took the leading role. I ran the simulation by using that data stream as the input to the digital circular polarizer and checked the correctness of the design in each stage. I showed Alan the simulation results from each stage and we discussed the observed facts with Alan supervising me in observing details. We observed together the correctness of the design. The results that validated the design are depicted in section 5.1.4 of this thesis. After we were sure that the design logic was correct, we decided to perform the second experiment (given in section 5.2 of this thesis) proposed in our theory to validate the algorithm finally.

We talked to Reinhard Keller for arrangement of the setup for the experiment. Reinhard Keller and Thomas Berenz of the RF lab did the setup for us as per Alan's guidance and I noticed all the details of the experiment along with discussing with Alan from time to time as Alan guided the experiment. We took some measurements. After the data were collected in the way described section 5.2.1 of this thesis, we started arrangements for processing the data using the firmware running in a simulator for verification prior to running the firmware in hardware in the FPGA. The data set was huge and very high computing power was needed along with an advanced simulation tool (ModelSim). Alan purchased an advanced simulation tool through IMPRS for me that can handle big designs and that run much much faster than the normal simulation tools. I broke the design into two parts; each of the two parts was replicated to be fed simultaneously with a part of the data collected to the simulator; all the parts of the collected data for all the parts of the replicated designs comprised of equal number of data points. I ran the simulation, which required a few days. The details of simulation is given in section 5.2.2 of this thesis. After simulation was complete I reported to Alan and we started analyzing the results with Alan supervising me in finding details. Most of the times during analysis I learned many details from Alan and accepted his reasoning after thinking thoroughly through them. The observed facts are described under section 5.2.3 of this thesis.

Then we were sure of the correctness of the algorithm and of the instrument and I proceeded towards implementing the whole logic in Xilinx ISE taking all constraints like the timing into account. I truncated bits in stages of the logic as per Alan's supervision and implemented the whole design with all constraints met in Xilinx; then only connecting the design to DBBC and loading of the design in FPGAs were remaining. The information on implementation of different parts of the design is given in section 4.3 of this thesis. The total time taken to design the digital circular polarizer addressing all technical difficulties was nearly one and a half years. Thanks to Alan as without his help the technical problems couldn't be solved.

We decided to publish the results and it took around three to four months in writing and polishing the paper. I wrote a first draft of the paper and gave that to Alan for corrections. Alan corrected the paper and gave me modifications at several places in the paper until he found the paper was ready for submission. He also included about two pages initially written by him with me discussing the detail and doing minor modification at one place. The experiment described in the paper (second experiment) and its results were written by us with Alan adding details about the experiment as he was the one who took the leading role in this experiment. He also added other details at several places in the paper. Along with reviewing our paper thoroughly and suggesting changes, he also suggested language modifications, which I followed. He also contributed towards polishing the paper and I followed his suggestions. Dorothea Sambtleben, my internal referee, helped in polishing the paper too. Gino Tuccari and Reinhard Keller also reviewed the paper. After the paper was published, I and Alan tried to see if we could get FPGA boards (DBBC boards) for loading the firmware but they were unavailable and hence we couldn't proceed further.

Next came thesis writing; Alan taught me the proper style of writing a thesis by teaching the guidelines that should be followed like the word limits. He also showed me the writing style in the first chapter where I had just written the points only properly in the beginning; he gave modifications again and again and I modified accordingly until it reached the level of satisfaction and I got the idea about how to write the later chapters and I followed the same way of writing in the rest of the chapters. I have then written the rest of the chapters in a way to entail minimum effort on Alan's part in doing corrections. He also gave me literature to read for the first and fifth chapters. He has done a thorough correction of all the chapters and I have implemented all his corrections. I am thankful to Alan for being there as a strong support while I was writing the thesis. Even though I have written the whole thesis, Alan's corrections were necessary and his support for completing the thesis was a source of moral strength to me. Finally, I am grateful to Alan Roy, Andreas Eckart and Anton Zensus for keeping positive attitude about completion of my thesis from the time when nothing were written up; it gave a lot of support to me on the basis of which I could start writing it up. I expect that the developed firmware gets implemented successfully in future to be used in radio telescopes.

ZUSAMMENFASSUNG

Zukünftige Radioastronomische Empfänger werden über einen erweiterten Frequenzbereich und damit auch eine höhere Bandbreite (Oktave) verfügen, um die Empfindlichkeit zu steigern und mehr Flexibilität bei der Auswahl des Frequenzspektrums zu haben. Dies stellt hohe Anforderungen an das Design eines analogen Frontends. Um bessere Polarisationseigenschaften zu bekommen, ist eine flacher Phasenverlauf über immer größere Bandbreiten nötig, was am einfachsten mit digitalen Methoden zu erreichen ist. Hier besitzt man die Möglichkeit eine zirkulare Polarisation mit perfekter Polarisationsverteilung über eine vorgegebenen Bandbreite zu formen, da mit digitalen Mitteln einfacher eine quadratur Phasenverschiebung zu erzeugen ist. In analogen Systemen ist die nötige Phasenverschiebung nicht exakt, sobald man von dem Frequenzpunkt abweicht für den das System entworfen wurde. Im Gegensatz dazu besteht bei digitalen Systemen die Möglichkeit, die exakte Phasenverschiebung durch Verrechnung der Signalvektroren jedes einzelnen Frequenzpunktes innerhalb des Frequenzbandes zu erzeugen. Daraus resultiert dann eine perfekte quadratur Phasenverschiebung innerhalb des kompletten Bandes. Der schnelle Fortschritt bei Field Programmable Gate Arrays (FPGA) bringt neben der nötigen Rechenleistung, einem günstigen Preis und der Portierbarkeit auch die Möglichkeit zur einfachen und schnellen Rekonfiguration des Systems, wodurch das Formen einer zirkularen Polarisation mit digitalen Mitteln auch praktisch sinnvoll wird. Dieses System kann dann für breitbandige Polarisationsmessung genutzt werden.

Zirkulare Polarisation wird aus Geometrischen- und Stabilitätsgründen bei der Radiointerferometrie mit sehr langen Basislängen (very long baseline interferometry, VLBI) genutzt. VLBI wird häufig bei der Untersuchung der Polarisation von Radiowellen verwendet. Die Polarisation dieser Wellen wird durch Syncrotron Effekte, Zeeman Effekte innerhalb von Atomen und Molekülen, Zyklotron Strahlung und Plasma Schwingungen in der solaren Atmosphäre hervorgerufen. Außerdem findet VLBI Anwendung bei Methoden der Synthese der Rotationsmessung, welche verwendet werden kann um die magnetische Feldstärke zu ermitteln. Weiterhin kann durch Beobachtung verschiedener Wellenlängen die Richtung des magnetischen Feldes bestimmt werden. Daher würde ein digital arbeitender Polarisator eine Vielzahl von Anwendungen in VLBI Systemen finden.

In dieser Arbeit untersuche ich die Effizienz eines digitalen zirkularen Polarisators. Wir entwarfen einen digitalen zikularen Polarisator in dem die Zwischenfrequenzsignale eines Empfängers mit ursprünglich linearer Polarisation abgetastet wurden. Diese wurden nach dem Abtasten in eine zirkulare Polarisation gewandelt. Die frequenzabhängige Phasen- und Amplitudendifferenz des Systems wurde mit Hilfe eines zugeführten Rauschsignals bestimmt. Dieses wurde auf beide linearen Polarisationen gegeben um die Übertragungsfunktion der beiden Polarisationskanäle abzugleichen. Dieser Abgleich wurde mit 512 Frequenzpunkten über eine Bandbreite von 500 MHz durchgeführt. Die zirkulare Polarisation wurde durch eine quadratur Phasenverschiebung und anschließende Summation des Signale erzeugt. Hierbei erzeugten wir über das ganze Band eine Polarisationsreinheit von -58 dB, was einem D-Wert von 0.0012 entspricht. Dieses D-Wert eine Obergrenze

Diese Technik ermöglicht die Entwicklung eines zirkularen Polarisators für VLBI, der mit einem breitbandigen radioastronomischen Empfänger mit linearer Polarisation arbeiten kann

ABSTRACT

Radio astronomical receivers are now expanding their frequency range to cover large (octave) fractional bandwidths for sensitivity and spectral flexibility, which makes the design of good analogue circular polarizers challenging. Better polarization purity requires a flatter phase response over increasingly wide bandwidth, which is most easily achieved with digital techniques. They offer the ability to form circular polarization with perfect polarization purity over arbitrarily wide fractional bandwidths, due to the ease of introducing a perfect quadrature phase shift. In analogue systems the quadrature phase shift is not accurate in the regions away from the design point or frequency. In digital systems on the contrary, it is possible to introduce the exact quadrature phase shift vectorially to each frequency point in the band thus producing a perfect quadrature phase shift throughout the band. Further, the rapid improvements in field programmable gate arrays provide the high processing power, low cost, portability and reconfigurability needed to make practical the implementation of the formation of circular polarization digitally. It will be possible to carry out broadband polarization observations.

Circular polarization is used in very long baseline interferometry (VLBI) due to geometrical and stability considerations. VLBI is often used to explore polarization of radio emission, which often occurs due to synchrotron mechanism, Zeeman effect in atoms and molecules, cyclotron radiation and plasma oscillations in the solar atmosphere. Also VLBI finds application in methods like rotation measure synthesis that can be used to find the magnetic field strength and whose multiwavelength observations determine the direction of magnetic field. So a digital circular polarizer would find a considerable application in VLBI systems.

Here I explore the performance of a circular polarizer implemented with digital techniques. I designed a digital circular polarizer in which the intermediate frequency signals from a receiver with native linear polarizations were sampled and converted to circular polarization. The frequency-dependent instrumental phase difference and gain scaling factors were determined using an injected noise signal and applied to the two linear polarizations to equalize the transfer characteristics of the two polarization channels. This equalization was performed in 512 frequency channels over a 500 MHz bandwidth. Circular polarization was formed by quadrature phase shifting and summing the equalized linear polarization signals. I obtained polarization purity of -58 dB corresponding to a D-term of 0.0012 over the whole bandwidth. This value of D-term is an upper limit.

This technique enables construction of broad-band radio astronomy receivers with native linear polarization to form circular polarization for VLBI.

CHAPTER 1

INTRODUCTION

In this chapter I provide the initial motivation to form circular polarization digitally in real time, which when implemented in practice would allow new receivers to have native linear polarization for broad frequency coverage. I simultaneously discuss the realization of circular polarization in analogue systems giving examples. Then I discuss the problem of obtaining a perfect 90º phase shift required to be introduced to the phase of one of the two linear polarizations for the formation of circular polarization in existing analogue systems providing an example from Effelsberg radio telescope and finally I describe our developed digital circular polarizer and provide the aims of our project solving the problems.

1.1 Motivation to form circular polarization

Circular polarizers play important roles in modern communication systems including those in radio astronomy. To obtain higher sensitivity and frequency coverage for spectral line observations, the radio antennas are moving to broad-band feeds and extremely broad bands are most easily realized with linearly polarized feeds due to the difficulty of producing 90º phase shift accurately over wide bandwidth. However, circular polarization is simplest for the application of very long baseline interferometry (VLBI), which enables astronomical sources to be resolved with sub-milliarcsecond synthesized beam widths, since linear dipoles do not generally remain parallel to each other in a global array due to different parallactic angles at different stations when observing the same source, causing loss of coherence in the cross-correlation products formed between stations. That loss could be recovered were one to compute also the cross-polarization cross correlation products to retain all information (doubling the correlator power needed), or one could rotate the receiver packages at each station to keep the dipoles parallel (requiring mechanical rotators). In contrast, use of circular polarization causes the parallactic angle differences between stations to add a simple phase rotation angle to the measured visibility, which can be predicted from the known observation geometry and subtracted in post-processing.

Circular polarizers with broad bandwidths have been realized in the past with a number of methods. Most common are as follows: 1) The septum polarizer originally designed by Davis et. al. (1967) and modified by Chen & Tsandoulas (1973); they introduced steps in the septum for better performance in terms of axial ratio and input port isolation. 2) Boifot et. al. (1990) is the origin of Boifot junction; they presented a broadband OMT (orthomode transducer used to couple out the two linear polarizations from waveguide into coax); an isolation better than 50 dB and a return loss less than -20 dB was achieved. 3) Linear quad-ridge OMT followed by a 90º hybrid junction or preceded by a corrugated waveguide phase shifter (Simmons 1955); these convert the linears into circular; Simmons provided analytical and experimental results of producing a differential 90º phase shift produced by phase delay and phase advance of two fundamental modes; it is done by loading the transmission line equivalent circuit of a rectangular or square waveguide with capacitance and inductance respectively; Srikant (1997) explains that corrugated phase shifter, which is not a polarizer on its own, can be used in conjunction with an OMT to form circular.

All are analogue techniques and produce a perfect 90º phase shift and hence perfect polarization purity at only one, two, or three frequencies and the phase errors grow larger at frequencies away from those design points, which ultimately limits the bandwidth of the devices. In contrast, digital techniques offer the possibility to produce an accurate 90º phase shift over broad bandwidths, but this potential has not yet been fully developed. One example is the Westerbork synthesis radio telescope, which converts native linear polarization to circular polarization by a combination of analogue and digital techniques for VLBI. During down conversion of the orthogonal linear signals, the 90º shift is added to the (analogue) LO for one polarization. After analogue-to-digital conversion, the (2 bit) signals are summed and differenced to form circular polarization, with a weight that corrects for the average receiver gain differences. The weights are determined by a separate measurement using a calibration noise source in each frontend and are updated every 10 s. This system operates on a bandwidth of 20 MHz, yielding one phase and amplitude correction for each 20 MHz of bandwidth (Boss 2007, private communication). The Westerbork system uses the pre-existing correlator and analogue phase rotation in the LO system, but most VLBI stations lack this equipment and so another, more general, solution is needed.

1.2 Realization of circular polarization in analogue systems

Conventionally, analogue circular polarizers are used to obtain circular polarization and all of them use some technique to impart a 90º phase shift between orthogonal linear field components.

An example is the very compact septum polarizer first devised by Davis et al. (1967). They used a sloping septum to obtain the polarization components and their phase relationships. Later Chen & Tsandoulas (1973) introduced steps in the septum to avoid reflection off the discontinuity that the septum creates in the waveguide. Here I provide an example of the septum polarizer described by Wade (2003), which is realized with a rectangular input waveguide containing both circuit input ports and two physically separated waveguide output ports that are physically quadratic (for our application). Fig.1.1 illustrates the septum polarizer. A septum polarizer can be considered as an equivalent four port

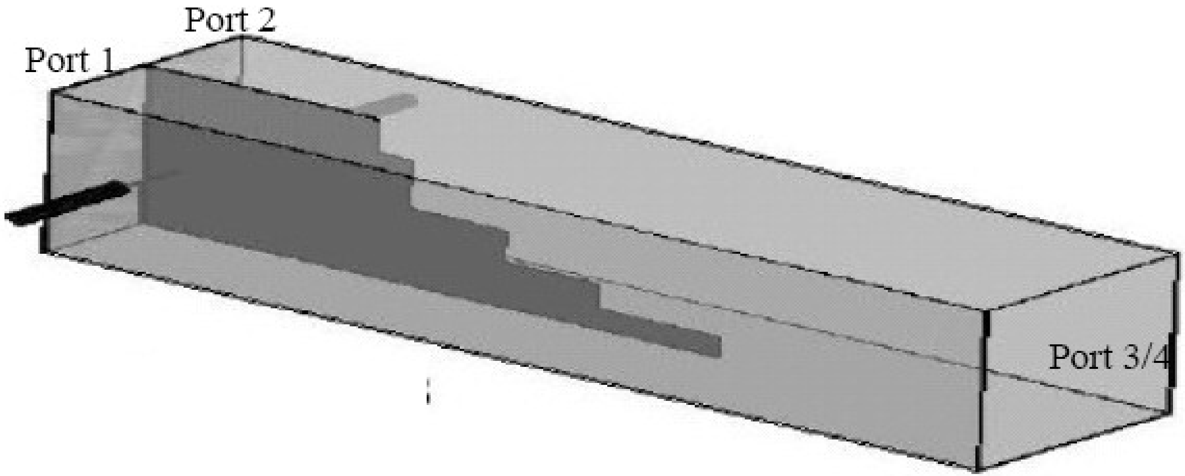


Fig. 1.1: Septum polarizer with four ports. Ports 3 and 4 are present in the same waveguide port and is only distinguishable conceptually. Ports 1 and 2 are physically separated (Wade 2003). The radiation enters port 3 and port 4 through a horn antenna in quadrature i.e there is a quadrature phase difference between the two circular polarization components and the two linear polarizations are obtained from port 1 and port 2. The thick black lines at the left side of the polarizer are output dipoles to extract the linear polarizations coming out. This septum polarizer responds to circular polarization in the sky and hence is relevant to our application.

microwave circuit. The two input ports (3,4), which are only conceptually distinguisahble, are contained in the same physical rectangular waveguide port and the two output ports (1,2) are physically separated with a fin. Let us consider a circularly polarized wave comprising of two polarization components with a 90° phase difference, entering the aperture (ports 3, 4). One of the components is parallel to the septum and the other is perpendicular to the septum. The septum divides the parallel component equally, which passes to the two rectangular output waveguides. The septum changes the cutoff frequency of the perpendicular component thereby shortening the wavelength of the perpendicular component. This means that the section of the waveguide containing the septum is electrically longer for the perpendicular component as compared to the parallel component. A path difference of $\lambda/4$ will render the vertical and horizontal component to be in phase at the output. However, there can still be phase differences as the band deviates from the centre frequency for which the septum polarizer is manufactured. The two output ports are isolated from each other. Constructive or destructive interference of the field components occurs at either side depending on the sense of circular polarization. The operating frequency is near the waveguide cutoff which leads to difference in electrical lengths between the two components. A circular waveguide can be used in place of the rectangular waveguide in Fig. 1.1 as the circular waveguide input is very useful especially while receiving signals from circular waveguide horn antennas having both LHC and RHC polarizations.

However, it is very difficult to obtain good isolation between the two orthogonal components. Fig. 1.2 illustrates an orthomode transducer developed by Dunning (2002). Good isolation prevents leakage of

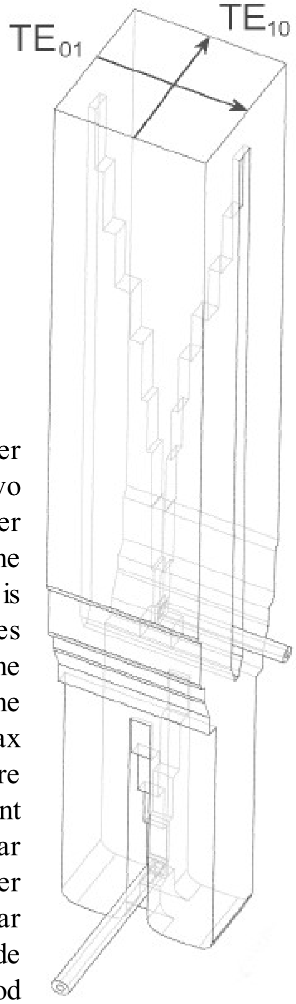
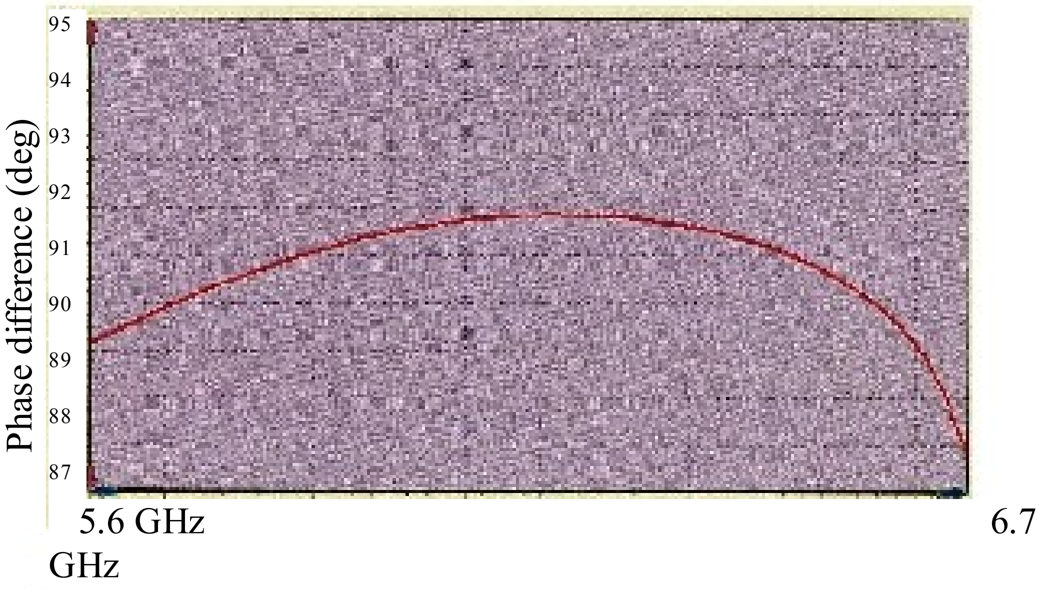


Fig. 1.2: Double ridged orthomode transducer by Dunning (2002). The two orthogonal linear components enter the square waveguide through the top end. One component, which is parallel to the plane of the ridges gets concentrated between the ridges at the center of the waveguide and is removed by a coax through one of the fins in the square waveguide. The other component reaches the square to rectangular transition and is removed by another coax through a fin in the rectangular waveguide. This orthomode transducer provides very good isolation between two polarizations.

one polarization component into another thereby facilitating the production of pure circular polarization without any ellipticity. The orthomode transducer developed by Dunning has good isolation between two polarizations and low insertion loss suitable for broad band radio astronomy receivers. It takes the two orthogonal field components excited at one common port and separates them. In that double ridged OMT, one polarization is concentrated between the ridges at the centre of the square waveguide and is removed by a coax inserted through one of the fins in that square waveguide. The other polarization is unaffected by the ridges and reaches a square to rectangular waveguide transition from where it is removed by another coax, which is inserted through one of the fins in the rectangular guide. A 90° hybrid could be connected to the output coax ports in order to convert to circular polarization. The excellent OMT developed by Dunning has achieved isolation better than 50 dB between ports and an insertion loss better than 0.3 dB for a 47 % fractional bandwidth.

Now, I will proceed to demostrate the difficulty in producing a quadrature phase shift between two orthogonal linear polarizations to obtain the circular polarization in existing analogue systems.

It is extremely difficult to obtain perfect 90º phase shifts using analogue polarizers. The quadrature phase shift is perfect only near the centre frequency. Fig. 1.3 shows the typical phase difference between the two orthogonal polarization components as measured for the polarizer in the 5 cm receiver at Effelsberg. The plot is provided to me by Alan Roy.



Keller. R. “Abgleich von Zircular Modenweichen”.

Fig. 1.3 : Phase difference between two orthogonal polarization components for the polarizer in the 5 cm receiver at Effelsberg. The phase difference varies across the band from 88.2º at the lower edge to 85.8º at the upper edge. It is around 90.9º near the centre frequency, 90º was chosen to be at two frequencies in order to distribute the errors and broaden the bandwidth. It is very difficult to obtain exact 90º phase difference over broad bands.

This is the phase difference between the two orthogonal components of the circular polarization to which the polarizer responds in the receiving system. It can be seen from the plot that the phase difference is around 90º at two frequencies for which the polarizer was optimized and it deviates as we move away in frequency towards the edges. The phase difference varies from 90.9º at the centre to

88.2º at the lower band edge and to 85.8º at the upper band edge. Hence it is very difficult to obtain an exact 90º phase shift over broad bands.

After discussing the analogue techniques I will now demostrate our digital technique to produce circular polarization from two linear orthogonal polarization components.

1.3 The digital circular polarizer project

In this project I have tried to form an exact 90º phase shift for a 200 % fractional bandwidth. I have tested that the polarizer works for 97 % fractional bandwidth. The aim was to obtain pure circular polarizations correcting all system imperfections leading to perfect 90º phase shift, which is one of the challenging aims of modern circular polarizers. This project is based on the idea of Alan Roy that if the circular polarization is formed in the digital domain then it might be possible to obtain flatter phase response over broad bands. I explored this by performing some initial simulations on test data before proceeding towards actual instrument development.

I have developed a self-contained digital processing system in which the correlator, channel equalization, phase rotation, gain scaling, quadrature phase shift and summation to form circular polarization are contained in a stand-alone unit. I and Alan arrived at the basic data flow shown in fig. 1.4. It accepts two intermediate frequency inputs with orthogonal linearly polarized signals, each of 500 MHz bandwidth and subdivides the band down to 1 MHz resolution. In each 1 MHz piece it measures the phase and amplitude differences between the orthogonally polarized channels using a

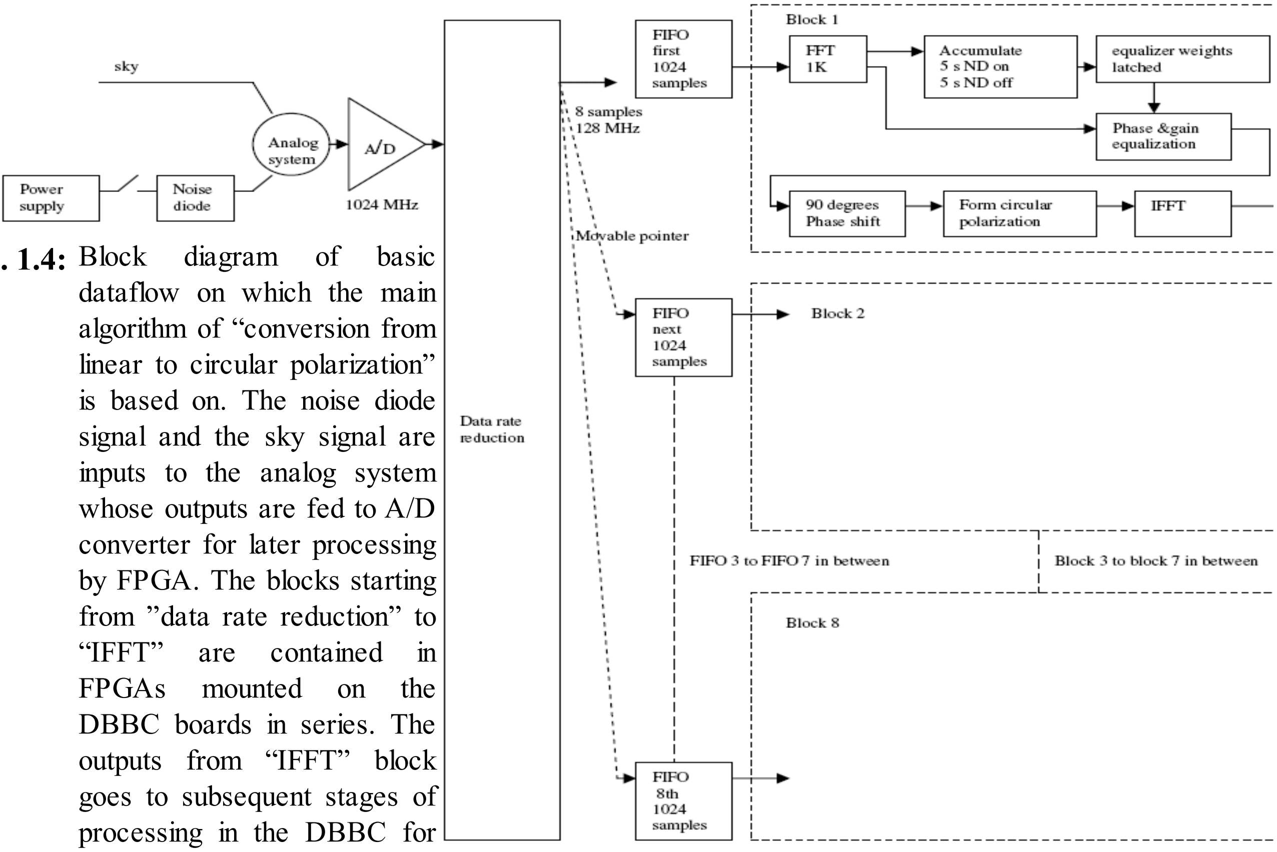


Fig. 1.4: Block diagram of basic dataflow on which the main algorithm of "conversion from linear to circular polarization" is based on. The noise diode signal and the sky signal are inputs to the analog system whose outputs are fed to A/D converter for later processing by FPGA. The blocks starting from "data rate reduction" to "IFFT" are contained in FPGAs mounted on the DBBC boards in series. The outputs from "IFFT" block goes to subsequent stages of processing in the DBBC for VLBI data acquisition. Accumulation is performed for 8.4 s to obtain good signal to noise ratio for accurate phase determination.

calibration noise source in the front end that is common to both polarizations. It then uses those measurements to equalize the channel phases and amplitudes during observations. After equalization it introduces an ideal 90^o phase shift into one polarization channel and forms sum and difference outputs that respond to orthogonal circular polarizations at the input. I perform an FFT to process the data in frequency domain for the simplicity to work in that domain.

We decided to use FPGAs since they have high computing power and are reconfigurable and portable. I took sampled time domain data and processed them in MATLAB as a quick preliminary test of the idea behind it, which came from my supervisor, before exploring the algorithms in detail. The results are shown later in the thesis. At this point I verified that the principle was sound and that the implementation could begin. The data flow shown in the figure is for each polarization state.

The signal received by the dipole is sampled using an analogue to digital (A/D) converter for later processing in FPGAs (DBBC boards, which are described later in this chapter, connected in series). Each eight consecutive samples coming in series at 1024 MHz are converted to eight parallel samples at 128 MHz by the data rate reduction block. There are eight FIFOs running in parallel to receive data from the data rate reduction block at the rate of 128 MHz to handle the speed of 1024 MHz. Until this point the blocks duplicate for the other polarization state.

After the above mentioned logic there are eight blocks, “block 1” to “block 8” each of which are connected to a pair of FIFOs only one of which is shown in the figure. The logic blocks inside these blocks are shown in the figure. The FFT performs a Fourier transform on the input polarizations which are fed to the two input channels of the FFT (I perform an FFT of 1024 samples to produce spectra with 1 MHz channel spacing and the whole algorithm involves computation of time separated spectra in the frequency domain) and the outputs are decoded and fed to power spectra accumulators and cross power spectra accumulators. The accumulators run for 8.4 s with noise diode, which is used as a calibration source, on and 8.4 s with noise diode off. The on-state accumulation and the off-state accumulation are carried out in separate accumulators.

In the next block the phase and gain scaling factors are determined and latched to be read out during equalization. After equalization of the FFT outputs by the block “phase and gain equalization”, one of the two polarizations is 90^o phase shifted and the circular power spectra are formed in the next block, which are then converted to time domain to feed later stages of DBBC for VLBI data acquisition. There is a noise diode control signal generated inside the FPGA (not shown in the figure). It goes high at the same time when the on-state accumulator is enabled and it goes low when the off-state accumulator is enabled. This signal is brought out of the FPGA and it controls the TTL (transistor-transistor logic) signal that controls the switching on and switching off of the noise diode signal.

After discussing the basic data flow I will now demonstrate the motivation to implement this design in FPGAs rather than using the conventional orthomode transducers to convert linear polarization to circular polarization.

Digital systems offer the ability to process continuous data in real time implementing automatic data processing algorithms. Data transmitted digitally are more resistant to external interference and hence digital devices supersede analogue counterparts at least where speed and signal purity are matters of concern. With the advent of logic devices like ASICs and FPGAs, it is possible to implement many complex algorithms, which would have been impossible otherwise and they also offer the user ease of replication compared to analogue systems. For these reasons our system to produce circular polarization generates almost perfect orthogonal field components in real time.

I have three options for implementation in the digital domain: 1) Software implementation, 2) ASIC (application specific integrated circuits) and 3) FPGA (field programmable gate arrays). Practical software implementation would require huge computing power to process the data in real time and it is difficult to have such computing power at the VLBI stations where I have aimed to form the circular polarization in real time. ASICs cannot be reconfigured and reconfiguration can be necessary in the future to increase bandwidth. ASICs also have large development and fabrication costs and for our application FPGAs have lower cost. Thus the most obvious option was to use FPGAs. The implementation in Xilinx software is complete with all timing constraints met and hence all the technical difficulties have been addressed. Also FPGA-based VLBI data acquisition systems, such as the Digital Base-Band Converter (DBBC, Tuccari 2004) are being deployed in many radio telescopes. The DBBC is a signal conditioning device used in VLBI observations and has ADCs and FPGAs for digital downconversion, digital filtering, and outputs 2 bit depth to VLBI recorders. It has the capability to sample at a rate of 1024 MHz clock rate. It produces eight parallel digital samples each of eight bits as intermediate output signals internal to the DBBC that are contained in eight buses each running at a clock rate of 128 MHz. These eight intermediate signals are the inputs of our project, which will also be placed inside the DBBC.

I have truncated bits in stages of the data flow in order to fit it inside the DBBC, as per Alan's Guidance, which has a predefined number of input output pins. A new DBBC board is being developed which would have a Virtex 7 mounted on it. So we have decided to implement the design in Virtex 7, which would encompass the bigger parts of the design and fewer number of FPGAs would be required. Thus it is best to use the DBBC with Virtex 7 as then the whole logic inside the FPGA will be utilized efficiently, that is, less approximations will be required since number of logic elements are much more than in Virtex 5 on which the design is already tested.

1.4 Aims of the project

This thesis acquaints the reader with the theory and derivation for obtaining instrumental phase and gain correction factors and applying them to the two received orthogonal linear polarizations to form phase and gain calibrated left hand circular (LHC) and right hand circular (RHC) polarizations. It explores limitations of system performance due to the most influential factors, which are the D-terms and receiver instabilities and explores the requirement for periodic recalibration to remove their detrimental effects on polarization purity. It also demonstrates the preliminary simulation of the idea to develop an FPGA based digital circular polarizer. It describes the digital signal processing in FPGA and the experimental verification of the technique to show that good polarization purity is obtained. It also describes the significance and application of the digital circular polarizer in VLBI in radio astronomy and finally visits the remaining topics to be discussed..

CHAPTER 2

THEORETICAL DEVELOPMENT

In this chapter I first provide an introductory paragraph on the significance of theoretical analysis. Then I provide an overview of the method to convert from linear to circular polarization. Next I describe phase and gain equalization after which I provide details on windowing of the resulting phase and gain equalization parameters. Then I describe the method to form circular polarization. Next is performance limitations due to D-term and temporal instability of receivers and subsequently I provide details on the stability of analogue receiver chains in Effelsberg telescope. Next I estimate the resulting polarization purity considering the sensitivity of polarization leakage to phase errors as obtained under performance limitation due to D-terms and existing phase errors in analogue receiver chains in Effelsberg. Finally I provide introductory text on the next chapter.

2.1 Significance of theoretical analysis

After exploring the preliminary idea to form circular polarization digitally, I proceeded to prove that this could be implemented in practice. Hence, it was needed to verify theoretically that the idea is sound so that no question is raised on its validity. All the formulations done by me unless specified are shown in order to manifest the understandability and to negate any ambiguity against the idea. The digital signal processing shown here encompasses the basis for FPGA implementation confirming that FPGA implementation could begin unquestionably. We have also taken into account the performance limitations due to D-terms and variability of analogue receiver transfer characteristics. Taking all the factors into account we came to the conclusion that excellent polarization purity can be achieved by this method. The channel width and the number of channels are variable and can be adapted to one's need. We decided to keep the numbers optimum covering 200 % fractional bandwidth (500 MHz) and keeping 1 MHz channel width so that the implementation can be done in reasonable number of FPGA chips and simultaneously achieving the goal to obtain better polarization purity than the analogue circular polarizers over broad bands.

2.2 Method overview

The noise diode signals during calibration pass through the same x and y receiving chains as does the astronomical signal later, and are sampled at IF at 1 GSamples/s (i.e. 500 MHz contiguous Nyquist bandwidth). The sampled $x(t)$ and $y(t)$ signals are processed in an FX correlator on the FPGA, which transforms to frequency domain with 1 MHz channel widths, cross multiplies each $X(\omega)$ spectrum against the corresponding $Y(\omega)$ spectrum and integrates for 8 s. We chose a spectral resolution of 1 MHz to allow for possibly rapidly changing channel phase differences with frequency. The result is a phase spectrum with low thermal phase noise that represents the phase difference between the x and y channels due to the transfer characteristics of the receiver chains. The phase spectrum is used during later astronomical observation for equalizing the (frequency-dependent) phase lengths of the x and y receiver chains. The x and y bandpass amplitude shapes are also equalized, using gains derived during calibration from total-power spectra of $X(\omega)$ and $Y(\omega)$ accumulated during the calibration stage. To form circular polarization from native linears during astronomical observations, we need likewise to transform the $x(t)$ and $y(t)$ time series to frequency domain in the same manner as during calibration, then equalize the transfer characteristics by applying a phase rotation to each frequency channel of one polarization and an amplitude scaling to each frequency channel of both polarizations in an equalizer stage, then simply add or subtract 90° (equivalent to exchanging real and imaginary in the complex spectra), and summing to form results that respond to the two hands of circular polarization.

2.3 Instrumental phase and gain calibration

The orthogonal time-domain field components *x(t)* and *y(t)* received by the crossed dipoles undergo unequal phase and amplitude distortions due to different frequency-dependent time delays and gains of the two receiving systems through which they pass. The phase and gain calibration aims to compensate the transfer characteristics of channel *x* and channel *y* to make them identical in *x(t)* and *y(t)*, reducing instrumental artifacts to zero. This approach is already used in software for calibrating radio astronomical data though those operate on stored data rather than in real time. Our effort is to extend it to the digital domain processing sampled IF signals in real time, calibrating with fine frequency channels, to enable formation of circular polarization in real time with more accurate phase and magnitude response than the analogue techniques can achieve over broad bandwidths. I have not considered channel non-linearities and multi-path effects. Non-linearity spreads the output spectrum beyond the input spectrum by introducing new frequency components and causes amplitude distortion. Therefore, radio astronomical receivers have to be designed to be linear. In the presence of strong RFI (radio frequency interference) non-linearity does occur and has to be blanked. Techniques for RFI mitigation are a sizable study in themselves and are beyond the scope of the present work. Nevertheless RFI mitigation techniques can easily be implemented in the same digital hardware. For the present development I assume linear transfer characteristics and Gaussian signal statistics. We have filtered the passband to avoid aliasing and kept signal levels in the linear regime. A linear time invariant system causes only pulse dispersion and amplitude scaling.

2.3.1 Phase equalization

Let us consider the noise diode signal during calibration as a broadband source radiating Gaussian random signals continuously in time, *s(t)*, and I receive and sample a finite number, N_s , of frames of time-domain samples each consisting of *N* samples spaced equally in time in two orthogonal linear polarization states, *x(t)* and *y(t)*. Let the sampled time series be represented by $x_i(t)$ and $y_i(t)$ and the noise diode signal at the sample times be $s_i(t)$ where *i* = 1, 2, 3, …., N_s . It is convenient to transform these time series into the frequency domain since the transfer characteristic of the receiving system used for calibration is frequency dependent. The Fourier transform produces N_s spectra, each consisting of *N* channels. N_s depends on the sampling rate, f_s , the number of samples in a spectrum, *N*, and the total integration time, T_{integ} , as

$$N_s=(f_s\times T_{integ})/N \tag{2.1}$$

Let a time-domain signal have length T_0 , then the corresponding frequency-domain spectrum will have channels spaced at an interval f_0 of $1/T_0$ frequency units. For a simplified analysis I consider one frequency component, the results from which hold good for all other spectral components in the band.

The signals $x_i(t)$ and $y_i(t)$ are represented by the equations

$$x_i(t)=h_x(t)*(s_i(t)+n_{xi}(t)) \quad , \tag{2.2}$$

$$y_i(t)=h_y(t)*(s_i(t-t_{xy})+n_{yi}(t)) \tag{2.3}$$

where $h_x(t)$ and $h_y(t)$ are the transfer functions of channel *x* and channel *y*, t_{xy} is the *x*-*y* propagation time difference from the dipoles to the receiver inputs (samplers for a digital receiver), and $n_{xi}(t)$ and $n_{yi}(t)$ are external unwanted signals (astronomical sources, thermal fluctuations and spurious sources). After transforming to the frequency domain these relations are

$$X_i(\omega)=|X_i(\omega)|e^{j\phi_X(\omega)} \tag{2.4}$$

$$= \quad |H_X(\omega)|e^{j\theta_X(\omega)}[S_i(\omega)+N_{Xi}(\omega)] \quad \text{, and} \tag{2.5}$$

$$Y_i(\omega)=|Y_i(\omega)|e^{j\phi_Y(\omega)} \tag{2.6}$$

$$= \quad |H_Y(\omega)|e^{j\theta_Y(\omega)}(S_i(\omega)e^{-j\omega t_{xy}}+N_{Yi}(\omega)) \tag{2.7}$$

where $\phi_X(\omega)$ and $\phi_Y(\omega)$ are the phases of $X_i(\omega)$ and $Y_i(\omega)$ respectively and $\theta_X(\omega)$ and $\theta_Y(\omega)$ are the phases of the transfer functions $H_X(\omega)$ and $H_Y(\omega)$ respectively. The phase difference $\phi_X(\omega)$ - $\phi_Y(\omega)$ is due to the initial and instrumental phase difference between $X(\omega)$ and $Y(\omega)$.

Now, let us consider the samples of $X_i(\omega)$ and $Y_i(\omega)$ at uniform intervals of $\omega_0=2\pi f_0$. If $X_i(r\omega_0)$ and $Y_i(r\omega_0)$ are the r^{th} (channel number) samples of $X_i(\omega)$ and $Y_i(\omega)$ respectively, where r= 0, 1, 2, 3,......, N-1, then equations 2.4, 2.5, 2.6 and 2.7 can be rewritten in discrete form (Cooley & Tukey 1965) as

$$X_i(r\omega_0)=|X_i(r\omega_0)|e^{j\phi_X(r\omega_0)} \tag{2.8}$$

$$= \quad |H_X(r\omega_0)|e^{j\theta_X(r\omega_0)}(S_i(r\omega_0)+N_{Xi}(r\omega_0)) \tag{2.9}$$

$$Y_i(r\omega_0)=|Y_i(r\omega_0)|e^{j\phi_Y(r\omega_0)} \tag{2.10}$$

$$= \quad |H_Y(r\omega_0)|e^{j\theta_Y(r\omega_0)}(S_i(r\omega_0)e^{-jr\omega_0 K_1 T_s}+N_{Yi}(r\omega_0)) \tag{2.11}$$

where K_1 is the fractional sample delay caused by t_{xy} and T_s is the sampling period. It is assumed that the system is linear. Thus the transfer function causes only linear distortion and no new frequency components are produced. The resulting pulse is dispersed in time and amplitude rescaled. The phase difference $\phi_X(r\omega_0)-\phi_Y(r\omega_0)$ is obtained from the accumulated cross power spectrum of $X_i(r\omega_0)$ and $Y_i(r\omega_0)$ in the frequency domain, which is expressed as

$$Z(r\omega_0)=\sum_{i=1}^{N_s} Z_i(r\omega_0) \tag{2.12}$$

$$= \quad \sum_{i=1}^{N_s}|X_i(r\omega_0)||Y_i(r\omega_0)|e^{j(\phi_X(r\omega_0)-\phi_Y(r\omega_0))}. \tag{2.13}$$

Using Eq.(2.9) and Eq.(2.11) in Eq.(2.13) I obtain the product $|X_i(r\omega_0)||Y_i(r\omega_0)|e^{j(\phi_X(r\omega_0)-\phi_Y(r\omega_0))}$, which consists of the summation of the following contributing terms

1. $|H(r\omega_0)|e^{j\theta(r\omega_0)}|S_i(r\omega_0)|^2 e^{jr\omega_0 K_1 T_s}$ (2.14)

2. $|H(r\omega_0)|e^{j\theta(r\omega_0)}N'_{Yi}(r\omega_0)N_{Xi}(r\omega_0)$ (2.15)

where $|H(r\omega_0)|e^{j\theta(r\omega_0)}=|H_X(r\omega_0)||H_Y(r\omega_0)|e^{j(\theta_X(r\omega_0)-\theta_Y(r\omega_0))}$.

$Z_i(r\omega_0)$ is accumulated for T_{integ} time. Since S_i is incoherent with N_{Xi} and N_{Yi} , their cross product terms will average to zero in the summation and so are not shown here. We have chosen, in practice, 8 s integration time corresponding to 8×10^6 frames of data so that the thermal fluctuations of the noise diode signal that is 5 % of the system temperature are averaged down to a fractional fluctuation of 4×10^{-4} in 1 MHz channels. Thus we attain our objective of measuring the transfer characteristic phase to 0.1° precision. At such high levels of precision, the measurement is sensitive to corruption by possible external sources or by RFI via the second term Eq. (2.15). I cancel this effect on our measurement of the transfer characteristic, provided the corrupting source remains constant, by performing the summation in Eq. (2.13) twice, first in the presence of $s_i(t)$ (noise diode switched on) and second with $s_i(t)=0$ (noise diode switched off) and differencing. Hence, considering only the first term (Eq. 2.14) I obtain the phase difference ϕ between the two polarization signals due to instrumental effects and due to any initial phase difference as follows

$$\phi=\phi_{instrument}+\phi_{initial} \tag{2.16}$$

where, $\phi_{instrument}=\theta(r\omega_0)$ and $\phi_{initial}=r\omega_0 K_1 T_s$. I have not considered any frequency down conversion in Eqs. (2.9) and (2.11). A down conversion will cause a frequency shift in ω by an amount ω_d where ω_d is the mixing frequency. Then the samples of ω in Eqs. (2.9) and (2.11) or $r\omega_0$ will represent the samples of $\omega-\omega_d$. The equations will remain unchanged except for the initial phase, which will now be $(r\omega_0+\omega_d)K_1 T_s$ so that the initial phase is unaffected by down conversion.

For accurate calibration, $\phi_{instrument}$ should be measured and used later to correct the signals for the effect of $\phi_{instrument}$. This should reduce the phase difference to zero, requiring that the initial phase difference be zero so that ϕ corresponds to instrumental phase errors only. This requires that the two orthogonal dipoles receive signal from a common source placed at a location equidistant from both the dipoles, which is accomplished by placing a noise source at 45° to the two dipoles. One could alternatively calibrate using a polarized astronomical source instead of the noise diode, and point on source and then off source for the two accumulations to be differenced. In the on-source position, the dish main beam must be pointed accurately to the source, placing the source on the focal axis. The dipoles are located in a plane accurately perpendicular to the focal axis and hence, the dipoles would be equidistant from the source to high accuracy, so $\phi_{initial}$ would be close to zero and can be ignored. $Z(r\omega_0)$ gives the angle through which one of the vectors say $Y(r\omega_0)$ must be rotated to make the phase lengths of the two polarization channels equal. The frequency-dependent rotation matrix elements $\cos\theta(r\omega_0)$ and $\sin\theta(r\omega_0)$ are obtained without trigonometric functions for computational efficiency using the relations

$$\cos\theta(r\omega_0)=\Re(Z(r\omega_0))/|Z(r\omega_0)| \quad , \tag{2.17}$$

$$\sin\theta(r\omega_0)=\Im(Z(r\omega_0))/|Z(r\omega_0)| \tag{2.18}$$

Let $Y'(r\omega_0)$ be the vector obtained after rotation of vector $Y(r\omega_0)$ by the phase difference

$\theta(r\omega_0)$. Then the phase difference between $X(r\omega_0)$ and $Y'(r\omega_0)$ reduces to zero or

$$\phi_X(r\omega_0)-\phi_{Y'}(r\omega_0)=0 \tag{2.19}$$

as required for a correctly calibrated system phase prior to formation of circular polarization.

2.3.2 Gain equalization

The amplitudes also undergo linear distortions due to different gains in the two channels due to different passband characteristics. To compensate the amplitude differences, all the spectral components of $X_i(\omega)$ and $Y_i(\omega)$ in the passband are scaled to one same level, chosen to be the maximum signal level, V_{max} in the passbands of $X_i(\omega)$ and $Y_i(\omega)$. Multiplying Eq. (2.5) with its complex conjugate, I obtain the power spectrum $|X_i(\omega)|^2$ and similarly Eq. (2.7) yields $|Y_i(\omega)|^2$. The scaling factors to equalize the magnitudes of $X_i(r\omega_0)$ and $Y_i(r\omega_0)$ are obtained from accumulating $|X_i(r\omega_0)|^2$ and $|Y_i(r\omega_0)|^2$ respectively for T_{integ} to reduce thermal noise fluctuations in the measurement of the passband shapes and are expressed by the following two equations:

$$|X(r\omega_0)|^2=\sum_{i=1}^{N_s}|X_i(r\omega_0)|^2 \tag{2.20}$$

$$|Y(r\omega_0)|^2=\sum_{i=1}^{N_s}|Y_i(r\omega_0)|^2 . \tag{2.21}$$

Expanding these using Eqs. (2.9) and (2.11) I find that each contains the following contributing terms for summation:

1. $|H_X(r\omega_0)|^2|S_i(r\omega_0)|^2$, (2.22)

   $|H_Y(r\omega_0)|^2|S_i(r\omega_0)|^2$, (2.23)

2. $|H_X(r\omega_0)|^2|N_{Xi}(r\omega_0)|^2$, (2.24)

   $|H_Y(r\omega_0)|^2|N_{Yi}(r\omega_0)|^2$ (2.25)

where $\phi_{N_{Xi}}(r\omega_0)$ and $\phi_{N_{Yi}}(r\omega_0)$ are the phases of $N_{Xi}(r\omega_0)$ and $N_{Yi}(r\omega_0)$ respectively.

The first term (Eqs. 2.22 and 2.23) provides the measurement of the bandpass shape that I seek (assuming that the noise diode produces white noise). The second term (Eqs. 2.24 and 2.25) does not average to zero to the extent that noise sources other than the noise diode (eg receiver noise, radio sources, atmospheric emission, cosmic microwave background, and RFI) are present. I cancel this term, provided the corrupting source remains constant, by performing the summation in Eqs. (2.20) and (2.21) twice, first in the presence of $s_i(t)$ (noise diode switched on) and second with $s_i(t)=0$ (noise diode switched off) and differencing. Hence, I am left with only the first term (Eqs. 2.22 and 2.23) where $|H_X(r\omega_0)|^2$ and $|H_Y(r\omega_0)|^2$ are the frequency-dependent amplitude scaling factors. To determine the gains $g_X(r\omega_0)$ and $g_Y(r\omega_0)$ that scale each $X_i(r\omega_0)$ and $Y_i(r\omega_0)$ to equalize the passband amplitudes to the same level, the following two equations are used

$$g_X(r\omega_0)=\sqrt{P_{max}/|X(r\omega_0)|^2}\quad , \tag{2.26}$$

$$g_Y(r\omega_0)=\sqrt{P_{max}/|Y(r\omega_0)|^2}\quad . \tag{2.27}$$

P_{max} in Eqs. (2.26) and (2.27) is obtained by comparing all $|X(r\omega_0)|^2$ and $|Y(r\omega_0)|^2$ of Eqs. (2.20) and (2.21) for *r* = 0,1, 2, 3,......, *N*-1 and finding the maximum power. The gains in Eqs. (2.26) and (2.27) are calculated using accumulated power spectra and are later applied to voltage spectra for equalization, which potentially introduces a small inaccuracy since the gains thus obtained are not the same as the actual gains obtained by accumulating individual voltage spectra consisting of absolute voltages in the denominators. However, if the absolute voltages, $|X_j(r\omega_0)|$ and $|Y_j(r\omega_0)|$, where $j=N_s+1, N_s+2, \ldots\ldots$ for the subsequent spectra acquired during observations, are accumulated for sufficient integration time then

$$g_X(r\omega_0)\approx V_{max}/\sum|X_j(r\omega_0)|\quad \text{, and} \tag{2.28}$$

$$g_Y(r\omega_0)\approx V_{max}/\sum|Y_j(r\omega_0)|\quad . \tag{2.29}$$

V_{max} in Eqs. (2.28) and (2.29) is the maximum of all $\sum|X_j(r\omega_0)|$ and $\sum|Y_j(r\omega_0)|$ for *r* = 0,1, 2, 3,......, *N*-1. Eqs. (2.28) and (2.29) are also confirmed in numerical simulation. Fig. 2.1 as shown below illustrates this.

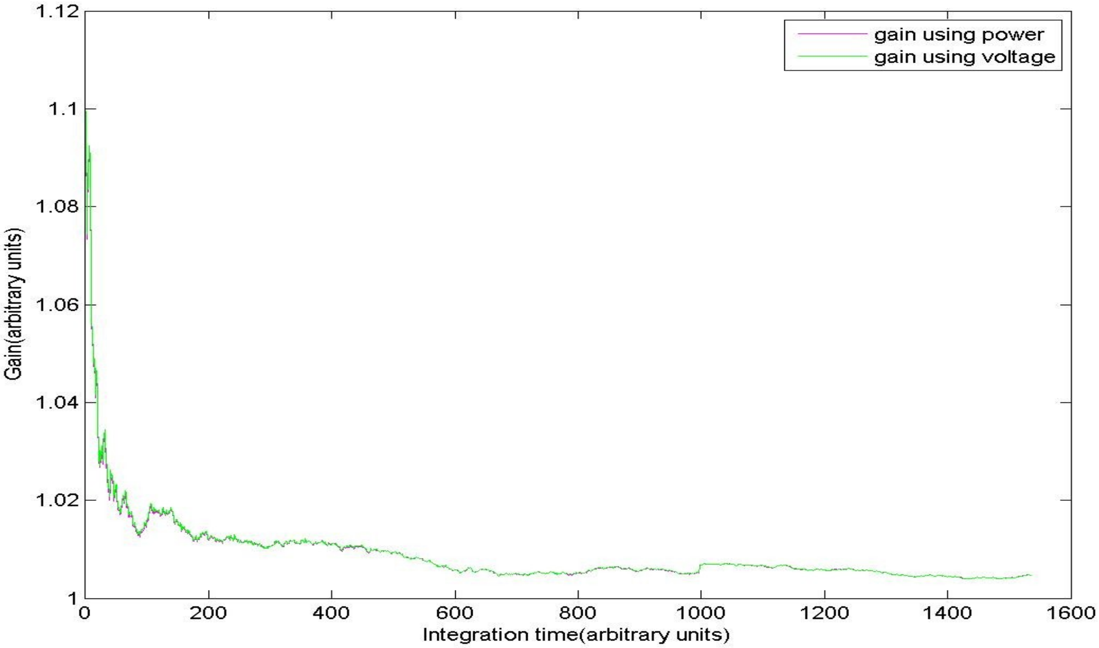


Fig. 2.1: Plots for gains using eqn. 2.26 (2.27), that is, power and eqn. 2.28 (2.29), that is, voltage for a single channel of one polarization as a function of integration time. Both are approximately same.

I have just plotted the gains for the first channel. But all of them show similar effects. The ratio of gain derived using power and gain derived using voltage is shown in fig. 2.2. The ratio approaches unity as the number of samples included in the summation increases or integration time increases.

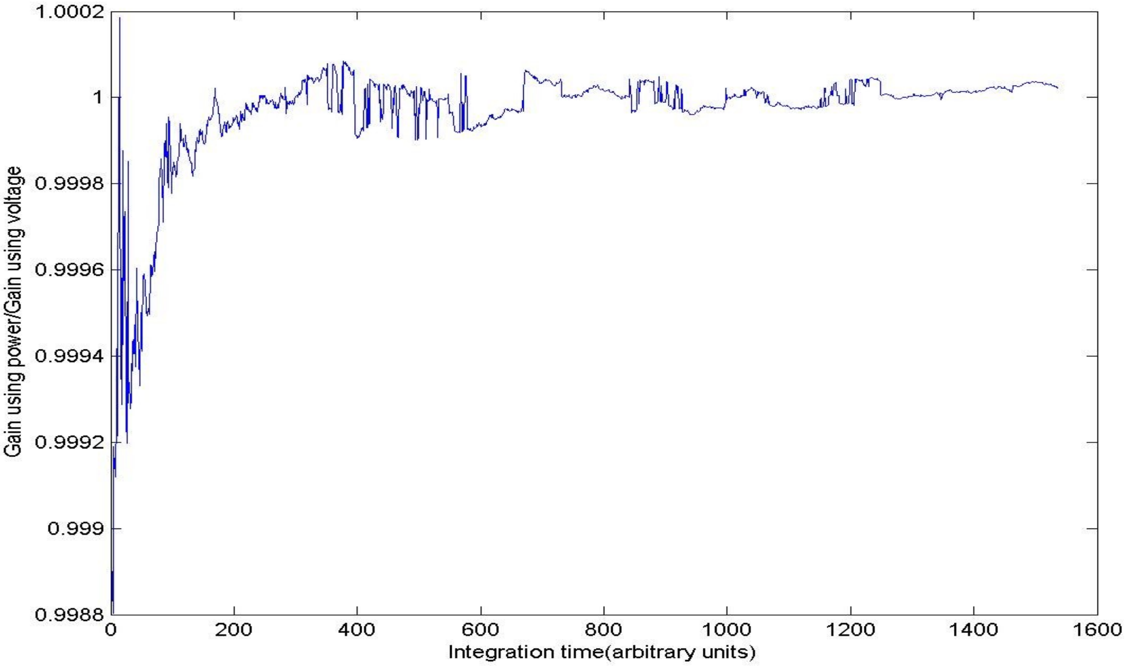


Fig. 2.2: Ratio of gains obtained by using eqn. 2.26 (power) and eqn. 2.28 (voltage). It approaches unity with increase in integration time.

The plots in figures 2.1 and 2.2 are generated in the following steps.

1) I generated a matrix of random numbers of dimension $N \times 1024$ and augmented all the elements by 5 to enhance the visibility of the plots. The number 1024 represents number of channels and N is the upper limit of the count up to which each channel's element can be incremented. N is taken as 1550. Another variable i takes the values from 5 to N, which means that at least 5 numbers pertaining to a particular channel should be added and then this number would increase up to N.

2) Formula 2.26 (2.27) is compared against 2.28 (2.29) for consecutively increasing integration time or count i and the result is shown in fig. 2.1. The two overlapping curves verify that 2.26 (2.27) and 2.28 (2.29) produces approximately the same results. The plot is for a single channel.

3) Fig. 2.2 shows the plot for $2.26 \div 2.28$ (same for $2.27 \div 2.29$) demonstrating that the two equations 2.26 and 2.28 (2.27 and 2.29) produce more and more identical results as the count is increased or their ratio tends towards unity.

2.4 Windowing

A window function is derived for the X_j and Y_j spectra to truncate the possible analogue filter flanks to avoid scaling up, by large factors, signals that have been strongly attenuated by band-limiting filters. The window function is conveniently obtained from the $|Z|$ spectrum since the Z spectrum contains the band common to both X and Y spectra thereby providing necessary frequency shift and bandwidth information for the window function. Spectral channels in which the signal level $|Z(r\omega_0)|$ is greater than one quarter of the maximum amplitude in the $|Z|$ spectrum are given unit weight and all others are given zero weight resulting in a frequency shifted rectangular function of unit amplitude. A unit delta function is added to this since we want to pass undisturbed the DC signal produced by the A/D converter. This window function is applied to both X_j and Y_j spectra resulting in rectangular band shape for the frequency band that is in common. The effect of the window function on the time-domain is to convolve the signal with a sinc function. Since, the frequency spectra are already band limited, the window function does not in itself introduce any new waveform characteristics. Rather it prevents contamination of the waveform that would arise were one to scale up, by large factors, frequency channels that had little signal.

2.5 Forming circular polarization

The gains $g_X(r\omega_0)$ and $g_Y(r\omega_0)$, the rotation parameters $\sin\theta(r\omega_0)$ and $\cos\theta(r\omega_0)$, and the window function $W(r\omega_0)$ are applied to the spectral components $X_j(r\omega_0)$ and $Y_j(r\omega_0)$ respectively. If $Y_j^{''}(r\omega_0)$ and $X_j^{'}(r\omega_0)$ are the resulting vectors after calibration then $Y_j^{''}(r\omega_0)$ is related to $Y_j(r\omega_0)$ by the product of gain, window function and rotation matrix as follows

$$\begin{vmatrix}\Re(Y_j{}''(r\omega_0))\\ \Im(Y_j{}''(r\omega_0))\end{vmatrix}=g_Y(r\omega_0)W(r\omega_0)\begin{vmatrix}\cos\theta(r\omega_0) & -\sin\theta(r\omega_0)\\ \sin\theta(R\omega_0) & \cos\theta(r\omega_0)\end{vmatrix}\times\begin{vmatrix}\Re(Y_j(r\omega_0))\\ \Im(Y_j(r\omega_0))\end{vmatrix} \tag{2.30}$$

where $\Re(Y_j^{''}(r\omega_0))$, $\Im(Y_j^{''}(r\omega_0))$ and $\Re(Y_j(r\omega_0))$, $\Im(Y_j(r\omega_0))$ are real and imaginary components of $Y_j^{''}(r\omega_0)$ and $Y_j(r\omega_0)$ respectively. Similarly,

$$\begin{vmatrix}\Re(X_j{}'(r\omega_0))\\ \Im(X_j{}'(r\omega_0))\end{vmatrix}=g_X(r\omega_0)W(r\omega_0)\begin{vmatrix}\Re(X_j(r\omega_0))\\ \Im(X_j(r\omega_0))\end{vmatrix} \tag{2.31}$$

The windowed, phase and gain calibrated $X_j^{'}(r\omega_0)$ and $Y_j^{''}(r\omega_0)$ are added in quadrature (± 90º) to obtain RHC and LHC polarizations.

2.6 Performance limitations

In this section we will discuss the implications of possible contamination of phase caused by the cross polar component or leakage of unwanted orthogonal polarization component (D-terms) and by the temporal instability of receiver transfer characteristics and their effects on polarization purity with approximate quantitative results to estimate those errors. we will also discuss the requirement of frequent recalibration due to the variations in the transfer characteristics of the receiver by observing the drifts in the most sensitive parameters, which are the channel phases. Since the gain fluctuations are much smaller than phase fluctuations, we can ignore their effects.

Effect of phase error on polarization purity:

Were one to introduce an imperfect 90º phase shift into one channel when forming circular polarization from perfectly orthogonal linearly polarized channels, the output voltages $\bar{V}_{RHC}$ and $\bar{V}_{LHC}$ would contain unwanted contributions from the opposite hand of polarization. The derivation below was originally given to me by Alan Roy and I did some modifications and we arrived at the following derivation together.

$$\bar{V}_{LHC}=V_{LHC}+\left(D_{LHC}V_{RHC}\right) \tag{2.32}$$

$$\bar{V}_{RHC}=V_{RHC}+\left(D_{RHC}V_{LHC}\right) \tag{2.33}$$

where D_{LHC} and D_{RHC} are the fractional voltage leakage factors from unwanted polarizations (D-terms).

The larger the phase error, the greater the contribution from the opposite hand. Consider the monochromatic case in which a linearly polarized wave is incident normally on crossed linear dipoles with the plane of the electric field oriented at 45° to the two dipoles. Then the voltages in the two dipoles are

$$V_x=V_0e^{j\omega t} \tag{2.34}$$

$$V_y=V_0e^{j\omega t} \tag{2.35}$$

After introducing an imperfect 90° phase shift to the *y* channel, one has

$$V_x=V_0e^{j\omega t} \tag{2.36}$$

$$\bar{V}_y=\pm jV_0e^{j\omega t}e^{j\epsilon} \tag{2.37}$$

where ϵ is the error in the 90° phase shift. Circular polarization is obtained by summing the *x* signal with the imperfectly phase-shifted *y* signal, giving

$$\bar{V}_{LHC}=V_0e^{j\omega t}\left(1-je^{j\epsilon}\right) \tag{2.38}$$

$$\bar{V}_{RHC}=V_0e^{j\omega t}\left(1+je^{j\epsilon}\right) \tag{2.39}$$

Had the 90° phase shift been perfect, then ϵ = 0° giving $\bar{V}_{LHC}=V_{LHC}$ and $\bar{V}_{RHC}=V_{RHC}$. Substituting Eq.(2.38) and Eq.(2.39) into Eqs.(2.32) and (2.33) respectively, with V_{LHC} and V_{RHC} obtained by setting $\epsilon=0$, we obtain the dependence of D_{LHC} and D_{RHC} on the phase error, ϵ :

$$D_{LHC}=-D_{RHC}=[1+\sin\epsilon-\cos\epsilon+j\left(1-\sin\epsilon-\cos\epsilon\right)]/2 \tag{2.40}$$

This result is used in section 2.8 for estimating the polarization purity.

2.7 Phase stability of the analogue receiver chain

This section and the following section taken from our paper (Das et. al. 2010) is contributed by Alan Roy. The phase and amplitude transfer characteristics of the receiver chains for the orthogonal polarization channels are known to drift with time, due primarily to temperature changes of filters and cables used in the receiver chains. Fortunately, most of that change is common to both orthogonal polarization channels as the equipment for both channels is housed in close proximity to each other, and the relative changes are small compared to the total. The effect of drift in the relative transfer characteristic is a degradation of the polarization purity, since the equalizer weights that were determined prior to an observation would no longer perfectly correct the channel differences, by the amount of the relative drift since that determination was made. This translates into a requirement that the equalizer weights be re-determined periodically to ensure that polarization impurity due to drift remains below a pre- determined level. We have estimated how often such re-determination would need to be made by measuring the relative phase drift in some existing receiver chains at Effelsberg and the VLBA. The measurements were made using the VLBI phase calibration system (Thompson 1991), which injects a pulse train in the front end and extracts them at the backend data acquisition rack or correlator, to monitor the phase length of the whole receiver system, from front end to the samplers. The measurements show that indeed the phase changes in the orthogonally-polarized channels track each other well (Fig 2.3 top and middle) and there are only occasional outliers, most likely related to

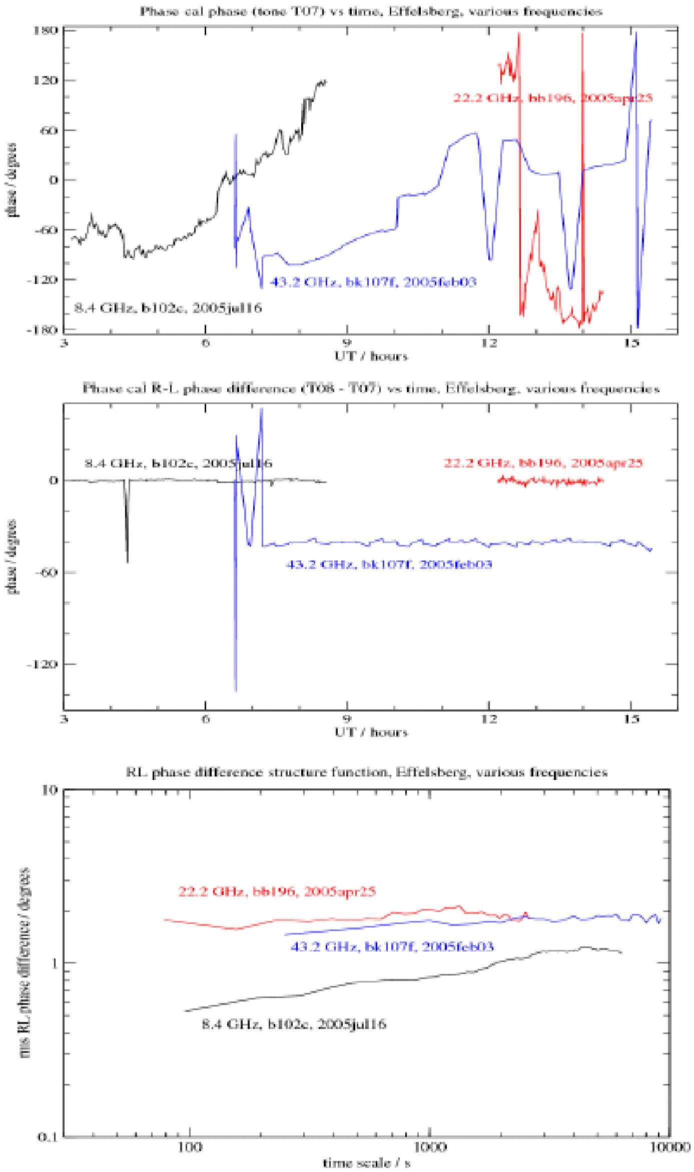


Fig. 2.3: Top: calibration system phase vs time for a single polarization channel for three different receivers at Effelsberg, showing phase changes of typically to over periods of hours. Middle: calibration system phase difference between orthogonal polarizations of the same receivers at Effelsberg, for the identical experiments as in the top plot with an arbitrary offset. Bottom: structure functions constructed from the relative phases presented in the middle plot. These show the rms of the phase difference vs time-scale.

phase-locked loop local oscillator used in the analogue base-band converters, which would not be present in a digital system. The drift in the relative phase is conveniently quantified using a structure function analysis, which converts the phase difference time series into the rms phase change as a function of time-scale (Fig 2.3 bottom). The result is that the rms phase difference due to drift is in the range 0.5º to 2º on time-scales of 100 s to 9000 s.

2.8 Expected polarization purity

The polarization purity to be expected from polarization conversion performed at IF can be derived by combining the two results from section 2.6 and 2.7 - the sensitivity of polarization leakage to phase errors and the typical phase errors in existing analogue receiver chains (0.5º to 2º rms).

The resulting D term is 0.006 (rms) for a 0.5º rms phase error, for which one must re-calibrate the equalizer every few minutes, rising to 0.025 (rms) for a 2º rms phase error, which one would obtain were one to calibrate the equalizer once and leave it fixed for many hours. These are smaller than the leakage D-terms measured for existing radio telescopes, which are commonly 0.05 to 0.15. However, the leakage in existing receivers is constant over long periods, since it occurs primarily due to tolerances in the manufacture of the analogue polarizers, and so can be calibrated using observations of astronomical polarization calibrators. That calibration reduces the effect of the leakage on the resulting polarization images by a factor of ten typically, and one typically sees residual polarization artifacts that are 0.005 to 0.015 rms times the peak flux density in the images. These values are comparable to those expected to be delivered from the digital polarization conversion without use of astronomical D-term calibration. However, the D-term from IF polarization conversion, though small, is expected to drift with time between equalizer re-calibrations due to drift in the relative phase of the orthogonal polarization receiver channels. Were one to want to improve on this by using astronomical calibration of the residual polarization leakage on time-scales between the equalizer re-calibration, then one must be able to derive the D-term from a snapshot observation. Such an algorithm is available and requires the use of an unpolarized calibrator source. However, the changing D-term requires that the post-correlation analysis software be able to handle time-varying D-terms. This will prevent the use of astronomical calibration of the residual polarization leakage, since the D-term calibration in use assume that the leakage is constant on a 12 h time-scale.

Hence, taking all the factors into account viz the theoretical validity of the digital circular polarizer, processing feasibility, limitations due to instabilities in the analogue receiving systems I concluded that the expected performance in terms of polarization purity of the digital circular polarizer would be better than the existing analogue polarizers while talking about broad bands. The next step would be to do a quick simulation using some test data and confirm the validity of the algorithm. The main objetive would be to observe if we can really make the phase difference due to instrumental polarization zero and equalize the gain with the described methods in this chapter. The preliminary verification in simulation is shown in the following chapter.

CHAPTER 3

PRELIMINARY TEST OF ALGORITHM

In this chapter, I will discuss the preliminary tests done to verify the phase equalization method, gain equalization method and windowing described in sections 2.3 to 2.5 for calibrating the two channels. I will discuss the results obtained from the tests to confirm the validity of our approach towards getting pure circular polarization in the end. I first demonstrate the experiment done to collect data for the preliminary test. Then I discuss the spectral characteristics of the data obtained at the outputs of the experiment, which we would also find in the real experiments. I also discuss processing of the data to separate the effects of noise from those of the instrument the steps being the same as described in section 2.3. I demonstrate how the instrumental phase difference between the two receiving channels could be extracted from the data as explained in section 2.3.1. I then show under phase equalization that the method to equalize the phase demonstrated in section 2.3.1 works as expected. Then I show the gain equalization method where I first demonstrate how the instrumental gains of the two receiving channels, following the method described in section 2.3.2, are obtained. Then I discuss windowing, which is also demonstrated in section 2.4. Next, I show the gain equalized and windowed spectra of the two channels simultaneously, which shows that gain equalization and windowing works as expected. After this point I provide the details of some preliminary questions that were discussed before proceeding towards actual instrument development. Finally I conclude this chapter based on the analysis done to confirm our approach.

3.1 Phase and gain equalization and windowing

Sections 2.3 to 2.5 highlight and describe various stages of conversion from linear polarization to circular polarization. Out of these stages I chose to perform preliminary simulation tests for the phase and gain equalization methods (one needs to go through sections 2.3 to 2.5 to have a clear understanding of this preliminary test) to confirm their practical validity. Once phase and gain equalization were done, adding a 90° phase shift to form circular polarization was trivial and hence was also not tested separately. I also included windowing in the test to remain consistent with the steps towards forming the circular polarization. The preliminary signal processing steps and the results of this section were discussed with Alan Roy and the detailed exploration was done by me.

3.1.1 Experiment to collect test data

Fig. 3.1 shows the experimental setup used for collecting data. A noise source was split and passed through two filters. Outputs were taken as X (channel 1) and Y (Channel 2) channels which were sampled with two channels of a digital storage oscilloscope (Tektronix DPO 7254). The noise diode consisted of an avalanche diode and RF amplifiers. The power splitter, two filters and internal resistance of the oscilloscope constituted the external circuit and load. The load resistance of the oscilloscope was 50 ohms. This method of data collection was employed to pass the same signals from a common source through two receiving channels as would happen in a practical calibration already discussed in section 2.3. Hence, the spectral characteristics obtained at the outputs of the two receiving channels would vary due to differences in the the systems through which they passed. Note that the sources of interference described in section 2.3 (radio sources, atmospheric emission, cosmic microwave background and RFI) except the receiver noise (here receiver refers to all the circuit elements through which the signal flows) were absent in this case. Hence, I didn't need to cancel the effects of those sources of interference in order to extract the information on the instrumental gain and phase differences; I would need to do so if calibration were done on sky. We should also note that the avalanche noise diode generates white noise in the RF range but the thermal fluctuations corrupt the amplitudes and phases of the spectrum. So I needed to account only for this receiver noise, which was purely random or thermal (as was found and will be discussed later).

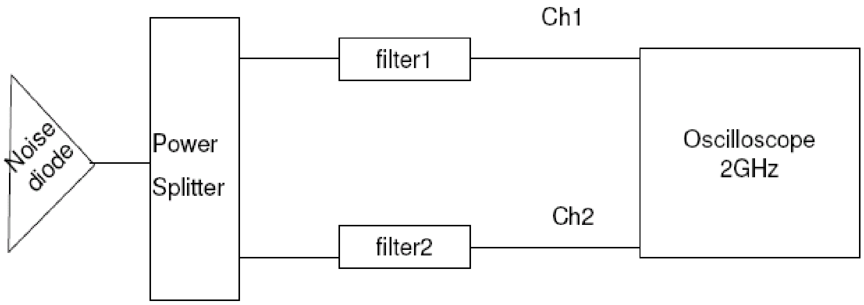


Fig. 3.1: Block schematic of the experimental setup to obtain sampled time series from the noise diode after passing the signal through two channels consisting of two filters using a 2.5 GHz sampling oscilloscope.

3.1.2 Description of spectral characteristics obtained by processing data in MATLAB

The sampled data from the experiment was saved to be processed in MATLAB to confirm the validity of the phase equalization method and the gain equalization method described in section 2.3. I performed an FFT in MATLAB to obtain the frequency spectra of the two channels and continue the test in the frequency domain. I covered the 1250 MHz Nyquist bandwidth with ≈ 2.44 MHz spectral-channel (represents one spectral component) width by using 1024 points FFT for 2500 MHz bandwidth (sampling frequency). By doing this I obtained 24 spectra in the frequency domain from the available 24 frames of the time domain signal for each channel.

The following plots (fig. 3.2 top left and top right) show the absolute values of the spectra obtained at the outputs of the two channels X and Y. These were single spectra of channel X and channel Y and hence they had a lot of noise fluctuations. Most of the plots in the incipient figures show an image of the negative frequency components in the upper half region of the 2500 MHz band. This is because of the following two reasons: 1) I operated on real data with a complex FFT where a spectrum of a real signal follows Hermitian symmetry, which means $|X(-\omega)|=|X(\omega)|$ and $\sphericalangle X(-\omega)=-\sphericalangle X(\omega)$. 2) Because of periodicity of the Fourier spectrum generated by the property of the DFT a spectrum repeats itself after every f_s (sampling frequency) intervals so do its negative frequency components.

Thus the spectrum repeated itself after 2500 MHz (1024 points). To avoid overlapping of these negative frequency components with the signal in the band of interest, one must sample at a frequency twice the bandwidth of interest, which I followed.

The bandshape of channel X (fig. 3.2 top left) and the bandshape of channel Y (fig. 3.2 top right) were different since the analogue filters, filter1 and filter2 in fig. 3.1 had different pass-band characteristics. Matching spectral components of the two channels (X and Y) would have a unique phase difference and a unique gain difference; I wanted to extract the phase difference and the gain difference between them due to difference in the transfer characteristics of the two channels. I needed to do this for all spectral components in the band of interest, which would provide the variations in the phase difference and the gain difference across the band. I multiplied the *X* spectrum with the conjugate of *Y* spectrum to obtain the cross-power or *Z* spectrum (fig. 3.2 bottom shows its magnitude). This spectrum contained the pass-band common to *X* and *Y* spectra and its phase represented the phase difference between channel *X* and channel *Y* as a function of frequency; the phase difference in this case means phase of *X* – phase of *Y*.

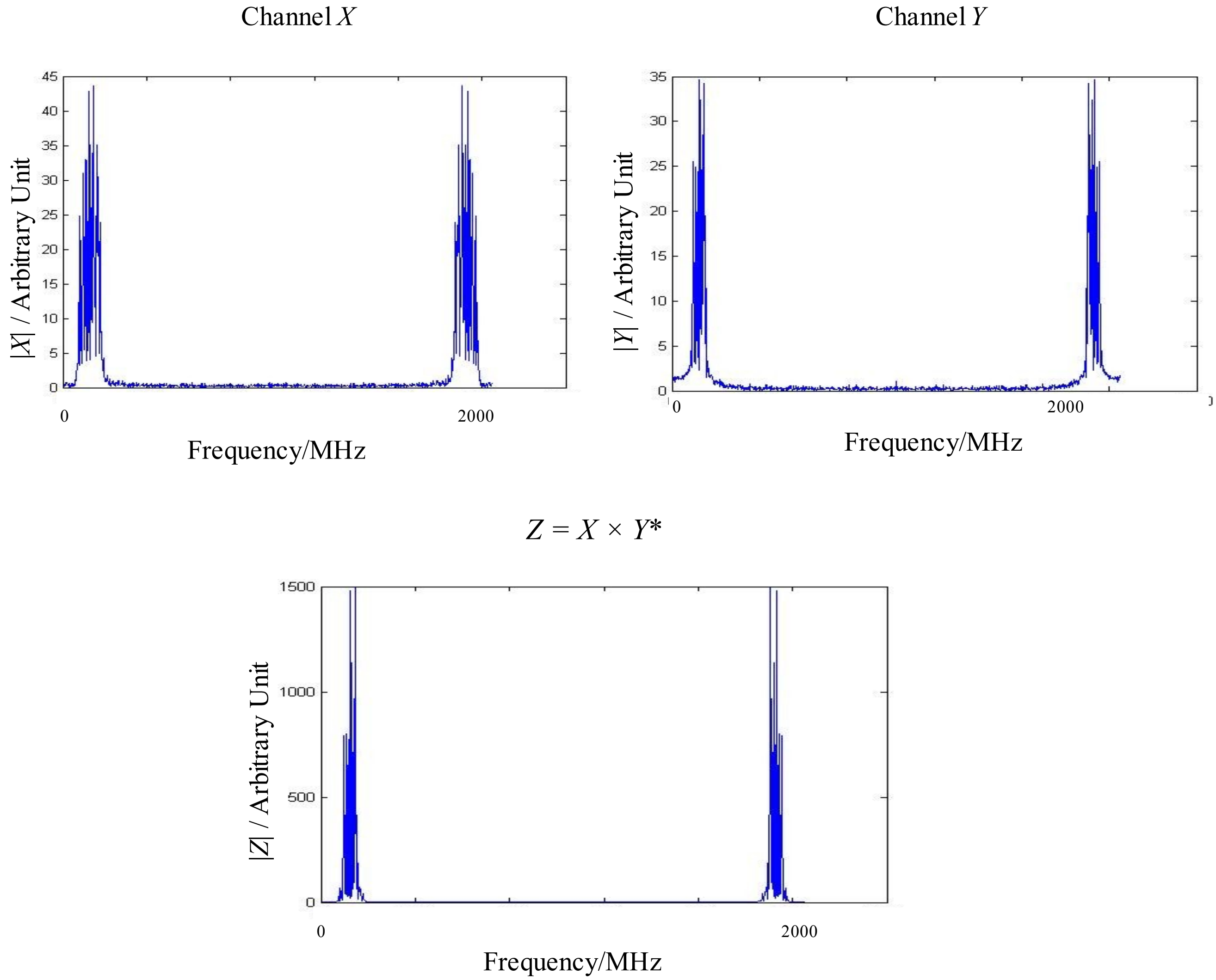
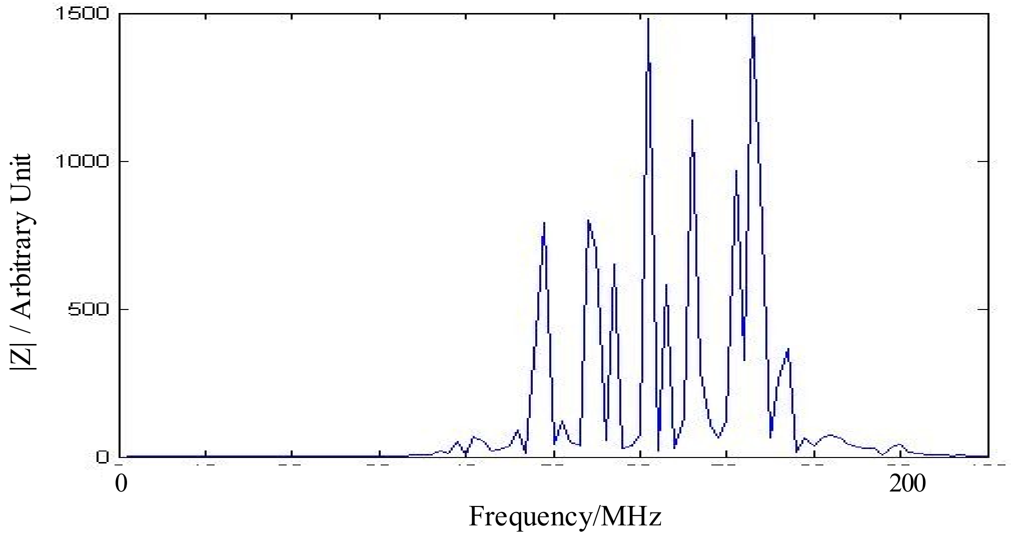


Fig. 3.2: Unaccumulated magnitude spectra for the channel *X (top left)*, channel *Y (top right)* and their cross-power spectrum *Z (bottom)* in the range from 0 MHz-1250 MHz. In each of these spectrum, the signal in the upper half of 2500 MHz band that is from 1251 MHz-2500 MHz represented the image of the negative frequency components of the spectrum (same explanation holds good for fig. 3.5, fig. 3.6 and fig. 3.7). The spectra are very noisy due to thermal fluctuations.

This is the same procedure as is used in FX correlators used in radio interferometers such as the VLBA and DiFX; the information on phase difference between two channels is extracted from the cross-power spectrum. However, as I see from the plots in fig. 3.3, the magnitude spectrum (fig. 3.3 top) and the phase spectrum (fig. 3.3 bottom) zoomed into the pass-band of the cross-power spectrum were highly distorted due to noise fluctuations. Since the same signals were passed through the two channels, any difference in phases of the two channels would correspond to instrumental phase difference. Further, since the two filters had linear phase characteristics, their difference would also have a linear phase response which was not evident from the phase spectrum. However, I was able to see the band somehow with one spectra.



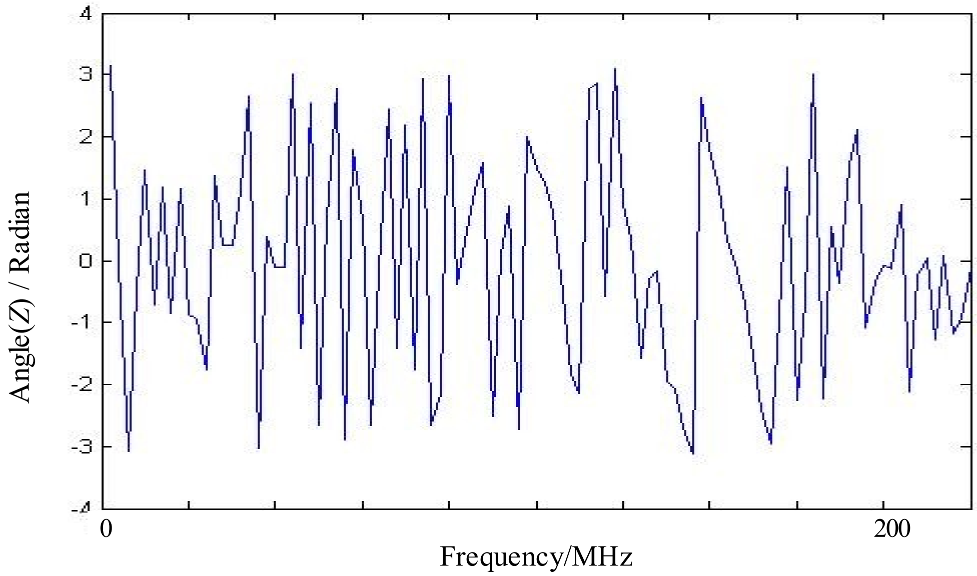


Fig. 3.3: *Top:* Magnitude of unaccumulated Z spectrum zoomed into the pass-band. *Bottom:* Phase of the same Z spectrum zoomed into the pass-band. Due to noise fluctuations the spectra are not clearly visible.

I observed the fluctuations after switching off the noise diode, which were due to the receiver noise since external noise sources were absent, and found that they were random. Hence, I decided not to perform any subtraction operation between noise diode on samples and noise diode off samples (see section 2.3.1 for details on the subtraction operation) to cancel the effects of any stationary noise source. Thus accumulation of a certain number of Z spectra could reduce the noise below a predetermined level (since accumulation of N spectra reduces the fluctuations by $\sqrt{N}$). Hence I proceeded to perform accumulation of Z spectra. I accumulated 24 Z spectra and the results obtained for the magnitude and phase spectra are shown in fig. 3.4 top and bottom respectively. Alan observed the results with me. It is very clear by comparing fig. 3.3 and fig. 3.4 that the responses for the magnitude and phase were much refined after accumulation - the difference in the pass-band characteristics is clearly visible in the phase spectrum that is the linear phase difference is visible as a ramp in the pass-band. In order to nullify this extracted instrumental phase difference between the two channels, as described in section 2.3.1, it is needed to rotate the phase of one channel by the amount of this phase difference. I proceeded to do the same and demonstrate this in the following subsection.

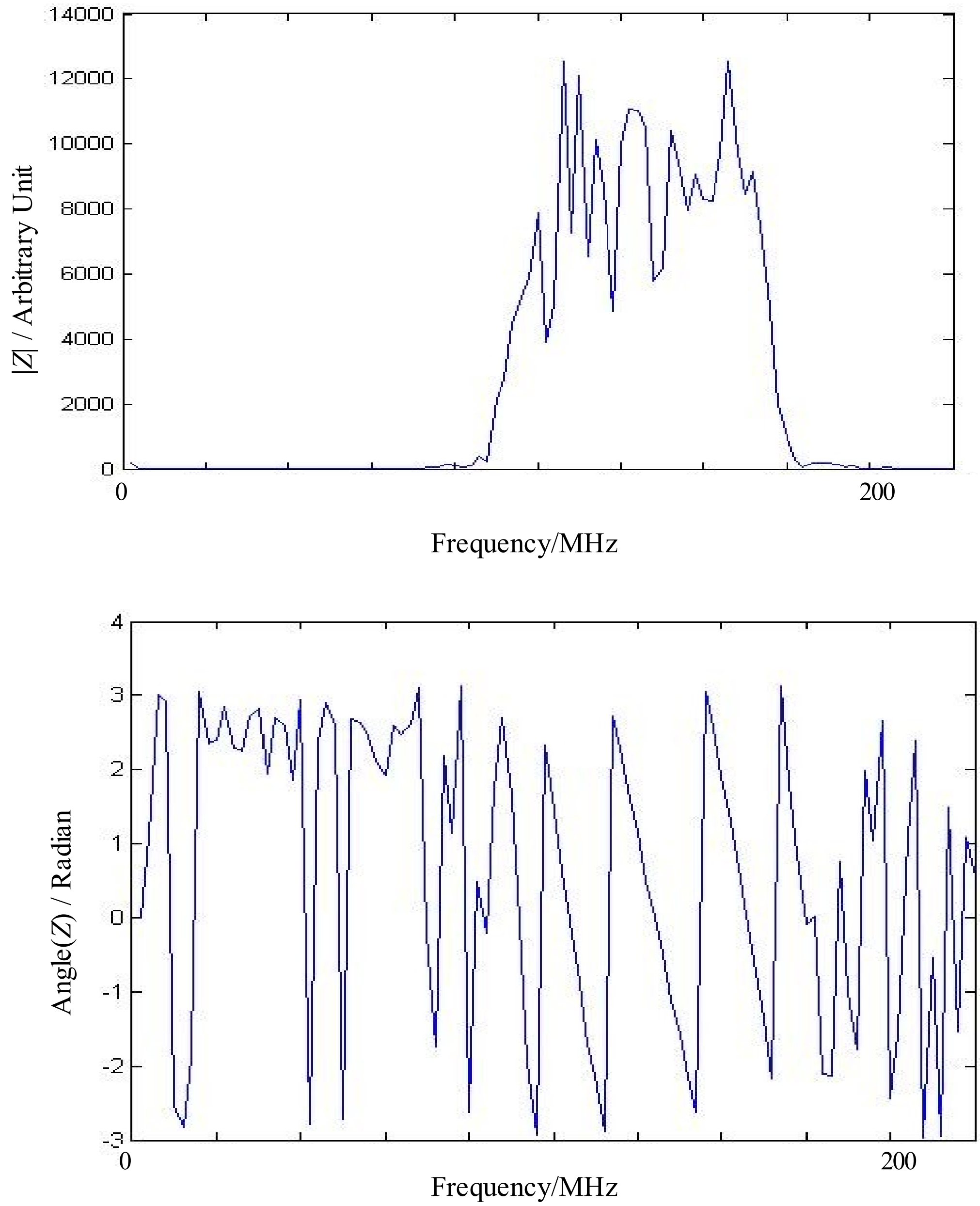


Fig. 3.4: *Top:* Magnitude of accumulated *Z* spectrum zoomed into the passband. *Bottom:* Phase of accumulated *Z* spectrum zoomed into the passband. Band is clearly visible or the spectra are much more refined due to reduction in noise after accumulation.

3.1.3 Phase equalization

In order to equalize the instrumental phases of the two spectra *X* and *Y*, it was needed to rotate the phase of one spectrum with respect to the other by the amount of phase difference shown in fig. 3.4 (bottom) . The rotation was meant to be done for each spectral channel. Phase rotation of *Y* was done by multiplying vector *Y* with the rotation matrix in eqn (2.30) whose elements were obtained from the Z spectrum as shown in eqs. (2.17) and (2.18). The next plot shows the phase difference between same

X and *Y* spectra from where the phase difference was extracted, after rotation of *Y* with respect to *X*. I would apply this rotation to any but different *Y* in reality acquired during observation and not during calibration using same instrument. This was done to confirm if the instrumental phase difference went to zero or not. As can be seen from fig. 3.5, the phase difference after rotation indeed went close to zero but not exactly zero due to thermal fluctuations. Hence I could proceed to implement this technique of instrumental phase equalization in FPGA without any uncertainty.

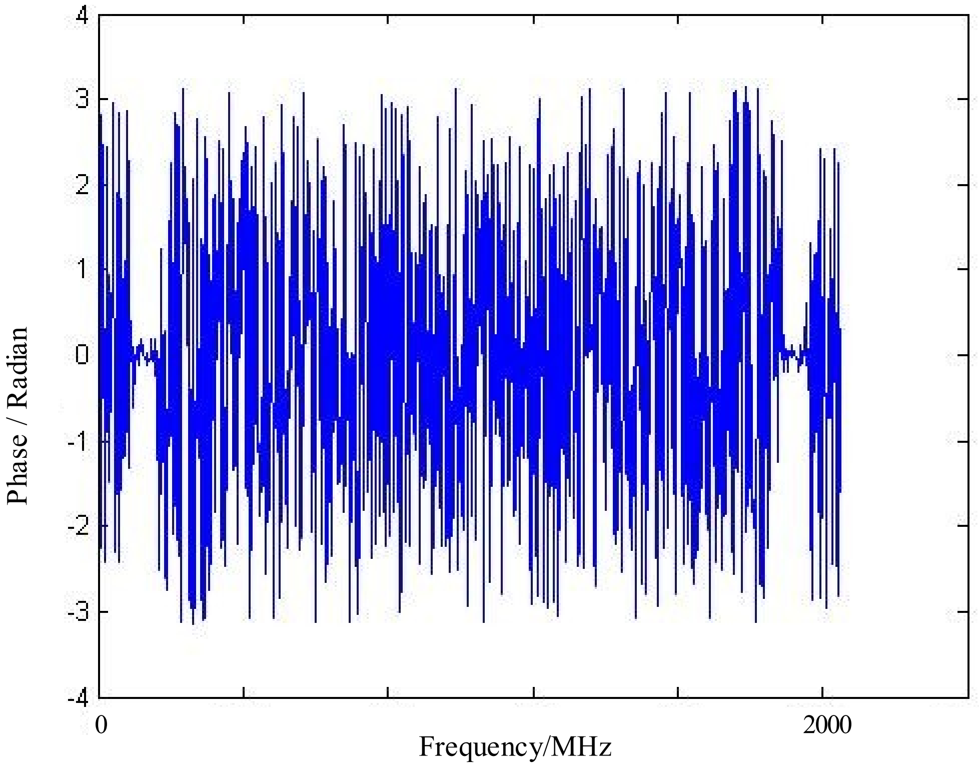


Fig. 3.5: Phase difference between *X* and rotated *Y* is zero. *Y* was rotated by the amount of phase difference between *X* and *Y* shown in fig. 3.4 (*bottom*).

3.1.4 Gain equalization

The next step was to perform gain equalization. This is done as even though the receiver components of the two channels are housed in close proximity to each other, there are imperfections and there are differences in the magnitude responses of the two systems. So one needs to equalize these magnitude responses to calibrate out the differences. Note that I am referring to instrumental gains; since equal amplitude signals were passed through the two channels, any difference in the magnitudes at the outputs of the two channels would represent instrumental gain difference. However, due to thermal noise fluctuations I again had to accumulate the 24 available *X* and *Y* power spectra (separately). The differences in the levels of these accumulated *X* and *Y* power spectra represented square of the instrumental gain difference as a function of frequency channel. I performed gain equalization as follows: I needed to find the amounts by which the two magnitude levels, of any pair of matching spectral channels of the two channels, needed to be changed to make them similar. These amounts would be the new additional gains for the two spectral channels. The magnitudes could be equalized by raising the signal levels of all the spectral channels to one same level; this common level could be conveniently taken as the maximum signal level in the pass-bands of the two spectra. Hence, I determined the maximum signal level among all the spectral channels in the pass-bands of the two accumulated power spectra. Then I found the ratio of this maximum level to the signal level of each spectral channel in a power spectra. The square root of this ratio represented the new gain as a function of frequency channel obtained for each channel *X* and channel *Y* as shown in fig. 3.6. Refer to fig. 2.1 and fig. 2.2, where I showed that the gain obtained by using square root of accumulated power spectra would be similar to that obtained by using accumulated magnitude spectra; the similarity increases with increase in the number of spectra accumulated. I multiplied these gains to the same *X* and *Y* spectra from where the gains were extracted. I did so to verify that all signal levels were raised to the same level indicating that instrumental gain differences were reduced to zero. In real experiments these gains would be applied to other *X* and *Y* spectra acquired during observation. I will show the effects of gain equalization (fig. 3.7) together with the effects of windowing in the following subsection.

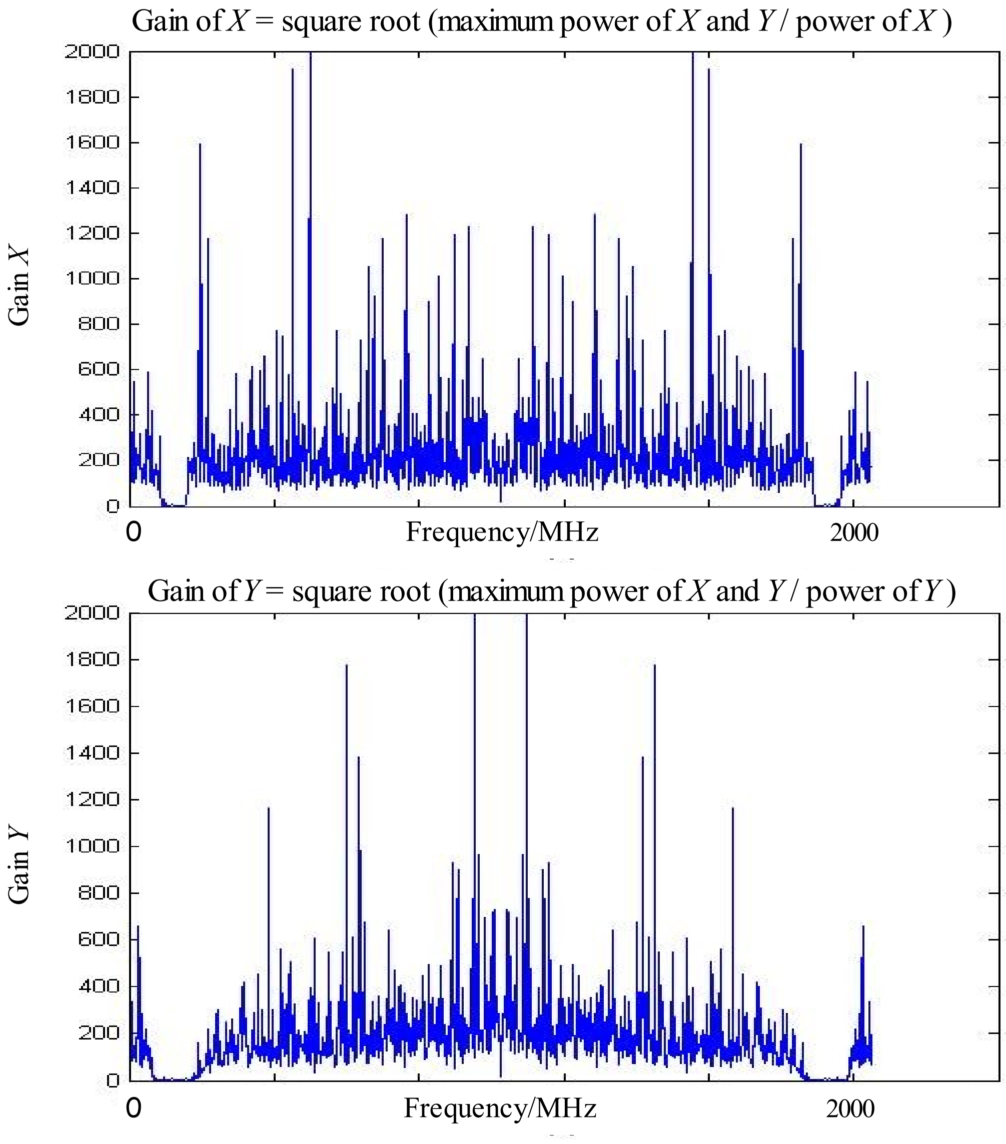


Fig. 3.6: *Top:* gain of *X* channel. *Bottom:* gain of *Y* channel. Each spectral channel gain is obtained from the square root of the ratio of maximum amplitude in the pass-bands of *X* and *Y* power spectra and the magnitude of corresponding power spectral channel.

3.1.5 Windowing

In order also to have a common pass-band of the two channels, I can pick the band in common, which can be conveniently obtained from the Z spectrum. Then I can produce a window function from the Z spectrum just to have the common band information. For that I cut down all signals that were less than or equal to one quarter of the maximum signal level in the pass-band of Z (magnitude) spectrum and then gave unity magnitude to the remaining spectral components. After multiplying with the gains obtained in 3.1.4 and with the window function thus obtained, the gain equalized and windowed X and Y magnitude spectra appear as shown in fig. 3.7.

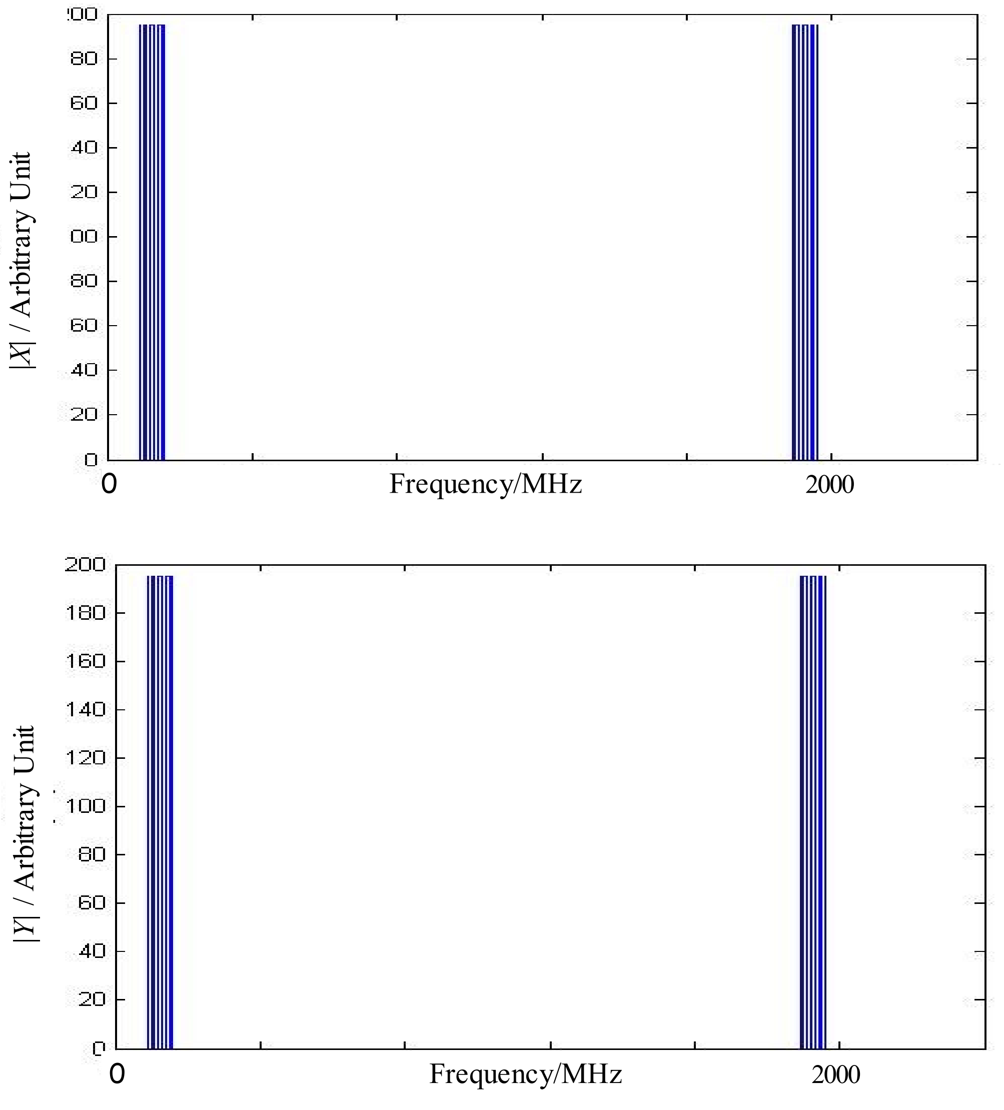


Fig. 3.7: *Top:* gain equalized and windowed $|X|$ spectrum. *Bottom:* gain equalized and windowed $|Y|$ spectrum. The spectra were obtained by multiplying $|X|$ and $|Y|$ spectra with the corresponding gains obtained in fig. 3.6 and with the window function. The window function was obtained by clipping off the signals that were less than or equal to one quarter of the maximum signal level in the pass-band of the Z spectrum and by assigning unity amplitude to the remaining spectral components.

The window function in the time domain is a sinc. This sinc function will be convolved with X and Y time series after equalization and inverse FFT. However, since the signal is band-limited, this window function does not introduce any new characteristics to the waveform.

All steps of phase, gain equalization and windowing seem to work ideally here. Now I will go into the details of the discussion that were done before proceeding towards actual instrument development.

3.2 Preliminary questions to be answered

The following points were discussed with Alan Roy before proceeding towards final implementation of the technique.

Q) What kind of source would we require while calibrating on sky in order that the electric field vector be equidistant from the two dipoles so that the signal from them will be totally coherent or correlated?

A) The vector should be oriented at 45° with respect to the two dipoles. Hence, one should use a polarized source for calibration.

Q) What would happen if the channels have non-linearities due to the system components whose characteristics vary with temperature?

A) Let us consider a transfer function *h(n)* where *n* is time as discrete variable. A transfer function can be approximated by series expansion like the Taylor series, Bessel series, sine series etc. The series expansion can be written as

$$h(n)=\sum_{k=1}^{N} a_k f_k(n)$$

where $f_k(n)$ represents the basis functions and a_k represents the expansion coefficients. The polynomials are generally used for the basis function approximation. The higher order terms of the basis functions will result in new frequency components. These new frequency components would distort the observed signal during calibration. There are several methods available to approximate the nonlinear characteristics and one can use those methods to model the non-linearities. However, it should be noted that the non-linearities may vary with time if the system components are temperature sensitive and no particular model will remain consistent, which will make it difficult to correct for its effects. Hence non-linearity can cause major difficulty in getting corrected phase and gain. We have observed that the Effelsberg system produces no non-linearities and hence I ignored this effect in our project.

Q) What would happen if there is time varying radio frequency interference (RFI)?

A) Following description follows Tuccari (2009): for a radio telescope, RFI is any unwanted signal interfering with the signals from the source under observation. The contributions to RFI are generally from ground communications, radio telescope equipment and by space communications. The bands allocated for radio observations are too narrow and hence the errors on the data from the observations should be minimum for the data to be useful. Generally the levels of RFI are much greater than the signal levels and thus the RFI can be identified from its level. Sometimes RFI is able to destroy the front-end LNAs and hence some filters prior to LNAs are required. There are many filters in use to eradicate the effects of known RFI. In VLBI generally the RFI is uncorrelated and hence it gets canceled during cross-correlation between the two stations. It may be possible to detect signal level change by appropriate algorithm before it enters the low-noise amplifiers and other RF stages and then clip off the band affected.

Q) How do D-terms affect measurements?

A) When there are manufacturing differences in the two dipoles, then they are not positioned accurately with respect to each other and then there would be a difference or error in the 90° position angle. In that case, the two dipoles will receive a fraction of the orthogonal polarization component. This cross-polar component has to be taken into account since it will contaminate the phase of the co-polar component. So there is this issue to correct for the effects of D-term or cross-polarization. The idea to minimize this already very low D-term is discussed in the conclusion of this thesis.

Q) I have performed a 1024 point FFT with 1 MHz channel spacing. If we operate the equalizer at 1 MHz channel spacings, what happens later when the correlator forms much finer frequency spectra? Perhaps one might see a sawtooth pattern in phase vs frequency?

A) Suppose there was a linear phase ramp vs frequency that the equalizer removed by subtracting off a phase offset from each frequency channel. Then the phase slope has been corrected in 1 MHz steps and not on finer scales, so one might see the uncorrected phase slope within 1 MHz with a step back to zero phase at each 1 MHz boundary. Applying a phase gradient vs frequency is equivalent to a fractional-sample time delay in the time domain. Suppose we correct the phase gradient in a 1024-point spectrum then transform it back to time domain, then the effect on that block of 1024 samples is to shift the data by a fraction of a sample. Suppose we then correct the next 1024-point spectrum by the same phase gradient then transform it back to the time domain then the effect on the second block is the same as on the first block; the data is shifted by a fraction of a sample. If the correlator then takes the time series and forms a finer spectrum, say with 0.5 MHz spacing, then the correlator must do a 2048 point transform and by using data that have all been shifted by the same fractional sample delay, and so the 0.5 MHz spectral points should be correctly corrected. Thus, one should not expect a sawtooth vs frequency in the correlator output as we might have feared.

3.3 Conclusion

Thus I showed in this chapter that the methods described in sections 2.3 to 2.5 to form pure circular polarizations from two orthogonal linear polarizations work as expected. I came to the conclusion that the method would yield expected results. Hence, it would be worth taking an effort to develop this FPGA based circular polarizer to contribute towards observational needs of VLBI broadband experiments.

CHAPTER 4

IMPLEMENTATION OVERVIEW

In this chapter I will provide the overview of the logic blocks that enables formation of two hands of circular polarizations from two orthogonal linear polarizations in real time. I will provide an overview of the FPGA-based circular polarizer describing the functionality of the comprising logical elements, which operate sequentially. In appendix B, which is a crucial part connected to this chapter I will go into the details of each of these logical elements describing the operations performed to meet the desired functionality along with providing details on timing of the operations. Also in appendix B I discuss how the implementation of the developed digital circular polarizer was carried out where I show the implementation summary pages generated by the Xilinx software after implementation of the logic, which manifests that the implementation was successful. So please refer to appendix B for details of the logic blocks in this chapter.

4.1 Overview of main logic blocks of the digital circular polarizer

Fig. 4.1 shows the layout of the logical blocks for converting from linear to circular polarization in real time with arrows showing the direction of data flow between the connected logic elements. The digital circular polarizer is designed to operate for data sampled at a rate of 1024 Msamples/s. Now I briefly describe the functionality of the logic elements in fig. 4.1 and detailed analysis of these elements along with the timing of the operations involved will be provided in the next section.

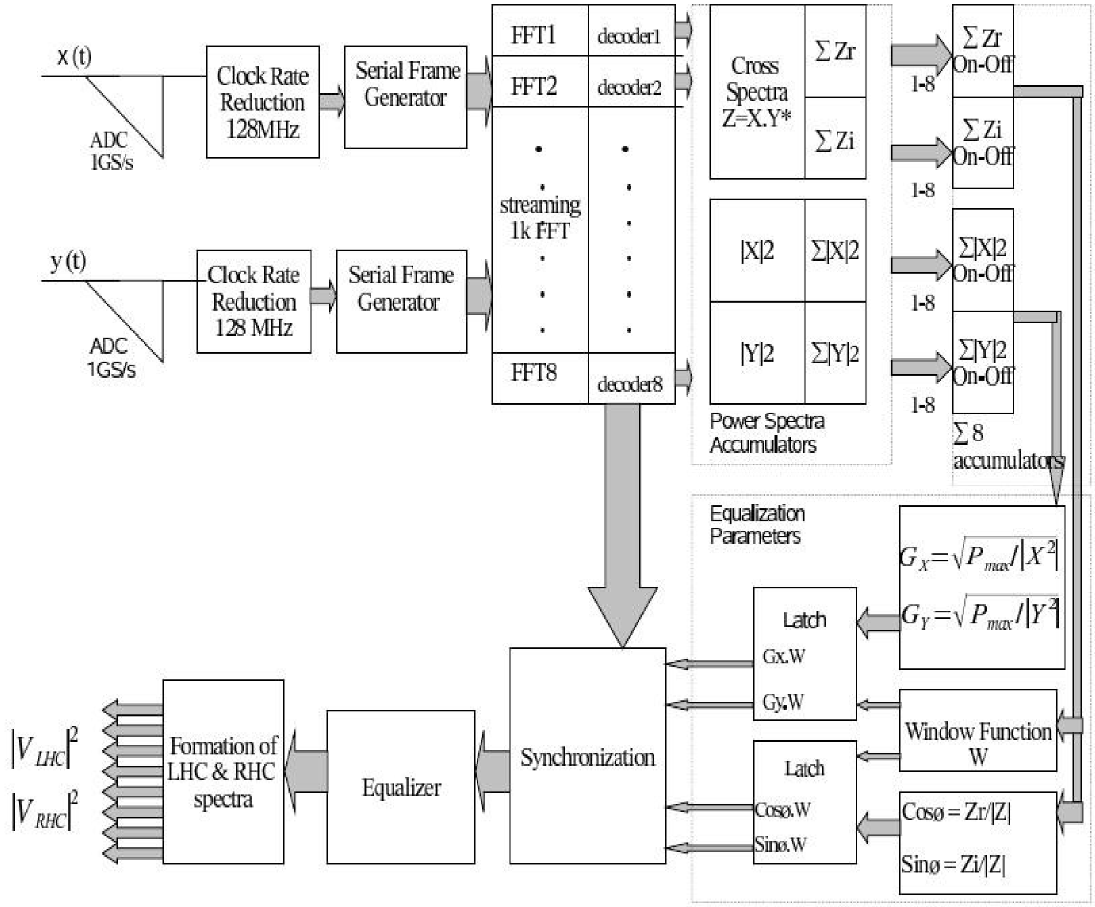


Fig. 4.1: Layout of the logic blocks for converting from linear to circular polarization in real time. The arrows depict the direction of dataflow between the connected logic elements.

1. ***Clock rate reduction logic*****:** Since the sampled time series at 1024 Msamples/s is too fast for processing, serial-to-parallel conversion of the streaming data has been implemented with a factor-eight fanout, generating eight parallel sample streams clocked at 128 MHz. This block was developed by Gino Tuccari in an earlier DBBC (Digital Base-Band Converter) project. The front-end of the system operates with a 1024 MHz sampling rate and the IF bandwidth is 512 MHz. The samples taken at 1024 MHz in the A/D converter are transferred in progressive steps in parallel to the next block, to reduce the clock rate, still maintaining a formal sampling clock of 1024 MHz. So in the serial to parallel conversion the data are demultiplexed from 1 sample at 1024 MHz to obtain 8 samples transferred in parallel on each clock cycle at 128 MHz. This system is already commercially available (Tuccari 2004).
Note- All the blocks described below operate at 128 MHz unless specified otherwise.

2. ***Serial frame generator*****:** This is a central block for enabling serial processing of real time data in parallel working at 128 MHz clock rate. To feed the eight identical streaming FFT blocks with frames of 1024 real time-domain samples, intermediate logic takes in eight samples in parallel at every clock edge and outputs them to one of eight buffers, eight samples in parallel at each clock edge. Once a complete frame of 1024 samples have been loaded into the buffer, the next buffer is selected to receive the next 1024 samples. The first buffer is read out serially into its corresponding FFT block at one eighth of the rate (128 MHz) at which it was filled. Reading from a buffer can start at the latest 128 clock pulses after writing the first eight samples is complete. In our case it starts two clock pulses after writing the first eight samples. The process of writing in one buffer after the previous buffer continues cyclically and the style of writing in a buffer and reading from the buffer remains the same and each buffer sends out data serially to the corresponding FFT block at a rate of 128 MHz without any overwriting.

3. ***FFT:*** The serial frame generator for the x polarization and for the y polarization each produce eight output lines to feed eight identical FFT blocks that run continuously and independently of each other, each processing successive frames of data to keep up with the real-time sampling rate. The FFT is a complex transform, but the data are purely real, and so I used the relation for the FFT of two real functions simultaneously, to save a factor of two in device resources by feeding the x and y polarization data to the real and imaginary channels of each FFT engine input. The streaming pipelined FFT is generated conveniently using a Xilinx IP core. It requires two's complement representation of the positive integers from the A/D converter. The real and imaginary terms of the X and Y spectra are retrieved from the FFT real and imaginary output channels by using a decoder at the output of each FFT. The method is equivalent to performing separate FFTs of the two real functions, but consumes half the space by performing them simultaneously in one transform.

4. ***Power spectra accumulator*****:** To calibrate the equalizer weights, I used total-power spectrum accumulators when determining the amplitude scaling factors needed for bandpass calibration and I used cross-correlation and accumulation to determine the phase difference between the X and Y polarizations in each frequency channel. The accumulators are dimensioned with enough bits to hold the accumulation results without overflow. To form the power spectra, each spectrum from each FFT (after decoding) is squared, forming $|X|^2$ and $|Y|^2$, and those are accumulated for nearly 8.4 s, which is determined by the goal of having thermal noise fluctuations that contribute at most 0.1° rms phase error during equalization. To measure the (frequency-dependent) phase difference between the X and Y polarizations, the cross-power spectra $Z = XY^*$ are formed and the real and imaginary terms, Zr and Zi, are also accumulated for nearly 8.4 s. Each of the four quantities ($|X|^2$, $|Y|^2$, Zr and Zi) are accumulated in eight pairs of accumulators, one following each FFT (after decoding), with each accumulator being paired to hold the noise diode on and off state results separately. After accumulation, the real-time processing of the calibration signal is stopped and calculations for the noise diode on and off states are performed sequentially on the accumulated results from all eight FFT engines. Differences are formed between the accumulators for the noise diode on and off states for each of the four quantities and the eight differences are accumulated in a new final accumulator to obtain the final integrated spectrum. Thus there are four final accumulators holding integrated spectra of the four quantities, which are read to obtain the equalizer gain and phase weights and the window function as described in the next two blocks.

Note- The following two blocks operate at two clock frequencies of 128 MHz and 64 MHz as required by the operations.

5. *Equalization parameters*: The phase and gain equalization parameters are calculated by using the outputs from the accumulators using equations (2.17), (2.18) and (2.26), (2.27) respectively. The division and square root operations are implemented using the Xilinx floating point IP core. The floating point operations are performed with a clock frequency of 64 MHz due to the frequency limitations of the IP core. The resulting phase and gain calibration parameters are latched in two respective registers for real-time equalization of the X and Y data streams during subsequent observations.

6. *Window Function*: A window function described in section (2.4) is determined by using outputs from the accumulators by following the same steps as in the description; After obtaining the window function it is latched along with the equalization parameters.

7. *Synchronization*: During observations the decoded FFT output samples are read out serially from the decoder and the corresponding gain, phase correction factors, and the window function are also read out serially from their respective latches and synchronization is required to ensure that the frequency channels of the spectrum and the equalization parameters are aligned. The synchronized values are passed to the next stage for phase and gain corrections.

8. *Equalization*: From this block again real-time operation starts. Phase and gain corrections and windowing are applied to each spectral channel of the decoder outputs, which implements equations (2.30) and (2.31). The resulting X and Y spectra are transferred to the next block for formation of circular spectra.

9. *Formation of circular polarization*: After equalization X and ±90º phase shifted Y are added to form the RHC and LHC polarizations. The ± 90º phase shifts of Y are implemented by exchanging the real and imaginary components of each spectral channel in the Y spectrum with a sign inversion of the real component for +90º phase shift and of the imaginary component for -90º phase shift. This block outputs the real and imaginary parts of the RHC and the LHC.

4.2 Conclusion

The logic demonstrated in this chapter and in appendix B is checked by simulating the design and is found to be correct. In the next chapter I will show the simulation results obtained by using data from an experiment. The simulations are carried out without any bit truncation using the design demonstrated in appendix B. The implementation that is carried out in the frontend of MPIFR and whose summary pages I have shown from fig. B.11 to B.13 have bit truncations; the bit truncations will become evident to the user by seeing the VHDL files and comparing with the ones enclosed in the CD or by seeing the design depicted in appendix B.

CHAPTER 5

EXPERIMENTS AND RESULTS

In this chapter I provide details of two experiments done to verify the design logic of the digital circular polarizer and to verify the polarization purity obtained from the digital circular polarizer. I simultaneously discuss the results obtained from the two experiments.

I first provide details on the design logic verification. In this section I first discuss the experimental setup to obtain the test data for simulating the design. Then I show the two time series or test data obtained from the digital oscilloscope used to sample the output signals from the two comprising channels in the experiment. I also discuss the resulting power spectra of the two channels. Next I show the simulation results of the verification; I present the plots obtained at the outputs of significant stages of the digital circular polarizer by passing the sampled test data through the design. I also compare these plots with mathematically obtained plots.

Next I provide details on the experiment done to verify the polarization purity obtained from our digital circular polarizer. In this section I first describe the signal flow through the experimental setup. Then I provide the measurement details, which includes measurement description, equalizer calibration, polarization purity measurement and data processing to form circular polarization. Next I discuss the simulation results where I show the two circular power spectra obtained at the outputs of the digital circular polarizer. I also present the plots manifesting polarization purity and facilitating the derivation of numbers like ellipticity and axial ratio. I provide numbers of cross-polarization or D-term and the formulas used to obtain them. Then I discuss the discrepancies in the observed response and the expected response from the digital circular polarizer.

Finally, I draw the conclusion from the results obtained in the experiments that it would be possible to obtain pure circular polarization over broad bands as required for radio astronomical applications.

5.1 Design logic verification and simulation results

We conducted a test to verify the design architecture described in chapter 4. I carried out the simulation for an ensemble of 24 spectra as that is sufficient to verify the design.

5.1.1 Lab setup for collecting test data

Fig. 5.1 shows the experimental setup. A noise source was split and passed through two filters. Outputs were taken as X (channel 1) and Y (Channel 2) channels which were sampled with two channels of a

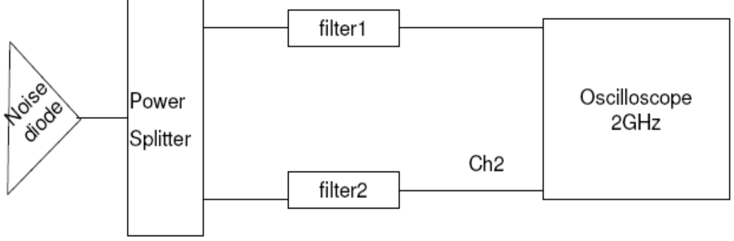


Fig. 5.1: Block schematic of the experimental setup to obtain sampled time series of the noise diode after passing it through the two channels consisting of two filters using a 2.5 GHz sampling oscilloscope.

digital storage oscilloscope (Tektronix DPO 7254). The noise source consisted of an avalanche diode and RF amplifiers. The power splitter, two filters, and internal resistance of the oscilloscope constituted the external circuit and load. The load resistance of the oscilloscope was 50 ohms.

5.1.2 Time series obtained from the digital oscilloscope in the experiment

After passing through the two filters, the two signals were sampled using a Tektronix DPO 7254 digital storage oscilloscope. An example of the time series obtained after sampling the signals from the two channels is shown below (fig. 5.2)

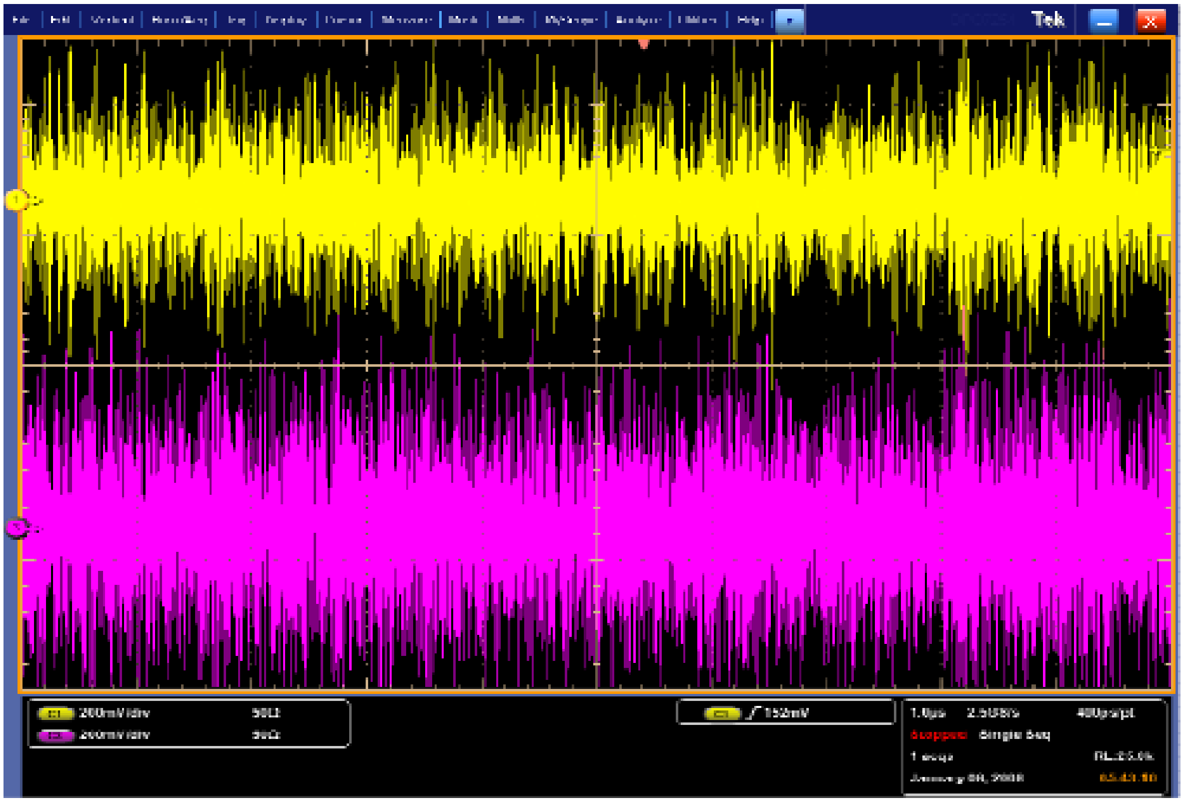


Fig. 5.2: Sampled time series from the *X* and *Y* channels using the Tektronix DPO 7254 oscilloscope. The sampling rate was 2.5 Gsamples/s producing a Nyquist bandwidth of 1.25 GHz.

The sampling rate was 2.5 Gsamples/s and hence the Nyquist bandwidth was 1.25 GHz. The time series thus obtained was converted to 10 bit binary numbers. These 10 bit binary numbers were used as test data to verify the functionality of the digital circular polarizer.

5.1.3 Power spectra from the two channels in the experiment

The following figures (fig. 5.3 top and bottom) show the pass-band for filter1 in channel 1 and for filter2 in channel 2. These are the power spectra of the two channels. The fluctuations are statistical since the input voltage is random. I performed accumulation of 24 spectra in MATLAB for getting these plots. The filter pass-bands are smooth with no such fluctuation. The fluctuations can be reduced by accumulating more spectra to reduce the noise by $\sqrt{N}$ where N is the number of spectra added.

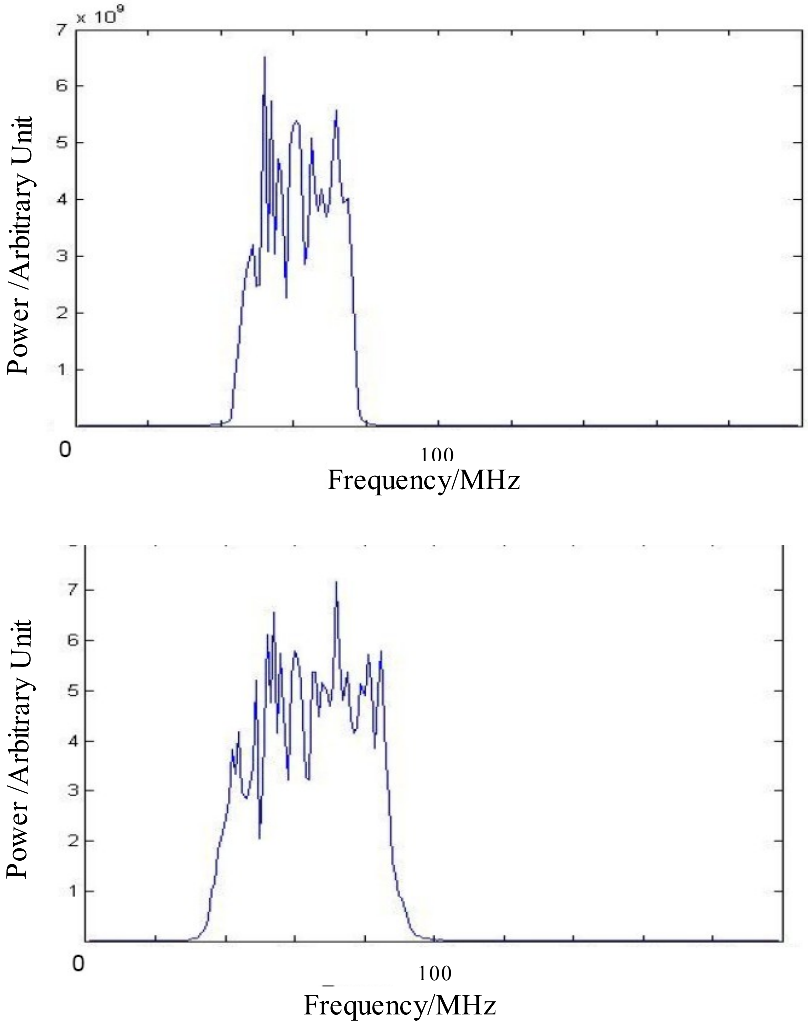


Fig 5.3: *Top:* Power spectrum from channel 1 in fig. 5.1 named as channel *X*. *Bottom:* Power spectrum from channel 2 in fig. 5.1 named as channel *Y*. These spectra are obtained after accumulating 24 spectra from corresponding channels.

5.1.4 Simulation results from the design logic

The verification of the design logic was done by checking correctness at two stages of dataflow in the design, which were after phase and gain equalization and after forming circular power spectra. These stages of dataflow can be referred from fig. 4.1. The correctness was checked by comparing the design logic simulation results against the simulation results in MATLAB without using the design but only the algorithm. The results from the two approaches were plotted on top of each other to show that the design logic was correct.

The verification of the logic was carried out in ModelSim, which is a simulation tool associated with Xilinx. ModelSim verifies the functionality of a design by verifying its outputs from known inputs. Internal signals in the design can also be viewed and verified. The bigger the design, the longer is the simulation run time. The inputs can be forced manually from the ModelSim wave window, which is the window for the plots or can be generated and forced into the design by connecting a custom designed data generator to the inputs of the design and the data pattern being similar to the actual data pattern at the inputs of the design. All the inputs, outputs and internal signals are visible in the wave window of ModelSim.

The simulation was carried out for 24 spectra each of which consisted of 1024 samples. The 24 spectra were fed to the design in the same manner as the digital circular polarizer would receive input data from the DBBC in real life, that is the data arrangements at the inputs were identical to that in practice. A data generator block was created whose output patterns look like the eight parallel 10 bit samples coming from the DBBC the samples being those from the time series of fig. 5.2. This data generator block was connected to the design to feed the inputs of the serial frame generator, which is the first block in the design shown in fig. 4.1. For more details on the functionality of the logic blocks please refer to chapter 4. The data passed through the various stages and reached the output terminals of the phase and gain equalizer from where the equalization parameters are taken to equalize the phases and gains of the two channels to form the two hands of circular polarization. The results from the outputs of phase and gain equalizer and from the outputs of the circular polarizer are compared with those from the algorithm in MATLAB. The resulting plots are shown in figures 5.4 to 5.12.

The curve labelled 'measured value' in the plots shows the result obtained by simulating the design and curve labelled 'true value' in the plots shows the result obtained in MATLAB simulation just using the algorithm. X represents channel 1 and Y represents channel 2 from fig. 5.4 to fig. 5.12.

-Plots verifying the design logic

Fig. 5.4 shows the phase difference between the two channels. This is the phase difference after phase equalization. It was expected that the phase difference would be 0 rad after equalization but there are deviations from 0 rad due to thermal noise fluctuations and due to using small number of spectra for accumulation which is not enough to cancel out the thermal noise fluctuations. However, our aim was to verify the design logic by comparing its outputs with those from the outputs of the algorithm, which is mathematical and the plots show that the two outputs are similar. Fig. 5.5 shows it clearly that the results obtained from the design and from the algorithm after phase equalization are identical. Fig. 5.6 shows the phase difference between the two channels before and after phase equalization and manifests how different they are. Thus the phase equalization has worked successfully. For more information on phase equalization please refer to section 2.3.1.

Fig. 5.7 verifies the design logic after gain equalization and windowing of channel X. The plot is for the pass-band of channel X. Again here the fluctuations are due to using small number of spectra for accumulation, which is not sufficient to reduce the noise fluctuations to 0. Fig. 5.8 shows the difference between gain equalized, windowed $|X|$ spectrum and the original $|X|$ spectrum before gain equalization and windowing and after accumulation. The fluctuations are much reduced after equalization. Fig. 5.9 and fig. 5.10 do the same for channel Y as fig. 5.7 and fig. 5.8 do for channel X respectively. Fig. 5.11 verifies the logic of gain equalized and windowed $|X|$ and gain equalized and windowed $|Y|$ spectra by comparing with the plots from the algorithm in MATLAB showing that they are identical. For more information on gain equalization and windowing of X and Y spectra please refer to section 2.3.2.

Finally, in fig.5.12 the two hands of circular power spectra are plotted. The minor variations between the two spectra are due to thermal noise fluctuations and should reduce with increased integration time by accumulating more spectra.

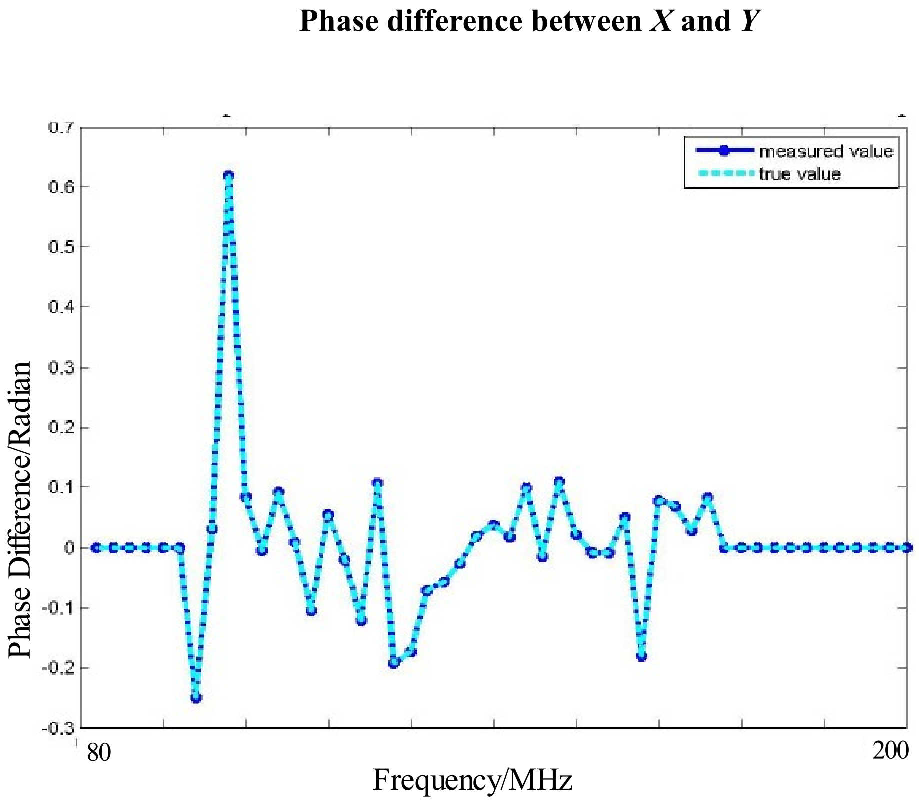


Fig. 5.4: Phase difference between the two channels obtained by using the design (measured value) and using MATLAB processed algorithm (true value) are compared. The plots show that they are identical.

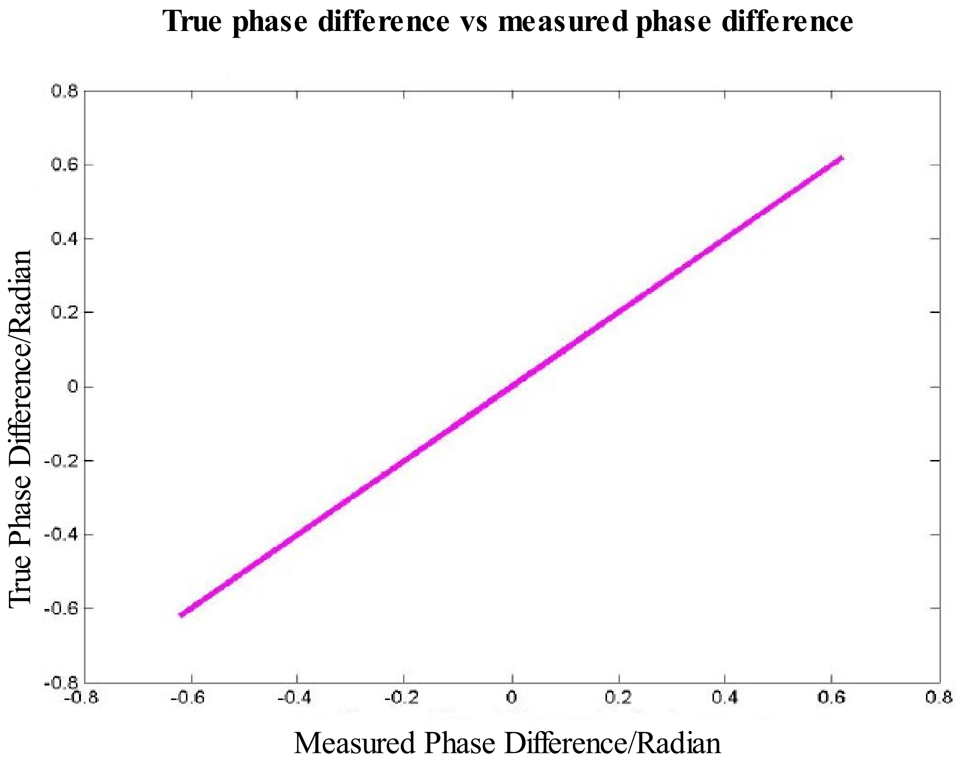


Fig. 5.5: The two plots of fig. 5.4 are plotted against each other to verify that they are exactly the same.

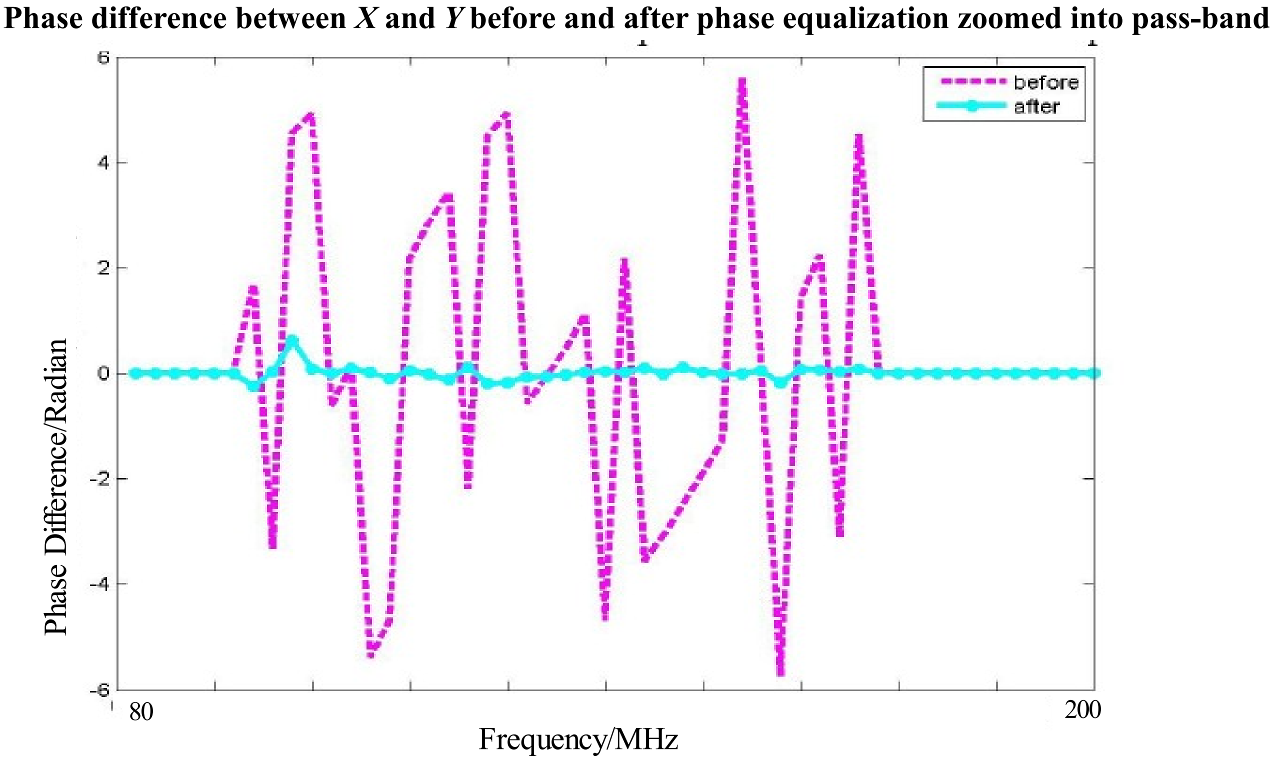

Phase difference between *X* and *Y* before and after phase equalization zoomed into pass-band

Fig. 5.6: The phase difference before and after phase equalization is shown. It is clear that the phase difference is very close to zero after equalization. It should be exact zero but due to thermal fluctuations there is variation, which reduces with increased integration time.

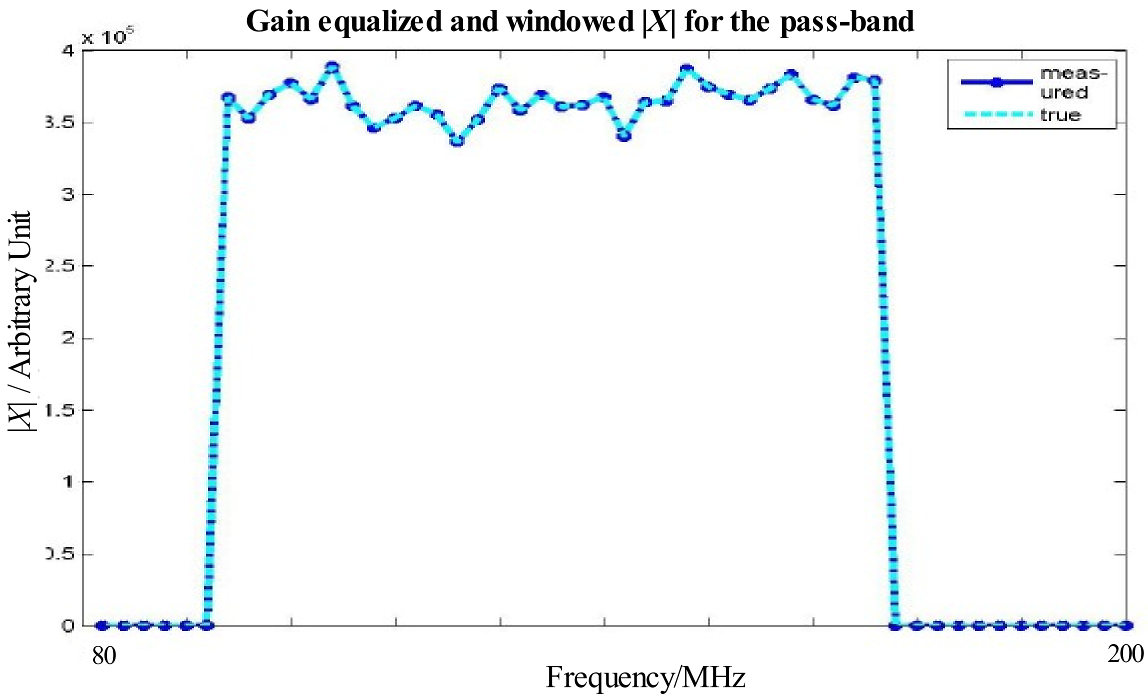

Gain equalized and windowed |*X*| for the pass-band

Fig. 5.7: Gain equalized and windowed channel *X* obtained by using the design (measured value) and using MATLAB processed algorithm (true value) are compared. They are similar.

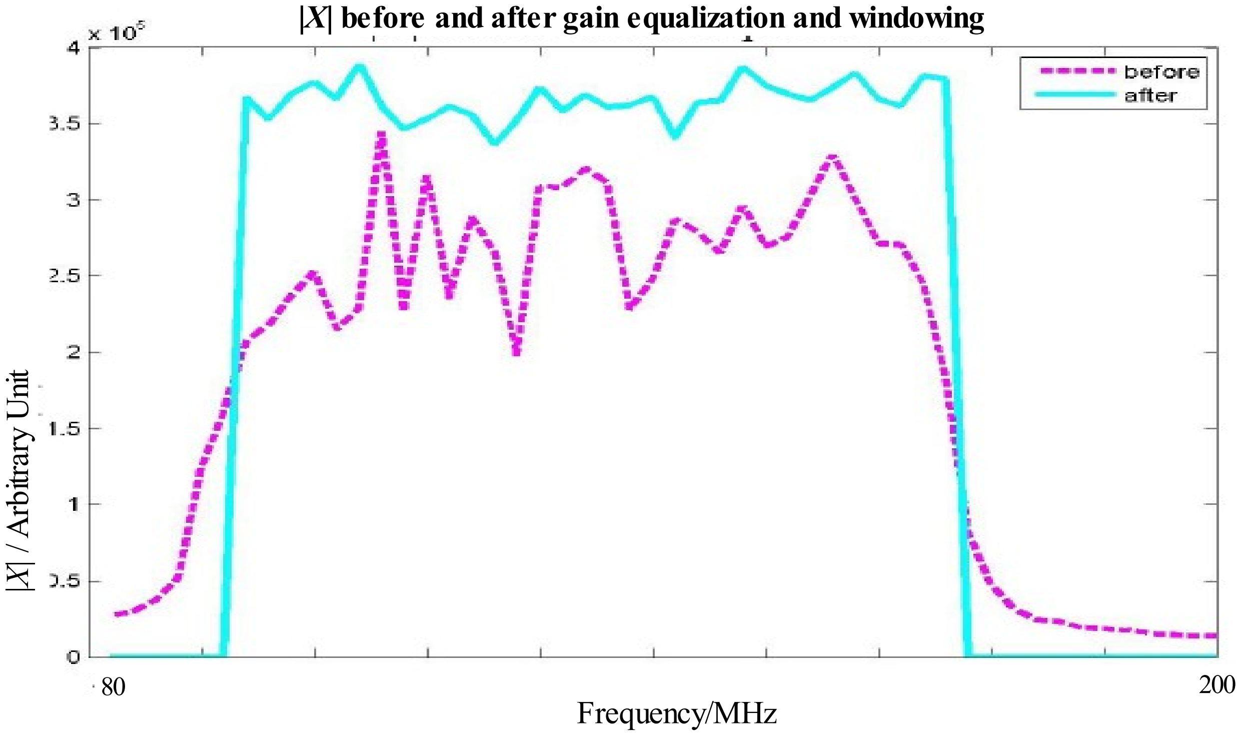


Fig. 5.8: Magnitude of accumulated *X* spectra before and after gain equalization and windowing. The fluctuations are much reduced after equalization and also the pass-band shape is prominent after windowing.

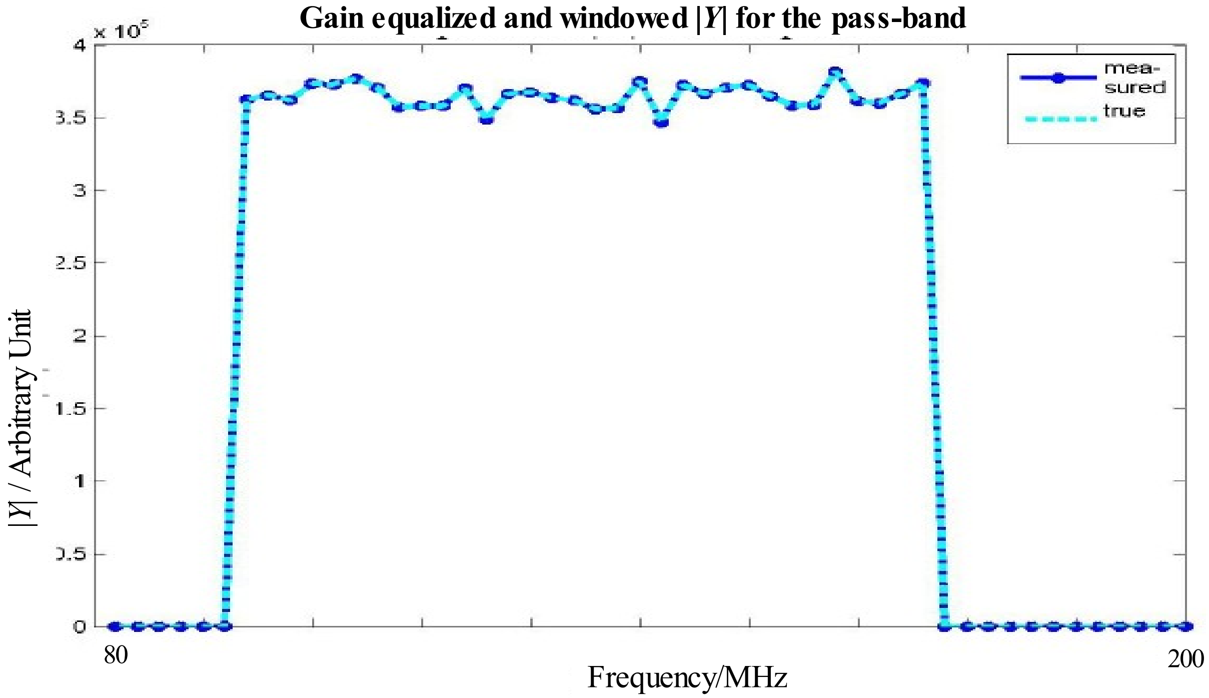


Fig. 5.9: Gain equalized and windowed channel *Y* obtained by using the design (measured value) and using MATLAB processed algorithm (true value) are compared. They are similar.

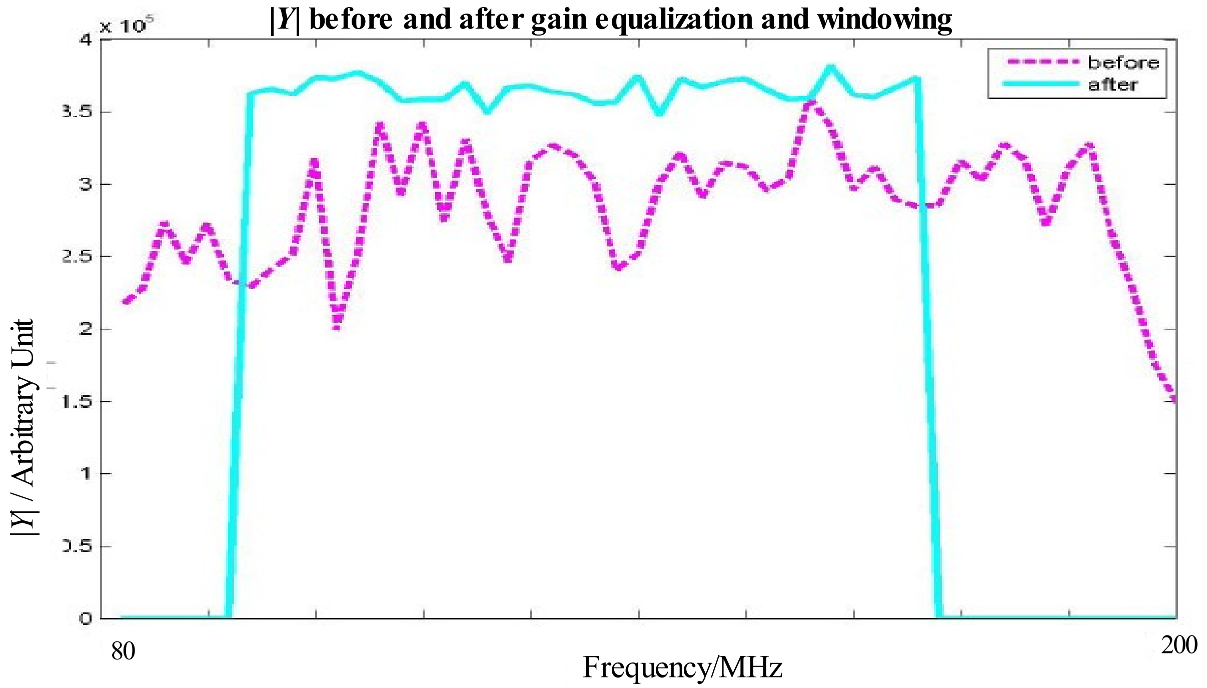


Fig. 5.10: Magnitude of accumulated *Y* spectra before and after gain equalization and windowing. The fluctuations are much reduced after equalization and also the pass-band shape is prominent after windowing.

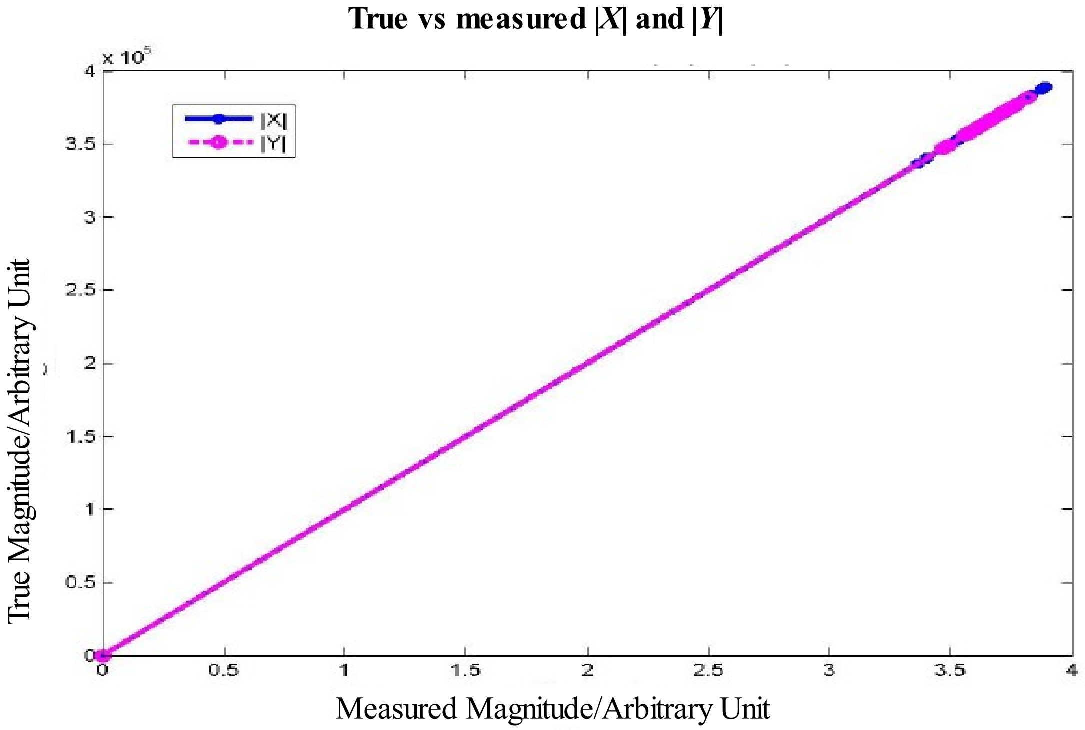


Fig. 5.11: Magnitudes of gain equalized and windowed |*X*| and |*Y*| spectra by using MATLAB processed algorithm and by using design are compared by plotting them against each other to show the similarity between the two.

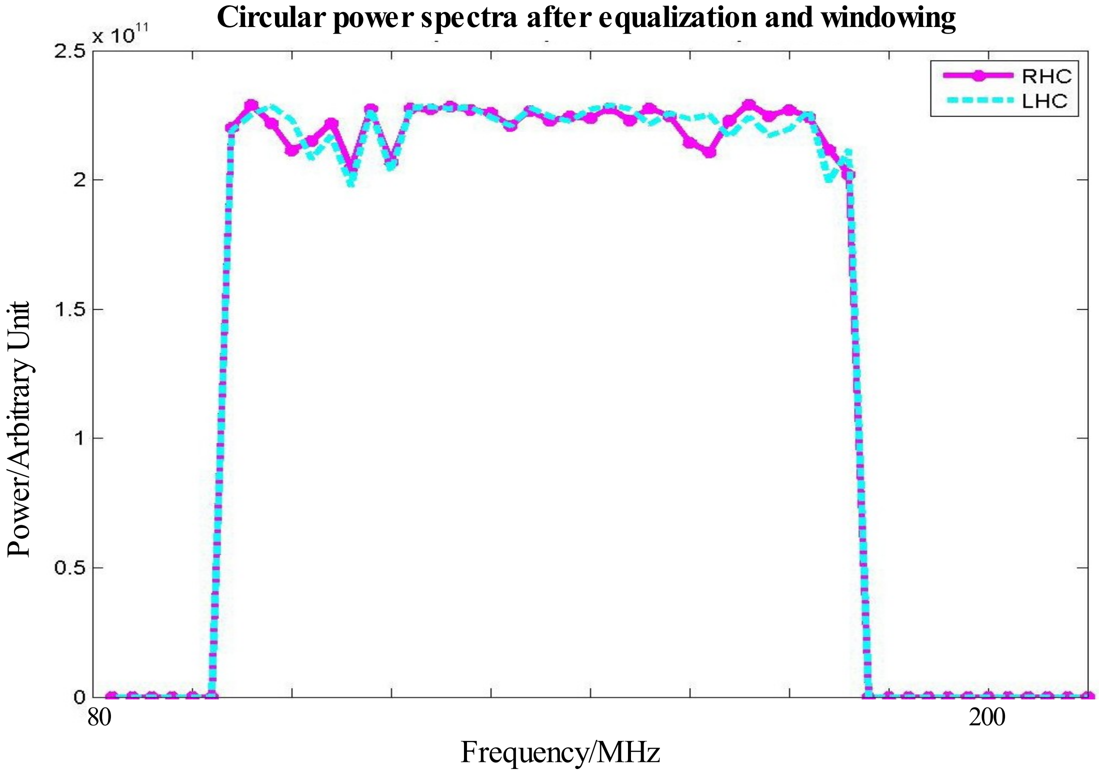


Fig. 5.12: Right-hand and left-hand circular polarization power spectra after phase and gain equalization and windowing. The difference is due to thermal noise fluctuations. These are the outputs from the two accumulators each containing the accumulated LHC and RHC power spectrum.

Thus the figures from 5.4 to 5.12 show that the design logic is correct.

After showing the verification for design logic I now demonstrate the experiment done to verify the polarization purity obtained from the digital circular polarizer. I now also show the simulation results obtained from the experiment.

5.2 Verification of polarization Purity

We performed the following experiment to measure the polarization purity obtained by this digital technique. The experiment was performed in an anechoic chamber by coupling linearly polarized broad-band noise into a circular waveguide by using a directional coupler and receiving with crossed linear dipoles at the end of the waveguide. However, the directional coupler coupled a linear polarization and a fraction of polarization orthogonal to the linear polarization. Since these two orthogonal polarizations had a phase difference, the resultant was elliptically polarized and was the one we were dealing with. While doing the setup the cross coupling was estimated to be better than -33.6 dB. This value of cross-coupling was determined by maximizing power in one dipole (the power in the dipole was maximized by aligning the major axis of the elliptical polarization parallel to it) and measuring the response in the other dipole. That response should have been 0 (no power). The power thus measured in the orthogonal polarization was -33.6 dB relative to or 33.6 dB less than the power in the parallel polarization. This cross-coupling thus included the effects of the orthogonal polarization coupled by the directional coupler and any imperfection in the 90° position angle between the two dipoles. The experimental setup is shown in fig. 5.13.

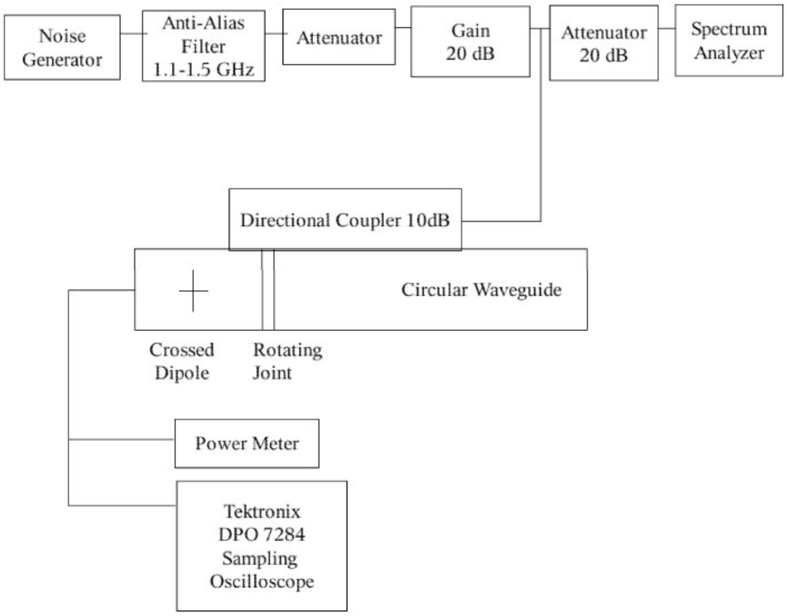


Fig. 5.13: Experimental setup to measure polarization purity in anechoic chamber. The broad-band noise from the noise generator was coupled into the circular waveguide through directional coupler and received by crossed linear dipoles at the end of the waveguide. The anti-alias filter limits the band and the attenuators and gains are for signal level adjustment. The rotating joint rotates the plane of polarization in the waveguide w.r.t the crossed dipoles. The power meter measures power in the two dipoles and the Tektronix sampling oscilloscope samples the data from the crossed dipoles.

5.2.1 Signal flow through the setup

The noise generator produced white Gaussian noise and the power spanned from 1160 MHz to 1462 MHz. The anti-alias filter was used to limit the band from 1.1 GHz to 1.5 GHz. After attenuating the signal it was passed through a directional coupler. The directional coupler coupled a fraction of the energy into the circular waveguide through probes inserted into the waveguide. The plane of the polarization was rotated using rotating joints in the waveguide and the signal was received by the crossed dipoles connected at the end of the waveguide. The crossed dipoles received signals for five position angles of the plane of polarization w.r.t the dipoles and the received signal was sampled by a digital storage oscilloscope and stored for later processing.

5.2.2 Measurement details

The spectrum of the broad-band noise that was coupled into the waveguide is shown in fig. 5.14. The noise power spanned from 1160 MHz to 1462 MHz (6 dB down from peak), representing a fractional bandwidth of 23 %. The power level at the edges of the Nyquist band, at 1000 MHz and 1500 MHz, were 37 dB and 30 dB below the peak in the band, and so very little power was aliased from outside the band. After receiving this signal with the crossed dipoles, the signal was under-sampled at a sample rate of 1000 Msamples/s by the digital oscilloscope, causing digital down conversion to baseband. Since the sample rate was slightly less than 1024 Msamples/s used by the design, the frequency channel width in this experiment was 0.976 MHz instead of 1 MHz. The input voltage range of -250 mV to +250 mV was translated to 0 to 1024 representing 10 bit positive integers before feeding the design for digital processing. No special effort was taken to match the complex gain or path length in the two channels from the receiving dipoles to the sampling oscilloscope. Thus, the RF band was translated to 160 MHz to 462 MHz, representing a fractional bandwidth of 97 % that was presented to the polarization converter.

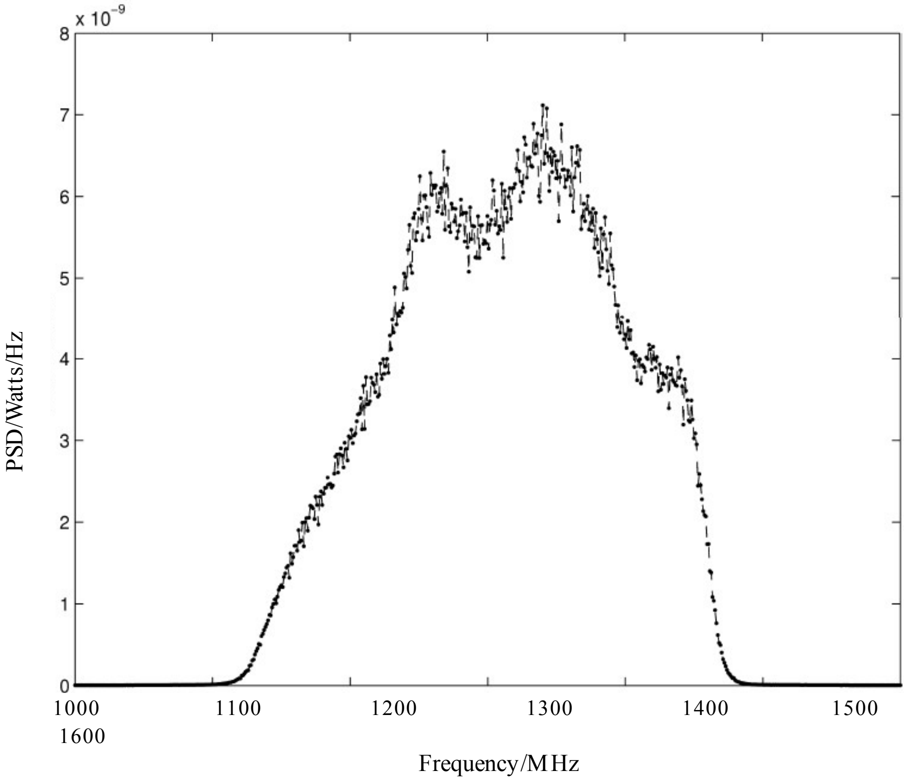


Fig. 5.14: Power spectrum of the broad-band noise measured at the input to the directional coupler by the spectrum analyzer shown in Fig. 5.13. The band was shaped using an anti-aliasing filter to ensure that the power level was low below 1000 MHz and above 1500 MHz to avoid aliasing of power from outside the third Nyquist zone during the later digital down conversion

1. Measurement Description:

The outputs of the crossed linear receiving dipoles were sampled with a digital storage oscilloscope for 4 ms and saved to file for later processing by the design. A rotating waveguide joint allowed us to rotate the plane of linear polarization with respect to the crossed receiving dipoles. Any ellipticity of the resulting circular polarization would show up as a change in the power in the circular polarization as the linear polarization is rotated.

2. Equalizer Calibration:

For equalizer calibration, the linear polarization was aligned at 45° to the two dipoles by connecting the dipole outputs to a two-channel power meter and rotated until equal power was measured in both channels. The resulting powers were +7.51 dBm and +7.50 dBm and drifted by ±0.14 dBm during the measurement, corresponding to a polarization rotational position angle uncertainty of 0.9°. For the noise diode off state, the data samples were simply set to zero rather than being measured, since the setup in fig. 5.13 was well shielded from RFI, in which case measuring with the noise diode off gives almost same results as setting the off-state samples to zero (verified in the previous experiment). The signals were sampled by the digital oscilloscope with the noise diode on and processed by the design to obtain the equalizer amplitude and phase weights. These were loaded into the equalizer for calibration of subsequent measurements, and are shown in fig. 5.15. The power spectrum in fig. 5.15 (top right) differs from the spectrum in fig. 5.14 since the transfer characteristics of the sampled spectrum are modified by the frequency response of each component through which the signal is transferred namely the directional coupler, circular waveguide, dipoles and the sampling oscilloscope.

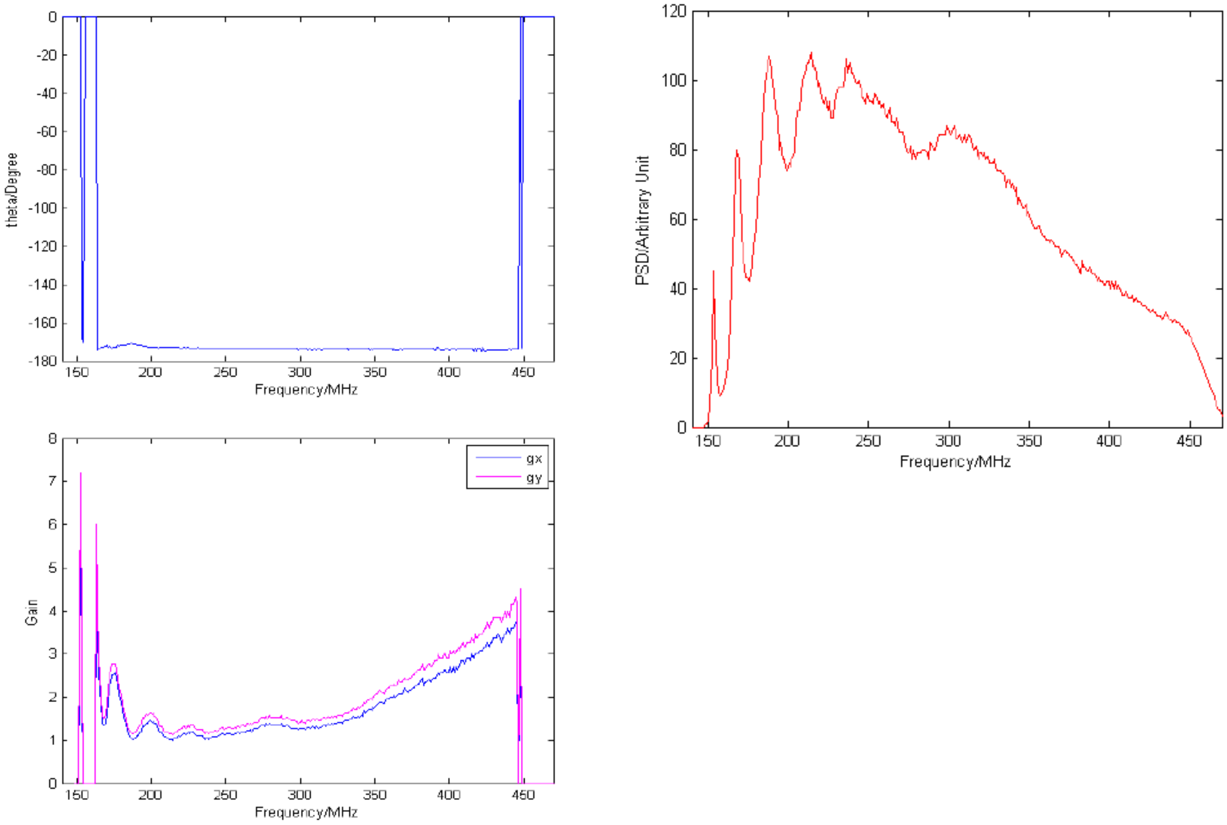


Fig. 5.15: Equalizer weights determined during calibration of the equalizer on the noise diode injected at the two receiving dipoles. *Top left*: phase difference between *X* and *Y* polarization channels. *Bottom left*: gains for *X* and *Y* polarization channels. *Top right*: power spectrum of the broad-band noise measured at the dipole outputs by the oscilloscope. The frequency range is labeled corresponding to the band after digital down conversion from the RF band of 1000 MHz to 1500 MHz to baseband of 0 MHz to 500 MHz. The gains are approximately the inverse of the square root of the noise spectrum at the top right as expected. The gains and phases are set to zero by the window function where the power dropped 6 dB below the maximum, which happened below 150 MHz and above 450 MHz. This is as expected, given the shape of the broad-band noise spectrum.

3. Polarization Purity Measurement:

For polarization purity measurement, the plane of input linear polarization was rotated to five positions with respect to the receiving dipoles and the received signals from both dipoles were sampled in each position. To adjust the rotational position angle accurately, the plane of input linear polarization was rotated to either minimize the power in one of the receiving dipoles (90º, 0º, -90º) or to obtain equal power in both dipoles (45º, -45º). The time elapsed between calibrating the equalizer and making all the measurements for determining the polarization purity was some 2 h. During this time, some drift in components might have occurred, but nevertheless good polarization purity was obtained.

4. Data Processing to Form Circular Polarization:

I processed the data by running the design in software logic simulation, using ModelSim SE on suse 10.3 Linux machines having 16 GB of RAM and 2.7 GHz clock speed. Processing of 4 ms of data from each position required 4 days of elapsed time on a single computer so I processed for each position 20 sets of $1/20^{th}$ of the data in parallel, which required one day.

5.2.3 Results

The resulting power spectra in LHC and RHC are shown in fig. 5.16 for one of the five position angles of the input linear polarization on top of each other. These show that the gain equalization flattened the spectra and that the window function truncated the spectra where the filters roll off. The total power is found to change very little with rotation of the input linear polarization, signaling a high purity circular polarization. To quantify the purity, I measured the total power in LHC and RHC as a function of rotation angle, and show this in fig. 5.17. This shows power level changes of around 1 part in 200 peak-to-peak over all position angles as a function of rotation angle.

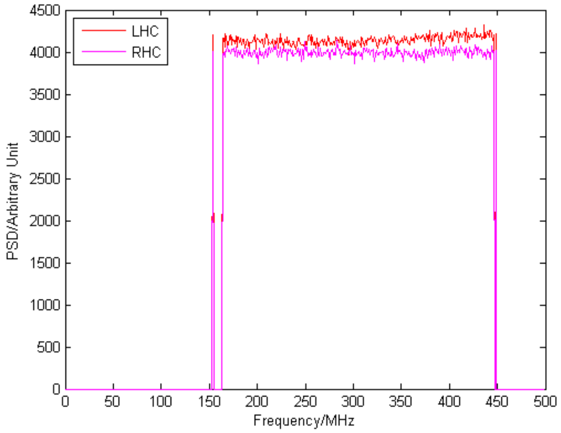


Fig. 5.16: Power spectral densities for LHC and RHC for a single position angle. The gains flattened the spectra and the window function truncated the spectra where the filters roll off giving a definite shape to the pass-band.

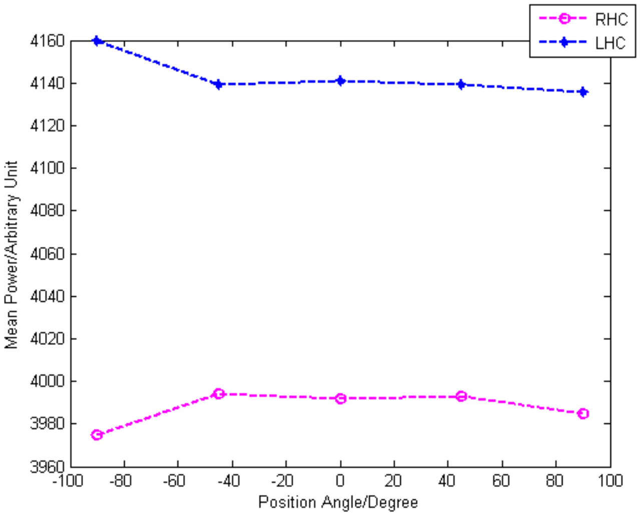


Fig. 5.17: Mean output powers in LHC and RHC as a function of the rotational position angle of the input linear polarization. The total power is found to change very little with rotation of the input linear polarization, signaling a high purity circular polarization.

We obtained ellipticities of 0.9971 and 0.9976, axial ratios of 1.0029 and 1.0024, and D-terms of 0.0014 and 0.0012 meaning cross-polarizations of -57.08 dB and -58.42 dB for LHC and RHC polarizations respectively.

Ellipticity and axial ratio:

Ellipticity is calculated from fig. 5.17. as

$$\sqrt{minimum\ power / maximum\ power}\ , \tag{5.1}$$

which gives 0.9971 and 0.9976 for LHC and RHC respectively and axial ratio is the reciprocal of ellipticity.

D-term or cross-polarization:

D-term is obtained by using the formula (Perley 2009)

$$|D| = (1 - \text{ellipticity}) / (1 + \text{ellipticity}), \tag{5.2}$$

which gives 0.0014 and 0.0012 for LHC and RHC respectively. The cross-polar response is simply the D-term in dB given by 20 log |D|, which gives -57.08 dB and -58.42 dB for LHC and RHC respectively. These values for D-terms are upper limits considering measurement errors.

Since the cross coupling of -33.6 dB caused by the directional coupler is more than the obtained cross-polar response, we infer that in the process of phase and gain equalization we have lowered the contribution of D-term to the measured power in the wanted polarization component somehow. The uncertainty on the polarization purity measurement has contributions from thermal noise, mechanical tolerance in the rotating waveguide joint, and in the repeatability of RF connections that we had to disconnect and reconnect during the measurement for connecting alternately the power meter and the oscilloscope. These can be estimated from the small changes of the power in the circular polarization formed when the input linear polarization was at -90º and at 90º. Those two positions are symmetric and the resulting powers should be equal, regardless of the ellipticity of the transmitting or the receiving antenna. We found fractional changes of 0.001 in the LHC power and 0.0004 in the RHC power between these two position angles. These are comparable to the peak power variations seen as we rotated the input linear polarization, and so the polarization leakage measurement is limited by mechanical tolerances in the apparatus. The thermal noise contribution was minor - the fractional error due to thermal noise fluctuations in the total power measurement was only 0.0006.

5.2.4 Discussion

A small systematic offset is seen between LHC and RHC powers (fig. 5.16). A small error in the magnitudes of the rotation matrix elements will cause the phase rotation of Y to introduce an offset between the LHC and the RHC powers. This can happen since truncation error can cause small differences in the obtained phase difference and the actual phase difference. The following equations show the dependence of the offset on the rotation angle error $\theta_{\epsilon}=\theta_y(r\omega_0)+\theta(r\omega_0)-\theta_x(r\omega_0)$.

$$V_{LHC}(r\omega_0)=X'(r\omega_0)-jY''(r\omega_0) \tag{5.3}$$

$$= \quad g_x(r\omega_0)|X(r\omega_0)|e^{j\theta_x(r\omega_0)} - j\,g_y(r\omega_0)|Y(r\omega_0)|e^{j\theta(r\omega_0)}e^{j\theta_Y(r\omega_0)} \tag{5.4}$$

$$= e^{j\theta_X(r\omega_0)}\left(g_x(r\omega_0)|X(r\omega_0)| - j\,g_y(r\omega_0)|Y(r\omega_0)|e^{j\theta_\epsilon(r\omega_0)}\right) \tag{5.5}$$

Therefore,

$$|V_{LHC}(r\omega_0)|^2 = g_x^2(r\omega_0)|X(r\omega_0)|^2 + g_y^2(r\omega_0)|Y(r\omega_0)|^2$$

$$+ 2\mathrm{g}_x(r\omega_0)\,g_y(r\omega_0)|X(r\omega_0)||Y(r\omega_0)|\sin\theta_\epsilon(r\omega_0) \tag{5.6}$$

Similarly,

$$V_{RHC}(r\omega_0) = X'(r\omega_0) + jY''(r\omega_0) \tag{5.7}$$

$$= g_x(r\omega_0)|X(r\omega_0)|e^{j\theta_x(r\omega_0)} + j\,g_y(r\omega_0)|Y(r\omega_0)|e^{j\theta(r\omega_0)}e^{j\theta_Y(r\omega_0)} \tag{5.8}$$

$$|V_{RHC}(r\omega_0)|^2 = g_x^2(r\omega_0)|X(r\omega_0)|^2 + g_y^2(r\omega_0)|Y(r\omega_0)|^2$$

$$- 2\mathrm{g}_x(r\omega_0)\,g_y(r\omega_0)|X(r\omega_0)||Y(r\omega_0)|\sin\theta_\epsilon(r\omega_0) \tag{5.9}$$

Hence, the offset between the two power spectra is given by

$$|V_{LHC}(r\omega_0)|^2 - |V_{RHC}(r\omega_0)|^2 = 4\mathrm{g}_x(r\omega_0)\,g_y(r\omega_0)|X(r\omega_0)||Y(r\omega_0)|\sin\theta_\epsilon(r\omega_0) \tag{5.10}$$

for a single channel.

The RHC power is almost flat across band but LHC showed a small change with frequency. To quantify this variation, I took equally spaced points from each power spectrum in the band of interest and plotted the deviation of those points from the mean of the respective power spectrum. Fig. 5.18 shows the deviation of LHC and RHC power spectra from their mean values. The plots are derived from LHC and RHC power spectra obtained by applying the equalization parameters to the same data samples from which the equalization parameters are obtained to show the deviations caused by bit truncation. From around 300 MHz onwards (fig. 5.18 right) the two polarizations deviate differently with increasing frequency. This difference can be explained in terms of truncation of numerical precision. I have truncated bits at stages before and after formation of LHC and RHC powers, determined by the input signal levels and by the aim to keep phase errors < 0.1°. The truncation error will cause offsets in the two circular power spectra shown in eqs. (5.6) and (5.9), to deviate unequally from their mean and hence, the deviation can be more in one than the other. I verified in simulations that the offsets and the distortions seen in LHC and RHC power spectra increases in proportion to the number of bits discarded. However, the effects are small and we nevertheless obtained good polarization purity.

The equalizer applies phase correction on 1 MHz channel spacing, so we must ensure that when the VLBI correlator later subdivides the spectrum into finer frequency channels that the phase equalization is smoothly interpolated and does not result in a sawtooth with 1 MHz spacing. I performed a numerical simulation that confirmed that when we applied equalizer weights at 1 MHz spacings, transformed back to time domain, then transformed to frequency domain with 0.5 MHz spacing (by adjoining two 1024 point frames of time domain data) then the intermediate points at n + 0.5 MHz, where n is an integer were seen also to have been phase corrected by the equalizer phase.

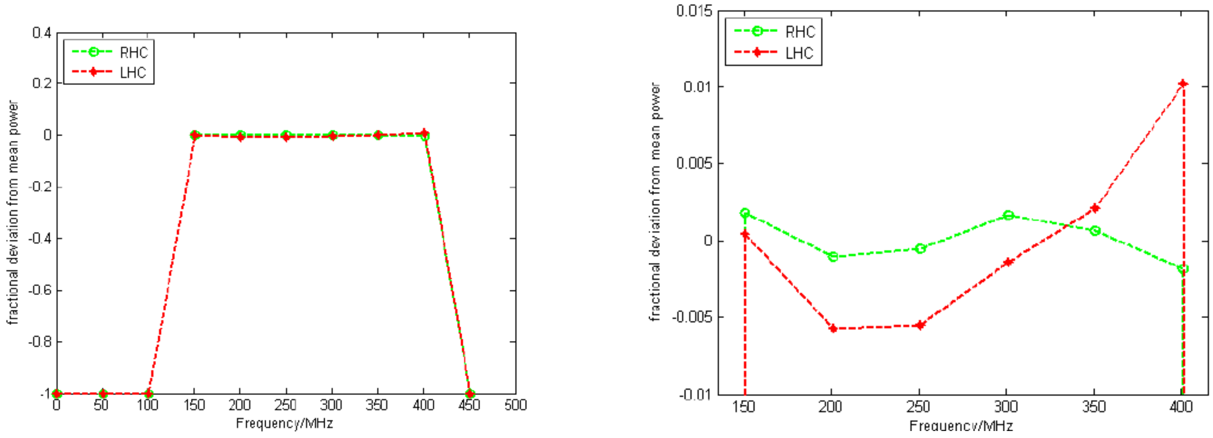


Fig. 5.18: *Left:* fractional deviations of LHC and RHC powers from their mean values across the band of interest. The spikes at 160 MHz in the two spectra (fig. 5.16) are excluded. *Right:* same as left, but zoomed in to show the trends of the deviations in the two power spectra.

5.3 Conclusion

The design logic is found to be correct and a polarization purity of -58 dB as obtained including all limiting factors in the measurement is good. This value of D-term is an upper limit. The method could yield almost ideal results, if the measurement limitations were absent. Thus it is verified that the digital circular polarizer would produce pure circular polarization over the whole broad band. Hence, it is worth deploying this FPGA based firmware in radio telescopes for observational accuracy.

CHAPTER 6

APPLICATIONS IN RADIO ASTRONOMY

In this chapter I will first discuss the significance of our digital circular polarizer in radio astronomical applications. Then I will discuss various astronomical phenomena that are associated with the generation of polarization to provide an overview of the fundamental processes whose explorations would be supported by polarization observations with our digital circular polarizer. I will also discuss various depolarizing effects, which should be taken into account while interpreting polarization data. Next I will go through various interesting and significant VLBI explorations and techniques where our digital circular polarizer would contribute to observing facilities. These include studies of Galactic magnetic fields by the Square Kilometer Array (SKA), studies of emission mechanism in Sagittarius A* (Sgr A*), studies of circular polarization in active galactic nuclei (AGN) and studies of cluster of galaxies like the Perseus cluster providing information on structure formation in the universe. Finally I conclude this chapter summarizing the utility of our digital circular polarizer in the described types of explorations.

6.1 Significance of the digital circular polarizer in radio astronomy

Polarization studies in radio astronomy are important to understand various fundamental phenomena and unveil cosmic source properties. VLBI, which covers a broad spectrum of radio astronomy and which facilitates astronomical sources to be resolved with sub-milliarcsecond synthesized beam widths, is simplest with circular polarization due to geometrical and stability considerations, which are described in section 1.1.

Also it is convenient to meet the bandwidth requirements for broad-band observations if native linear dipoles are used for receiving the signals since circular feeds do not meet the bandwidth requirements and hence a method to convert the received linear polarization into circular polarization would find significant application in VLBI. The method used by us can produce pure circular polarization for the whole frequency range irrespective of how broad the band is unlike present analogue circular polarizers. I have developed the digital circular polarizer for 500 MHz bandwidth with 1 MHz channel widths, which can be used for broadband applications.

Before going into the details of various interesting VLBI explorations which will be supported technically by our digital circular polarizer alleviating several observational inaccuracies, I would introduce the mechanisms responsible for producing polarization in the sky so that the reader gets an overview of the fundamental processes which can be unveiled by using polarization observations.

6.2 Different astronomical phenomena generating polarization

Polarization of radio emission often occurs due to the cyclotron and synchrotron mechanism, Zeeman effect in atoms and molecules, plasma oscillations in the solar atmosphere, Thompson scattering and Brewster angle effects. I will now go into the details of how polarization is associated with each of these processes.

6.2.1 Cyclotron and synchrotron emission

In a cyclotron a charged particle acquires high enough energy by crossing an electric field under the effect of a perpendicular magnetic field, which changes its direction to cross the same electric field repeatedly. It gets accelerated and emits an electromagnetic wave. The oscillating frequency is the same as the frequency of motion of the charged particle under the influence of the two fields. Synchrotron emission is a special case of cyclotron emission where the magnetic field that is responsible for changing the direction of the moving particle and the electric field that is responsible to accelerate the particle are synchronized that is they are changed in order to keep the path of the particle constant even when it gains energy due to the acceleration. These are laboratory definitions from where the terms cyclotron and synchrotron came.

In nature, one gets cyclotron radiation if the radiating electron has sub-relativistic speed, and synchrotron emission if the speed is relativistic under the influence of an acceleration perpendicular to its velocity. There is a third kind of emission called cyclo-synchrotron emission which occurs if the speed of the electron is trans-relativistic that is the speed lies in between that of cyclotron and synchrotron. The resulting emission spectra are different in the three cases. Synchrotron emission shows a continuous spectrum while cyclotron emission occurs mostly at the cyclotron frequency with small fraction of energy at higher harmonics. In cyclotron the emission is linearly polarized when observed perpendicular to the magnetic field and is circularly polarized when observed parallel to the magnetic field. For relativistic speeds like in synchrotron (see Jackson 2009), polarization seen parallel to the plane of motion is much stronger in intensity than polarization seen perpendicular to the plane of motion. The intensity increases with decrease in angle between the line of sight and plane of motion. The polarization is mostly linear and very small circular polarization can be seen. The degree of circular polarization increases with increase in angle between the line of sight and plane of motion accompanied with reduced power emission.

6.2.2 Plasma frequency and plasma oscillations

For any electromagnetic wave to be able to travel through a plasma, the frequency of the wave must be greater than the plasma frequency (arising due to natural oscillations in plasma) otherwise the wave number would be purely imaginary and the wave would be attenuated away. For more details refer to Griffiths (1997). Hence, a plasma is transparent to the waves whose frequency is greater than the plasma frequency and is opaque to those having a frequency lower than that of it. The plasma oscillations resulting from the motion of a certain number of charged particles per unit volume results in polarized electromagnetic waves though this polarization is different from the polarization of the electromagnetic wave traveling through the plasma.

6.2.3 Zeeman effect

Another phenomenon that can lead to polarization is the Zeeman effect. In the Zeeman effect, the electron configurations having the same energies, that give rise to single spectral line in case of transition between them, reconfigures in the presence of a magnetic field. The magnetic field breaks this degeneracy and modifies the energy of the electrons according to their quantum numbers. Hence the energies of these configurations are now different instead of being the same. A transition between these states would produce very closely spaced spectral lines. The Zeeman effect also exhibits polarization when observed longitudinal or transverse to the magnetic field. For different transitions, different polarization states are observed.

6.2.4 Thompson scattering

Thompson scattering is caused by elastic scattering of electromagnetic wave by a free charged particle (it is non relativistic Compton scattering). When an electromagnetic wave is incident on a particle, it gets accelerated due to the electric field of the wave. The acceleration causes emission of a wave whose frequency is same as that of the incident wave. The wave is polarized in the direction of motion of the charged particle and the radiation is perpendicular to the direction of motion.

6.2.5 Brewster angle effects

Polarization also results from reflection at the Brewster angle. When an electromagnetic wave encounters a boundary between two media with different refractive indices then a fraction of it is reflected and another fraction is transmitted. For polarization parallel to the plane of incidence if the angle of incidence is the Brewster angle, then the reflected component is zero. If a wave of arbitrary polarization is incident on a medium at the Brewster angle then the reflected wave is totally plane polarized with the electric field vector in the direction perpendicular to the plane of incidence. Polarization due to Brewster angle effects is observed at planetary surfaces.

After discussing various astronomical phenomena that can cause polarization I will now describe several depolarization effects, which are very common in polarization studies and which are responsible for changing the polarization angles of the observed radiation.

6.3 Depolarization effects

6.3.1 Depolarization due to Faraday rotation

If a linearly polarized wave travels through a lossless plasma in the direction of an existing magnetic field, then the phase velocities of the two circular polarization components of the linear polarization will be different. This is due to the fact that the magnetic field causes motion of the charged particles present in the plasma. These charged particles will move in the same direction as one hand circular polarization but opposite to that of the other hand. Hence, the index of refraction for one will be lower than that of the other. This difference in the indices of refraction leads to change in phase difference between the two circular polarization components which is equivalent to changing the original position angle of the linear polarization (a linear polarization with position angle ψ has two circular polarization components with phase difference 2ψ), which is called Faraday rotation of the electromagnetic wave. Faraday rotation occurring within a source may depolarize the emergent radiation. This is due to the fact that radiation emitted from different depths in a source undergoes different amounts of Faraday rotation and the percentage polarization of the observed radiation is reduced.

6.3.2 Bandwidth depolarization

Bandwidth depolarization occurs when the polarization angles change significantly across the frequency band and the polarization of the observed emission is reduced. It is given by $\mathrm{sinc}(2RM\,\lambda^2\,\delta\nu / \nu)$ (e.g. Tabatabaei et al. 2008) where RM is the rotation measure (slope of polarization angle versus wavelength squared), λ is the wavelength of observation, $\delta\nu$ is the bandwidth of observation and ν is the frequency of observation.

6.3.3 Beam depolarization

If the orientation of the polarization vectors varies within the telescope beam then the percentage of the observed polarization is reduced and this effect is called beam depolarization.

Hence polarization studies and observations can provide information on the above described astrophysical phenomena. Also observations at different frequencies yield information on the source properties. Now I will proceed to go through various significant VLBI explorations that require advanced techniques for correct interpretation of data and our digital circular polarizer would contribute towards the technical needs of such explorations.

6.4 Explorations and underlying techniques requiring polarimetric observations in VLBI

In this section I will go through significant explorations done by and lying ahead of the astronomy community that entailed or entails advanced polarimetric observational methods and techniques. I will also go through the methods and techniques facilitating the explorations. The digital circular polarizer is developed for observational convenience of these kinds of experiments and to contribute towards meeting their scientific goals.

6.4.1 Magnetic field studies by the SKA

In this section we review “Observations of magnetic fields in the Milky Way and in nearby galaxies with a Square Kilometre Array” studied by Beck at al. (2004).

Studies of magnetic fields often provide insight to the structure and evolution of galaxies and of the interstellar medium in the universe. For instance, the structure of magnetic field provides information on source distribution and the magnetic field topology of sources tells about their origin. The astrophysical processes like Faraday rotation, Zeeman splitting, synchrotron and cyclotron radiation occur in the presence of magnetic field and polarization of radio emission is often associated with these processes and hence polarization studies of the galaxies and interstellar medium can provide information on the configuration of Galactic magnetic fields.

Rotation measure (*RM*), which is a measure of change in polarization angle (due to Faraday rotation) with respect to change in squared wavelength or $RM = \Delta\chi/\Delta\lambda^2$ where χ is the polarization angle and λ is the corresponding wavelength, is a stronger technique for exploring magnetic fields in the interstellar medium as compared to synchrotron emission or Zeeman effect in the regions away from the sites of active star formation. This is due to the fact that synchrotron emission can only be identified in regions of high density of cosmic rays and strong magnetic fields and similarly Zeeman effects entail strong magnetic fields. On the other hand Faraday rotation occurs due to dual refraction of the traveling radiation in the plasma in the presence of a magnetic field, which may occur even in diffuse regions and the change in position angle of the wave can be detected by multi-frequency and polarimetric observations. A change in polarization angle with change in frequency following a common trend will indicate the presence of a magnetic field. Single sources generally show smooth *RM* gradients but sources with complex distributions show more complex *RM* distribution. Further by carrying out rotation measure synthesis (described later in section 6.4.4) emissions at different Faraday depths (each Faraday depth is responsible for a certain rotation measure) can be observed.

By following Burn (1966), Thompson et al. (2001), Beck et al. (2004) and Brentjens et al. (2005), I see that the definitions for *RM* and Faraday depth are inconsistent. What is called as *RM* in Thompson et al. (2001) and Beck at al. (2004) is called as Faraday depth in Burn (1966) and Brentjens et al. (2005). However since each Faraday depth is responsible to produce a certain *RM,* accounting for a certain *RM* would mean to account for the corresponding Faraday depth. So I remain consistent with Beck et al. (2004).

RM in rad/m^2 is defined as

$$RM = K\int B\cos\theta\, n_e\, dl \tag{6.1}$$

where $K \approx 0.81$ rad m^{-2} pc^{-1} cm^{3} μG^{-1}. B, θ and n_e are the magnetic field strength, angle between the magnetic field and line of sight of the observer and number density of thermal electrons respectively. The integration is carried along the line joining the source and the observer.

Further, let us consider a wave emitted at a polarization angle χ_0 corresponding to $\lambda=0$ and is observed at a wavelength λ. Then the measured polarization angle χ is given by

$$\chi(\lambda^2)=\chi_0+RM\,\lambda^2 \tag{6.2}$$

where λ is in m, χ_0 and χ are in radians. This is clearly the eqn. of a straight line having a slope *RM*.

Note- In eqn. (6.2), $\chi(\lambda^2)-RM\,\lambda^2$ corresponds to derotation of the polarization vector at wavelength λ with respect to the polarization vector at wavelength 0 ($\lambda=0$). If χ_0 is replaced by $\chi(\lambda_0^2)$ where λ_0^2 is a point in λ^2 axis and $\chi(\lambda_0^2)$ is the corresponding polarization angle then a derotation by an amount $\chi(\lambda^2)-RM(\lambda^2-\lambda_0^2)$, which is equal to $\chi(\lambda_0^2)$ would mean to derotate $\chi(\lambda^2)$ with respect to the polarization vector at wavelength λ_0 accounting for the same *RM*. This description will be useful in section 6.4.4.

Measurements of *RM* will provide information on the direction and strength of magnetic field along the line of sight of the observer provided we have some information about the number density of thermal electrons. Further we can group diffuse polarized emission measured in narrow bands over a broad frequency range to enable Faraday tomography to be carried out where different layers of polarized emission are penetrated by different frequencies. Thus by multi-frequency observations any field structure along the line of sight can be identified uniquely. With large number of channels (required for fitting *RM*) *RM* synthesis becomes feasible, which is described clearly in section 6.4.4.

If enough polarized sources can be observed in the background material then the magnetic field geometry and strength in the foreground material can be explored, which is again due to the fact that the emissions from different regions in the sky having different backgrounds can be separated and contribution to *RM* by different regions can be inferred. This will become more clear when I will go through the details of *RM* synthesis in section 6.4.4. For now I will concentrate towards the aims of the *RM* survey, which is planned to be conducted by the SKA project. SKA will have enhanced observational capabilities with two orders of magnitude greater sensitivity than the existing radio telescopes. It will cover large instantaneous field-of-view. It will have an effective collecting area of one million square metres, frequency range from 100 MHz to 25 GHz, sensitivity gain of 100 relative to the current radio interferometers. It will also have unique polarimetric capabilities.

SKA plans to create *RM* grid consisting of nearly placed *RM* measurements in any direction in the sky. To accomplish this goal it is required to increase the density of polarized background radio sources dramatically so that the foreground region can be studied. Considering all the technical challenges, it will take one year of observation time with 1 h integration time per pointing covering 10,000 deg^2 of the sky. This experiment entails the Stokes parameters be available for broad frequency band. Since the density of background polarized sources is expected to be high, this puts an upper limit to the angular resolution to get a clear picture of the polarized sky. Accounting for this angular resolution and an optimum field-of-view there is a need of a certain number of channels with a certain channel width to get rid of bandwidth smearing. There are other issues like bandwidth depolarization, which can be accounted for by observing with narrow channels. Also the dynamic range required for detecting faint polarized sources is taken into account. To get more information on the specifications please refer to Beck et al. (2004)

Now I will go through the details of specific projects that would be supported by the *RM* measurement experiment by the SKA.

-Projects supported by SKA *RM* measurements

1. Study of Milky Way:

Studies of the Milky Way provide information on the structure and evolution of Galactic magnetic fields. However, due to sparse sampling of *RMs* and due to limited sensitivity this opportunity is not yet fully explored. The proposed SKA Galactic and extra-Galactic *RM* survey will overcome all these limitations and it will be possible to obtain the full geometry of the Galactic magnetic field.

Turbulence in the ISM is another topic of interest. Turbulence originates as a results of astrophysical processes and enters the ISM. The size of the turbulence can be characterized in terms of its extent in space. This size represents the size of its spatial power spectrum. It enters the ISM with a certain spatial extent and this extent decreases monotonically with time before being dissipated as heat. The size varies from scales greater than or equal to 1 kpc to a fraction of an AU before it vanishes completely. It is possible to obtain continuous power spectra of turbulence using dense sampling of RM. Fluctuations in the magnetic fields are coupled to those in electron density for producing the turbulence but the relation is not yet known. It will also be possible to obtain the spatial power spectrum of the turbulence in two dimensions, which will be a section of the three dimensional turbulence, as a function of Galactic latitude and longitude. However, since different sources lie at different distances along the line of sight, information on variation in the third dimension may also be obtained. At high latitudes the optical extinction is distinctively less so information on mean electron density can be disentangled from that of mean magnetic field if information on Hα emission from diffuse ionized gas coupled with dispersion measures of pulsars present in globular cluster is used with *RM* measurements of background sources.

2. Study of Galactic supernova remnants:

The magnetic fields play a major role in studying shocks in SNRs and there are theories that strong magnetic fields result in shocks in SNRs. There are many phenomena occurring in SNRs like heating, turbulence and acceleration of particles, which can be uncovered by studying magnetic fields. It is difficult to know the strength of magnetic fields from *RM* measurements only though the direction can be obtained from the polarization direction. The young adiabatic SNRs have magnetic field strengths about or less than 1 mG; since they have small compression ratio, the reason for this field strength is unknown. Generally the contribution from the field strength and from the number density of cosmic ray electrons are not the same and hence the magnetic field strength can only be inferred if we have some additional information on density of cosmic rays from other sources of information like X-ray emission. Also magnetic field strength may be obtained when there is Zeeman-splitting of OH masers, which is due to interaction with the shocks.

Further SNR shocks inject significant amount of turbulent energy arising from magneto-hydrodynamic turbulence to its environment and these can be tested and studied by *RM* measurements of background sources in the vicinity of SNRs. It has been difficult to disentangle the effects of SNRs from their complex surroundings until now due to sparse sampling of *RMs*. SKA *RM* measurements would yield dense *RM* grid thus alleviating this problem.

3. Study of Galactic HII regions:

A study of magnetic fields and motion and arrangement of gas molecules in HII regions will provide clue to how magnetic fields are affecting the gas flows, its amplification and compression. This is because the densities of HII regions are distributed over a wide range facilitating detailed analysis from variable information or data. It will be possible to determine magnetic fields in diffuse regions by *RM* measurements of background sources, which otherwise is very difficult. This is because the effects of these diffuse regions on *RM* of diffuse polarized emission from the Galactic background can easily be identified whereas other sources to detect magnetic fields like the Zeeman splitting can only yield results for excessively compact regions. It is possible to obtain information on electron density from Hα or continuum emission, which when combined with *RM* measurements can yield the magnetic field strength. SKA will be having wide field-of-view thus enabling increase in the number of background sources dramatically. Also compression of gas and magnetic fields by ionization fronts and contribution of magnetic field towards causing turbulence in the interiors of HII regions can be studied.

4. Study of nearby galaxies:

It is difficult to measure magnetic field strengths in the regions away from the sites of active star formation in the outer parts of galaxies without *RM* measurements of polarized background sources. With the current available background sources no galaxies beyond M 31 can be mapped. With the SKA, the polarized background sources will increase dramatically yielding detailed maps of magnetic structure within the fields of M 31, the LMC or the SMC. Smoothing of *RM* grid will result in high sensitivity to *RM* measurements and even the weak fields will be detected by the *RM* measurements of SKA.

5. Study of origin of magnetic fields

RM measurements will provide clue to dynamo or primordial field origin. Large scale *RM* patterns in many galaxies indicate an organized direction for the regular magnetic field and hence it cannot be caused by compression or expansion in gas flows. The two models for generation of magnetic fields- the dynamo model and the primordial field model predict different azimuthal and vertical symmetries for the magnetic field. Each such pattern represents a mode and thus each mode has a unique structure and orientation of magnetic field lines. In dynamo modes coherent magnetic fields are preserved, which are the superposition of modes while the primordial fields are difficult to be preserved due to diffusion and reconnection caused by differential rotation as time passes in a galaxy's lifetime. The conditions for excitation of different modes are different. Observations of nearby galaxies made so far reveal a mixture of modes, which cannot be determined reliably due to limits on angular resolution and signal to noise ratios but SKA will overcome these limitations. The modes generate a Fourier spectrum of azimuthal *RM* patterns, which can be reliably determined by the SKA since it has the required sensitivity and spatial resolution. Also the weaker polarization emissions will be mapped by the SKA *RM* measurements, which will enable determination of field patterns in diffuse regions. Primordial and dynamo models predict different *RM* patterns and hence *RM* measurements can yield information on the mechanism of field origin.

6.4.2 Studies of Sgr A*

In this section we review “ The linear polarization of Sagittarius A* I. VLA spectro-polarimetry at 4.8 and 8.4 GHz”, “The linear polarization of Sagittarius A* II. VLA and BIMA polarimetry at 22, 43 and 86 GHz”, and “Interferometric detection of linear polarization from Sagittarius A* at 230 GHz” studied by Bower et al. (1999a), Bower et al. (1999b) and Bower et al. (2003) respectively.

Synchrotron emission from active Galactic nuclei (AGN) often manifests high polarization. So observations have been carried out to detect linear polarization from the nearest AGN, Sgr A*, which would strongly support synchrotron mechanism as its radiation process. Sgr A* is recognized as a massive black hole candidate. Stellar proper motion studies indicate the presence of a mass of $2.6\times10^{6}M_{\odot}$ within 0.01 pc of Sgr A*. Also a lack of detected motion of Sgr A* suggests that at least $10^{3}M_{\odot}$ is associated with it. High temperature of $10^{9}K$ and compactness of the radio source revealed by VLBI observations at millimeter wavelengths support emission of cyclo-synchrotron radiation. Linear polarization is expected to be the outcome of this cyclo-synchrotron emission that produces the radio-millimeter wavelength spectrum.

Information that we have are as follows: a fractional polarization of 70 % is manifested by a homogeneous and optically thin synchrotron source under the influence of a uniform magnetic field. The fractional polarizations measured in AGN are usually a few percent at wavelength shorter than 6 cm though sometimes regions with increased polarization are also observed in VLBI polarization images. The polarization fractions are observed to increase with frequency in the cores which may be due to large *RMs* in the cores, enhanced visibility of the shocked regions and decreased opacity of the synchrotron. Further, observations of AGN have revealed correlation between the evolution of linear polarization and total intensity asserting the presence of shocks in the relativistic jets of AGN. Also comparison of the variations in the polarized and total intensity may provide clue to other ongoing processes. Hence this information should be used along with the data from Sgr A* to unveil its structure and radiation mechanism.

We should also note the effects of depolarization in case of Sgr A* since it is surrounded by thermal plasma and the region has high magnetic field strength and high density of electrons which will cause Faraday rotation of a polarized signal from it. Faraday rotation may cause depolarization of the emitted radiation as it travels through different depths in the interstellar matter. Also large *RM*s may cause depolarization of a polarized signal from Sgr A* especially when the polarization angle wraps through more than one turn, $n\pi$ ambiguities make it difficult to detect *RMs* through linear least squares fit. Further, bandwidth depolarization will occur if the polarization angle rotates by more than one radian or if *RM* exceeds

$$RM_{max}=\nu/(2\lambda^{2}\Delta\nu) \tag{6.3}$$

In eqn. (6.3) if $\Delta\nu$ is divided into $\delta\nu$ channels then a search for *RMs* exceeding RM_{max} can be made. The minimum *RM* detectable in spectro-polarimetric measurements is around the same as the maximum *RM* for the continuum measurements for the same bandwidth.

By taking into account all the information stated above, I now go through some relevant observations carried out to detect linear polarization from Sgr A* and the inferences drawn from them.

-Observations to detect linear polarization from Sgr A*

Now I will briefly go through each observation carried out to detect linear polarization from Sgr A*. In these observations the complex visibility data, P = Q + iU was Fourier transformed and mapped with respect to λ^2, which enables detection of large *RMs* without sensitivity loss. The rotation of the polarization vector is equivalent to a rotation in the two-dimensional Stokes Q and U space. The polarization angle χ is given by

$$\chi = ½ \tan^{-1}(U/Q) \tag{6.4}$$

Since *RM* is the slope of χ as a function of λ^2, a measure of P in the λ^2 space would yield a measure of *RM*. The polarization angle for zero Faraday rotation (*RM* = 0), which is also the emitted position angle (see eqn. (6.2)) is obtained by extrapolating χ with respect to λ to yield its value (χ_0) at $\lambda = 0$. Polarization fractions were obtained by imaging Sgr A* with and without *RM* corrections. It is the ratio of polarized intensity to the total intensity.

VLA (Very Large Array) polarimetric observations at 4.8 and 8.4 GHz:

Both continuum and spectro-polarimetric observations were made at 4.8 GHz for a bandwidth of 50 MHz. In the continuum observation, a polarization fraction of 0.1 % was detected for a maximum detectable *RM* of around 10^4 rad/m^2. The spectro-polarimetric observation was made in 8 consecutively spaced frequency bands each of which was 6.25 MHz wide and was further divided into 32 frequency channels. A polarization fraction of 0.2 % was detected. The range of *RM* covered was from 10^4 rad/m^2 to 3.5×10^6 rad/m^2.

The spectro-polarimetric observation at 8.4 GHz was carried out in 7 frequency bands each of which was 6.25 MHz wide and was divided into 32 frequency channels. A fractional polarization of 0.1 % was detected. The range of RM was 2400 ± 37000 rad / m^2.

Amplitude, phase and polarization calibrations were performed using calibrator sources and by following standard practices for all these observations. The measured values were upper limits as indicated by off-source peaks and residual instrumental polarization. For detailed information on calibration refer to Bower et al. (1999)

VLA continuum observations at 22 GHz and 43 GHz:

This showed a polarization fraction of 0.2 % for 22 GHz and 0.3 % for 43 GHz. The bandwidth was 50 MHz. The maximum *RM* detectable were 1.3×10^6 rad/m^2 and 8.4×10^6 rad/m^2 respectively.

Amplitude, phase and polarization calibrations were performed using calibrator sources by following standard practices. There were polarization errors due to D-term errors, smaller number of antennas and poorer performance at these frequencies. The errors (off-source peaks) were comparable to the measured fractional polarizations. Hence the measured fractional polarizations were upper limits. For detailed information on calibration refer to Bower et al. (1999b).

BIMA (Berkeley-Illinois-Maryland Association) continuum observations at 86 GHz and 90 GHz:

A fractional polarization of around 1 % was inferred from these observations. The bandwidth was 800

MHz. The maximum detectable *RM* was 4.8 × 10^6 rad/m^2.

Amplitude, phase and polarization calibrations were performed using calibrator sources by following standard practices. The polarization errors were estimated from the off-source peaks in the polarization maps. The measured polarization fraction was an upper limit. For detailed information on calibration refer to Bower et al. (1999).

BIMA continuum observation at 230 GHz

A linear polarization of 7.2 % ± 0.6 % was observed at a position angle 139º ± 4º. The bandwidth was 800 MHz. The detected *RM* was – 4.5 × 10^5 rad/m^2 ± 1.6 × 10^5 rad/m^2.

The resolution was high enough to isolate the emission from Sgr A* from those of other sources. Phase and leakage calibrations were applied following usual methods. Antenna gains were observed to be stable so no further amplitude calibration was carried out. The sources of errors were atmospheric coherence and antenna pointing errors and both of these contributed equally to polarized and unpolarized emissions. So polarization fractions were more reliable measurements. Magnitudes of these errors were estimated from observations of calibrator sources with different arrays. For detailed information on calibration refer to Bower et al. (2003).

Note: All the above mentioned observations switched to circular polarizations in the receivers as required for reasons already discussed in 1.1.

-Inferences from the results of observations

For dust emission whose typical intensity values and distribution are known to account for the observed polarization intensity and polarization fraction at 230 GHz, the dust needs to be significantly clumped and highly polarized at the location of Sgr A*, which is extremely unlikely.

Absence of linear polarization at and below 112 GHz (found in another observation) may occur due to depolarization as a result of internal field disorder or foreground beam depolarization.

The *RM* variations observed in Galactic centre (GC) region are much less than that required to depolarize Sgr A*. Also the conditions on magnetic field strength and density of electrons necessary to depolarize Sgr A* in the GC scattering regions taking all factors like propagation path lengths and equilibrium conditions between the magnetic field and thermal components into account are extreme. So depolarization is unlikely to occur in the scattering region.

Absence of linear polarization at 112 GHz for 800 MHz bandwidth excludes the possibility of bandwidth depolarization at 112 GHz since it will require an *RM* greater than 1×10^7 rad/m^2, which is much greater than the measured *RM*, which is around – 4.3 × 10^5 rad/m^2 ± 1.6 × 10^5 rad/m^2 at 230 GHz. The measured *RM* can be just sufficient to cause angular depolarization by changing the position angle by more than 180º at 112 GHz. A fully turbulent accretion region whose scales are comparable to source size can depolarize the source at 112 GHz.

Another explanation could be the presence of two sources one of which is polarized and the other unpolarized and the polarized source dominates the spectrum above 230 GHz. All models include at least two components.

There are several models like Bondi-Hoyle accretion model, advection-dominated accretion flows (ADAFs) models and convection-dominated accretion flows (CDAFs) models, which are trying to account for the measured *RM* and the observed spectrum by different accretion rates. No accretion rate determined by these models agreed with the measured *RM* and the observed spectrum simultaneously. That is the measured *RM* produces accretion rates that are too low to produce the observed spectrum. If there are magnetic field reversals and variations from equipartition between particle and field energy then a low *RM* similar to that measured may account for the observed spectrum since the same models may produce lower *RMs* (similar to that measured) for the rms magnetic field than they will do for a uniform magnetic field. But as calculated the number of magnetic field reversals required to meet the condition is extreme. So ADAFs and CDAFs are implausible.

There are jet models accounting for increase in polarization fraction with frequency where the emission originates from shock accelerated particles near the base of the jet and the order of magnetic field increases towards the base of the jet due to the shock. The base of the jet being optically thick is transparent to only high frequencies while the regions away from the base being optically thin are transparent to all frequencies implying more emission in the high frequency regime. The position angle of $181^\circ \pm 2^\circ$ of the electric field vector implies a north-south direction, which is perpendicular to the magnetic field (as in case of synchrotron; see section 6.2) and thus the direction of alignment of the jet axis is north-south being perpendicular to the compressed magnetic fields as per these models. In most models the jet axis is aligned with the electric field vector. However, an inclination angle for the jet axis close to the line of sight is possible since it is not well constrained in case of Sgr A* and since highly inclined sources show strongest polarization. Also then the radio/millimeter emission will be relativistically beamed resulting in the absence of a visible jet.

-Other explorations

Along with modeling the radiation process there are other explorations to be carried out like precession of the accretion disk causing changes in polarization angle where the precision is resulting from spin of associated black hole, polarization angle differences at different frequencies and measurements of polarization fraction as a function of frequency, which will contribute towards constraining or relaxing the models. *RM* variations with time with time-scales from hours to years will probe changes in the accretion or outflow rates. Further, VLBI imaging of polarized emission will also be able to probe general relativity effects near the black hole.

Further, strong and variable circular polarization has been detected between 1.4 GHz and 43 GHz with no accompanying linear polarization. There are models accounting for this observation through conversion of linear to circular polarization and the linear polarization being bandwidth-depolarized with low energy electrons and through models having plasma modes near a black hole. This topic needs further investigations.

6.4.3 Studies of circular polarization in AGN

In this section we review “Circular polarization in AGN” studied by Macquart (2001).

Circular polarization observed in AGN is typically less than 1 %. In order to detect this small circular polarization, which is mainly present in the compact regions, high resolution polarimetry is required otherwise there would be beam depolarization. VLBA (Very Long Baseline Array) has the required high angular resolution. It has also been able to provide information on the location of circular polarization with respect to the core and jets of an AGN. With ATCA (Australia Telescope

Compact Array) it is possible to do high precision, short time scale monitoring of intra-day variable (IDV) radio sources thus using source variability information to obtain the angular resolution information. It can measure better than 0.01 % degree of circular polarization. Coupled with an understanding of effects of interstellar scintillation, the IDV monitoring by ATCA has enabled milli-arcsecond resolution polarimetry.

-Origin of circular polarization

Circular polarization may arise from synchrotron mechanism, cyclotron emission, coherent emission by several localized patches, conversion to circular polarization from linear polarization in magnetized relativistic plasma and by scintillation effects. However, all these processes require constraints either for the generation or for the detection of circular polarization. The constraints can be to decrease the angle of magnetic field to the line of sight required to increase the degree of circular polarization but that in turn decreases power emitted in case of synchrotron, on the orientation of the plane of motion of particles and consideration of only fundamental frequency in case of detection of circular polarization in cyclotron and on the geometry of infalling rays with respect to magnetic field coupled with the requirement of Faraday rotation or that the incident radiation originates in a region having different direction of magnetic field than that in the region where the conversion takes place in case of generation of circular polarization in relativistic magnetized plasma. Lastly there are scintillation models to account for generation and variability of circular polarization. However, a sign change in the circular polarization on a time scale in accordance with the scintillation pattern as expected is not observed. Relaxing some of the assumed constraints of the model would affect both the magnitude of the effect and the time scales on which the changes in sign are expected to occur. This theory may account for the variable circular polarization seen at cm wavelengths in some AGN and in Sgr A*. There are many more specific constraints along with those mentioned that highly complicates these issues and it seems these mechanisms would work only under very special cases, which may happen but the probability of their existence will decrease along with increase in constraints. None of these mechanisms seem to be an obvious one. Now I would proceed to go through the inferences from the results obtained from observations carried out to detect circular polarization.

-Inferences from the results of observations

Detection and variability observations of circular polarization by VLBA and ATCA has provided valuable information on the origin of circular polarization. Observations reveal that circular polarization in AGN is variable. However, the variability of circular polarization measured by its timescale and magnitude is not same as that of linear polarization or of total intensity. The shorter time scales of circular polarization variability as compared to that of total intensity indicate that the circular polarization originates from a more compact region compared to the total unpolarized bulk emission. However, if this region is too compact then the circular polarization may get beam depolarized even on VLBI scales and hence this topic needs further investigations. Further some sources exhibit high circular polarization, which is not correlated to the extent of linear polarization, which may be due to the fact that linear polarization gets Faraday depolarized. High circular polarization is tentatively associated with IDV AGN as many of these show substantial and variable circular polarization. Observations show that the degree of circular polarization is a few percent. Also some sources show same handedness of circular polarization for decades and some do not but the reality of these changes is not yet fully explored.

One effect that may account for the variation observed in circular polarization is scintillation. However, since it affects circular polarization, linear polarization and total intensity equally, the source structure must account for the differences observed. A source model consisting of a polarized and an unpolarized component may account for the differences observed if the angular separation between the sources is comparable to the angular scale of scintillations since then the fluctuations due to each component are uncorrelated. Further, since variability in Stokes V arises from one component and that in Stokes I arises mostly from the other component, the variations in V and I are uncorrelated or marginally correlated. The correlation pattern changes due to change in the direction of scintillation velocity with respect to the line joining the two components during the course of a year. This is due to Earth's change in velocity arising due to its orbital motion, which changes the apparent scintillation velocity significantly. In reality the source structures are more complicated than two component model. It will be useful to know the power spectrum of source brightness distribution in each of the Stokes parameters to uncover the structural details. Since the power spectrum of the fluctuations observed is the product of scintillation power spectrum of a point source that is a known parameter and the power spectrum of the source angular brightness distribution, the source structure (angular brightness distribution) can be obtained.

Since VLBI and scintillation measurements indicate that circular polarization originates from very compact regions, high resolution polarimetric imaging techniques are required to detect this polarized emission. Scintillation imaging is the best technique at cm wavelengths to meet this objective. Scintillation observations can provide information on the extent of circular polarization and its location with respect to the polarized and unpolarized emission. The spectral slope of circular polarization can limit the origin of circular polarization since different frequencies can trace different regions. Such detection will require high precision polarimetric observations over a broad frequency range. However, the complex inhomogeneity and source structure limit this scope.

6.4.4 Studies of the Perseus cluster

In this section we review “Faraday rotation measure synthesis” and “Diffuse polarized emission associated with the Perseus cluster” studied by Brentjens et al. (2005) and de Bruyn et al. (2005) respectively.

Polarization studies of clusters can contribute significantly to deriving the magnetic field strength (if number density of thermal electrons can be derived by some other means) and structure along with facilitating explorations of particle energy distribution, shocks, jets and various emitting structures and modeling polarization generating mechanisms.

Polarization studies of clusters can provide information on structure formation in the universe. Current simulations for structure formation indicate that small masses collapsed and the larger masses accreted smaller ones growing bigger in size. The outcome of these kinds of processes is huge amount of gas flows, which produce shocks at their intersections. These shocks can be identified by studying different structures in a cluster for example the cluster relic sources, which are bubbles of magnetized plasma are dormant due to their very low electron densities owing to adiabatic expansion of the bubble and are visible only at very low frequencies. The electrons in the bubble may get energized by a structure formation shock that compresses the bubble adiabatically. The compression will reconfigure the magnetic field and may result in polarized emission. Also highly polarized radio sources in the outskirts of galaxies may provide information on shocks at the common boundary between clusters and super-cluster filaments accreting into them. Polarization studies of the Perseus cluster can provide information on thermal gas, relativistic gas and magnetic fields. Low frequency observations can yield information on low density regions if they are under the influence of a magnetic field since low gas densities and low temperatures result in reduced emission in the high frequency regime.

However, low frequency observations are limited by off-axis instrumental polarization, ionospheric Faraday rotation, internal depolarization, bandwidth depolarization and beam depolarization. Hence these factors need to be accounted for to make correct estimates.

In studying Perseus cluster Brentjens and de Bruyn used a very novel technique called rotation measure synthesis (*RM* synthesis), which is also derived by them. They carried out this technique in the image plane (to go to the image plane from the visibility data please refer to Thompson et al. (2001) or any standard textbook on radio-synthesis imaging). The advantage of this technique is different structures can be identified at different Faraday depths (we should note that each Faraday depth is responsible for a certain *RM* and hence it seems okay to even replace *RM* with Faraday depth. If an *RM* is not caused by a certain assumed Faraday depth then the emissions corresponding to that *RM* will not add coherently in the assumed Faraday depth otherwise they will) for a large field of view and also the morphology of the structures can provide information on their origin. Now I will go through this technique for clear understanding of the observations that follow.

In *RM* synthesis a range of values of Faraday depths are assumed. Each value of Faraday depth constitute an *RM* frame consisting of many pixels in two dimensions of the sky coordinates. An *RM* cube is constructed by placing the *RM* frames in parallel with the third dimension being the Faraday depth. A Faraday depth of ϕ causes a Faraday rotation of $\phi\lambda^2$ for a wavelength λ . In the k^{th} *RM* frame the i^{th} frequency channel's polarization vector in each pixel (one pixel here represents one measured spectrum in the two dimension of sky coordinates. This is obtained by collapsing the pixels corresponding to a certain location of sky coordinates in different channel images. One pixel has one data point in each channel image. Hence after collapsing each pixel will have *N* data points where *N* is the number of channels) is derotated by an amount of $\phi_k\lambda_i^2$, where ϕ_k is the Faraday depth of the k^{th} frame and λ_i is the wavelength of the i^{th} frequency channel, to determine the polarization angles at a Faraday depth ϕ_k . Note that this derotation corresponds to placing all polarization vectors in their positions with respect to $\lambda=0$ polarization vector accounting for a Faraday depth ϕ_k . If the derotation is with respect to some other polarization vector at $\lambda=\lambda_0$ then the corresponding amount of derotation will be $\phi_k(\lambda_i^2-\lambda_0^2)$, which is the general and practiced form (see the text following eqn. (6.2)). The concept of λ_0^2 will become more clear in the text that follows. Thus the emissions originating at a particular Faraday depth will add coherently in the associated *RM* frame and all other emissions will add only partly coherently reducing the sensitivity to emissions not originating at the assumed Faraday depth. After derotation and addition in each pixel, each *RM* frame shows $\tilde{F}(\phi_k)$, which is the reconstructed Faraday dispersion function or reconstructed polarized surface brightness per unit Faraday depth, for the assumed Faraday depth (ϕ_k). In order to get the general formulations of *RM* synthesis I will first write the quantities in terms of continuous variables ϕ and λ and then I will show the corresponding discrete functions for the practical cases.

$\tilde{F}(\phi)$, where ~ represents observed quantities, is given by

$$\tilde{F}(\phi)=K\int_{-\infty}^{\infty}\tilde{P}(\lambda^2)e^{-2i\phi\lambda^2}d\lambda^2 \tag{6.5}$$

where the observed polarized surface brightness

$$\tilde{P}(\lambda^2)=P(\lambda^2)W(\lambda^2) \tag{6.6}$$

where $P(\lambda^2)$ is the actual polarized surface brightness and $W(\lambda^2)$ is the weighting function corresponding to the beam in the λ^2 domain. Thus the Faraday dispersion function in the ϕ domain is the Fourier transform of polarized surface brightness in the λ^2 domain. Then I can write by Fourier transforming eqn. (6.6)

$$\tilde{F}(\phi)=F(\phi)*R(\phi) \tag{6.7}$$

where $R(\phi)$ is the Fourier transform of the weighting function $W(\lambda^2)$ normalized to unity at $\phi=0$ or

$$R(\phi)=K\int_{-\infty}^{\infty} W(\lambda^2)e^{-2i\phi\lambda^2}d\lambda^2 \tag{6.8}$$

It is also known as the rotation measure transfer function (RMTF). *K* is the normalization factor given by

$$K=1/\int_{-\infty}^{\infty} W(\lambda^2)d\lambda^2 \tag{6.9}$$

Eqs. (6.5) and (6.8) correspond to the case when the response of RMTF is parallel to the polarization vector at $\lambda=0$ and all other polarization vectors are derotated to their positions relative to the polarization vector at $\lambda=0$ at a Faraday depth ϕ . However, if derotation is with respect to some other vector at $\lambda=\lambda_0$ then eqs. (6.5) and (6.8) are generalized respectively as

$$\tilde{F}(\phi)=K\int_{-\infty}^{\infty} \tilde{P}(\lambda^2)e^{-2i\phi(\lambda^2-\lambda_0^2)}d\lambda^2 \tag{6.10}$$

and

$$R(\phi)=K\int_{-\infty}^{\infty} W(\lambda^2)e^{-2i\phi(\lambda^2-\lambda_0^2)}d\lambda^2 \tag{6.11}$$

In eqs. (6.10) and (6.11) all vectors are derotated to their positions with respect to the position of polarization vector at $\lambda=\lambda_0$ accounting for a Faraday depth ϕ . No information is lost by derotating with respect to polarization vector at $\lambda=\lambda_0$ and not with respect to $\lambda=0$. The response of the entire main peak of the RMTF should be parallel to the polarization vector at $\lambda=\lambda_0$. Hence, the value of λ_0 should be such that the orthogonal response of the RMTF at $\phi=0$ is minimized and it is found by setting the derivative of imaginary part (orthogonal response) of the RMTF to 0 at $\phi=0$. This is reasonable since at $\phi=0$, there is no Faraday rotation. In this way λ_0^2 is obtained as

$$\lambda_0^2=\int_{-\infty}^{\infty} W(\lambda^2)\lambda^2 d\lambda^2 | \int_{-\infty}^{\infty} W(\lambda^2)d\lambda^2 \tag{6.12}$$

Or, λ_0^2 needs to be the weighted average of all λ^2 .

Until now all equations are in terms of continuous variables. Now I will return to the original case where the *RM* frames represent the Faraday depth samples of the Faraday dispersion function and are obtained by Fourier transforming the sampled polarized surface brightness function in the squared wavelength domain. Eqs. (6.5), (6.6), (6.7), (6.8), (6.9), (6.10), (6.11) and (6.12) can be rewritten respectively in discrete form as

$$\tilde{F}(\phi_k)=K\sum_{i=1}^{N}\tilde{P}(\lambda_i^2)e^{-2i\phi_k\lambda_i^2} \quad \text{(for rotation with respect to } \lambda=0 \text{ vector)} \tag{6.13}$$

$$\tilde{P}(\lambda_i^2)=P(\lambda_i^2)W(\lambda_i^2) \tag{6.14}$$

$$\tilde{F}(\phi_k)=(F*R)(\phi_k) \tag{6.15}$$

$$R(\phi_k)=K\sum_{i=1}^{N}W(\lambda_i^2)e^{-2i\phi_k\lambda_i^2} \quad \text{(for rotation with respect to } \lambda=0 \text{ vector)} \tag{6.16}$$

where $K=\sum_{i=1}^{N}W(\lambda_i^2)$ (6.17)

$$\tilde{F}(\phi_k)=K\sum_{i=1}^{N}\tilde{P}(\lambda_i^2)e^{-2i\phi_k(\lambda_i^2-\lambda_0^2)} \quad \text{(for rotation with respect to } \lambda=\lambda_0 \text{ vector)} \tag{6.18}$$

$$R(\phi_k)=K\sum_{i=1}^{N}W(\lambda_i^2)e^{-2i\phi_k(\lambda_i^2-\lambda_0^2)} \quad \text{(for rotation with respect to } \lambda=\lambda_0 \text{ vector)} \tag{6.19}$$

and

$$\lambda_0^2=\sum_{i-1}^{N}W(\lambda_i^2)\lambda_i^2|\sum_{i=1}^{N}W(\lambda_i^2) \tag{6.20}$$

After describing the method of *RM* synthesis, I will now go through the observations and the inferences drawn.

-Spectro-polarimetric observation of Perseus cluster with WSRT

RM frames for a range of Faraday depths from -300 rad/m² to +300 rad/m² were constructed out of which 126 images were used for the construction of the *RM* cube. Several features were observed in different *RM* frames. Observation was carried out for 80 MHz bandwidth centred at 350 MHz. The band was divided into 8 independently tunable bands of 10 MHz each. These bands were further divided into 64 channels.

To reduce side lobes and increase spectral resolution, Hamming tapering was used and to reject RFI, combinations of odd-even channels were used. There were some problems due to ionospheric fluctuations and calibration was done on a channel by channel basis taking ionospheric models into account. Total intensity was self calibrated. The focus was only in the inner 3° of the cluster, which was not affected by pointing errors. On-axis polarization calibration was done using a calibrator source to align the phases of the two orthogonal polarizations. Leakage corrections were done with an unpolarized calibrator source. The main sources of errors were off-axis instrumental polarization leakages, which cause spurious polarization signals in the location of strong sources and falls with increase in *RM* (ϕ) values especially when the polarization is frequency independent causing them to add incoherently at large *RMs*, multiplicative errors in the uv plane that results into convolution patterns in the image plane, polarized grating lobes from Cas A although it was insignificant and instrumental artifacts giving rise to structures like the detected whiskers whose intensity fell with increase in distance from 3C 84. All these sources of errors were easily detected to isolate them from signals of celestial origin. For detailed information on calibration refer to de Bruyn et al. (2005).

-Detected features and inferences from the observation

Celestial signals were clearly identified in the *RM* cube frames covering a wide range of Faraday depths between 0 rad/m² and 90 rad/m². At low Faraday depths from 0 rad/m² ≤ ϕ ≤ 15 rad/m² diffuse structures with slowly varying polarization angles of the order of several tens of arcmins were observed and at high Faraday depths from 30 rad/m² ≤ ϕ ≤ 90 rad/m² distinct large structures with sizes of the order of a degree and having granularity in the polarization angles of the order of few arcmins were observed. No significant emission was detected between 15 rad/m² to 30 rad/m².

The diffuse emission at low ϕ was inferred to be produced by the Galactic foreground as similar *RM* was observed in another observation of a source located at around same latitude (l) but opposite longitude (b). Thus the low ϕ emission was inferred to be originating from the Galactic foreground.

High ϕ emissions showed richer spatial structures. A weak front-like structure was observed at φ = 30 rad/m² extending from $\alpha \approx 3^h 16^m$, $\delta \approx 40^o 24'$ to $\alpha \approx 3^h 10^m$, $\delta \approx 41^o 36'$. A stronger linear feature with a slight change in the polarization angle was observed at 42 rad/m². A bright circular doughnut like structure of diameter around 7' was observed at $\alpha \approx 3^h 15^m 35^s$, $\delta \approx 41^o 42.3'$. A lenticular feature lying southwest of the doughnut was observed at φ = 52 rad/m². The position angle of the lens was similar to that of the linear feature. Bright extended emission was shown by φ = 60 rad/m², φ = 69 rad/m² showed mottled emission centered around the area between NGC 1275 (3C 84) and NGC 1265 (both of these sources lie within the Perseus cluster and a pictorial representation can be found in de Bruyn et al. (2005)). At φ = 78 rad/m² significant emission was observed- a horizontal bar at $\alpha \approx 3^h 20^m$, $\delta \approx 42^o 25'$ lying north east of NGC 1265. This emission faded towards the cluster. No structure could be detected beyond φ = 100 rad/m² at the observing resolution of 2' to 3'.

Total intensity counterparts for the features observed in the polarized intensity maps were not detected. They attributed this to the fact that the sensitivity in Stokes I (1.5 mJy/beam) was significantly poorer than that in P (0.1 mJy/beam). This manifested a high polarization percentage.

In order to determine the location of high ϕ emission from *RM* measurements snapshot observations of other fifteen background sources, all of which avoid the Perseus cluster along the line of sight except two, were taken. Except three of these sources (excluded for their complex brightness distribution) all others showed a smooth spatial *RM* gradient with the *RM* values increasing towards the east. This gradient was clearly due to the Galactic foreground. The *RM* was 0 rad/m² just west of the Perseus cluster where the front, lens and the doughnut were located. The background sources which showed this 0 rad/m² *RM* just avoid the Perseus cluster along the line of sight. The *RM* of the diffuse emission from the Galactic background is around 10 rad/m². So for the total integrated *RM* towards these background sources to end up to 0 rad/m², a screen of -10 rad/m² is needed somewhere in our galaxy along the same line of sight which might be possible. However across 2° diameter area of the high ϕ emission, an *RM* of 60 rad/m² was observed. The estimated contribution from the Galactic foreground towards this region of the Perseus cluster is around 10 rad/m² to 20 rad/m². Subtracting the contribution from the Galactic foreground this showed an *RM* of 40 rad/m² to 50 rad/m² in excess. This emission was observed to be terminated at 1.5° from the pointing centre which may be associated with beam attenuation though chances are less. So in order to compensate for this *RM* if caused by our Galaxy, a screen of *RM* = - 40 rad/m² was required towards these background sources but no such emitting screen was found in the *RM* cube. Other arguments were given to negate the possibilities of 40 rad/m² originating from cloud in the Perseus arm of our galaxy that just covered the Perseus cluster or from north and east of the cluster centre. In the first case the argument was based on the comparison of the observed Hα surface brightness with Hα surface brightness that would be caused by the number density of electrons corresponding to *RM* of 40 rad/m² for an assumed approximate magnetic field strength and path length. The second possibility was ruled out by considering the fact that the emission from the north and east of the cluster centre showed a considerable drop in surface brightness much before primary beam attenuation initiates. Thus they concluded that the high ϕ emission was associated with the Perseus cluster of galaxies.

They also considered the possibility of Thompson scattering where they excluded the core whose activity being new cannot cause the Thompson echo to reach us. They only considered the 3C 84 30" component and the halo for Thompson scattering. This emission can be there and can be detected if the polarized intensity is more than the sensitivity of the telescopes but for the high ϕ emission from the large scale structures like the lens to be caused by Thompson scattering, the luminosity 3 million to 6 million years ago should be a factor 100 to 500 higher and the density of electrons should be at least 5 $\times 10^{-4}$ cm^{-3}at a distance of 1 Mpc to 2 Mpc from the cluster. Both these conditions are implausible. Further, excess *RMs* of 20 rad/m² and 60 rad/m² were shown by NGC 1275 and IC 310 respectively. A value of excess 60 rad/m² indicated that IC 310 is located deep within the cluster. The excess 40 rad/m² was interpreted to be located at the periphery near NGC 1275.

The morphology of the observed linear structures resembled relics of shock fronts. In order to avoid depolarization, these structures must be located at the periphery in the near side of the cluster. Similar alignment of the front and the lenticular structure indicated their co-location. Further the structure of the doughnut was similar to the structure and topology of the magnetic field in a pre-shock bubble. A blob was also detected to the north of NGC 1265 whose curved shape resembled the curvature of the emission from NGC 1265 indicating the possibility of the blob being associated with a previous phase of activity in NGC 1265 and thus being its detached bubble.

Exploration of other clusters and detection of similar structures would provide clue to their origins. Observation over a wide range of frequencies would enable determination of internal plasma density and internal magnetic field structure. Further, emissions at low frequencies would be detected by SKA and LOFAR enabling detection of cosmological shock waves below 500 MHz.

After discussing various new findings and several associated future goals in section 6.4, I will now proceed to conclude this chapter where I will describe the benefits of using our digital circular polarizer in these kinds of observations.

6.5 Conclusions

Thus we see that these kinds of observations rely mostly on polarimetric techniques. It is vital to measure the polarization magnitudes and angles for each frequency channel in a broad-band with high precision as in case of *RM* measurements. In VLBI presently linear to circular polarization conversion is done using analogue techniques but the quadrature phase shift is not perfect for the frequencies away from the design points (see fig. 1.3), which means that the phase difference between the two linear polarizations is not 90º for the whole band. Hence, only at the centre frequency the phase difference between the two hands of circular polarization will correspond to the angle of linear polarization in the sky (a linear polarization with position angle ψ has two circular polarization components with phase difference 2ψ) whereas the phase difference at any other frequency away from the centre frequency will not correspond to the polarization angle in sky thus requiring further corrections to be applied. In our case we obtain pure circular polarization for the whole band and hence for each frequency channel, the phase difference between two hands of circular polarization will provide the angle of linear polarization in the sky. Further, this digital circular polarizer can be used in any frequency range having a total bandwidth of 500 MHz so the same unit can be used for different frequency observations unlike the analogue polarizers designed for a certain centre frequency and a frequency range. It will also be possible to detect circular polarization with equal ease and measurements of circular polarization with high precision for broad bandwidths as required in case of scintillation imaging (refer to section 6.4.3) would be possible. Thus this digital circular polarizer would contribute significantly towards accurate polarization measurements for broad-bandwidths. It should also be noted that even though I have implemented the method for a 500 MHz bandwidth with 1 MHz channel width, this method can be adapted to one's need.

CHAPTER 7

REMAINING TOPICS OF DISCUSSION

In this chapter I will discuss about the polarization ellipse in detail. First I will provide a brief overview of the antenna system used in the radio astronomy receivers. Then I will discuss the polarization ellipse as a response of the antenna having all ideal characteristics that is when the orientation of the component dipoles of a crossed dipole are purely orthogonal and also there is no leakage of voltage or power from one dipole to another. I will go into the details of the general equation of the ellipse whose major axis or minor axis is oriented at an angle with respect to the dipole elements. Then I will see if I can arrive at the linear polarization whose antenna response due to presence of cross-polarization is an ellipse. Next I will discuss the response of a crossed dipole whose component dipoles are not in exact quadrature or they are not purely orthogonal. Next I will discuss the ellipse in our experiment described in section 5.2. Next I will discuss the effects of D-terms in the antenna responses and then I will discuss the relationship between ellipticity and D-term used in eqn (5.2). Finally I conclude this chapter summarizing the important details

7.1 Brief overview of the antenna system in radio telescopes

The antennas in the radio telescopes are mounted with crossed dipoles to receive electric fields. Each dipole is connected to a transmission line at the receiving end and is flared out in the other end that is in space. A balun is used to connect a dipole with the transmission line. The whole antenna circuit (an equivalent circuit comprising of the antenna and other elements of the receiver or transmitter) comprises of a source voltage, a source impedance and a load impedance connected in series, which is called the Thevenin's equivalent circuit. The load impedance is the free space impedance plus the antenna impedance, which may consist of reactive elements also. The load impedance closes the circuit by including the free space intercepted by flared out dipoles. The antenna circuit is reciprocal that is it behaves identically in the transmit and receive mode. In the receive mode there is no source voltage but a voltage same as the source voltage as in the transmit mode is generated if the transmitted electric field resembles totally with the electric field being received. Now I will discuss the polarization ellipse that is the response of antennas in radio telescopes and is of special interest since it is always present in all radio telescopes and causes deviations from the real signal being received or transmitted. Hence it must be a known quantity to deal with the observables during calibration of the amplitudes and phases of the received signal.

7.2 Polarization ellipse: response of radio telescopes in ideal cases

Now I will discuss the polarization ellipse and while discussing I will remain in the receive mode of the antenna. Whenever a crossed dipole in an antenna is in the receive mode of a linear polarization oriented at an angle ψ with respect to the horizontal or X dipole, the dipoles generate a response orthogonal to the same linear polarization called the cross-polar response; this cross-polar response has a phase difference say $\Delta\phi$ with the linear (wanted) polarization. The phase difference between the wanted polarization and the cross polarization is possibly dependent on the orientation of the wanted polarization with respect to the dipoles and magnitude of the wanted polarization coupled with the receiver characteristics to produce the orthogonal or cross-polar component. This is because different orientations of the linear (wanted) polarization, with respect to the dipoles, having different magnitudes produce different phase differences between the wanted and the cross-polar component.

Thus in space I have two orthogonal polarizations with a phase difference $\Delta\phi$ and whose amplitudes are also unequal. Neither of the two polarizations as discussed already is aligned with either dipole as is required for the general case. So I would have received two components of the linear polarization if the cross-polar response were not there as

$$X'(t)=A\cos\omega_0 t\cos\psi \tag{7.1}$$

and

$$Y'(t)=A\cos\omega_0 t\sin\psi \tag{7.2}$$

where *A* is the amplitude of the linear polarization, *X'(t)* and *Y'(t)* are the components of the linear polarization for *X* and *Y* dipole respectively at time *t*; ω_0 is the angular frequency considered. However, since the cross-polar response is there an elliptical polarization is formed in space and the *X* and *Y* dipole do intercept at distinct two points in the ellipse. Instead of receiving the signals in eqs (7.1) and (7.2), I receive the following two components in the *X* and *Y* dipoles respectively:

$$X''(t)=B\cos\omega_0 t \tag{7.3}$$

and

$$Y''(t)=C\cos(\omega_0 t+\Delta\chi) \tag{7.4}$$

Where *B* and *C* are the amplitudes received by *X* and *Y* dipoles respectively and $\Delta\chi$ is the phase difference between the signals received by *X* and *Y* dipoles. So the components in eqs (7.3) and (7.4) are received by the *X* and *Y* dipoles at time *t* instead of the components of linear polarization (eqs (7.1) and (7.2) respectively). These are the components of the ellipse that are intercepted by the *X* and *Y* dipoles in space. The quantities *B* and *C* are not the same as $A\cos\psi$ and $A\sin\psi$ respectively. The major axis, minor axis and the orientation of the major or minor axis will be dependent on the parameters *B, C* and $\Delta\chi$. The dependence is given for $B \neq 0$, $C \neq 0$ and $\Delta\chi\neq 0$ as

$$a^2=[(B^2+C^2)\pm\sqrt{(B^2+C^2)^2-4B^2C^2\sin^2\Delta\chi}]/2 \tag{7.5}$$

where *a* is the semi major axis. The semi minor axis *b* can be found from the relationship

$$1/b^2=1/(B^2\sin^2\Delta\chi)+1/(C^2\sin^2\Delta\chi)-1/a^2 \tag{7.6}$$

and the angle of inclination of the major axis α with respect to *X* axis can be obtained from the following equation by using the values of *a* and *b* obtained from eqs (7.5) and (7.6). In the following equation $\zeta=\cos\alpha$.

$$\zeta=\pm[a\sqrt{1-b^2/(B^2\sin^2\Delta\chi)}]/\sqrt{a^2-b^2} \tag{7.7}$$

Note that all semi major axis, semi minor axis and the orientation of the ellipse are dependent on all the parameters viz. B, C and $\Delta\chi$. A change in the orientation of the ellipse changes the phase difference between the two received components of *X* and *Y* and also the magnitudes of *X* and *Y.* The derivation for obtaining these parameters that are *a, b* and α from the parameters B, C and $\Delta\chi$ is given in section A.3; the parameter *A* in section A.3 is replaced by *B* here, the parameter *B* in section A.3 is replaced by *C* here; the parameter $\Delta\phi$ in section A.3 is replaced by $\Delta\chi$ here.

At this point I would want to analyze the effects of rotation of the same ellipse with respect to the receiving dipoles in terms of the magnitudes of the received components and the phase difference between them. Combination of eqs (7.3) and (7.4) by eliminating any term containing $\omega_0 t$ I arrive at the following equation of ellipse:

(After replacing in eqn (A.23), *A* by *B* here, *B* by *C* here and $\Delta\phi$ by $\Delta\chi$ here, I arrive at the following equation)

$$Y''^2(t) = C^2 \sin^2\Delta\chi - (C^2/B^2) X''^2(t) + (2C/B) X''(t) Y''(t) \cos\Delta\chi \tag{7.8}$$

or, $$Y''^2(t) - (2C/B) X''(t) \cos\Delta\chi \, Y''(t) + (-C^2 \sin^2\Delta\chi + (C^2/B^2) X''^2(t)) = 0 \tag{7.9}$$

Eqn (7.9) is a quadratic equation of *Y''(t)* when *X''(t)* is given. *Y''(t)* is given by

$$Y''(t) = [(2C/B) X''(t) \cos\Delta\chi \pm \sqrt{((2C/B) X''(t) \cos\Delta\chi)^2 - 4((-C^2 \sin^2\Delta\chi + (C^2/B^2) X''^2(t)))}]/2 \tag{7.10}$$

I know *X''(t)* ranges from -*B* to +*B*.

In MATLAB I assume the following values of B, C and $\Delta\chi$ in table 7.1 to observe the values of *a, b* and α .

CASE	B	C	$\Delta\chi$
1	5	3	$\pi/12$
2	5	3	$\pi/4$
3	2	4	$\pi/4$

Table 7.1: Three cases showing three different values of B, C and $\Delta\chi$ of eqn (27).

I take *X''(t)* from - 6 to 6 (more than what *X''(t)* assumes) in steps of 0.1 to get corresponding *Y''(t)*. The plot in fig. 7.1 is obtained for the three ellipses for the three cases. The ellipse produced in the first

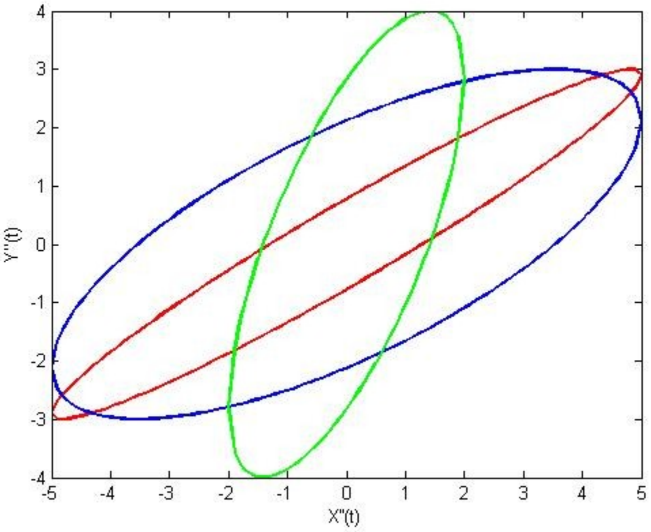


Fig.7. 1: Three ellipses corresponding to the three cases in table 1. The red colour showing case 1 with B = 3, C= 5 and $\Delta\chi = \pi/12$; blue colour showing case 2 with B = 3, C = 5 and $\Delta\chi = \pi/4$; green colour showing case 3 with B = 2, C = 4 and $\Delta\chi = \pi/4$. In the plot the *X* and *Y* axes are not equally spaced; there is a little difference in spacing. Still placing a protractor, to measure angles, at the lower tip of the major axis of each ellipse, the angle of inclination can be confirmed. It is almost same what we calculated in table 7.2. The little difference is due to unequally spaced axes.

case is red in colour in the plot; the ellipse produced in the second case is blue in colour in the plot; the ellipse produced in the third case is green in colour in the plot.

The magnitudes of the semi major axis, semi minor axis and the orientation of each ellipse can be determined using the eqs (7.5), (7.6) and (7.7) by putting in the values of B, C and $\Delta\chi$ for each case. And the corresponding values are given in table 7.2:

CASE	*a*	*b*	α
1	5,79	0,67	30.55°
2	5,5	1,92	26.48°
3	4,27	1,32	68.34°

Table 7.2: Three cases showing three different obtained values of a, b and α corresponding to the three cases in table 1 respectively obtained by putting values of B, C and Δχ in eqs (7.5), (7.6) and (7.7) for the corresponding case.

From the plot from case 1 and case 2 I see that if I keep the values of *B* and *C* the same and just change value of $\Delta\chi$ then both the ellipse and its orientation changes; in other words the ellipse in case 2 has different major axis, minor axis and orientation with respect to the coordinates when compared to the ellipse in case 1 even when case 1 and case 2 have same values of *B* and *C*; if I keep $\Delta\chi$ same as in case 2 and case 3 changing the values of *B* and *C* then also I arrive at a different ellipse at different orientation. Now the question is how to arrive at the same ellipse at different orientations? I have the three equations eqs (7.5), (7.6) and (7.7) for the two axes and the orientation of the ellipse. I need to determine those values of *B, C* and $\Delta\chi$ for which *a* and *b* remains constants and only α changes, which can be done. As I have already mentioned change in orientation of an ellipse changes the magnitudes of the received components in the *X* and *Y* dipoles and also the phase difference between the received components.

7.2.1 Retrieving linear polarization in the sky from elliptical response of antenna

So until now I have discussed the general equation of the polarization ellipse. Now suppose I want a desired *X* polarization and then I will orient *X* dipole parallel to the original linear polarization vector in the sky. As already stated due to the elliptical response of the antenna, the antenna will see the ellipse in response to the desired linear polarization vector; the ellipse will have maximum power along its major axis and I can consider this major axis as the wanted signal if the minor axis is negligibly small (discussed in section 7.4), which I am not considering now.

Orienting the *X* dipole along the major axis means having the original linear polarization make an angle with the *X* dipole; the major axis will have a different amplitude as compared to the original linear polarization when intercepted by the *X* dipole. The wave due to this major axis has the maximum amplitude among all other component waves of the ellipse. The *Y* dipole will receive the minor axis, which has a phase difference of 90° with respect to the major axis in the *X* dipole. I have a power loss from the original linear polarization component into its orthogonal counterpart. I will now proceed to the general case where the two dipoles intercept at two distinct points of the ellipse not the major and minor axes specifically as described in this paragraph, which will include this specific case also. In the general case the original linear polarization is inclined to the dipoles at any arbitrary angle and I will see if I can arrive at the magnitude and orientation of the original linear polarization, which was having no orthogonal counterpart as depicted by eqs (7.1) and (7.2), from the elliptical response. In this section since the two dipoles are exactly orthogonal, there is no voltage or power leakage from one dipole to another and also for now I am ignoring other external effects responsible for voltage or power leakage from one dipole to another. So now I have the *X* and *Y* dipoles intercepting two arbitrarily distinct points of the ellipse.

If the polarization is oriented at an angle ψ as depicted by eqs (7.1) and (7.2) and I have to detect the angle of inclination of the polarization then it really becomes difficult as the orientation of the ellipse produced is really not the orientation of the linear polarization unless the ellipse is highly elongated along the major axis (described later in section 7.4) where approximate the major axis as the wanted linear polarization. Let the components received by the *X* and *Y* dipoles be described by eqs (7.3) and (7.4) respectively then the intercepted *X* and *Y* components will have the phase difference ΔX . One may cross-correlate the signals in *X* and *Y* dipoles to get the lag at which the cross-correlation coefficient is maximum; this lag multiplied by the frequency under consideration will determine the phase difference between the *X* and *Y* components and then this lag needs to be compensated to keep the *X* and *Y* signals in phase. I am proceeding to see if it is possible to detect the correct magnitude of the linear polarization and the correct orientation of the linear polarization with respect to the *X* dipole from the information that I have from the elliptical polarization.

From eqs (7.1) and (7.2) I see that the linear polarization with magnitude *A* and making an angle ψ should have magnitudes $A\cos\psi\cos\omega_0 t$ at time *t* in the *X* dipole and $A\sin\psi\cos\omega_0 t$ at time *t* in the *Y* dipole if there is no power leakage from the linear polarization to its orthogonal counterpart. But I receive $B\cos\omega_0 t$ and $C\cos(\omega_0 t+\Delta X)$ at time *t* in the *X* and *Y* dipoles respectively. The total power in the linear polarization should be the same as the total power in the received elliptical polarization. Total power would mean power in the *X* receiving chain + power in the *Y* receiving chain. have already compensated the phase difference between the received *X* signal and the received *Y* signal by compensating the lag or by reducing the phase difference between the *X''(t)* and *Y''(t)* to zero.

Now I want to determine the total power in the linear polarization and in the received elliptical polarization. I find the power spectrum in the frequency domain, which involves multiplication of the absolute value of the spectrum with itself. Hence, first taking the Fourier transform of eqs (7.1) and (7.2) I arrive at the two following frequency domain signals respectively

$$X'(\omega)=A(\cos\psi)\pi(\delta(\omega-\omega_0)+\delta(\omega+\omega_0)) \tag{7.11}$$

and

$$Y'(\omega)=A(\sin\psi)\pi(\delta(\omega-\omega_0)+\delta(\omega+\omega_0)) \tag{7.12}$$

So the power in eqn (7.11) is

$$2A^2\cos^2\psi\pi^2 \tag{7.13}$$

and the power in eqn (7.12) is

$$2A^2\sin^2\psi\pi^2. \tag{7.14}$$

Hence the total power is the summation of powers in eqs (7.13) and (7.14), which is

$$= \quad 2A^2\pi^2 \tag{7.15}$$

Similarly by taking the Fourier transform of eqs (7.3) and (7.4) I obtain the following two frequency domain signals respectively.

$$X''(\omega)=B\pi(\delta(\omega-\omega_0)+\delta(\omega-\omega_0)) \tag{7.16}$$

and

$$Y''(\omega)=C\pi(e^{j\Delta X}\delta(\omega-\omega_0)+e^{-j\Delta X}\delta(\omega+\omega_0)) \tag{7.17}$$

Therefore the power from *X''(t)* and *Y''(t)* is obtained by summing the powers of eqs (7.16) and (7.17), which is

$$= \quad 2B^2\pi^2 + 2C^2\pi^2 \tag{7.18}$$

The power in eqn (7.15) and the power in eqn (7.18) must be equal and hence I obtain

$$A^2=B^2+C^2 \tag{7.19}$$

So I have got the information on the magnitude of linear polarization from the elliptical polarization. However, the orientation of the linear polarization remains unknown. I have now the linear polarization $A\cos\omega_0 t$ without any information on its orientation. I am able to obtain the parameters of the elliptical polarization that are its major axis, minor axis and angle of orientation of major axis from eqs (7.5), (7.6) and (7.7). There are many software packages available which can simulate a practical antenna by incorporating the same transfer characteristics in terms of the two orthogonal responses as the real antenna and thus producing the same radiation pattern in response to an electric field as would the real antenna; in that case I need not take measurements on the telescope to determine radiation patterns all the time. I can simulate the effects of rotation of the linear polarization $A\cos\omega_0 t$ with respect to the *X* dipole and see if I can arrive at the received elliptical polarization. This would mean to model the antenna characteristics in terms of its two orthogonal responses. However, if the two orthogonal responses of the antenna are totally indeterminable then I cannot obtain the orientation of the original linear polarization from the ellipse. If the responses are time varying then I may radiate $A\cos\omega_0 t$ at different orientations to one of the dipoles during the same experiment when the ellipse is being received to see if I arrive at the received ellipse; for this measurement the antenna needs to be switched to transmit mode. The relationship between the voltage amplitude and the electric field amplitude must be used to obtain the desired magnitude *A* of the transmitted linear polarization. The relation can be found in Kildal (2000) or other antenna fundamental textbooks.

7.3 Polarization ellipse: response from imperfectly oriented dipole elements

Until now I have been discussing the components of elliptical polarization received by a crossed dipole whose component dipoles are perfectly orthogonal and there is no leakage from one dipole to other. The situation becomes more difficult when the component dipoles are not exactly orthogonal to each other as then I do not receive the two orthogonal components of the elliptical polarization but two components, which are the components of two different coordinate systems. Here also I exclude the effects of leakage of one polarization state into another. However, since the two dipoles are not the components of a single coordinate system, the signal received by one dipole of the unwanted coordinate will have a component of signal in the other dipole of the wanted coordinate. This will be clear in the following description.

Let us consider an imperfect crossed dipole whose coordinates are *X* and Y_{new} where the orthogonal counterpart of *X* is *Y* and the orthogonal counterpart of Y_{new} is $X_{new.}$ So X_{new} and *Y* are not physically present as dipole elements. Now I will consider the case of receiving the elliptical polarization whose *X* and *Y* components are represented by eqs. (7.3) and (7.4). So I will always measure two signals in the *X* and Y_{new} dipoles. For now I will not consider leakage of signal from one dipole to another, which arises due to several random physical reasons. Now I will just consider the effects of imperfectly oriented dipoles without any arbitrarily added D-term. Let us rewrite the equations for the *X* and *Y* components of the ellipse for visual ease.

$$X''(t)=B\cos\omega_0 t \tag{7.3}$$

and

$$Y''(t)=C\cos(\omega_0 t+\Delta X) \tag{7.4}$$

Now in this case the signal *Y''(t)* (eqn (7.4)) is not received at all since I do not have a dipole oriented in the direction of *Y* .

If I know the angle of imperfection of the crossed dipole that is the angle between *Y* and Y_{new} or *X* and X_{new} , which is θ (say) then I can express the signal in Y_{new} dipole for the same frequency component as

$$Y_{new}''(t)=X''(t)\sin\theta+Y''(t)\cos\theta \tag{7.20}$$

and I am receiving this signal. So in order to get the orthogonal counterpart of the signal received in the *X* dipole, which is an imaginary (not physically present) *Y* dipole, I need to find *Y''(t)*.

$$Y''(t)=[Y_{new}''(t)-X''(t)\sin\theta]/\cos\theta \tag{7.21}$$

$Y_{new}''(t)$ is a measured quantity, *X''(t)* is also measured and θ is known so I should be able to get *Y''(t)* in principle, which is the required other component of the elliptical polarization.

Note that I could also take the signal component of Y_{new} dipole as genuine and find the orthogonal component of its signal $Y_{new}''(t)$ that is $X_{new}''(t)$. That would had given the same ellipse in terms of $X_{new}''(t)$ and $Y_{new}''(t)$ with the orientation of the ellipse with respect to X_{new} and Y_{new} coordinates.

7.4 Analysis of the ellipse in the experiment described in section 5.2

In the experiment described in section 5.2, we obtained an ellipse as a response produced in the waveguide to be received by the crossed dipole. Let us consider here that the dipole elements are orthogonal for simplicity. Then the equations (7.3) and (7.4) are received by the *X* and *Y* dipoles respectively. The ellipse produced inside the circular waveguide was highly elongated. I will show here that for such an elongated ellipse where the minor axis and thus semi minor axis $b \rightarrow 0$, the major axis mimics a linear polarization oriented at an angle of orientation of the major axis.

From eqn (7.6) I have $1/b^2 = 1/(B^2 \sin^2 \Delta\chi) + 1/(C^2 \sin^2 \Delta\chi) - 1/a^2$

or, $b^2 = (B^2 a^2 C^2 \sin^2 \Delta\chi)/(a^2 C^2 + B^2 a^2 - B^2 C^2 \sin^2 \Delta\chi)$ (7.22)

or, $b = (BaC \sin \Delta\chi)/\sqrt{a^2 C^2 + B^2 a^2 - B^2 C^2 \sin^2 \Delta\chi}$ (7.23)

In the limit $b \rightarrow 0$, the quantity $BaC \sin \Delta\chi \rightarrow 0$. Now the major axis a is not zero and hence B and C are also not zero that is $a \neq 0$, $B \neq 0$ and $C \neq 0$; that is the system is designed in such a manner that the major axis is there and the minor axis tends to zero and thus semi minor axis $b \rightarrow 0$. Then from eqn (7.23) I have $\sin \Delta\chi \rightarrow 0$, which also means that $\Delta\chi \approx n\pi$ where n is 0, 1, 2, 3,.... and for these values of $\Delta\chi$, $\cos \Delta\chi \approx 1$. So the signal received by the Y dipole in eqn (7.4) is modified as $Y''(t) \approx C \cos \omega_0 t$ and the orientation of the major axis or of the equivalent linear polarization is $\tan^{-1} C/B$. Thus we could work with the highly elongated ellipse in place of a linear polarization. The currents induced in the two dipoles by this ellipse was in phase.

7.5 Effects of D-term in the received voltage by the crossed dipole

The real world is not so simple and there are several physical processes occurring near the antenna or inside the antenna circuits, which give rise to coupling of voltages from one dipole to another. Note that I have the contribution to the leakage or to the D-terms due to the presence of nearby objects to the antenna which receives a fraction of a polarization state and then radiates back to the dipole receiving the orthogonal polarization state; this phenomenon of leakage is valid even when the dipoles are perfectly orthogonal to each other; leakage due to this phenomenon is totally random to be modeled at all.

Leakage from one dipole to another can also happen due to mutual induction when the dipoles are not exactly orthogonal and the inductance of the coil is not zero, which is generally the case as the inductance is not tuned out for all frequency components. If the dipoles were orthogonal then there will be no mutual coupling of magnetic flux. The induced voltages in a dipole due to mutual inductance corrupts the phases of the voltage being received by the dipole. The amount of coupling due to mutual inductance from one dipole to the next for both the dipoles are the same. Finally, the leakages happen back and forth that is from one dipole to the next and then back to the previous dipole and so on. The very general equations of the signals received by the X and Y dipoles that includes the effects of D-terms are given in all literature of radio astronomy covering the basics of D-terms as

$$\bar{X}(t) = X(t) + (D_X Y(t)) \tag{7.24}$$

$$\bar{Y}(t) = Y(t) + (D_Y X(t)) \tag{7.25}$$

Where $\bar{X}(t)$ and $\bar{Y}(t)$ are the received signals from X and Y dipoles respectively and *X(t)* and *Y(t)* are the signals without D-terms corresponding to X and Y dipoles respectively. So I only receive these two terms in the LHS of eqs (7.24) and (7.25) from the antenna. Since there are a redundant number of such equations formed from the too many values of $\bar{X}(t)$ and $\bar{Y}(t)$, I get many solutions for the D-terms. Similar equations in terms of LHC and RHC are there, which are already provided in chapter 2, namely eqs (2.32) and (2.33) respectively. Until now there is no model that can solve for the D-terms from the above equations. Note that D_{LHC} and D_{RHC} in eqn (2.32) and (2.33) have different values when compared to D_X and D_Y of the same case. I will call D_{LHC} and D_{RHC} as circular D-terms and D_X and D_Y as linear D-terms.

7.5.1 Discussion on relation between circular D-terms and ellipticity

We have shown in eqn (5.1) for the RHC and for the LHC that is for the two hands of circular polarizations, the ellipticity, ϵ , is defined as $\sqrt{minimum\ power / maximum\ power}$ where power (mean power) is measured for different orientations of the dipoles with respect to the received plane of electric field. Now I will provide the reasons for the validity of eqn (5.2) provided by Perley, which can also be written as

$$\epsilon = (1-|D|)/(1+|D|) \tag{7.26}$$

Where D is D_{LHC} for desired LHC and is D_{RHC} for a desired RHC. The derivation (follows Kildal (2000)) of eqn (7.26) follows as below:

$$\epsilon = E_{min} / E_{max} \tag{7.27}$$

where E_{max} is the maximum electric field magnitude in the polarization ellipse and E_{min} is the minimum electric field magnitude in the polarization ellipse. Therefore,

$$\epsilon = (|E_{desired}| - |E_{cross-polar}|)/(|E_{desired}| + |E_{cross-polar}|) \tag{7.28}$$

where $E_{desired}$ is the electric field of the desired circular polarization and $E_{cross-polar}$ is the other undesired component of circular polarization.

Dividing numerator and denominator of RHS of eqn (7.28) by $E_{desired}$ I get

$\epsilon = (1-|D|)/(1+|D|)$ where $|D| = |E_{cross-polar}| / |E_{desired}|$. Thus eqn (7.26) is proved. For a desired LHC (or RHC) the $E_{desired}$ is the electric field of LHC (or RHC) and $E_{cross-polar}$ is the undesired component of RHC (or LHC). Eqn. (7.26) is not valid for linear D-terms.

7.6 Conclusion

In this chapter I visited the ellipse produced as a response from the antennas in the radio telescopes. I discussed various properties of the ellipse and saw in 7.2.1 that with additional simulations or experiments it is possible to derive the original signal, which is a linearly polarized wave from the elliptical response. However, if the ellipse is highly elongated, that is its major axis >> minor axis and the minor axis tends towards zero, then we can work with linear polarization approximation of the elliptical polarization, that is the major axis is equivalent to a linearly polarized signal. I also discussed the relation between D-term and ellipticity. Thus the remaining topic after 6 chapters of this thesis, which was the elliptical response of the antennas in radio telescopes and in our experiment described in section 5.2, is complete in this chapter. Next I will provide future work and three appendices: appendix A, appendix B, appendix C and the references.

FUTURE WORK

Though this technique to convert to two hands of circular polarizations from two orthogonal linear polarizations was demonstrated using logic simulator in software, it has been implemented using Xilinx software generating firmware that can be loaded into FPGAs. The details on the design and implementation can be found on chapter 4 of this thesis. An implementation on the digital baseband converter (Tuccari, 2008) is in preparation, which will make this technique available for use at many radio observatories for VLBI, and with minor extension, for measuring Stokes parameters. This could enable the sensitive search for circular polarization in active galactic nuclei. As demonstrated in chapter 6, the polarizer can be used to explore all phenomena generating linear or circular polarizations. It is very essential to obtain the angle of linear polarization correctly to determine rotation measures; a polarization angle oriented at an angle ψ will produce two hands of circular polarization having equal magnitudes and phase difference 2 ψ (derivation provided in section A.4); thus from the measurements of phase difference between the two circular polarizations produced by our digital circular polarizer, we can arrive at the inclination of the linear polarization. Also circular polarization can be detected with equal ease as the linear polarization. Later measurements at the telescope should confirm that excellent polarization purity is achieved in real applications, as it was in the anechoic chamber. One can also confirm the stability of the transfer characteristics and decide on a re-calibration interval for operational use. One can also characterize the typical phase response of receivers and confirm that the choice of 1 MHz frequency spacing is well matched to the existing systems. The effect of RFI on the system can be explored, to give recommendations on tolerable RFI levels and required performance of mitigation strategies. The trend in next-generation receivers for radio astronomy is to move the samplers as close as possible to the front end, which will benefit this system of polarization conversion since the time variable path length changes due to analogue cables and filters and amplifiers will be much reduced, yielding even better polarization purity.

APPENDIX A: BASIC POLARIZATION DERIVATIONS

Polarization arises due to spin of photons. A photon can assume two states of spin angular momentum given by $\pm\sqrt{2}h/(2\pi)$ with the $+\sqrt{2}h/(2\pi)$ state corresponding to LHC and $-\sqrt{2}h/(2\pi)$ corresponding to RHC. A photon is generally in superposition of these two states and the sense of polarization will depend on the relative proportion of the two states being superposed. There are three cases of polarization arising due to the superposition of the two states. One is linear polarization, the other is circular polarization and the third is elliptical polarization. Here I derive the equations of a straight line, circle and ellipse from them respectively at all times which means the equations are valid at all times of wave propagation. Lastly I show the conversion from circular to linear polarization and vise versa. The derivations are as follows.

A.1 Equation of straight line from equation of linear polatization

In this section and in the following sections *t* is an instant of time and *ω* is angular frequency chosen. The *X* and *Y* components of linear polarization with amplitude *A* and orientation *θ* w.r.t *X* axis are

$$\overline{X}(t)=A\cos\omega t\cos\theta\,\hat{x} \tag{A.1}$$

$$\overline{Y}(t)=A\cos\omega t\sin\theta\,\hat{y} \tag{A.2}$$

Therefore, $Y(t)=(X(t)/\cos\theta)\times\sin\theta$ (A.3)

where *X (t)* and *Y(t)* are magnitudes of $\overline{X}(t)$ and $\overline{Y}(t)$ respectively.

or, $Y(t)/X(t)=\tan\theta$ (A.4)

or, $Y(t)=\tan\theta\times X(t)$ (A.5)

Eqn. (A.5) is the equation of a straight line with slope $\tan\theta$ or *Y* polarization is related to *X* polarization by this identity at ant time *t.*

A.2 Equation of circle from equation of circular polarization

Now I will derive the equation of a circle from the *X* and *Y* components of circular polarization.

The *X* and *Y* components of circular polarization are

$$\overline{X}(t)=A\cos\omega t\,\hat{x} \tag{A.6}$$

$$\overline{Y}(t)=A\cos(\omega t\pm\pi/2)\,\hat{y} \tag{A.7}$$

Where *A* is the amplitude of *X* and *Y* components. Equating the magnitudes of eqn (A.7) I have

$$Y(t)=A\cos(\omega t\pm\pi/2) \tag{A.8}$$

$$=\pm A\sin\omega t \tag{A.9}$$

From eqn (A.6) after equating the magnitudes $\sin\omega t = \sqrt{1 - X^2(t)/A^2}$

Therefore, $Y(t) = \pm A\sqrt{1 - X^2(t)/A^2}$ (A.10)

$$= \pm\sqrt{A^2 - X^2(t)} \tag{A.11}$$

Squaring both sides of eqn (A.11) I have

$$Y^2(t) = A^2 - X^2(t) \tag{A.12}$$

or, $X^2(t) + Y^2(t) = A^2$ (A.13)

Eqn (A.13) is the equation of a circle at any time *t.*

A.3 Equation of ellipse from equation of elliptical polarization

Finally, I will derive the equation of ellipse from the *X* and *Y* components of elliptical polarization. The derivation follows for phase difference $\Delta\phi \neq 0$ between received *X* and *Y* with amplitudes *A* and *B* respectively where A ≠ 0 and B ≠ 0.

$$\bar{X}(t) = A\cos\omega t\,\hat{x} \tag{A.14}$$

$$\bar{Y}(t) = B\cos(\omega t \pm \Delta\phi)\,\hat{y} \tag{A.15}$$

Equating the magnitudes of both sides in eqn (A.15)

$$Y(t) = B[\cos\omega t\cos\Delta\phi \mp \sin\omega t\sin\Delta\phi] \tag{A.16}$$

From eqn (A.14) after equating magnitudes I get $\sin\omega t = \sqrt{1 - X^2(t)/A^2}$ and $\cos\omega t = X(t)/A$ and hence using these in eqn(A.16) I get

$$Y(t) = B[(X(t)/A)\cos\Delta\phi \mp (1/A)\sqrt{A^2 - X^2(t)}\sin\Delta\phi] \tag{A.17}$$

or, $Y(t) = B(X(t)/A)\cos\Delta\phi \mp (B/A)\sqrt{A^2 - X^2(t)}\sin\Delta\phi$ (A.18)

or, $\pm(B/A)\sqrt{A^2 - X^2(t)}\sin\Delta\phi = B(X(t)/A)\cos\Delta\phi - Y(t)$ (A.19)

Squaring both sides of eqn (A.19) I get

$$(B^2/A^2)\sin^2\Delta\phi(A^2 - X^2(t)) = B^2(X^2(t)/A^2)\cos^2\Delta\phi + Y^2(t) - 2(B/A)X(t)Y(t)\cos\Delta\phi \tag{A.20}$$

or, $B^2\sin^2\Delta\phi - (B^2/A^2)X^2(t)\sin^2\Delta\phi = B^2(X^2(t)/A^2)\cos^2\Delta\phi + Y^2(t)$

$$-2(B/A)X(t)Y(t)\cos\Delta\phi \tag{A.21}$$

or, $B^2\sin^2\Delta\phi = B^2(X^2(t)/A^2)\cos^2\Delta\phi + (B^2/A^2)X^2(t)\sin^2\Delta\phi + Y^2(t) - 2(B/A)X(t)Y(t)\cos\Delta\phi$
(A.22)

or, $B^2\sin^2\Delta\phi=(B^2/A^2)X^2(t)+Y^2(t)-2(B/A)X(t)Y(t)\cos\Delta\phi$ (A.23)

or, $X^2(t)/(A^2\sin^2\Delta\phi)+Y^2(t)/(B^2\sin^2\Delta\phi)-2((X(t)Y(t))/(AB))(\cos\Delta\phi/\sin^2\Delta\phi)=1$ (A.24)

which is the equation of an ellipse..

The equation of an ellipse whose major and minor axes coincide with those coordinates which make an angle α with the Cartesian coordinate system *X* and *Y* (rotating *X/Y* by angle α counterclockwise) is given by

$$(X(t)\cos(\alpha)-Y(t)\sin(\alpha))^2/a^2+(X(t)\sin(\alpha)+Y(t)\cos(\alpha))^2/b^2=1 \quad \text{(A.25)}$$

where *a* is semi major axis and *b* is semi minor axis. This form eqn (A.25) is taken from Kalman (2008)

then

$$(\cos^2\alpha/a^2+\sin^2\alpha/b^2)X^2(t)-2\cos\alpha\sin\alpha(1/a^2-1/b^2)X(t)Y(t)$$

$$+(\sin^2\alpha/a^2+\cos^2\alpha/b^2)Y^2(t)=1 \quad \text{(A.26)}$$

Comparing eqn(A.26) with eqn(A.24) I have

$$\cos^2\alpha/a^2+\sin^2\alpha/b^2=1/(A^2\sin^2\Delta\phi) \quad \text{(A.27)}$$

$$\sin^2\alpha/a^2+\cos^2\alpha/b^2=1/(B^2\sin^2\Delta\phi) \quad \text{(A.28)}$$

and

$$\cos\alpha\sin\alpha(1/a^2-1/b^2)=\cos\Delta\phi/(AB\sin^2\Delta\phi) \quad \text{(A.29)}$$

Thus eqs (A.27), (A.28) and (A.29) have three unknowns *a, b* and α . So solving these three equations would yield the values of *a, b* and α in terms of A, B, $\Delta\phi$ and thus the semi major, semi minor and angle of inclination of ellipse can be determined in terms of A, B, $\Delta\phi$.

Solution with major steps of calculations:

Let $\cos\alpha=\zeta$ then $\sin\alpha=\sqrt{1-\zeta^2}$. Therefore eqs (A.27) and (A.28) becomes the following respectively.

$$\zeta^2/a^2+(1-\zeta^2)/b^2=1/(A^2\sin^2\Delta\phi) \quad \text{(A.30)}$$

$$(1-\zeta^2)/a^2+\zeta^2/b^2=1/(B^2\sin^2\Delta\phi) \quad \text{(A.31)}$$

Adding eqs (A.30) and (A.31) I get

$$1/a^2+1/b^2=1/(A^2\sin^2\Delta\phi)+1/(B^2\sin^2\Delta\phi) \quad \text{(A.32)}$$

Eqn (A.29) turns into

$$\sqrt{\zeta^2(1-\zeta^2)}(1/a^2-1/b^2)=\cos\Delta\phi/(AB\sin^2\Delta\phi) \tag{A.33}$$

Squaring eqn (A.33) I get

$$(\zeta^2-\zeta^4)(a^2-b^2)^2/(a^4b^4)=\cos^2\Delta\phi/(A^2B^2\sin^4\Delta\phi) \tag{A.34}$$

From eqn (A.30) I get

$$\zeta=\pm[a\sqrt{1-b^2/(A^2\sin^2\Delta\phi)}]/\sqrt{a^2-b^2} \tag{A.35}$$

In eqn (A.34) let us solve first for the term ζ^4 by putting value of ζ fron eqn (A.35) then

$$\zeta^4=(\zeta^2)^2=(a^2-(a^2b^2/(A^2\sin^2\Delta\phi)))^2/(a^2-b^2)^2 \tag{A.36}$$

$$\zeta^2=(a^2-(a^2b^2/(A^2\sin^2\Delta\phi)))/(a^2-b^2) \tag{A.37}$$

By solving eqn (A.34) by putting values of ζ^2 and ζ^4 from eqs (A.37) and (A.36) respectively I arrive at the following equation

$$1/a^2[A^2\sin^2\Delta\phi-(A^4\sin^4\Delta\phi)/b^2+(a^2A^2\sin^2\Delta\phi)/b^2-a^2]/A^2=\cos^2\Delta\phi/B^2 \tag{A.38}$$

Now let us solve for the terms $(A^4\sin^4\Delta\phi)/b^2$ and $(a^2A^2\sin^2\Delta\phi)/b^2$ of eqn (A.38) to eliminate b by writing these terms in terms of a, which can be done by using eqn (A.32)

So I get

$$(A^4\sin^4\Delta\phi)/b^2=A^4\sin^4\Delta\phi(1/(A^2\sin^2\Delta\phi)+1/(B^2\sin^2\Delta\phi)-(1/a^2)) \tag{A.39}$$

$$= A^2\sin^2\Delta\phi+(A^4/B^2)\sin^2\Delta\phi-(A^4\sin^4\Delta\phi/a^2) \tag{A.40}$$

and

$$(a^2A^2\sin^2\Delta\phi)/b^2=a^2A^2\sin^2\Delta\phi(1/(A^2\sin^2\Delta\phi)+1/(B^2\sin^2\Delta\phi)-(1/a^2)) \tag{A.41}$$

$$= a^2+a^2A^2/B^2-A^2\sin^2\Delta\phi \tag{A.42}$$

By replacing with the obtained expressions in eqs (A.40) and (A.42) for the terms $(A^4\sin^4\Delta\phi)/b^2$ and $(a^2A^2\sin^2\Delta\phi)/b^2$ respectively in eqn (A.38) I get

$$1/a^2[-(A^2\sin^2\Delta\phi)/B^2+(A^2\sin^4\Delta\phi)/a^2+a^2/B^2-\sin^2\Delta\phi]=\cos^2\Delta\phi/B^2 \tag{A.43}$$

By solving eqn (A.43) I arrive at the following equation, which is a quadratic equation of a^2

$$\sin^2\Delta\phi\, a^4-a^2\sin^2\Delta\phi(A^2+B^2)+A^2B^2\sin^4\Delta\phi=0 \tag{A.44}$$

In eqn (A.44) let

$$g=-\sin^2\Delta\phi(A^2+B^2) \tag{A.45}$$

$$f=\sin^2\Delta\phi \tag{A.46}$$

and

$$h=A^2B^2\sin^4\Delta\phi \tag{A.47}$$

Therefore, the quadratic eqn (A.44) becomes

$$f a^4+g a^2+h=0 \tag{A.48}$$

or,

$$a^2=[-g\pm\sqrt{g^2-4\mathrm{fh}}]/2\mathrm{f} \tag{A.49}$$

or, $$a^2=[(A^2+B^2)\pm\sqrt{(A^2+B^2)^2-4A^2B^2\sin^2\Delta\phi}]/2 \tag{A.50}$$

and *a* is given by square root of eqn (A.50). The parameter *b* can be found by putting the obtained value of *a* in eqn (A.32) and the parameter ζ ($\cos\alpha$) can be found by putting the obtained values of *a* and *b* in eqn (A.35). I see that *a* changes with change in $\Delta\phi$, which means the ellipse changes when the phase difference between the *X* and *Y* polarization components is changed.

A.4 Conversion from circular polarizations to linear polarization and vice versa

Conversion from two circular polarizations of opposite hands having equal magnitudes and phase difference $\theta_2-\theta_1$ to a linear polarization whose angle of orientation will be $(\theta_2-\theta_1)/2$ is shown in the following derivation.

From Kildal (2000), the two hands of circular polarization directions are given in terms of unit vectors. The RHC polarization direction is given by $(\hat{x}-j\hat{y})/\sqrt{2}$ and the LHC polarization direction is given by $(\hat{x}+j\hat{y})/\sqrt{2}$.

An LHC polarization with amplitude *A* and phase θ_1 in vector form is given as

$$A e^{j\theta_1}(\hat{x}+j\hat{y})/\sqrt{2} \tag{A.51}$$

Now an RHC polarization with amplitude *A* and phase θ_2 in vector form is given as

$$A e^{j\theta_2}(\hat{x}-j\hat{y})/\sqrt{2} \tag{A.52}$$

adding them both I get

$$[A e^{j\theta_1}(\hat{x}+j\hat{y})+A e^{j\theta_2}(\hat{x}-j\hat{y})]/\sqrt{2} \tag{A.53}$$

$$= \quad A/\sqrt{2}[\hat{x}(e^{j\theta_1}+e^{j\theta_2})+j\hat{y}(e^{j\theta_1}-e^{j\theta_2})] \tag{A.54}$$

$$= \quad A/\sqrt{2}[\hat{x}(e^{j\theta_1}+e^{j\theta_2})+\hat{y}(e^{j(\pi/2+\theta_1)}-e^{j(\pi/2+\theta_2)})] \tag{A.55}$$

$$= A/\sqrt{2}\,\hat{x}[(\cos\theta_1+j\sin\theta_1+\cos\theta_2+j\sin\theta_2)]$$

$$+A/\sqrt{2}\,\hat{y}[(\cos(\pi/2+\theta_1)+j\sin(\pi/2+\theta_1)-\cos(\pi/2+\theta_2)-j\sin(\pi/2+\theta_2))] \tag{A.56}$$

$$= \quad A/\sqrt{2}\,\hat{x}[\cos\theta_1+\cos\theta_2+j(\sin\theta_1+\sin\theta_2)]$$

$$+A/\sqrt{2}\,\hat{y}[\cos(\pi/2+\theta_1)-\cos(\pi/2+\theta_2)+j(\sin(\pi/2+\theta_1)-\sin(\pi/2+\theta_2))] \tag{A.57}$$

$$= \quad A/\sqrt{2}\,\hat{x}[(2\cos(\theta_1+\theta_2)/2\times\cos(\theta_1-\theta_2)/2+j(2\sin(\theta_1+\theta_2)/2\times\cos(\theta_1-\theta_2)/2))]$$

$$+A/\sqrt{2}\,\hat{y}[(2\sin(\pi+\theta_1+\theta_2)/2\times\sin(\theta_2-\theta_1)/2)+j(2\cos(\pi+\theta_1+\theta_2)/2\times\sin(\theta_1-\theta_2)/2)] \tag{A.58}$$

$$= \quad A/\sqrt{2}[2\,\hat{x}\cos(\theta_1-\theta_2)/2(\cos(\theta_1+\theta_2)/2+j\sin(\theta_1+\theta_2)/2)]$$

$$+A/\sqrt{2}[2\,\hat{y}\sin(\theta_1-\theta_2)/2(-\cos(\theta_1+\theta_2)/2-j\sin(\theta_1+\theta_2)/2)] \tag{A.59}$$

$$= \quad \sqrt{2}\,Ae^{j(\theta_1+\theta_2)/2}[\hat{x}\cos(\theta_2-\theta_1)/2+\hat{y}\sin(\theta_2-\theta_1)/2] \tag{A.60}$$

Thus from eqs (A.53) and (A.60) I see that if I have two opposite hands of circular polarization with equal amplitudes and phase difference $\theta_2-\theta_1$ (from eqn (A.53)) then the vectorial addition of these two circular polarizations would yield a linear polarization as in eqn (A.60) with position angle $(\theta_2-\theta_1)/2$. This is also in accordance with the satement in TMS (Thompson et al. 2001) that "A linearly polarized wave with position angle ψ can be decomposed into right and left circularly polarized waves of equal amplitudes and phase difference 2ψ ."

Note- By the convention given in IEEE, the tip of the vector of the electric field in RHC rotates clockwise when looked in the direction of propagation of wave.

Now I will find out how to get the magnitudes and phases of the two initial circular polarizations from the linear polarization in eqn (A.60).

I take the X component that is $\sqrt{2}\,Ae^{j(\theta_1+\theta_2)/2}[\cos(\theta_2-\theta_1)/2]$ and Y component that is $\sqrt{2}\,Ae^{j(\theta_1+\theta_2)/2}[\sin(\theta_2-\theta_1)/2]$ of eqn (A.60) and form $X\pm jY$. First I will see if $X-jY$ really gives the magnitude and phase of one hand of circular polarization or not.

$$X-jY=\sqrt{2}\,A\cos(\theta_2-\theta_1)/2(\cos(\theta_1+\theta_2)/2+j\sin(\theta_1+\theta_2)/2)$$

$$-j\sqrt{2}\,A\sin(\theta_2-\theta_1)/2(\cos(\theta_1+\theta_2)/2+j\sin(\theta_1+\theta_2)/2) \tag{A.61}$$

$$= \quad \sqrt{2}\,A[\cos(\theta_2-\theta_1)/2\cos(\theta_1+\theta_2)/2+\sin(\theta_2-\theta_1)/2\sin(\theta_1+\theta_2)/2]$$

$$+j\sqrt{2}\,A[\cos(\theta_2-\theta_1)/2\sin(\theta_1+\theta_2)/2-\sin(\theta_2-\theta_1)/2\cos(\theta_1+\theta_2)/2] \tag{A.62}$$

$$= \quad \sqrt{2}\,A[\cos(\theta_1)+j\sin(\theta_1)] \tag{A.63}$$

$$= \quad \sqrt{2}\,Ae^{j\theta_1} \tag{A.64}$$

Thus I see from eqn (A.64), which is obtained by forming $X - \mathrm{j}Y$ where X and Y are the components of linear polarization in eqn (A.60), I arrive at the phase of the starting LHC that is of $A\,e^{j\theta_1}(\hat{x}+j\,\hat{y})/\sqrt{2}$ (given in eqn (A.51)) and I also obtain the magnitude ($\sqrt{2}\,A$) as twice the magnitude of the LHC, which is $A/\sqrt{2}$. Similarly $X + \mathrm{j}Y$ formed using X and Y components of the linear polarization in eqn (A.60) will yield the magnitude of the RHC (eqn(A.52)) multiplied by 2 and the phase of the RHC.

APPENDIX B: DESIGN AND CODE DESCRIPTION OF LOGIC BLOCKS OF CHAPTER 4

B.1 Individual block details

In this section I provide all the details involved in the individual blocks and in their connections along with providing details on the timing of operations and of dataflow. While demonstrating the VHDL implementation I describe the simulated logic, which has different bit widths since for the implementation I truncated the bit. I enclose a CD containing the main design without any bit truncation and its details will be provided at the end of this chapter. The implemented design is available in the frontend of MPIFR. Further, until simulation was complete I knew that each input sample will have 10 bit since that was originally told to me and hence the main/simulated design has 10 bit input samples. However, after I finished simulation, I was told that each input sample will have 8 bit (probably changed to 8 bit from 10 bit) and since I had already finished the simulation, I only changed to 8 bit while implementing the design. The blocks in B.1.1 and in B.1.2 and all connections between them duplicate and the two copies work in parallel one for each polarization state. Further the blocks from B.1.1 to B.1.3 are contained in a single module named as *dout2acmf* in the VHDL code and can be found in the enclosed CD.

B.1.1 Clock rate reduction logic

The details of this block are published by Tuccari (2004) and are not repeated here. The outputs from this block are passed to the next block as inputs.

B.1.2 Serial Frame Generator

Note- All elements operate at 128 MHz whether mentioned or not.

This block is discussed with Alan Roy. From the previous block 8 samples come in at a rate of 128 MHz, which is equivalent to 1 sample coming in at 8×128 = 1024 MHz. It is possible in our case to keep the incoming sample rate the same (1024 MHz) by dividing into 8 identical stages each operating at 128 MHz with certain time delay between them predecided by the incoming data rate. So I divided into 8 stages having the same serial data processing elements. The serial frame generator enables this division by sending 8 data lines to the 8 stages. The following method is used to feed without any sample loss the 8 FFTs all of which require time-domain frames to be fed serially sample by sample.

I have chosen to do 1024 point FFT to obtain a spectral resolution of 1 MHz. So one spectrum or frame has 1024 points or samples. Time-domain frames with samples coming in serially need to be fed continuously to each FFT at 128 MHz. So I take 1024 words deep register that stores the 8 incoming samples together at one clock pulse and sends out one sample at one clock pulse and as evident I need eight such registers each to feed one FFT. A total of 128 clock pulses are required to fill in a register whereas 1024 clock pulses are required to read out the same register. So the data rate at the output of the register is reduced to 1/8 times the input data rate. However, the data from a location has to get out before or at the same time the next data for that location arrives otherwise the existing sample in that location would be overwritten. It can be shown that if writing in the eight registers occur consecutively and cyclically and if the reading operation in each register starts at the latest 128 clock pulses after writing of first 8 samples is complete, then there is no overwriting. Each register receives a data frame whose serial number is the serial number of existing data frame in the register incremented by 8. Each register sends out data serially at a rate of 128 MHz to be processed serially by the corresponding FFT and the successive blocks.

B.1.2.1 Simple case illustration

Now I will go into the details of a simple identical case, which employs the same method. To generalize the method I represent the operating clock frequency by M (128 in our case); I represent the sampling rate by S (1024 in our case), which also represents the number of samples in a frame being consistent with our case; I represent the number of incoming samples by N (8 in our case), such that $M \times N = S$; the units remain the same. So if N samples are coming in parallel at M frequency units then our objective is to transfer frames continuously in N lines (sequence of frames doesn't matter) at a rate of 1 sample at M frequency units without any data loss. To do so I would need N registers each of which is S words deep operating at M frequency units and the delay between any consecutive register would be M number of clock pulses.
Now let us take the case when $S = 8$, $N = 2$ and $M = 4$. So in this case there are 2 registers each 8 words deep and 2 samples arrive at one clock pulse (4 MHz) maintaining the incoming data rate of 8 MHz. The writing and reading operations occur as follows:

I begin with frame number 1, which represents the frame arriving with the start of the serial frame generator. Here I start with the clock pulse with the start of writing operation in the serial frame generator. Writing operation starts with the designated first register. Here I will start reading M clock pulses after I start writing in a register, which is the maximum delay I can incur between the start of read and write operation in a register without any overwriting; this delay can vary from 1 to M and the upper limit is picked up to show that the numbers from 1 to M are safe for this delay. So the delay is 4 clock pulses here in the following demonstration.

Clock cycles 1, 2, 3, 4 (writing of first frame in the first register): first register is written. Writing in the first register starts by filling first two locations simultaneously at one clock pulse with the first two respective and parallel samples of the first frame. The next two locations get the next two samples in the next clock pulse in the same way and so on. So in the fourth clock pulse seventh and eighth locations are written. Thus writing of the first frame is complete and the register is full.

Clock cycles 5, 6, 7, 8 (writing of second frame in the second register): second register is written. The second frame enters this register the same way as the first frame enters the first register. It gets filled in the eighth clock pulse. So writing of the second frame is complete.

Clock cycles 5, 6, 7, 8, 9, 10, 11, 12 (reading of first frame from the first register): first register is read out. The samples are read out consecutively at a rate one sample per clock pulse starting from the first sample/location. So in the twelfth clock pulse the last sample of first frame leaves.

Clock cycles 9, 10, 11, 12, 13, 14, 15, 16 (reading of second frame from the second register): second register is read out as the first register. So in the sixteenth clock pulse the last sample of the second frame leaves.

Clock cycles 9, 10, 11, 12 (writing of third frame in the first register): The process of writing in the first register repeats and again it is written with frame number 3. In clock pulse number 9 first and second locations of first register which are empty then are written. In clock pulse number 10 third and fourth locations which are empty then are written. In clock pulse number 11 fifth and sixth locations, which are empty then are written and in clock pulse number 12 seventh and eighth locations are written with the eighth sample of third frame entering at the same time when the eighth sample of first frame is leaving and the two can occur in one clock pulse with no overwriting. So it is only the last sample of third frame that enters when the last sample of first frame leaves the buffer. All other samples of third frame enter after the corresponding samples of first frame have left being totally safe.

Clock cycles 13, 14, 15, 16 (writing of fourth frame in the second register): The process of writing in the second register repeats and again it is written with frame number 4 with no overwriting as is shown above for second time writing of first register. So it is only the last sample of fourth frame that enters when the last sample of second frame leaves the buffer. All other samples of fourth frame enter after the corresponding samples of second frame have left being totally safe.

Clock cycles 13, 14, 15, 16, 17, 18, 19, 20 (reading of third frame from the first register): The process of reading from the first register repeats and the last sample of the third frame leaves the first register in the twentieth clock pulse (in twentieth clock pulse last sample of fifth frame also enters and all other samples of fifth frame enter after the corresponding samples of third frame have left).

Clock cycles 17, 18, 19, 20, 21, 22, 23, 24 (reading of fourth frame from the second register): The process of reading from the second register repeats and the last sample of the fourth frame leaves the second register in the twenty fourth clock pulse (in twenty fourth clock pulse last sample of sixth frame also enters and all other samples of sixth frame enter after the corresponding samples of fourth frame have left).

If x is the serial number of any frame input to a register then at its output this frame will be followed by a frame having serial number $x + 2$. I have shown two rounds of writing in and reading from each of the two registers. I see that each of the two registers are reading out frames serially sample by sample and continuously at the operating clock frequency maintaining the incoming data rate without any data loss due to overwriting. If I continue then there will be several such rounds and I get continuous frames with samples arriving serially at the outputs of the two registers.

Now if I extend this concept to the case when S =1024 MHz, M = 128 MHz and N = 8 samples, which is our case then I will get eight output lines each from one of the eight registers sending out continuous frames of time-domain samples at the rate of one sample per clock pulse. If x is the serial number of any frame input to a register then at its output this frame will be followed by a frame having serial number $x + 8$ ($x + N$ for the general case). Further, in our case I can start reading a register 128 (M for the general case) clock pulses after I have started writing in the register but I start reading a register 2 clock pulses after I start writing in that register being totally safe. Now I will go into the design details of this block. I will describe the logic implemented in VHDL. I will also provide details on the timing of the signals in the logic elements of the block.

B.1.2.2 VHDL implementation

The following figure (fig. B.1) shows the layout of the top module of the serial frame generator implemented in VHDL. It is named as *doutf2*. The layout shows the components and the signals in the top module with their names. One can find the same names in the VHDL code.

Note : In this module and in the following modules whenever I refer to an operation occurring at a clock pulse, I mean the operation occurs at the rising edge of the clock pulse. All signals are initialized to 0 in binary in this module and in the following ones unless stated otherwise.

Module *doutf2* (refer to fig. B.1)*:*

Inputs:

clock: system clock running at 128 MHz.
We: 1 bit user defined active-high control signal to start *doutf2*.
X/Y: Represents a group of 8 parallel input lines getting data from the 8 parallel output lines from the 'clock rate reduction' logic corresponding to *X/Y* polarization. The line numbers are mapped 1 to 1 with those of the 'clock rate reduction' logic block. Each line receives one sample and is configured with 10 bit.

Outputs:

rQ1-rQ8/iQ1-iQ8: 8 output lines corresponding to *X/Y* polarization. Each line is configured with 11 bit with 0 in the MSB (concatenated to MSBs of *X/Y* respectively) representing positive integers in two's complement form.

Component modules:

1. dfftdatn2: With the first clock pulse after *we* goes high, the present *X* and *Y* samples are transferred simultaneously to *sdata1* (10 bits for each sample) and *sdata2* (10 bits for each sample) respectively. In the same clock pulse next *X* and *Y* samples arrive and get transferred to *sdata1* and *sdata2* respectively in the following clock pulse. This continues as in any sequential logic.

2. control_n: With the first clock pulse after *we* goes high *sel*, which is a 3 bit control signal generated by *control_n* enters its first state ("000") out of 8 states. Its state gets incremented by 1 after every 128 clock pulses after *we* is high. The signal *re1* (1 bit) goes high two clock pulses after *we* goes high. The signals from *re1-re8* (1 bit each) go high with a delay of 128 clock pulses between any two consecutive signals.

3. counter_7bitac: This is a 7 bit up counter. It starts counting from 0 from the next clock pulse after *we* goes high. The count gets incremented every clock pulse. This count is used as the signal *s1* (7 bit).

4. counter_10bitac: This is a 10 bit up counter. There are 8 such counters as shown in the figure each triggered/initiated by one of the lines from *re1-re8* in the same way as *we* triggers *counter_7bitac*. The signals *read_address1-read_address8* (each 10 bit) get the counts from the counters triggered by *re1-re8* respectively in the same way as *s1* gets the count in *counter_7bitac*.

5. data_demuxtest: This is the central block of a *serial frame generator*. I will discuss one of the two such blocks shown in fig. B.1 as both operate parallelly. Each operate for one polarization channel. So I go into the details of the one getting *sdata1* corresponding to *X* polarization channel only. This block has 8 buffers each having 1024 locations and each location is configured with 10 bit width. Each state of *sel* discussed earlier is used to select a buffer in *data_demuxtest*.

As *we* goes high, *sel* (used for writing only) enters "000" state in the next clock pulse, which means to select the first out of 8 buffers in *data_demuxtest;* the consecutive states of *sel* select the consecutive buffers. Since *sel* remains in this state for 128 clock pulses, the first buffer remains selected for these 128 clock pulses.

The first 8 samples appear in *sdata1* one clock pulse after *we* goes high and at the same time first write address for writing in the first 8 locations in the selected buffer appears in the bus *s1*. Thus in the following clock pulse or after two clock pulses after *we* goes high these 8 samples in *sdata1* enter the first 8 locations and it is then when next 8 samples also appear in *sdata1* and when *sl* increments by 1 pointing the next 8 locations. In following clock pulse these next 8 samples enter the next 8 locations and this continues until *sel* changes its state to “001” selecting the second buffer and the whole process of writing repeats for the second buffer and then for the third buffer and goes on cyclically.

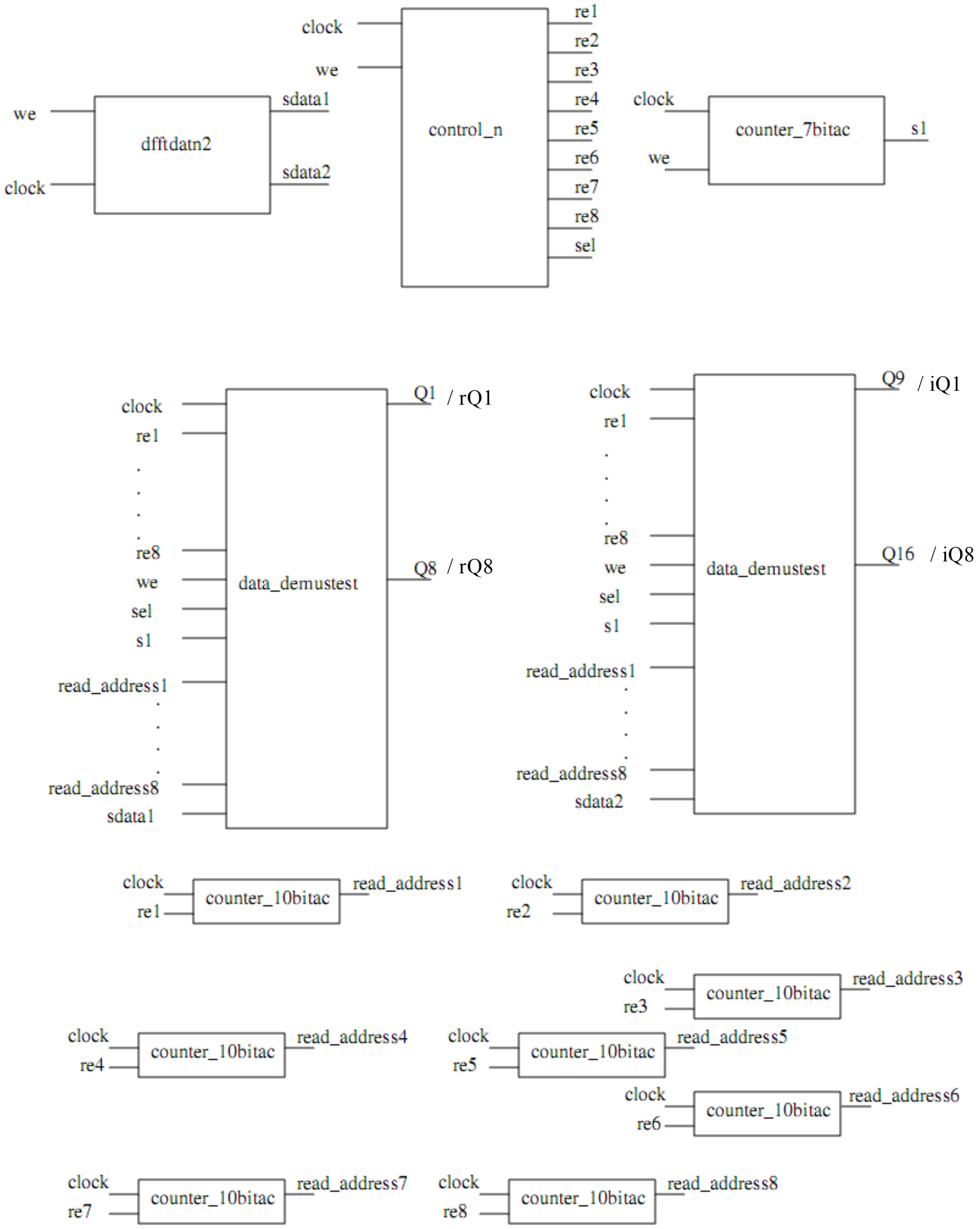


Fig. B.1: The *doutf2* module in the VHDL code of the digital circular polarizer. This module implements the serial frame generators for both *X* and *Y* polarizations. The names of the components shown are same as in the VHDL code. *X* and *Y* signals in the figure are inputs to *doutf2* corresponding to *X* and *Y* polarization channels respectively. The signals *rQ1* to *rQ8* are outputs corresponding to *X* polarization and the signals *iQ1* to *iQ8* are outputs corresponding to *Y* polarization from *doutf2*. The *clock* in the figure represents the input clock and the signal *we* is a user defined control signal. All other signals are internal to the module; the signals in the LHS of each component in the figure represent the inputs to that component and the signals in the RHS of each component represent the outputs from that component.

The signal *re1* goes high two clock pulses after *we* goes high. One clock pulse after *re1* (used for first buffer only) goes high, the first read address for reading the first buffer appears in *read_address1* pointing the first location with “0000000000”; the consecutive states of *read_address1* select the consecutive locations in the first buffer. In the next clock pulse after the first read address appears, the output line *Q1* of the first buffer receives the first sample from the first location and in the same clock pulse *read_address1* increments by 1 pointing the next (second) location in the buffer. In the following clock pulse after the second read address (*read_address1* = “0000000001”) has appeared, *Q1* receives the second sample with the first sample moving forward and *read_address1* again gets incremented by 1 pointing the third location. Thus *Q1* receives one sample per clock pulse with samples proceeding to one of the FFT blocks.

The signal *re2* goes high to read from the second buffer 128 clock pulses after *re1* goes high as there is a delay of 128 clock pulses between writing operations in consecutive buffers cyclically. Similarly *re3* follows *re2*, *re4* follows *re3* and so on and reading from all buffers occur in the same way.

The signal *rQ1* is the output line from the first buffer after concatenating 0 in the MSB of *Q1* (representing the two's complement form of *Q1), rQ2* is from the second buffer,...., *rQ8* from the eighth buffer. So *rQ1* to *rQ8* each sends continuous frames at the rate of one sample (11 bit) per clock pulse to the corresponding FFT block. There is a delay of two clock pulses between starting the write operation for a frame and starting the read operation for the same frame in a buffer.

So I am getting 8 output lines (*rQ1-rQ8*) corresponding to *X* polarization and 8 output lines (*iQ1-iQ8*) corresponding to *Y* polarization from *doutf2*. The lines *rQ1* and *iQ1* feed the first out of 8 identical stages, *rQ2* and *iQ2* feed the second,...., *rQ8* and *iQ8* feed the eighth. Each of the stages has an FFT as the first processing element receiving the corresponding two output lines from *doutf2*. So as evident from the timing discussed in B.1.2.2, *rQ1/iQ1* to *rQ8/iQ8* start sending data with a delay of 128 clock pulses between any two consecutive lines and hence the 8 identical stages must also start with a delay of 128 clock pulses between any two consecutive stage. Starting the digital circular polarizer starts the first stage and 128 clock pulses later the second stage starts and so on. The *doutf2* block starts two clock pulses after the digital circular polarizer starts and *rQ1/iQ1* starts sending data four clock pulses after *doutf2* starts and thus 6 clock pulses after the digital circular polarizer starts. The eight stages are named as *combunitfft1*, *combunitfft2*,....,*combunitfft8* in the VHDL code. Now I will proceed to describe one of the 8 stages (*combunitfft1*).

B.1.3 FFT to accumulators of first out of eight stages

Now I describe the first stage out of eight stages receiving two parallel outputs from the first of the eight pairs of output lines of the previous block where each pair corresponds to the *X* and *Y* from the two corresponding simultaneous buffers. The two output lines feed the FFT block described below sample by sample at a rate of 128 MHz.

1. FFT

A streaming pipelined FFT is used to process the serially arriving time frames. For details of this block refer to Xilinx documentation for FFT version 5.

I feed the incoming real data for the *X* and *Y* polarizations to the real and imaginary channels of the FFT respectively since I can thus perform FFT of the two real functions simultaneously. After a certain delay the FFT starts sending outputs sample by sample at a rate of 128 MHz and thus frame by frame.

However, at the outputs of the FFT I get a combination of *X* and *Y* with the real channel representing $X_r(F,K)-Y_i(F,K)$ and imaginary channel representing $X_i(F,K)+Y_r(F,K)$ where *K,* which is also transferred in an output line*,* represents the index of frequency-domain sample number of a frame *F*; the outputs $X_r(F,K)-Y_i(F,K)$, $X_i(F,K)+Y_r(F,K)$ and *K* are parallel; the value of *K* in any data line increments by 1 every clock pulse representing the next sample in a frame and repeats every 1024 clock pulses; the value of *F* in any data line remains the same for 1024 clock pulses from the time (clock pulse) this value initiates and after those 1024 clock pulses it increments by 8 representing a new frame; $X_r(F,K)$, $X_i(F,K)$, $Y_r(F,K)$ and $Y_i(F,K)$ represent the real and imaginary parts of *X(F,K)* and *Y(F,K)* respectively. So I need to extract the real and imaginary parts of the *X* and *Y* polarizations from their combination at the outputs of the FFT. The following logic block is used to extract the real and imaginary parts of *X* and *Y.*

2. Decoder

This receives the inputs $X_r(F,K)-Y_i(F,K)$ say *R(F,K),* $X_i(F,K)+Y_r(F,K)$ say *I(F,K)* and *K* simultaneously at the same clock pulse from the FFT. So it needs to extract, from the inputs *R(F,K)* and *I(F,K)*, the output quantities $X_r(F,K)$, $X_i(F,K)$, $Y_r(F,K)$ and $Y_i(F,K)$ simultaneously and transfer these extracted quantities without any sample loss. These four quantities are extracted by implementing the following four equations respectively:

$$X_r(F,K)=R(F,K)/2+R(F,1024-K)/2 \tag{B.1}$$

$$X_i(F,K)=I(F,K)/2-I(F,1024-K)/2 \tag{B.2}$$

$$Y_r(F,K)=I(F,K)/2+I(F,1024-K)/2 \tag{B.3}$$

$$-Y_i(F,K)=R(F,K)/2-R(F,1024-K)/2 \tag{B.4}$$

So the decoder needs to transfer *R(F, K)*, *R(F, 1024-K)*, *I(F, K)* and *I(F, 1024-K)* in parallel with the values of *F* and *K* changing as described earlier and this is accomplished the following way.

There are two units one working to transfer *R(F, K)*, *R(F, 1024-K)* and the other to transfer *I(F, K), I(F, 1024-K).* Both the units work identically and parallelly. Hence, I will only go into the details of one of them and I pick up the one transferring *R(F, K)* and *R(F, 1024-K)* simultaneously.

This unit has two buffers each having 1024 locations. One clock pulse after this unit starts, the spectral data point *R(N, 0),* where *N* represents the first frame from the FFT, arrives at its input and it is then when first buffer gets selected for writing. In the next clock pulse (two clock pulses after this unit starts), *R(N,0)* enters the location with address 0 (address is pointed by *K*, which is 0 then) in the first buffer with *R(N,1)* at the input and one clock pulse after *R(N,0)* has entered the buffer, *R(N,1)* enters the buffer in location whose address is 1 and this continues until *R(N,1023)* enters 1024th location with address 1023 filling the first buffer. When *R(N,1023)* enters the first buffer, the second buffer gets selected for writing in the same way and the first buffer for reading with *R(N+8, 0)* at the input. Hence one clock pulse after *R(N+8, 0)* arrives at the input, it gets written in the location 0 of the second buffer and *R(N,0)* and *R(N, 1024-0)=R(N,0)* from the first buffer are read out. In this way while writing of *R(N+8, K)* is happening in the second buffer, parallel reading of *R(N, K)* and *R(N,1024-K)* are happening from the first buffer. When *R(N+8, 1023)* enters 1024th location of the second buffer, first buffer is again selected for writing and the second buffer for reading with *R(N+16, K)* at the input. So from the next clock pulse after frame *N+8* has fully entered second buffer, *R(N+8, K)* and *R(N+8, 1024-K)* are read out from the second buffer with *R(N+16, K)* entering the first buffer parallely. In this way when one buffer is written, the other buffer is read out and this continues cyclically. Thus at the output I get *R(F, K)* and *R(F, 1024-K)* with values of *F* and *K* changing as usual.

So from two such units I obtain *R(F, K)*, *R(F, 1024-K), I(F, K), I(F, 1024-K)* in parallel to implement the four equations (B.1), (B.2), (B.3) and (B.4) simultaneously to obtain $X_r(F,K)$, $X_i(F,K)$, $Y_r(F,K)$ and $Y_i(F,K)$ in parallel and transfer these extracted quantities without any sample loss.

Each of these four parallel outputs from the decoder passes to the next two parallel elements (one for cross-power spectra accumulation and the other for power spectra accumulation); I also transfer the K in an output line in parallel with the four outputs. The next two blocks demonstrate the cross-power spectra accumulation and the power spectra accumulation respectively.

3. Cross-power spectra accumulation

This block receives the parallel inputs $X_r(F,K)$, $X_i(F,K)$, $Y_r(F,K)$, $Y_i(F,K)$ and *K* at 128 MHz from the decoder. It cross multiplies *X(F, K)* and *Y(F, K)* to obtain *Z(F, K)* (cross-power) that is $Z_r(F,K)$ and $Z_i(F,K)$ where $Z_r(F,K)$ is the real part and $Z_i(F,K)$ is the imaginary part of *Z(F, K)*; $Z_r(F,K)$ and $Z_i(F,K)$ are obtained parallely in two lines; *K* is also transferred parallely with each of them. $Z_r(F,K)$ and $Z_i(F,K)$ are accumulated parallely and separately frame by frame which means to add $Z_r(F,K)$ / $Z_i(F,K)$ for a particular *K* but different *F*. So the output of accumulation will have 1024 results corresponding to 1024 different values of *K*. The accumulation process of both $Z_r(F,K)$ and $Z_i(F,K)$ are identical so I will describe only one for neatness. So let us take the case of accumulating $Z_r(F,K)$, which is described as follows.

The line transferring $Z_r(F,K)$ from the output of cross multiplication is further broken into two lines. The first line is active (transferring $Z_r(F,K)$) while first 1048576 frames (I need to accumulate 1048576 frames) of Z_r are being transferred and during this time the second line is inactive (transferring zeros). So the first line is active for 1048576×1024 clock pulses, counting from the clock pulse when $Z_r(F,K)$ starts appearing at the output of complex multiplication, after which this line becomes inactive and the second line becomes active. Or, the second line is active when next 1048576 frames of Z_r are being transferred. So the second line remains active for 1048576×1024 clock pulses, counting from clock pulse when the next frame after the first 1048576 frames of Z_r have appeared at the output of the complex multiplication, after which this line becomes inactive. So the first 1048576 frames of Z_r , which correspond to the noise diode on state, are transferred through one line; the next 1048576 frames of Z_r , which correspond to the noise diode off state, are transferred through the other line. Further *K* is transferred in parallel to each of the two lines.

There are two accumulators for accumulating $Z_r(F,K)$, one getting input from the line corresponding to the noise diode on state and the other getting input from the line corresponding to the noise diode off state. The two accumulators also receive *K* along with $Z_r(F,K)$. So the first line transfers $Z_r(F,K)$ to the location with address *K* of the on-state accumulator, to get added to the present data (initially 0) in that location. In this way $Z_r(F,K)$ for a particular *K* accumulates in time in the location with address *K* of the on-state accumulator until the first line input to this accumulator becomes inactive and the second line becomes active to transfer $Z_r(F,K)$ to the location with address *K* of the off-state accumulator, to get added to the present data (initially 0) in that location. In this way $Z_r(F, K)$ for a particular *K* accumulates in time in the location with address *K* of the off-state accumulator until the second line input to this accumulator becomes inactive. As evident each accumulator has 1024 locations each for one frequency channel. Thus once 1048576 frames are accumulated in each of the two accumulators, the accumulation of $Z_r(F, K)$ stops and the results are ready to be read out. Similarly $Z_i(F, K)$ gets accumulated in another pair of accumulators with the on-state accumulators/off-state accumulators of both working in parallel.

4. Power spectra accumulation

This block receives the parallel inputs $X_r(F, K)$, $X_i(F, K)$, $Y_r(F, K)$, $Y_i(F, K)$ and *K* at 128 MHz from the decoder. It forms power $|X(F, K)|^2$ and $|Y(F, K)|^2$; $|X(F, K)|^2$ and $|Y(F, K)|^2$ are obtained in parallel in two lines; *K* is also transferred in parallel with each of them. Both powers $|X(F, K)|^2$ and $|Y(F, K)|^2$ are accumulated in parallel and separately frame by frame which means to add $|X(F, K)|^2/|Y(F, K)|^2$ for a particular *K* but different *F*. So the output of accumulation will have 1024 results corresponding to 1024 different values of *K*. The accumulation process of both $|X(F, K)|^2$ and $|Y(F, K)|^2$ are identical so I will describe only one for neatness. So let us take the case of accumulating $|X(F, K)|^2$, which is described as follows.

The line transferring $|X(F, K)|^2$ from the output of power formation is further broken into two lines. The first line is active (transferring $|X(F, K)|^2$) while first 1048576 frames (I need to accumulate 1048576 frames) of $|X|^2$ are being transferred and during this time the second line is inactive (transferring zeros). So the first line is active for 1048576×1024 clock pulses, counting from the clock pulse when $|X(F, K)|^2$ starts appearing at the output of power formation, after which this line becomes inactive and the second line becomes active. Or, the second line is active when next 1048576 frames of $|X|^2$ are being transferred. So the second line remains active for 1048576×1024 clock pulses, counting from the clock pulse when next frame after the first 1048576 frames of $|X|^2$ have appeared at the output of power formation, after which this line becomes inactive. So the first 1048576 frames of $|X|^2$, which correspond to the noise diode on state, are transferred through one line; the next 1048576 frames of $|X|^2$, which correspond to the noise diode off state, are transferred through the other line. Further *K* is transferred in parallel with each of the two lines.

There are two accumulators for accumulating $|X(F, K)|^2$, one getting input from the line corresponding to the noise diode on state and the other getting input from the line corresponding to the noise diode off state. The two accumulators also receive *K* along with $|X(F, K)|^2$. So the first line transfers $|X(F, K)|^2$ to the location with address *K* of the on-state accumulator, to get added to the present data (initially 0) in that location. In this way $|X(F, K)|^2$ for a particular *K* accumulates in time in the location with address *K* of the accumulator until the first line input to this accumulator becomes inactive and the second line becomes active to transfer $|X(F, K)|^2$ to the location with address *K* of the off-state accumulator, to get added to the present data (initially 0) in that location. In this way $|X(F, K)|^2$ for a particular *K* accumulates in time in the location with address *K* of the off-state accumulator until the second line input to this accumulator becomes inactive. As evident each accumulator has 1024 locations each for one frequency channel. Thus once 1048576 frames are accumulated in each of the two accumulators, the accumulation of $|X(F, K)|^2$ stops and the results are ready to be read out.

Similarly $|Y(F, K)|^2$ gets accumulated in another pair of accumulators with the on-state accumulators/off-state accumulators of both ($|X(F, K)|^2$ and $|Y(F, K)|^2$) working parallely.

Note- Outputs from the power formation and cross-power formation appear parallely. The on-state/off-state accumulators of the quantities $Z_r(F,K)$, $Z_i(F,K)$, $|X(F, K)|^2$ and $|Y(F, K)|^2$ operate parallely.

B.1.3.1 VHDL implementation

The module containing FFT to accumulators for the first stage out of eight identical stages is named as *combunitfft1*. So I will now go into the details of *combunitfft1*. The remaining seven stages are named as *combunitfft2*, *combunitfft3*, *combunitfft4*, *combunitfft5*, *combunitfft6*, *combunitfft7* and *combunitfft8*. The only difference between these blocks is the timing to start the read operation from the accumulators. The signal controlling the read operation in each stage takes care of the time when the read operation of the previous stage has stopped, that is read operation of a stage starts two clock pulses after the read operation of the previous stage has stopped. The read operation of the first stage starts two clock pulses after writing in all eight stages has stopped. I now go into the details of *combunitfft1*.

Module *combunitfft1* (refer to fig. B.2)

Inputs:

clk: 128 MHz input clock.
ce: 1 bit user defined control signal to start *combunitfft1*.
blank: 1 bit user defined control signal to pause accumulation by commanding to start transfer of zeros to the accumulators.
rout, imout: 11 bit data (all positive) in two's complement form from the first output lines of the serial frame generators corresponding to the *X* and *Y* polarizations respectively.

Outputs:

oacr, oaci, oacmx, oacmy: 69 bit $Z_r(F,K)$, $Z_i(F,K)$, $|X(F,K)|^2$, $|Y(F,K)|^2$ respectively after subtraction of off-state accumulator results from on-state accumulation results.
indexr, indexi, indexx, indexy: 10 bit indices of data in *oacr, oaci, oacmx* and *oacmy* respectively.
weoutr, weouti, weoutx, weouty: 1 bit control signal that controls writing of read data *oacr*, *oaci*, *oacmx*, *oacmy* to the four respective accumulators. (There are four final accumulators for accumulating *oacr*, *oaci*, *oacmx*, *oacmy* respectively from the eight identical stages.).

Bidirectional lines:

sstart: 1 bit control signal connected to *start* of Xilinx FFTv5.
sfwd_inv_we: 1 bit control signal connected to *fwd_inv_we* of Xilinx FFTv5.

Component modules:

1. fftdecacmcontrolfb: It is started by *ce*. One clock pulse after *ce* goes high, s*fwd_inv_we* (1 bit *fwd_inv_we* of Xilinx FFT) goes high and it remains high for one clock pulse after which it goes low. At the same clock pulse when *fwd_inv_we* goes low, *sstart* (1 bit *start* of Xilinx FFT) goes high. The signal *sstart* remains high for one clock pulse after which it goes low. This square wave pulse of *sstart* repeats every 1024 clock pulses required to initiate FFT block for the corresponding time frame. For details on the relation between *sstart* and initiation of FFT block please refer to Xilinx documentation

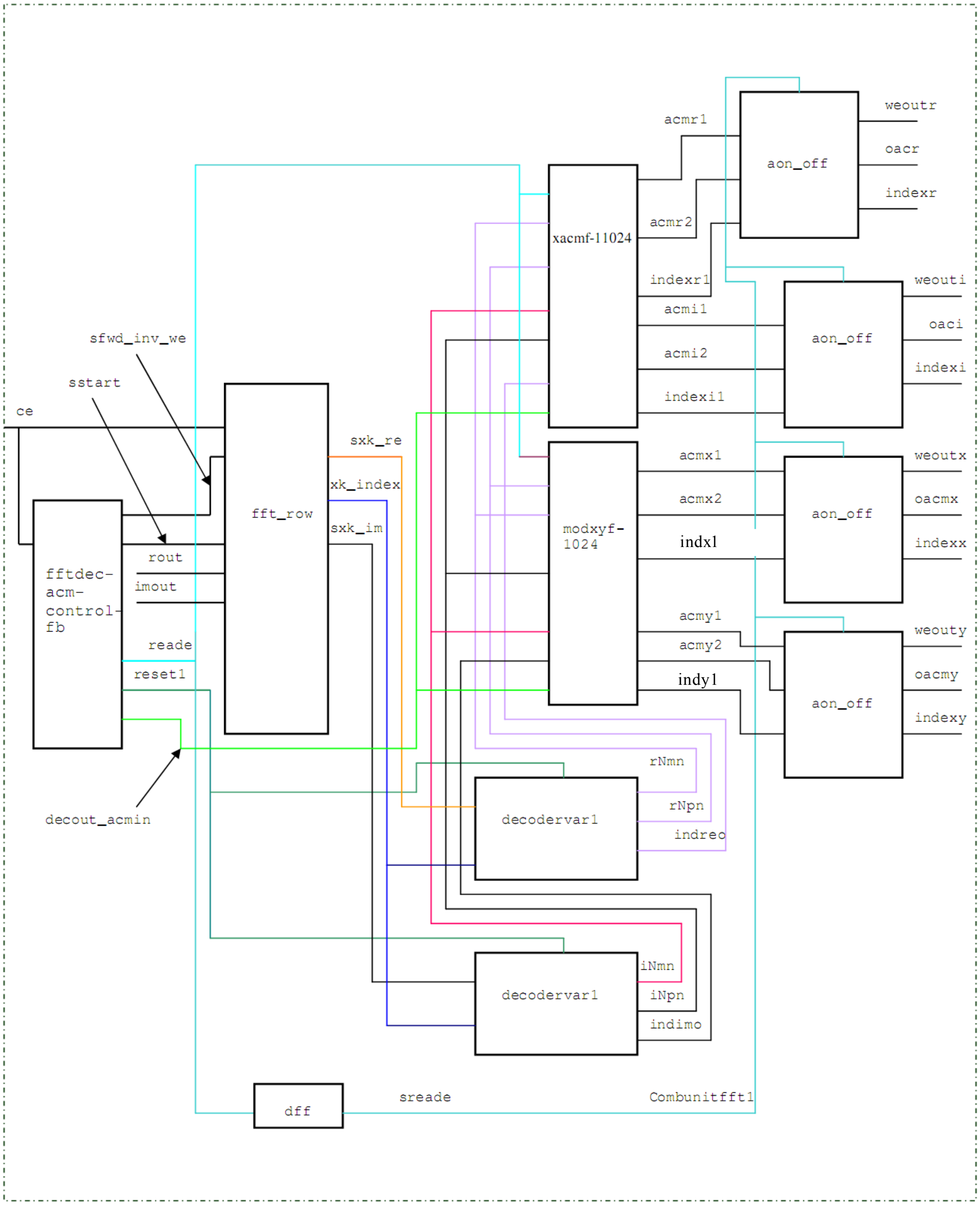


Fig. B.2: Top module *combunitfft1* depicting dataflow from FFT to the subtraction operation between the on-state accumulation results and off-state accumulation results. The lines in the L.H.S represent the inputs to the logic blocks and the lines in the R.H.S represent the outputs from the logic blocks in the figure.

for FFT v5. 2173 clock pulses after *sstart* is high for the first time, *reset1* (1 bit reset of the two *decodervar1s*) goes high since after 2174 clock pulses after *sstart* is high, FFT outputs start appearing to enter the two *decodervar1s* with the real channel of the FFT (*X* polarization) entering one *decodervar1* and the imaginary channel of the FFT (*Y* polarization) entering the other *decodervar1*. 1025 clock pulses after *reset1* goes high, *decout_acmin* (1 bit control signal to start *xacmf11024* and *modxyf1024*) goes high and it remains high for 1048576 × 1024 × 2 + 2 clock pulses to allow accumulation in the on-state and off-state accumulators consecutively after which it goes low. The signal *reade* (1 bit) goes high two clock pulses after *decout_acmin* goes low to start reading from the four accumulators and *reade* remains high for 1026 clock pulses. (This *reade* is only for the first out of eight identical stages. Other *reade* signals for the rest seven stages takes care when the *reade* signal of the previous stage has stopped to start reading).

2. fft_row: The signals of this block *ce, sfwd_inv_we, rfd, sstart, fwd_inv, sdv, done, clk, busy, edone, rout, sxk_im, xn_index, sxk_re, imout, xk_index* are connected to *ce, fwd_inv_we, rfd, start, fwd_inv, dv, done, clk, busy, edone, xn_re, xk_im, xn_index, xk_re, xn_im, xk_index* of Xilinx FFTv5 respectively.

3. decodervar1: As shown in fig. B.2 there are two such modules working in parallel one receiving s*xk_re* corresponding to *R(F, K)* generating quantities in eqs (B.1) and (B.4) as outputs and the other receiving s*xk_im* corresponding to *I(F, K)* generating quantities in eqs (B.2) and (B.3) as outputs. So I discuss below only the one receiving s*xk_re* whose details is shown in fig. B.3.

Inputs:

clock: 128 MHz clock , *clk* of *combunitfft1*.
reset: 1 bit control signal to start *decodervar1, reset1* of *combunitfft1*.
address: 10 bit address line, *xk_index* of *combunitfft1* representing *K* of input data.
datain: 22 bit data line, s*xk_re* of *combunitfft1* representing *R(F, K)*.

Outputs:

dout1: 22 bit output line, *rNpn* of *combunitfft1* representing $R(F,K)/2+R(F,1024-K)/2$.
dout2: 22 bit output line, *rNmn* of *combunitfft1* representing $R(F,K)/2-R(F,1024-K)/2$.
index: 10 bit index line representing *K* of the data in *dout1* and *dout2, indreo* of *combunitfft1.*

decodercontrol1024: It generates *s1* (1 bit) and *s2* (1 bit). Counting from the next clock pulse after *reset* goes high, the state of s1 toggles every 1024 clock pulses. In the next clock pulse after *reset* goes high *s1* starts with state '0'. The signal *s2* is the inverted version of *s1*.

dffdec: It delays a signal by one clock pulse. There are 1024 such elements connected in series with the first element getting *reset* as its input. So the *reset* is delayed by *n* clock pulses by the *n* elements where *n* = 1, 2, 3, …., 1024. When *n* = 1, output is *reset1(0)*, when *n* =2, output is *reset1(1)* and so on. Thus when *n* =1024, output is *reset1(1023)*.

counter_t1024: This is a 10 bit up counter initiated by *reset1(1023)*. It starts counting from 0 one clock pulse after *reset1(1023)* goes high with the count incrementing by 1 every clock pulse. The signal *addressi* (10 bit) represents this count.

dffadd1: It is started by *reset1(1023).* It delays *addressi* by one clock pulse and sends it as *index* (10 bit).

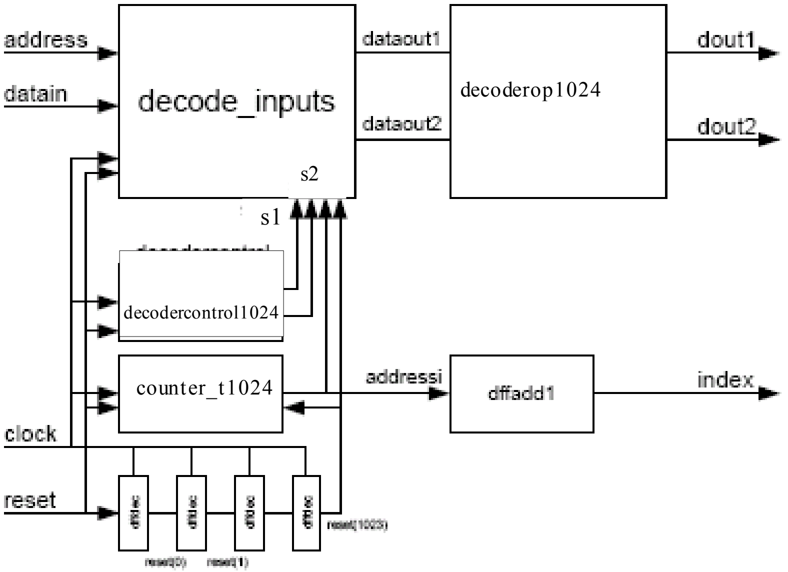


Fig. B.3: Logic details of the block *decodervar1*. Two such blocks are required to extract *X* and *Y* polarization from their combination at the outputs of the FFT. The input line *address* represents the index (of samples) coming out from the FFT and the input line *datain* represents the corresponding data coming out in the real channel from the FFT (the imaginary channel of the FFT feeds another identical and parallel decodervar1). This block is used to extract the real part of *X* polarization and imaginary part of *Y* polarization from the input data (the other *decodervar1* working similarly extracts real part of *Y* and imaginary part of *X* from its input data). The output lines *dout1* and *dout2* output real part of *X* polarization (real part of *Y* in the other case) and imaginary part of *Y* polarization (imaginary part of *X* in the other case) respectively.

decode_inputs210241: This is the central element of *decodervar1*. It has two buffers each having 1024 locations and each location is 22 bit wide. It is started by *reset*. Inputs *datain* and *address* starts appearing one clock pulse after *reset* goes high and it is then when *s1*, which is used to select between the two buffers for writing, enters '0' state selecting the first buffer for writing and *s2*, which is used to select between the two buffers for reading, enters '1' state selecting the second buffer (empty or filled with zeros then) for reading; state '0' of *s1/s2* selects first buffer and state '1' of *s1/s2* selects second buffer.

So the first buffer is written starting from the next clock pulse after its selection sample by sample as demonstrated under decoder of B.1.3 with a sample entering the location pointed by the corresponding value in *address*. Reading need not start in the second buffer the next clock pulse after its first time selection after *reset* goes high since being empty it doesn't need to be read then; hence I do not generate the read address, *addressi* (representing *K* of data to be read out), used for reading the buffers until the first frame from the FFT has entered the first buffer. Reading is initiated by *reset1(1023)*, which controls the generation of *addressi* the next clock pulse it goes high. When the last sample of this first frame enters the last location of the first buffer, *s1* toggles to '1' state selecting the second buffer for writing with the first sample of the next frame and its address at the inputs, *datain* and *address* respectively; it is then when *s2* toggles to '0' selecting first buffer for reading sample by sample as demonstrated under decoder of B.1.3; the two output lines *dataout1* and *dataout2* get data simultaneously from the locations pointed by *addressi* and *1024-addressi* respectively in the clock pulse following selection of the buffer to be read. So as required and already demonstrated, *addressi* starts appearing (and continues to appear after that in its way) when the last sample of the first frame from the FFT enters the last location of the first buffer or 1025 clock pulses after *reset* goes high.

So while the second buffer is being written (sample enters a location), the first buffer is being read out (sample goes out of a location) sample by sample; reading from a buffer and writing in the other buffer corresponds to the same sample number/location number in a frame/buffer. In this way writing and reading in a buffer after another continues cyclically and I get in the two output lines *dataout1* and *dataout2*, *R(F, K)* and *R(F, 1024-K)* respectively and in parallel.

decoderop1024: This unit takes *dataout1* and *dataout2* from *decode_inputs210241* as two parallel inputs, forms *(dataout1 + dataout2)/ 2* and *(dataout1 - dataout2)/ 2* in parallel and sends them as outputs *dout1* and *dout2* respectively. Thus *dout1* and *dout2* represent *R(F,K)/2 + R(F,1024-K)/2* and *R(F,K)/2 – R(F,1024-K)/2* respectively. There is no delay in generating these outputs since the logic is combinatorial. As can be figured out the *index* represents the *K* of data in *dout1/dout2*.

As already stated two *decodervar1* work in parallel one sending quantities in eqs (B.1) and (B.4) and the other sending the quantities in eqs (B.2) and (B.3) and I get all four of them in parallel.

Note: The outputs from the other decoder are named as *iNpn* and *iNmn* in *combunitfft1* (fig. B.2) corresponding to I*(F,K)/2 + I(F,1024-K)/2* and I*(F,K)/2 – I(F,1024-K)/2* respectively.

4. xacmf11024* (refer to fig. B.4)*:

Inputs:

clk: 128 MHz clock, *clk* of *combunitfft1*.
reset: 1 bit active high control signal to start *xacmf11024, decout_acmin* of *combunitfft1*.
reade: 1 bit active high control signal to start reading from the constituting accumulators, *reade* of *combunitfft1*.
blank: 1 bit active high user control signal to pause accumulations for certain duration when *xacmf11024* is in running condition. This duration refers to the time required to change the pointing of telescope to the source from the calibrator, *blank* of *combunitfft1*.
xr and xi: Real (*rNpn* of *combunitfft1*) and imaginary (*iNmn* of *combunitfft1*) parts of *X* polarization (representing *X(F, K)*) from *decodervar1* respectively. Each is 22 bit wide.
yr and yi: Real (*iNpn* of *combunitfft1*) and imaginary (*rNmn* of *combunitfft1*) parts of *Y* polarization (representing *Y(F, K)*) from *decodervar1* respectively. Each is 22 bit wide.
Indexi: 10 bit index of samples in *xr/xi/yr/yi, indreo* of *combunitfft1*.

Outputs:

oacr1, oacr2, oaci1, oaci2: 69 bit output lines from the accumulators accumulating $Z_r(F, K)$ corresponding to noise diode on state, $Z_r(F, K)$ corresponding to noise diode off state, $Z_i(F, K)$ corresponding to the noise diode on state and $Z_i(F, K)$ corresponding to noise diode off state respectively, *acmr1*, *acmr2*, *acmi1*, *acmi2* respectively of *combunitfft1*.
indexr1, indexr2, indexi1, indexi2: 10 bit output lines representing indices of data in *oacr1*, *oacr2*, *oaci1* and *oaci2* respectively. Same names in *combunitfft1*.

Component modules:

cscombf1024n: It is started by *reset*. It generates two signals *reset3* (1 bit) and *reset4 (1 bit)*. The signal *reset3* goes high one clock pulse after *reset* goes high and remains high for 1048576×1024 + 2 clock pulses after which it goes low. The signal *reset4* goes high two clock pulses before *reset3* goes low and it remains high for 1048576×1024 + 2 clock pulses after which it goes low.

complex_multiplier1024: It is started by *reset.* Its inputs *xr*, *xi*, *yr*, *yi* and *indexi* start arriving one clock pulse after *reset* goes high. It multiplies the complex quantities *xr + jxi* and *yr + jyi* (since *-yi* is received as evident from eqn (B.4)) to form the cross power *zr1 + jzi1* where *zr1* (45 bit) represents $Z_r(F, K)$ and *zi1* (45 bit) represents $Z_i(F, K)$. Both *zr1* and *zi1* are obtained in parallel at its outputs three clock pulses after *reset* goes high that is the input-output latency is two clock pulses. The output indices *indor* is transferred in parallel with *zr1* and *indoi* is transferred in parallel with *zi1*. This block also takes in the input *blank*, which when active transfers zeros in the output lines.

dff1: This element delays a 1 bit signal by one clock pulse. There are 3 such elements two of which are used to delay *reset* by two clock pulses to produce *reset2* (1 bit).

clockdividerac30: This element combined with the third *dff1* generates a signal *sel2* (1 bit). It is started by *reset2* which goes high one clock pulse after *reset3* goes high. Counting from the next clock pulse after *reset2* goes high, *sel2* toggles every 1048576×1024 clock pulses. In the next clock pulse after *reset2* goes high, *sel2* starts with '0'.

demux4ac1024: There are two such units working in parallel, one for transferring *zr1* another for transferring *zi1*. Since the operation of both are identical I will go into the details of only one that is the one for transferring *zr1* as follows: It receives *zr1* and *indor* from the *complex_multiplier1024*. It is started by *reset1*. If *reset1* is '1' then if *sel2* is '0' then the output lines *zr11* and *indor1* get the inputs *zr1* and *indor* respectively and the output lines *zr12* and *indor2* receive zeros. If *reset1* is '1' and if *sel2* is '1' then the output lines *zr12* and *indor2* get the inputs *zr1* and *indor* respectively and the output lines *zr11* and *indor1* receive zeros. The data arrive 2 clock pulses after *reset1* goes high and the output is received the same clock pulse the input arrives the logic being combinatorial. In the other unit receiving *zi1*, all the above description is applicable with *zr1*, *indor*, *zr11*, *zr12*, *indor1*, *indor2* replaced by *zi1*, *indoi*, *zi11*, *zi12*, *indoi1* and *indoi2* respectively. The remaining input/output signal lines are common to both.

acm1024: There are four such units of which two work to accumulate *zr11* and *zr12* respectively and the other two work to accumulate *zi11* and *zi12* respectively. Accumulation process of *zi11* and *zi12* is same as accumulation process of *zr11* and *zr12* respectively, which also means *zi11* and *zr11* (*zi12* and *zr12*) are accumulated in parallel. So I will only go into the details of accumulation of *zr11* and *zr12*.

Accumulation of *zr11* in one *acm1024*: *zr11* and *indor1* starts appearing two clock pulses after *reset3* goes high. The signal *reset3* is the write enable to initiate writing in this accumulator. One clock pulse after *zr11* appears, it gets added to the contents in the location pointed by *indor1* (representing *K* of *zr11* and of the location where *zr11* enters). Consecutive *zr11* get added to the contents of the corresponding location (pointed by *indor1*) one by one as described under topic 3 of B.1.3. Since I need to accumulate 1048576 spectra, *reset3* must remain high for 1048576 × 1024 (number of samples entering the accumulator) + 2 (delay between reset3 going high and *zr11* appearing as already mentioned) = 1073741826 clock pulses as it does (see *cscombf1024n*). So writing in this accumulator stops when *reset3* goes low. The signal *reade* is there, which when active enables reading in the output line *oacr1* and the index appearing in the output line *indexr1.*

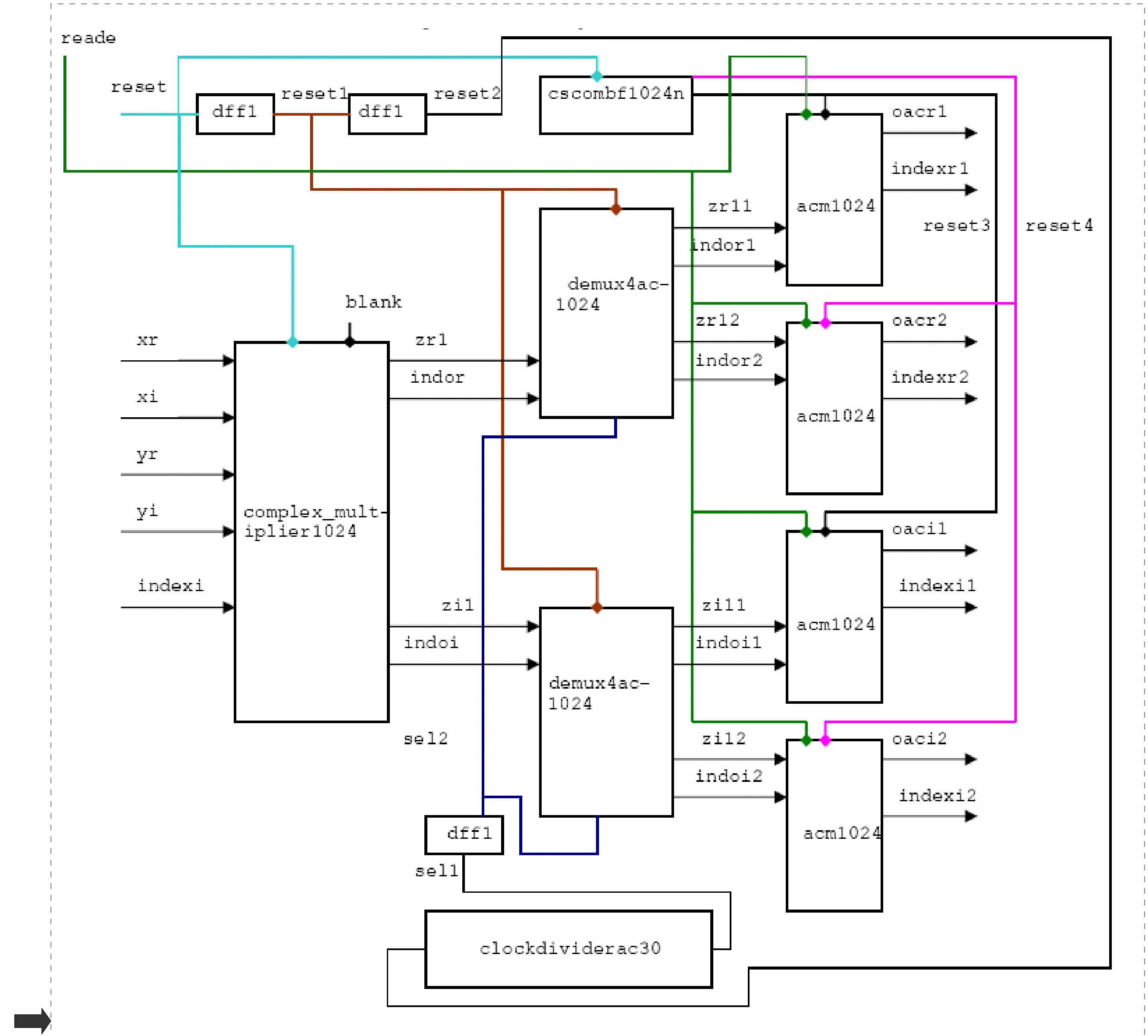


Fig.B.4: Logic block *xacmf11024*, which receives the real and imaginary parts of the *X* and *Y* polarization named as *xr, xi, yr* and *yi* respectively in the figure from the two decodervar1s described under the previous block detail of *combunitfft1*. It cross multiplies *X* and *Y* polarizations and accumulates the resulting *Z* spectrum for both on and off states of the noise diode. The accumulation results are read when desired. The four *acm1024* in the figure are four accumulators whose signals to the right are the outputs of this block. The signals to the RHS of the blocks in the figure are outputs from those blocks and the signals to the LHS of the blocks are the inputs to those blocks. The signal *clk* in the left most bottom corner is the clock (128 MHz), which feeds all the logic elements in the figure.

Accumulation of *zr12* in another *acm1024*: It is started by *reset4*. The signal *zr12* arrives two clock pulses after *reset4* goes high (at the same time when *reset3* goes low). This is also when *sel2* has toggled to 1 sending output in *zr12*. So *zr12* is written in this accumulator following the same way as *zr11* has been written with *indor2* pointing the location for writing in this accumulator. So *reset4* (as *reset3*) also remains high for $1048576 \times 1024 + 2$ clock pulses to accumulate 1048576 spectra. The signal *reade* is there, which when active enables reading in the output line *oacr2* and the index appearing in the output line *indexr2.*

Accumulation of *zi11* in a third *acm1024*: *zi11* is accumulated in parallel with *zr11*. All operations for accumulating *zi11* in this accumulator remain the same as for *zr11* in its accumulator with the input/output lines *zr11*, *indor1*, *oacr1*, *indexr1* replaced by *zi11*, *indoi1*, *oaci1*, *indexi1* respectively and all other input/output lines to these accumulators are common to both.

Accumulation of *zi12* in the fourth *acm1024*: *zi12* is accumulated in parallel with *zr12*. All operations for accumulating *zi12* in this accumulator remains same as for *zr12* in its accumulator with the input/output lines *zr12*, *indor2*, *oacr2, indexr2* replaced by *zi12*, *indoi2*, *oaci2*, *indexi2* respectively and all other input/output lines to these accumulators are common to both.

The common *reade* signal to all the above accumulators, when high initiates reading from the four accumulators in parallel.

5. modxyf1024:

Note: The layout for modxyf1024 is similar to the layout sown in fig. B.4 with the complex_multiplier1024 responsible to cross multiply X and Y to produce real and imaginary parts of Z is replaced by power multiplier to produce X and Y powers correspondingly.

Inputs:

clk: 128 MHz clock, *clk* of *combunitfft1*.
reset: 1 bit active high control signal to start *modxyf1024, decout_acmin* of *combunitfft1*.
reade: 1 bit active high control signal to start reading from the constituting accumulators, *reade* of *combunitfft1*.
blank: 1 bit active high user control signal to pause accumulations for certain duration when *modxyf1024* is in running condition. This duration refers to the time required to change the pointing of telescope to the source from the calibrator, *blank* of *combunitfft1*.
xr and xi: Real (*rNpn* of *combunitfft1*) and imaginary (*iNmn* of *combunitfft1*) parts of *X* polarization (representing *X(F, K)*) from *decodervar1* respectively. Each is 22 bit wide.
yr and yi: Real (*iNpn* of *combunitfft1*) and imaginary (*rNmn* of *combunitfft1*) parts of *Y* polarization (representing *Y(F, K)*) from *decodervar1* respectively. Each is 22 bit wide.
Indexi: 10 bit index of samples in *xr*/*xi*/*yr*/*yi, indimo* of *combunitfft1.*

Outputs:

oacmx1, oacmx2, oacmy1, oacmy2: 69 bit output lines from the accumulators accumulating $|X(F,K)|^2$ corresponding to noise diode on state, $|X(F,K)|^2$ corresponding to noise diode off state, $|Y(F,K)|^2$ corresponding to the noise diode on state and $|Y(F,K)|^2$ corresponding to noise diode off state respectively, *acmx1*, *acmx2*, *acmy1*, *acmy2* respectively of *combunitfft1*.
indexr1, indexr2, indexi1, indexi2: 10 bit output lines representing indices of data in *oacmx1*, *oacmx2*, *oacmy1* and *oacmy2* respectively, *indx1*, *indx2*, *indy1*, *indy2* respectively of *combunitfft1*.

Component modules:

cscombf1024n: It is started by *reset*. It generates two signals *reset3* (1 bit) and *reset4 (1 bit).* The signal *reset3* goes high one clock pulse after *reset* goes high and remains high for 1048576×1024 + 2 clock pulses after which it goes low. The signal *reset4* goes high two clock pulses before *reset3* goes low and it remains high for1048576×1024 + 2 clock pulses after which it goes low.

mod_xy_square: It is started by *reset.* Its inputs *xr*, *xi*, *yr*, *yi* and *indexi* start arriving one clock pulse after *reset* goes high. It forms $xr^2 + xi^2$ power (named as *zr1*) and $yr^2 + yi^2$ power (named as *zi1*)where $xr^2 + xi^2$ (45 bit) represents $|X(F, K)|^2$ and $yr^2 + yi^2$ (45 bit) represents $|Y(F, K)|^2$. Both $xr^2 + xi^2$ and $yr^2 + yi^2$ are obtained in parallel at its outputs three clock pulses after *reset* goes high that is the input-output latency is two clock pulses. The output index *indor* (obtained from *indexi*) is transferred in parallel with $xr^2 + xi^2$ and the output index *indoi* (obtained from *indexi*) is transferred in parallel with $yr^2 + yi^2$. This block also takes in the input *blank*, which when active transfers zeros in the output lines.

dff1: This element delays a 1 bit signal by one clock pulse. There are 3 such elements two of which are used to delay *reset* by two clock pulses to produce *reset2* (1 bit). The *reset* delayed by one clock pulse is *reset1*.

clockdividerac30: This element combined with the third *dff1* generates a signal *sel2* (1 bit). It is started by *reset2* which goes high one clock pulse after *reset3* goes high. Counting from the next clock pulse after *reset2* goes high, *sel2* toggles every 1048576×1024 clock pulses. In the next clock pulse after *reset2* goes high, *sel2* starts with '0'.

demux4ac1024: There are two such units working in parallel, one for transferring *zr1* ($xr^2 + xi^2$) another for transferring *zi1* ($yr^2 + yi^2$). Since the operation of both are identical I will go into the details of only one that is the one for transferring *zr1* as follows: It receives *zr1* and *indor* (index of *zr1*) from the *mod_xy_square*. It is started by *reset1*. If *reset1* is '1' then if *sel2* is '0' then the output lines *zr11* and *indor1* get the inputs *zr1* and *indor* respectively and the output lines *zr12* and *indor2* receive zeros (*zr11* receive on state $xr^2 + xi^2$). If *reset1* is '1' and if *sel2* is '1' then the output lines *zr12* and *indor2* get the inputs *zr1* and *indor* respectively (*zr12* receives off-state $xr^2 + xi^2$) and the output lines *zr11* and *indor1* receive zeros. The data arrive 2 clock pulses after *reset1* goes high and the output is received the same clock pulse the input arrives the logic being combinatorial. In the other unit receiving *zi1*, all the above description is applicable with *zr1*, *indor*, *zr11*, *zr12*, *indor1*, *indor2* replaced by *zi1*, *indoi*, *zi11* (on-state $yr + yi^2$), *zi12* (off-state $yr + yi^2$), *indoi1* and *indoi2* respectively. The remaining input/output signal lines are common to both.

acm1024: There are four such units of which two work to accumulate *zr11* and *zr12* respectively and the other two work to accumulate *zi11* and *zi12* respectively. Accumulation process of *zi11* and *zi12* is same as accumulation process of *zr11* and *zr12* respectively, which also means *zi11* and *zr11* (*zi12* and *zr12*) are accumulated in parallel. So I will only go into the details of accumulation of *zr11* and *zr12*.

Accumulation of *zr11* in one *acm1024*: *zr11* and *indor1* starts appearing two clock pulses after *reset3* goes high. The signal *reset3* is the write enable to initiate writing in this accumulator. One clock pulse after *zr11* appears, it gets added to the contents in the location pointed by *indor1* (representing *K* of *zr11* and of the location where *zr11* enters). Consecutive *zr11* get added to the contents of the corresponding location (pointed by *indor1*) one by one as described under topic 3 of B.1.3. Since I need to accumulate 1048576 spectra, *reset3* must remain high for 1048576 × 1024 (number of samples entering the accumulator) + 2 (delay between *reset3* going high and *zr11* appearing as already mentioned) = 1073741826 clock pulses as it does (see cscombunitf1024n). So writing in this accumulator stops when *reset3* goes low. The signal *reade* is there, which when active enables reading in the output line *oacmx1* and the index appearing in the output line *indexr1*.

Accumulation of *zr12* in another *acm1024*: It is started by *reset4*. The signal *zr12* arrives two clock pulses after *reset4* goes high (at the same time when *reset3* goes low-two clock pulses after *reset4* goes high). This is also when *sel2* has toggled to 1 sending output in *zr12*. So *zr12* is written in this accumulator following the same way as *zr11* has been written with *indor2* pointing the location for writing in this accumulator. So *reset4* (as *reset3*) also remains high for $1048576 \times 1024 + 2$ clock pulses to accumulate 1048576 spectra. The signal *reade* is there, which when active enables reading in the output line *oacmx2* and the index appearing in the output line *indexr2*.

Accumulation of *zi11* in a third *acm1024*: *zi11* is accumulated in parallel with *zr11*. All operations for accumulating *zi11* in this accumulator remain the same as for *zr11* in its accumulator with the input/output lines *zr11*, *indor1*, *oacmx1*, *indexr1* replaced by *zi11*, *indoi1*, *oacmy1*, *indexi1* respectively and all other input/output lines to these accumulators are common to both.

Accumulation of *zi12* in the fourth *acm1024*: *zi12* is accumulated in parallel with *zr12*. All operations for accumulating *zi12* in this accumulator remain the same as for *zr12* in its accumulator with the input/output lines *zr12*, *indor2*, *oacmx2*, *indexr2* replaced by *zi12*, *indoi2*, *oacmy2*, *indexi2* respectively and all other input/output lines to these accumulators are common to both.

The common *reade* signal to all the above accumulators, when high initiates reading from the four accumulators in parallel.

6. dff1: This is a delay flip flop to delay *reade* by one clock pulse to yield *sreade*.

7. *aon_aoff:* There are four such units for the four quantities $Z_r(F,K)$, $Z_i(F,K)$, $|X(F,K)|^2$, $|Y(F,K)|^2$. The inputs corresponding to the four quantities are *acmr1*, *acmi1*, *acmx1*, *acmy1* respectively from noise diode on-state accumulators and *acmr2*, *acmi2*, *acmx2*, *acmy2* respectively from off-state accumulators. The indices of the four quantities input to these units are *indexr1*, *indexi1*, *indx1*, *indy1* respectively. Each of these units performs subtraction between the on-state accumulation results and off-state accumulation results. The output after subtraction comes out two clock pulses after *sreade* goes high. The signal *sreade* is connected to the output signal *weoutr/ weouti/ weoutx/ weouty* (each for one of the four quantities). So the outputs appear as *oacr/ oaci/ oacmx/ oacmy* with indices *indexr/ indexi/ indexx/ indexy* two clock pulses after *weoutr/ weouti/ weoutx/ weouty* goes high.

B.1.4 Combination of eight identical stages

Each Z_r , Z_i , $|X|^2$ and $|Y|^2$ from the accumulators of 8 identical stages (of FFT to accumulators in B.1.3) are further accumulated in a single accumulator to yield four final accumulated quantities $Z_r(K)$, $Z_i(K)$, $|X(K)|^2$ and $|Y(K)|^2$ respectively. The parameter F is missing since it has no meaning after accumulation. Since the accumulation process is described once I will see the detailed operation that is accumulation of 8 stages for each quantity under VHDL description of this stage only.

B.1.4.1 VHDL implementation

Module *dout2acmf* (from serial frame generator to accumulators)*:*

Inputs:

clk: 128 MHz input clock.
ce: 1 bit user control signal to start *dout2acmf*.
X/Y: Represents a group of 8 parallel input lines getting data from the 8 parallel output lines from the 'clock rate reduction' logic corresponding to *X/Y* polarizations. The line numbers are mapped 1 to 1 with those of the previous block. Each line receives one sample and is configured with 10 bit. These are also the inputs to *doutf2*.
blank: Same signal (1 bit user defined) in *combunitfft1* also passed to *combunitfft2* to ***combunitfft8.***

Outputs:

oacr, oaci: 69 bit outputs from the final accumulators accumulating $Z_r(F,K)$ (on-state-off-state) from 8 stages and $Z_i(F,K)$ (on-state-off-state) from 8 stages respectively.
oacmx, oacmy: 69 bit outputs from the final accumulators accumulating $|X(F,K)|^2$ (on-state-off-state) from 8 stages and $|Y(F,K)|^2$ (on-state-off-state) from 8 stages respectively.
indexr, indexi: 10 bit indices of data in *oacr* and *oaci* respectively.
Indexx, indexy: 10 bit indices of data in *oacmx* and *oacmy* respectively.

Component modules:

1. combunitfft1 to combunitfft8: As already manifested that each of the eight identical stages receive a pair of inputs from the serial frame generator (*doutf2*) corresponding to *X* and *Y* polarization. First pair of output lines from *doutf2* corresponding to *X* and *Y* polarizations is received by *combunitfft1*, second pair by *combunitfft2* and so on. Two clock pulses after the first stage starts *doutf2* is started. The data appears at the inputs of the FFT of the first stage 6 clock pulses after the start of the first stage or 4 clock pulses after *doutf2* starts. Other stages follow the same process with a delay of 128 clock pulses between the start of two consecutive stages with a stage delayed by 128 clock pulses with respect to the previous stage. In the topmost module *dout2acmf*, *combunitfft1* has the input/output names as *clk, ce0, blank, rout0, imout0, oacr0, oaci0, oacmx0, oacmy0, indexr0, indexi0, indexx0, indexy0, weoutr0, weouti0, weoutx0, weouty0, sstart0, sfwd_inv_we0* respectively as compared to the input/output names given under *combunitfft1* in B.1.3.1. The second stage *combunitfft2* has the corresponding names with 0 replaced by 1 in the R.H.S of the names in *combunitfft1*. The third stage has 0 replaced by 2, fourth stage has 0 replaced by 3 and so on. The rest are common to all the 8 stages. The reading of accumulators for a particular quantity among $Z_r(F,K)$ (on-off state), $Z_i(F,K)$ (on-off state), $|X(F,K)|^2$ (on-off state), $|Y(F,K)|^2$ (on-off state) for 8 stages takes place consecutively that is reading of second stage starts two clock pulses after reading of first stage and so on.

2. cufftdout: It is started by *ce0*. A signal *resdout* is generated two clock pulses after *ce0* goes high. The signals *ce1* to *ce8* are generated with *ce1* delayed by 128 clock pulses as compared to *ce0*, *ce2* delayed by 128 clock pulses as compared to *ce1* and so on.

3. doutf2: The details of this block is already given in B.1.2.2. The corresponding input/output lines are named as *clk, resdout, X/Y, rout0 - rout7/imout0 - imout7* respectively.

4. mux4ac8: There are four such units each for transferring one of the four quantities $Z_r(F,K)$ (on-off state), $Z_i(F,K)$ (on-off state), $|X(F,K)|^2$ (on-off state), $|Y(F,K)|^2$ (on-off state) from all 8 stages (*combunitfft1* to *combunitfft8*) consecutively. I will only go into the details of transferring $Z_r(F,K)$ (on-off state) from all the 8 stages that is transferring *oacr0* to *oacr7* and the transfer of the rest viz. *oaci0* to *oaci7, oacmx0* to *oacmx7, oacmy0* to *oacmy7* will not be described as they follow the same process. So the *mux4ac8* for $Z_r(F,K)$ (on-off state) generates a signal *selo1*, which goes high when any of the signals from *weoutr0* to *weoutr7* is high and otherwise *selo1* is low. The output data line from this block receives data from the lines *oacr0* to *oacr7*. The selection between *oacr0* to *oacr7* to enter the output line is made by checking the high state of *weoutr0* to *weour7* respectively. Thus the output data line always receives the data two clock pulses after the corresponding signals *weoutr0* to *weour7* is high. The output data line is named as *doutr*. The indices of the data are transferred in parallel from the lines *indexr0* to *indexr7* to the line *waddr*. The output data and indices of the rest three units are *douti*, *doutx*, *douty* and *waddi*, *waddx*, *waddy* with *selo1* replaced by *selo2, selo3, selo4* respectively. The four units work in parallel.

5. readacm: This block generates a signal *reade8ac* two clock pulses after *weoutr7/ weouti7/ weoutx7/ weouty7* goes low as then writing in the final accumulators stop. The status of *weoutr7* is only checked to generate this signal.

6. acm10248plus: There are four such units working in parallel to accumulate *doutr, douti, doutx* and *douty* respectively with write enables *selo1, selo2, selo3* and *selo4* respectively. The write addresses of the four units are *waddr, waddi, waddx* and *waddy* respectively. The accumulation process is the same as *acm1024* under *xacmfl1024* or *modxyf1024*. The signal *reade8ac* starts reading from the four accumulators in parallel to pass the outputs or contents of the accumulators to the next stages. The four output lines are *oacr, oaci, oacmx* and *oacmy* respectively with indices *indexr, indexi, indexx* and *indexy* respectively.

B.1.5 Equalization parameters for formation of circular polarization

Now I will go into the details of how the phase and gain equalization parameters and window function are obtained and how they are then applied to the real time *X* and *Y* polarization channels. The outputs from the stage B.1.4 (the VHDL module *dout2acmf*) and the outputs from the stage B.1.3 block decoder (VHDL module *decodervar1* (two of them)) are the inputs to this stage. I will also describe the formation of circular polarization. In this section first I will describe the window function then rotation parameters and then gain parameters and finally formation of circular polarization.

1. Window function

I have already described the accumulation of cross power spectra and power spectra of channel *X* and *Y* in section B.1.4. The real and imaginary parts of cross power that are $Z_r(F,K)$ and $Z_i(F,K)$ and two power spectra that are $|X(F,K)|^2$ and $|Y(F,K)|^2$ where *F* is the frame number and *K* is the channel number, gets accumulated in four separate accumulators. Let the outputs from the four final accumulators of the stage B.1.4 (module *dout2acmf)* be $Z_r(K)$, $Z_i(K),$, $|X(K)|^2$ and $|Y(K)|^2$ respectively since after accumulation the parameter *F* does not exist the frames being averaged together. The window function is produced from the absolute value of the accumulated cross power that is mentioned above and it does so as described in detail in the next paragraph; the window function has zero magnitude where the magnitude of the cross power is less than one quarter of the maximum magnitude in the band of cross power and the window function has a magnitude one where the magnitude of the cross power is more than or equal to one quarter of the maximum magnitude in the band of cross power; it is generated as described below.

Each of the quantities $Z_r(K)$, $Z_i(K)$, , $|X(K)|^2$ and $|Y(K)|^2$ is the result of accumulation of 8 stages (on-off state for each stage). The window function gets the parallel inputs $Z_r(K)$ and $Z_i(K)$ from *dout2acmf* in two data lines with the value of *K* for each of the two parallel data lines incrementing by 1 per clock pulse as usual representing a new sample. Each of the quantities $Z_r(K)$ and $Z_i(K)$ is written in a buffer having 1024 locations corresponding to 1024 values of *K* at 128 MHz clock rate consecutively by writing one sample per clock pulse starting from the location having address 0 pointed by the value of *K* (*K*= 0). $Z_r(K)$ and $Z_i(K)$ enters the location pointed by *K* and hence it takes 1024 clock pulses to write 1024 samples of $Z_r(K)$ and $Z_i(K)$. Note that a sample enter the buffer one clock pulse after the sample arrives in the data line. Writing in this pair of first buffers take place at 128 MHz clock rate in parallel.

After writing of $Z_r(K)$ and $Z_i(K)$ are finished, these buffers are ready to be read out and the reading starts after a certain number of clock pulses. Reading from these buffers occurs in parallel at 64 MHz clock rate and each buffer sends out samples serially at 64 MHz starting from the sample in the location with address 0 and going down to address 1023 by just incrementing the read address by 1 per clock pulse. Reading happens one clock pulse after the read address appears to point to the location for reading. Note at this point that the process of writing in a buffer and reading from a buffer using the value of *K* for pointing to the location to be written or read out and the timing between arrival of data in its data line and writing in a location or between pointing a location to be read and data appearing at the output line remains the same all the time; so this will not be described in future (this though has been described in previous logic blocks like in B.1.3: decoder). Thus $Z_r(K)$ and $Z_i(K)$ are passed in parallel to the next stage to form $|Z(K)|^2$ at the rate of 64 MHz.

The resulting $|Z(K)|^2$ is written in another buffer at 64 MHz clock rate. After writing is finished in this second buffer, the maximum among all the entered values is determined by comparing them. The comparison is done at 64 MHz clock rate. For comparison a temporary register is taken holding the value 0 initially. The content of this register is compared with the stored $|Z(K)|^2$ consecutively starting from *K* = 0 with $|Z(K)|^2$ replacing the content of the register if $|Z(K)|^2$ is greater than the present content of the register. Thus after determining the maximum among all the entered $|Z(K)|^2$, again the stored $|Z(K)|^2$ is compared with this obtained maximum starting from *K* = 0 and if the $|Z(K)|^2$ is greater than or equal to the 1/4th of the found maximum then a 1 is returned at the output *W(K)* (window function) else if $|Z(K)|^2$ is less than 1/4th of the maximum found then a 0 is returned to *W(K)*. Thus *W(K)* is either 0 or 1 and is passed serially for *K* = 0 to *K* =1023 through its data line at 64 MHz clock rate. The serially arriving *W(K)* is written into another buffer having 1024 locations for 1024 values of *K* and thus is latched and ready to be read at 128 MHz clock rate when required.

2. Rotation parameters:

The rotation parameters function gets the parallel inputs $Z_r(K)$ and $Z_i(K)$ from the stage B.1.4 (module *dout2acmf)*. Individual $Z_r(K)$ or $Z_i(K)$ are transferred serially; I have discussed until now many times that value of *K* in any data line increases by 1 per clock pulse representing a new sample of the quantity being transferred in the data line; in future I will not further demonstrate serial transfer of any quantity in a data line as that is evident by now. Each of the quantities $Z_r(K)$ and $Z_i(K)$ is written in a buffer having 1024 locations for 1024 values of *K*. After writing of $Z_r(K)$ and $Z_i(K)$ at 128 MHz clock rate are finished, these buffers are ready to be read out and the reading starts after a certain number of clock pulses.

Reading from these buffers occur in parallel at 64 MHz clock rate. Thus $Z_r(K)$ and $Z_i(K)$ are passed in parallel to the next stage to form $|Z(K)|^2$ at the rate of 64 MHz. Then this arriving $|Z(K)|^2$ is converted to the floating point format and let us call that $|Z(K)|^2_{float}$, which is passed further on. Note that conversion to floating point format and square root and division operations described next use Xilinx floating point IP core. Square root of $|Z(K)|^2_{float}$ is taken to obtain $|Z(K)|_{float}$. Similarly $Z_r(K)$ and $Z_i(K)$ are also converted to floating point format to produce $Z_r(K)_{float}$ and $Z_i(K)_{float}$ respectively, which are passed in parallel further on. It is evident that it takes more time to obtain $|Z(K)|_{float}$ than to obtain $Z_r(K)_{float}$ and $Z_i(K)_{float}$. However, for our purpose these three quantities should arrive in parallel and hence $Z_r(K)_{float}$ and $Z_i(K)_{float}$ are delayed equally by the difference in the number of clock pulses between their generation and generation of $|Z(K)|_{float}$. Then the divisions $Z_r(K)_{float} \div |Z(K)|_{float} = \cos\theta(K)_{float}$ and $Z_i(K)_{float} \div |Z(K)|_{float} = \sin\theta(K)_{float}$ are performed in parallel to obtain the rotation parameters shown in the matrix of eqn. 2.30 (θ is the phase difference between the *X* and *Y* channels).

Then $\cos\theta(K)_{float}$ and $\sin\theta(K)_{float}$ are converted to fixed point representation $\cos\theta(K)$ and $\sin\theta(K)$ respectively; the conversion of the two quantities happen in parallel. The floating point conversion of the quantities $Z_r(K)$, $Z_i(K)$ and $|Z(K)|^2$ were carried out since the square root and division operations require the usage of the Xilinx floating point IP core, which takes floating point inputs and produces floating point outputs. All floating point operations are carried out at 64 MHz clock rate. Each of the quantities $\cos\theta(K)$ and $\sin\theta(K)$ is written in a buffer having 1024 locations at 64 MHz clock rate for 1024 values of *K*. Thus 1024 values of each $\cos\theta(K)$ and $\sin\theta(K)$ are latched in the respective locations of a buffer to be read when required at 128 MHz clock rate.

3. Gain parameters:

The gain parameters function gets the parallel inputs $|X(K)|^2$ and $|Y(K)|^2$ from the stage B.1.4 (module *dout2acmf)*. Each of the quantities $|X(K)|^2$ and $|Y(K)|^2$ is written in a buffer having 1024 locations for 1024 values of *K*. Writing in this pair of first buffers take place at 128 MHz clock rate. After parallel writing of $|X(K)|^2$ and $|Y(K)|^2$ are finished in their respective buffers, the maximum among all the entered values of each $|X(K)|^2$ and $|Y(K)|^2$ is determined by comparing them. The comparison is done at 64 MHz clock rate. For comparison a temporary register is taken in each of the two cases for $|X(K)|^2$ (case 1) and $|Y(K)|^2$ (case 2) holding the value 0 initially. In case1 the content of the register is compared with the stored $|X(K)|^2$, or in case 2 with the stored $|Y(K)|^2$ consecutively starting from *K* = 0 with $|X(K)|^2$ (in case 1) or $|Y(K)|^2$ (in case 2) replacing the content of the register if $|X(K)|^2$ (in case 1) or $|Y(K)|^2$ (in case 2) is greater than the present content of the register.

Thus after determining the maximum say $P1_{max}$ or $P2_{max}$. among all the values of $|X(K)|^2$ or $|Y(K)|^2$ in case 1 or case 2, the two buffers are ready to be read out and the reading starts after a certain number of clock pulses. Reading from these buffers occurs at a rate of 64 MHz clock rate. Thus $|X(K)|^2$, $|Y(K)|^2$, $P1_{max}$, $P2_{max}$ are read and passed in parallel to a stage where $P1_{max}$ and $P2_{max}$ are compared and whichever is greater is transferred to the output terminal of this stage and renamed as P_{max} at the output terminal; the inputs $|X(K)|^2$ and $|Y(K)|^2$ to this stage are also transferred in parallel to P_{max} through other two output lines; in this stage it is checked if $|X(K)|^2$ or $|Y(K)|^2 = 0$ as then a 1 is transferred to the corresponding output line for that value of *K*; each of the three parallel outputs from this stage P_{max} , $|X(K)|^2$ and $|Y(K)|^2$ is sent to one of the three similar stages to convert from fixed point representation to floating point representation using Xilinx floating point IP core. The outputs from the three parallel and similar stages for converting from fixed point to floating point formats are $P_{maxfloat}$, $|X(K)_{float}|^2$ and $|Y(K)_{float}|^2$. These outputs are paired as $P_{maxfloat}$, $|X(K)_{float}|^2$ and $P_{maxfloat}$, $|Y(K)_{float}|^2$ for next two parallel and identical stages; one stage performs $P_{maxfloat} \div |X(K)_{float}|^2$ to produce $g_x^2(K)_{float}$ and the other stage performs $P_{maxfloat} \div |Y(K)_{float}|^2$ to produce $g_y^2(K)_{float}$; each of these two parallel quantities $g_x^2(K)_{float}$ and $g_y^2(K)_{float}$ is transferred to one of the two parallel and identical stages performing square root operation of the input quantity. So at the outputs of these two stages (which perform square root operation of the input quantity) I obtain $g_x(K)_{float}$ and $g_y(K)_{float}$ respectively.

Now each $g_x(K)_{float}$ and $g_y(K)_{float}$ is transferred to one of the two identical and parallel next stages that convert from floating point representation to fixed point representation and at the outputs of these two stages I obtain $g_x(K)$ and $g_y(K)$ respectively, which are fixed point quantities. The quantities $g_x(K)$ and $g_y(K)$ are obtained in parallel. The floating point conversion of the quantities were carried out since the square root and division operations require the usage of the Xilinx floating point IP core, which takes floating point inputs and produces floating point outputs. All floating point operations are carried out at 64 MHz clock rate. Each of the quantities $g_x(K)$ and $g_y(K)$ is written in a buffer having 1024 locations at 64 MHz clock rate. Thus 1024 values of each $g_x(K)$ and $g_y(K)$ are entered in the respective locations of a buffer being latched and to be read when required at 128 MHz clock rate.

4. Formation of circular polarization

From this point again real time operation starts with a clock frequency of 128 MHz. After obtaining the equalization parameters as stated above, I now divide into 8 stages each receiving inputs from one of the 8 decoders from 8 identical stages of FFT to accumulators (described in B.1.3, VHDL modules from combunitff1 to combunitfft8 respectively representing 8 identical stages). These 8 stages work with a delay of 128 clock pulses between any two consecutive stage starting from the first one as is the case in our design. I will only discuss one of the 8 identical stages. The stage receives $X_r(F,K)$, $X_i(F,K)$, $Y_r(F,K)$ and $Y_i(F,K)$ in parallel from the corresponding decoder (thus these *X* and *Y* are the ones acquired during observation). Each of the equalization parameters $\cos\theta(K)$, $\sin\theta(K)$, $g_x(K)$ and $g_y(K)$ is read from their latches at a rate of 128 MHz to be multiplied with *W(K),* which is also read in parallel, from its latch at a rate of 128 MHz, with the equalization parameters, to obtain windowed equalization parameters $\cos\theta_w(K)$, $\sin\theta_w(K)$, $g_{xw}(K)$ and $g_{yw}(K)$ respectively. After multiplication each of these quantities are entered in a buffer having 1024 locations for 1024 values of *K*. Thus 1024 values of each of these quantities are stored in a buffer and hence there are four buffers holding the four quantities. The inputs $X_r(F,K)$, $X_i(F,K)$, $Y_r(F,K)$ and $Y_i(F,K)$ are delayed in such a way and the quantities $\cos\theta_w(K)$, $\sin\theta_w(K)$, $g_{xw}(K)$ and $g_{yw}(K)$ are read from their buffers in such time that all the eight quantities $X_r(F,K)$, $X_i(F,K)$, $Y_r(F,K)$, $Y_i(F,K)$, $\cos\theta_w(K)$, $\sin\theta_w(K)$, $g_{xw}(K)$ and $g_{yw}(K)$ are aligned in terms of *K* that is all are parallel. These parallel quantities are then passed to a stage to equalize phase and gain of *X* and *Y* channels and form circular polarization. To accomplish phase and gain equalization and formation of circular polarization, following operations are performed in a stage:

$$X_{rnew}(F,K)=g_{xw}(K)\times X_r(F,K) \tag{B.5}$$

$$X_{inew}(F,K)=g_{xw}(K)\times X_i(F,K) \tag{B.6}$$

$$Y_{rnew}(F,K)=g_{yw}(K)\times Y_r(F,K)\times\cos\theta_w(K)-g_{yw}(K)\times Y_i(F,K)\times\sin\theta_w(K) \tag{B.7}$$

$$Y_{inew}(F,K)=g_{yw}(K)\times Y_r(F,K)\times\sin\theta_w(K)+g_{yw}(K)\times Y_i(F,K)\times\cos\theta_w(K) \tag{B.8}$$

The quantities in the left hand side of eqs (B.5) to (B.8) are the phase and gain equalized real and imaginary parts of X and Y polarizations respectively and are obtained in parallel. The following operations are performed using these quantities to form circular polarization.

$$LHC_r(F,K)=X_{rnew}(F,K)+Y_{inew}(F,K) \tag{B.9}$$

$$LHC_i(F,K)=X_{inew}(F,K)-Y_{rnew}(F,K) \tag{B.10}$$

$$RHC_r(F,K)=X_{rnew}(F,K)-Y_{inew}(F,K) \tag{B.11}$$

$$RHC_i(F,K)=X_{inew}(F,K)+Y_{rnew}(F,K) \tag{B.12}$$

where LHC_r , LHC_i , RHC_r and RHC_i are real and imaginary parts of *LHC* and *RHC* respectively and are obtained in parallel.

B.1.5.1 VHDL implementation

Now I will go into the VHDL implementation of B.1.5.

Module *circularacm2plustest*:

Inputs:

clock: 128 MHz input clock.
reset: 1 bit control signal to start *circularacm2plustest.*
blank: 1 bit control signal which when active pauses accumulation by transferring zeros.
rez, imz: 31 bit inputs representing *oacr, oaci* (38 LSBs truncated) from *acm10248plus* of *dout2acmf* (B.1.4.1) respectively.
addressin: 10 bit indices of *rez* and *imz* (same as *indexr* of *acm10248plus* of *dout2acmf*).
modx2, mody2: 69 bit inputs representing *oacmx, oacmy* from *acm10248plus* of *dout2acmf* (B.1.4.1) respectively.
sindx: 10 bit index of *modx2* and *mody2* (same as *indexx* of *acm10248plus* of *dout2acmf*).

Outputs:

LHCreal1-LHCreal8, LHCimag1-LHCimag8: 8 output lines each 48 bit wide representing real and imaginary parts of LHC respectively.
RHCreal1-RHCreal8, RHCimag1-RHCimag8: 8 output lines each 48 bit wide representing real and

imaginary parts of RHC respectively.
Indexout1-indexout8: 8 output lines representing 10 bit indices of *LHCreal1-LHCreal8* or *LHCimag1-LHCimag8* or *RHCreal1-RHCreal8* or *RHCimag1-RHCimag8* respectively.

Component modules:

1. signalreade: It is started by *reset*. It generates a signal *reade* (1 bit) 8000 clock pulses after *reset* goes high. The signal *wenable* (1 bit) is generated 3 clock pulses after *reade* is high. This *wenable* signal remains high for 1026 clock pulses after which it goes low.

2. zwindowtop1nc* (refer to fig. B.5)*:

Inputs:

clock: 128 MHz input clock. The *clock* of *circularacm2plustest*.
reset: 1 bit control signal to start *zwindowtop1nc*. The *reset* of *circularacm2plustest*.
reade: 1 bit control signal to start reading from the last stage buffer of this block. The *reade* of *circularacm2plustest*.
rez, imz: 31 bit inputs representing the *rez, imz* of *circularacm2plustest* respectively.
addressin: 10 bit index of *rez* and *imz*. The *addressin* of *circularacm2plustest.* Not used.

Outputs:

Q2: 2 bit output from the last stage buffer of this block. A signal *swinfunc* of *circularacm2plustest.*
index: 10 bit index of *Q2*. A signal *sindexwin* of *circularacm2plustest.*

Component modules:

clockdiv2: It takes in the clock *clock* (128 MHz) and generates a 64 MHz clock named *clock1*.

zwindow1csc: It generates a signal *enable* (1 bit) 1030 *clock1* pulses after another signal *enableo* (1 bit) generated in the next block goes high. This signal *enable* remains high for 1026 *clock1* pulses after which it goes low.

zwindow1* (refer to fig. B.6)*:

Inputs:

rez, imz: The *rez, imz* of *zwindowtop1nc* respectively.
clock: 128 MHz input clock.
clock1: 64 MHz input clock. The *clock1* of *zwindowtop1nc.*
reset: 1 bit control signal to start *zwindow1*. The *reset* of *zwindowtop1nc*.
enable: 1 bit control signal. The *enable* of *zwindowtop1nc.*
reade: 1 bit control signal to start reading from the last stage buffer of this block. The *reade* of *zwindowtop1nc*.

Outputs:

enableout: 1 bit output signal same as *enableo* of *zwindowtop1nc* to control generation of *enable*.

Q2: 2 bit output from the last stage buffer representing the window function. The *Q2* of *zwindowtop1nc*.
index: 10 bit index of *Q2*. The *index* of *zwindowtop1nc*.

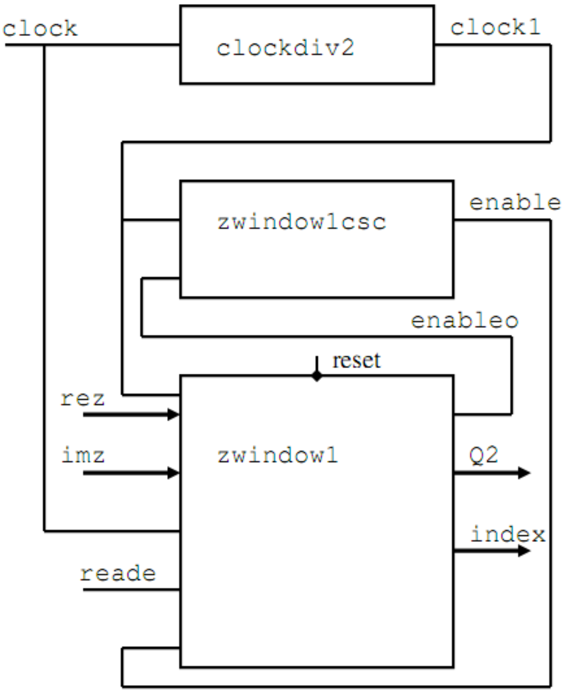


Fig. B.5: The top module *zwindowtop1nc* for window function. This logic receives the inputs from the *dout2acmf* block, which are the final accumulated real and imaginary parts of the *Z* spectrum named as *rez* and *imz* respectively in the figure. The block *zwindow1* generates the window function to be latched and read in the line *Q2* when required with *index* line representing the index of *Q2*. The signal lines to the left of the blocks and on the top of the blocks in the figure represent inputs to those blocks and the signal lines in the RHS of the blocks represent the outputs from those blocks.

Component modules:

testcs1n1024: It is started by *reset*. It generates the signals *sreset1* (1 bit), *sreade* (1 bit) and *senable* (1 bit). The signal *sreset1* goes high with *reset* and remains high for 1025 clock pulses after which it goes low. 3 *clock1* pulses after *sreset1* is low, *sreade* goes high and remains high forever. One *clock1* pulse after *sreade* is high, *senable* goes high and remains high forever. The signal *senable* is *enableout*.

windowarraygtop2: There are two such elements working in parallel one for holding and transferring *rez* and the other for holding and transferring *imz*. Each consists of a buffer having 1024 locations each of which is 31 bit wide. I will only discuss the one for *rez*. The signal *sreset1* enables writing in the buffer. One clock pulse after *sreset1* goes high, *rez* appears at its inputs and one more clock pulse later writing of *rez* in the buffer starts with 1024 values of *rez* entering 1024 consecutive locations starting from the first location. In 1025 clock pulses after *sreset1* goes high, writing in the buffer finishes. Reading from the buffer is enabled by *sreade* and the data from the buffer is transferred consecutively starting from the first location to the output line *s3* (31 bit) 2 *clock1* pulses after *sreade* is high at a rate of 64 MHz (reading is triggered by *clock1*). The output line representing the index of *s3* is kept open as that is a redundant signal when index of *s4* (corresponding signal of *s3* of the other buffer for *imz*) named as *addressin* (10 bit) is passed. All the signals for the other buffer remain the same except *rez*, *s3* and *open* (index of *s3*) replaced by *imz*, *s4* and *addressin* respectively.

squaremz1: It is started by *senable*. The input clock is *clock1*. The signals *s3, s4* and *addressin* are the inputs to this block, which arrive 1 *clock1* pulse after *senable* goes high. It forms $rez^2 + imz^2$ that is $|Z|^2$, which is transferred to its output line *s2* (64 bit) with the index of *s2* appearing in the output line *s1* (10 bit) in parallel to *s2*. The input-output latency is 3 *clock1* pulses.

cs31024: It is started by *senable* with input clock *clock1*. It generates a signal *reset1* (1 bit) 3 *clock1* pulses after *senable* goes high. This signal *reset1* remains high for 1025 *clock1* pulses after which it goes low.

acmwin1: It consists of two buffers each having 1024 locations (66 bit wide). Writing in the first buffer is enabled by *reset1* and *s2* and *s1* arrive at its inputs one *clock1* pulse after *reset1* is high. The signal *s2* (changed to 66 bit signed Boolean) is written in the location pointed by *s1* in the first buffer consecutively starting from the first location; this writing starts two *clock1* pulses after *reset1* goes high. It takes 1025 *clock1* pulses to write *s2* in this buffer after *reset1* goes high. The signal *enable,* which is generated 1030 *clock1* pulses after *senable* goes high is then used to trigger finding the maximum among all the entered *s2* in the buffer. The maximum is found in the following way: A temporary register of 66 bit is taken initialized to 0 and in the next *clock1* pulse after *enable* is high, its value is compared with the value in the first location of the buffer containing *s2.* If the value of the data in the first location is greater than the contents of the register then the register gets the data from the first location else it keeps its previous value thus updating of the register after comparison is done. In the next *clock1* pulse after comparing with first location and updating the register, its value is compared with the value in the second location of the buffer containing *s2*. If the value of the data in the second location is greater than the present contents of the register then the register gets the data from the second location else it keeps its previous value and so on and each time the register content is updated to get the maximum after all 1024 comparisons are finished. The signal *enable* remains high for 1026 *clock1* pulses enough to find the maximum (it takes 1024 *clock1* pulses to find the maximum). In the 1025th *clock1* pulse after *enable* is high this buffer is triggered to be read.

Reading happens consecutively starting from the first location of the buffer at 64 MHz. The output line (2 bit) of the buffer receives a value of 1 if the read data is greater than or equal to 1/4th of the found maximum and receives a 0 if the read data is less than 1/4th of the found maximum. This 2 bit output data is written in another buffer (last) having 1024 locations each 2 bit wide at 64 MHz. Writing in this last buffer is triggered by a signal which goes high one *clock1* pulse after the previous buffer is triggered to be read as the 2 bit output data from the previous buffer arrives two *clock1* pulses after the buffer is triggered to be read. This signal triggering the writing in this buffer remains high for 1025 *clock1* pulses as required to fill in the buffer. Then the buffer is ready to be read at the rate of 128 MHz using the clock *clock* with the output line *Q2* sending the read data (the window function) and the output line *index* representing the index of *Q2.* The reading will be enabled by the signal *reade*.

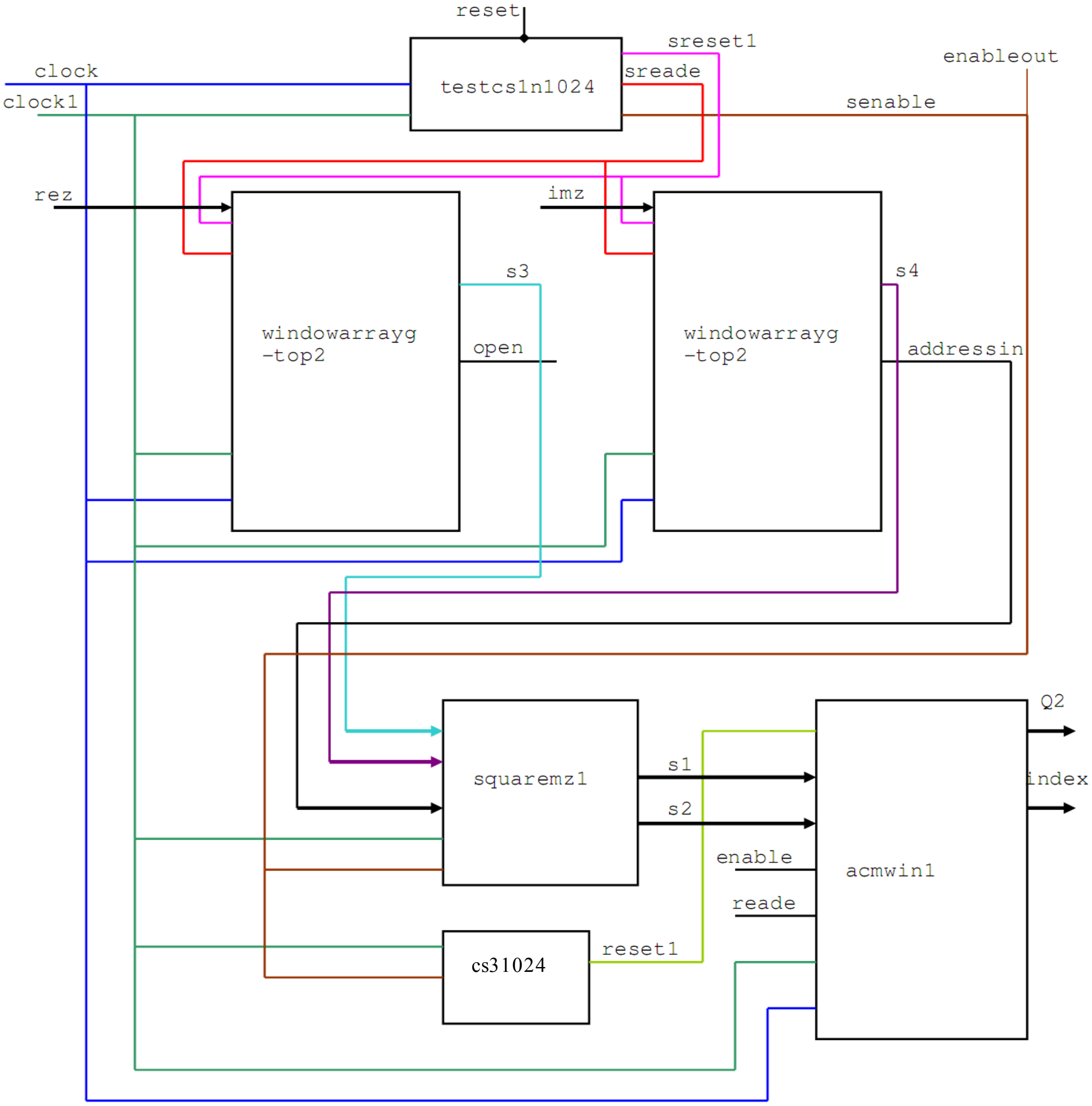


Fig. B.6: Detailed view of *zwindow1* (fig. 4.5). This block is the central to form $|Z|^2$ from the last stage accumulators of Z in *dout2acmf* and obtain the window function by replacing those values of $|Z|^2$ by 1, which are greater than or equal to 1/4th of the maximum of all 1024 $|Z|^2$ and by replacing those values of $|Z|^2$ by 0, which are less than 1/4th of the maximum of all 1024 $|Z|^2$. The window function thus obtained is latched to be read when required. The signal lines in the LHS and on the top of the logic blocks in the figure represent input lines to those blocks and the signal lines in the RHS of the blocks in the figure represent the output lines from those blocks. The inputs to this block as discussed in 4.5 are *rez* and *imz* and the output (window function) from this block is *Q2* with its index in the output line *index*.

3. rotparam2top1c* (refer to fig. B.7)*:

Inputs:

clock: *128 MHz input clock.*
reset: 1 bit control signal to start *rotparam2top1c*. Same as *reset* in *circularacm2plustest*.
reade: 1 bit control signal to start reading from the last stage buffer of this block. The *reade* of *circularacm2plustest*.
zr, zi: 31 bit inputs same as *rez, imz* respectively of *circularacm2plustest*.

Outputs:

cosn, sinn: 22 bit outputs representing $Z_r(K)\ /\ |Z(K)|$ and $Z_i(K)\ /\ |Z(K)|$ respectively. Signals named as *scosn* and *ssinn* in *circularacm2plustest.*
index1, index2: 10 bit indices of *cosn* and *sinn* respectively. These terminals are open in *circularacm2plustest*.

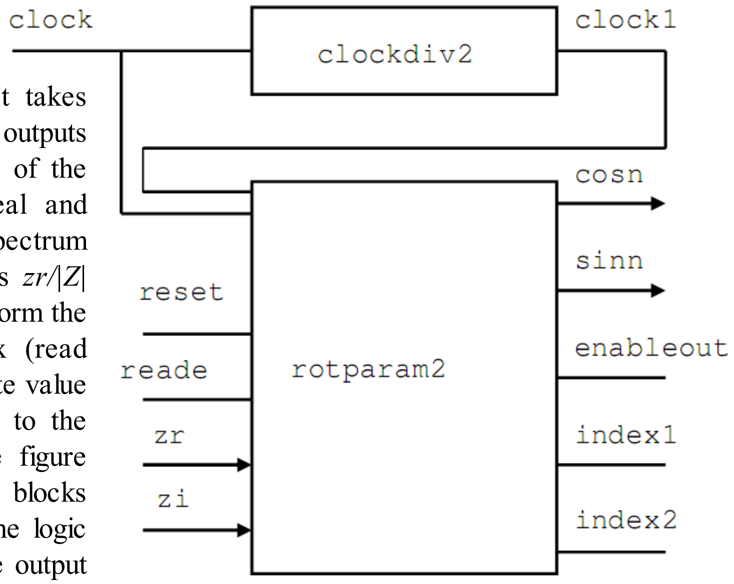


Fig. B.7: The module *rotparam2top1c*. It takes the inputs *zr* and *zi* which are the outputs from the two final accumulators of the module *dout2acmf* holding real and imaginary parts of the *Z* spectrum respectively. This block produces $zr/|Z| = cos\theta$ and $zi/|Z| = sin\theta$, which form the elements of the rotation matrix (read section 4.2.5, 2). $|Z|$ is the absolute value of the *Z* spectrum. The signals to the LHS of the logic blocks in the figure represent the input lines to those blocks and the signals in the RHS of the logic blocks in the figure represent the output lines from those blocks.

Component modules:

clockdiv2: It takes the clock *clock* (128 MHz) as input and generates *clock1* (64 MHz).

***rotparam2* (refer to fig. B.8*)**

**Note- Two blocks of rotparam2 are not shown in the figure (fig. B.8) to keep neatness and since they are buffers that I have described many times.*

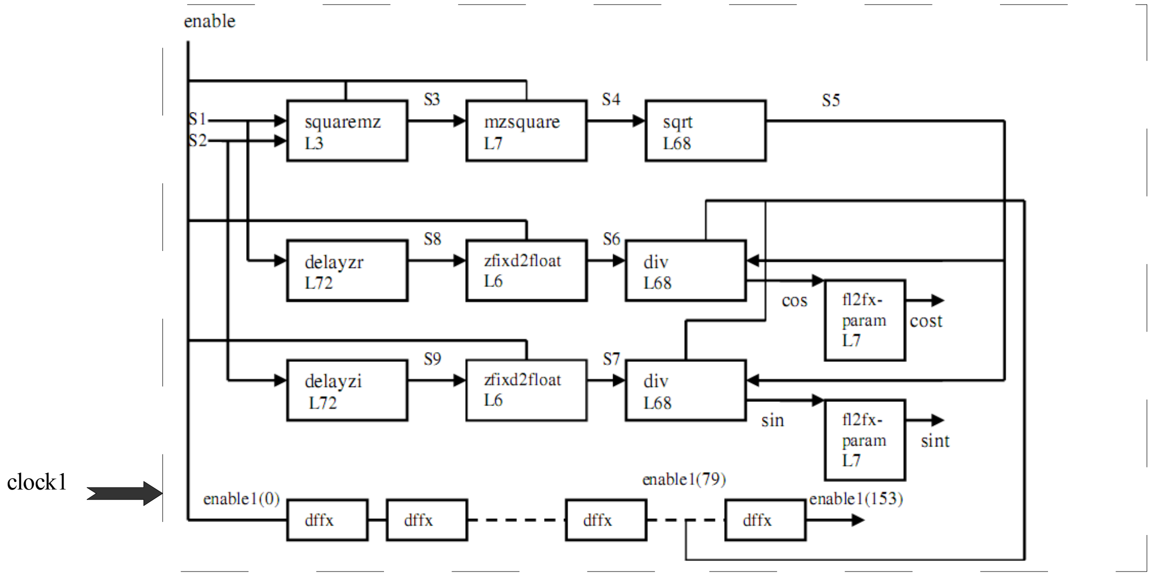


Fig. B.8: The figure shows the blocks from *squaremz* to *fl2fxparam* of the module *rotparam2* under *rotparam2top1c* (sec 4.2.5.1). The blocks before *squaremz* and after *fl2fxparam* are not shown in the figure since they consist of only registers and few control signals, which are easy to understand. The clock *clock1* feeds the module. The outputs from the buffer or register before *squaremz* are *s1* and *s2*, which are real and imaginary parts of input *Z* spectrum respectively. This block generates the rotation parameters *cost* and *sint* (where *t* has the same meaning as θ in 4.2.5, 2), which are stored in a buffer to be read when required. All operations shown in the figure are done at 64 MHz clock rate. And the operations not shown that are writing in the first buffer before *squaremz* and reading from the last buffer after *fl2fxparam* occur at 128 MHz clock rate. The number beside *L* inside the blocks in the figure represent the latency of that block in number of clock pulses. The signals to the LHS of the logic blocks in the figure represent the input lines to those blocks and the signals in the RHS of the logic blocks in the figure represent the output lines from those blocks.

Inputs:

clock: 128 MHz input clock.
clock1: 64 MHz input clock. Same as *clock1* of *rotparam2top1c*.
reset: 1 bit control signal to start *rotparam2*. Same as *reset* of *rotparam2top1c*.
reade: 1 bit control signal to start reading from the last stage buffer of this block. The *reade* of *rotparam2top1c*.
zr, zi: 31 bit inputs same as *zr* and *zi* respectively of *rotparam2top1c*.

Outputs:

cosn, sinn: 22 bit outputs representing $Z_r(K)$ / $|Z(K)|$ and $Z_i(K)$ / $|Z(K)|$ respectively. Same as *cosn* and *sinn* of *rotparam2top1c*.
enableout: 1 bit control signal generated in *rotparam2*. Not used in *rotparam2top1c*.
index1, index2: 10 bit indices of *cosn* and *sinn* respectively. Same as *index1* and *index2* of *rotparam2top1c*.

Component modules:

testcs1n1024: This block is started by *reset* and generates a signal *sreset1* (1 bit), which goes high with *reset* and remains high for 1025 clock (128 MHz) pulses. Three *clock1* pulses after *sreset1* goes low, *sreade* (1 bit) goes high and remains high forever. One *clock1* pulse after *sreade* is high, *enable* (1 bit) goes high and remains high forever. The output *enableout* gets the signal *enable*.

dffx: There are 154 such elements connected in series with the first element getting *enable* named as *enable1 (0)* as its input. Each element delays its input by one *clock1* pulse. The delayed signals from the elements are named as *enable1 (1), enable1 (2),....., enable1(154)* respectively.

windowarraygtop2: It consists of a buffer of 1024 locations and each location is 31 bit wide. There are two such elements working in parallel one for storing and transferring *zr* and the other for storing and transferring *zi.* I will only go into the details of the one used for *zr.* The signal *zr* arrives at its input one clock pulse after *sreset1* goes high and enters the buffer 2 clock pulses after *sreset1* goes high. Writing in the buffer with arriving *zr* occurs consecutively at each clock pulse starting from the first location. Writing finishes in 1025 clock pulses after *sreset1* goes high and it is also when *sreset1* goes low. Three *clock1* pulses after writing is finished, the buffer is triggered to be read by the signal *sreade* at the rate of 64 MHz using the clock *clock1*. Reading happens consecutively starting from the first location. The output data from this block appears in the line *s1* (31 bit) two *clock1* pulses after *sreade* is high. The line representing the index of *s1* is *open* since it is not used. The corresponding signals to *zr*, *s1, open* (10 bit index of *s1*) for the other *windowarraygtop2* are *zi, s2, open* (10 bit index of *s2*). The rest of the signals are common to both.

squaremz: It is started by *enable*. The input clock to this block is *clock1.* The signals *s1* and *s2* arrive at its inputs in parallel one *clock1* pulse after *enable* goes high. It forms $zr^2 + zi^2$ that is $s1^2 + s2^2$, which is transferred as output. The first output appears 4 *clock1* pulses after *enable* goes high or the input-output latency is 3 *clock1* pulses. In case $s1^2 + s2^2$ is 0, a 1 is returned at the output line. The output is named as *s3* (64 bit).

mzsquare: Xilinx floating point v4 IP core is used to generate this component. The signal *s3* is input to this block and is connected to the input terminal *A* of the IP core. The signal *enable* is used to start this block and is connected to the input line *CE* of the IP core. The input clock to this block is *clock1* and is connected to *CLK* of the IP core. The operation *fixed to float* is selected in the IP core since I want to represent *s3* as a floating point binary. For details on the conversion from fixed to float please refer to Xilinx documentation on floating point v4 IP core. This block generates the output *s4* (*RESULT* terminal of the IP core), which is floating point representation of *s3* having an exponent width 8 and a fraction width 64. So the total width of *s4* is 72. The input-output latency is 7 *clock1* pulses.

sqrt: This element is generated by Xilinx floating point v4 IP core. It is used to obtain $|Z(K)|$ by taking the square root of *s4*. The signal *s4* is connected to the input terminal *A* of the IP core. The function *square root* is selected in the IP core. The input clock to this block is *clock1* and is connected to *CLK* of the IP core. This block generates output $|Z(K)|$ by taking the square root of *s4* and the output is named as *s5* (72 bit floating point representation with 8 bit exponent and 64 bit fraction), which is the terminal *RESULT* of the IP core. For details on the square root operation please refer to Xilinx documentation on floating point v4 IP core. Input-output latency is 68 *clock1* pulses.

delayzr: It takes *s1* as its input with the input clock *clock1*. The signal *enable* is used to start this block. This block delays *s1* by 72 *clock1* pulses to produce *s8* (31 bit).

delayzi: This block works in parallel to *delayzr*. It takes *s2* as its input with the input clock *clock1*. The signal *enable* is used to start this block. This block delays *s2* by 72 *clock1* pulses to produce *s9* (31 bit).

zfixd2float: This block is generated by Xilinx floating point v4 IP core. There are two such blocks working in parallel one to convert *s8* into floating point number and the other to convert *s9* to floating point number. I will discuss both as for *s8* in case 1 and *s9* in case 2. Input clock is *clock1* to these blocks and is connected to *CLK* of the two IP cores. Each of the inputs *s8* in case 1 and *s9* in case 2 is connected to the input terminal *A* of the corresponding IP core. The function *fixed to float* is selected in the two IP cores. The outputs from the blocks are *s6* in case 1 and *s7* (each *s6* and *s7* is 72 bit floating point representation with 8 bit exponent and 64 bit fraction) in case 2 and each output is connected to the output terminal *RESULT* of the corresponding IP core. Input-output latency is 6 *clock1* pulses.

div: This block is generated by Xilinx floating point v4 IP core. There are two such elements working in parallel one getting the input *s6* and *s5* to obtain *s6 ÷ s5* and the other getting the inputs *s7* and *s5* to obtain *s7 ÷ s5*. I will only discuss the one getting the inputs *s6* and *s5*. The function *divide* is selected in the IP core. The signal *s6* is connected to the input terminal *A* of the IP core, which is the dividend and signal *s5* is connected to the input terminal *B* of the IP core, which is the divisor. The input clock to this block is *clock1* and it is connected to *CLK* of the IP core. The signal *enable1(79)* is used to start this block and is connected to *CE* of the IP core. The output is *s6 ÷ s5 = cos* (72 bit floating point representation with 8 bit exponent and 64 bit fraction) and is connected to the output terminal *RESULT* of the IP core. For details on the division operation please refer to Xilinx documentation on floating point v4 IP core. Input-output latency is 68 *clock1* pulses. The corresponding signals to *s6, s5* and *cos* in the other identical element are *s7, s5* and *sin* respectively. All other signal are common to both.

fl2fxparam: This block is generated by Xilinx floating point v4 IP core. There are two such elements working in parallel one to receive *cos* and convert it to fixed point representation and the other to receive *sin* and convert it to fixed point representation. I will discuss both as case 1 for *cos* and case 2 for *sin*. The input clock to each of these blocks is *clock1* and is connected to *CLK* of each IP core. Each of the signals *cos* and *sin* is connected to the input terminal *A* of the corresponding IP core. The function *float to fixed* is selected in the two IP cores. The floating point represented *cos* in case 1 or *sin* in case 2 is converted to fixed point represented *cost* in case 1 or *sint* (each *cost* and *sint* is 2 bit integer and 20 bit fraction so the total width is 22 bit) in case 2 . For more details on the fixed point representation please refer to Xilinx documentation on floating point v4 IP core. Each output *cost* and *sint* is connected to the output terminal *RESULT* of the corresponding IP core. Input-output latency is 7 *clock1* pulses.

testcsn1024: It is started by *enable1(153)*. The input clock is *clock1*. It generates *sreset* (1 bit), which goes high with *enable1(153)* and remains high for 1025 *clock1* pulses.

windowarraytop1: There are two such elements working in parallel one receiving *cost* and the other receiving *sint*. I will discuss the one receiving *cost*. It consists of a buffer having 1024 locations and each location is 22 bit wide. The signal *cost* arrives at its input one *clock1* pulse after *sreset* goes high and from the next *clock1* pulse or after two *clock1* pulses after *sreset* goes high, arriving *cost* starts entering the buffer at each *clock1* pulse consecutively starting from the first location. So the buffer is written at 64 MHz clock rate. In 1025 *clock1* pulses after *sreset* is high, writing in the buffer stops and it is ready to be read using the read enable signal *reade*. This *reade* signal when high will trigger reading of data from the buffer at each clock pulse consecutively starting from the first location using the clock *clock* (128 MHz) in the output line *cosn* (22 bit). The output line *index1* (10 bit) represents the index of *cosn*. The signals in the other *windowarraytop1* corresponding to the signals *cost, cosn* and *index1* in this *windowarraytop1* are *sint, sinn* and *index2* respectively. All other signals are common to both.

4. gxytop2n1c **(refer to fig. B.9)***:*

Inputs:

clock: 128 MHz input clock.
reset: 1 bit control signal to start *gxytop2n1c*. Same as *reset* of *circularacm2plustest.*
sdat1, sdat2: 31 bit data corresponding to $|X(K)|^2$ and $|Y(K)|^2$ from the final accumulators in *dout2acmf*. These are *modx2* and *mody2* respectively of *circularacm2plustest.*
sindx: 10 bit index of *sdat1* and *sdat2* respectively. Same as *sindx* of *circularacm2plustest*.
reade: 1 bit control signal to start reading from the last stage buffer of this block. The *reade* of *circularacm2plustest*.

Outputs:

gxout, gyout: 22 bit gains of *X* and *Y* channels respectively. Signals named as *sgxout* and *sgyout* respectively in *circularacm2plustest*.
xindex, yindex: 10 bit indices of *gxout* and *gyout* respectively. These terminals are open in *circularacm2plustest*.

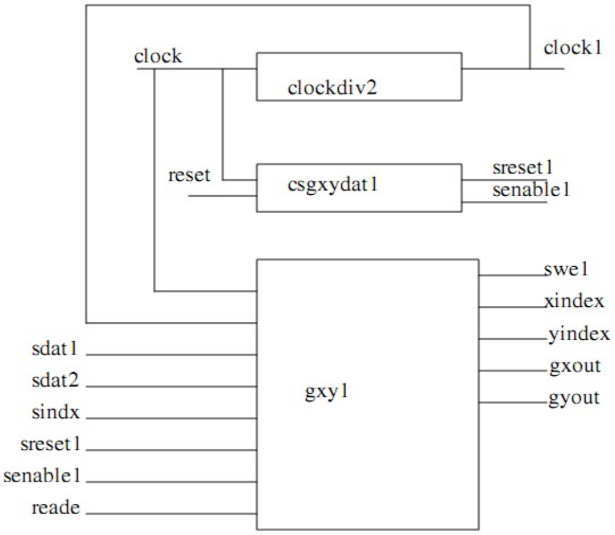


Fig. B.9: The figure shows the module *gxytop2n1c*. This block takes in the inputs *sdat1* and *sdat2* which are the outputs from the last stage power spectra accumulators of *dout2acmf* that are the power spectra of *X* and *Y* channels respectively. This block obtains the gain parameters for *X* and *Y* channels from these inputs. The gain parameters are the gains by which the absolute values of *X* and *Y* channels must be raised to equalize the channels' magnitude responses. The outputs are *gxout* (gain of *X* channel) and *gyout* (gain of *Y* channel) with their indices in *xindex* and *yindex* respectively. The signals to the LHS of the logic blocks in the figure represent the input lines to those blocks and the signals in the RHS of the logic blocks in the figure represent the output lines from those blocks. Ports having same signal names are interconnected.

Component modules:

clockdiv2: It takes the clock *clock* (128 MHz) as input and generates *clock1* (64 MHz).

csgxydat1: It is started by *reset.* It generates a signal *sreset1* (1 bit), which goes high with *reset* and remains high for 1025 clock pulses. It also generates a signal *senable1* (1 bit) 2 clock pulses after *sreset1* goes low. This signal *senable1* remains high forever.

gxy1* (refer to fig. B.10)*:

Inputs:

clock: 128 MHz input clock
clock1: 64 MHz clock. Same as *clock1* of *gxytop2n1c*.
D1, D2: 31 bit inputs. Same as *sdat1* and *sdat2* respectively of *gxytop2n1c*.
write_address: 10 bit address of *D1* and *D2*. Same as *sindx* of *gxytop2n1c*.
we: 1 bit control signal to start writing in the first stage buffer of *gxy1*. Same as *sreset1* of *gxytop2n1c*.
enable: 1 bit control signal to trigger finding of maximum of all 1024 *D1* or 1024 *D2*. Same as *senable1* of *gxytop2n1c*.
re: 1 bit control signal to start reading from the last stage buffer of *gxy1*. Same as *reade* of *gxytop2n1c.*

Outputs:

swe1: 1 bit output signal. Not used in *gxytop2n1c*.
gxout, gyout: 22 bit gains of *X* and *Y* channels respectively. Same as *gxout* and *gyout* respectively of *gxytop2n1c*.
xindex, yindex: 10 bit indices of *gxout* and *gyout* respectively. Same as *xindex* and *yindex* respectively of *gxytop2n1c*

Component modules:

dff1: There are 1108 such modules connected in series each delaying its 1 bit input by one *clock1* pulse. A signal *e(0)* gets the signal *enable* and is the input to the first *dff1*. The outputs from all the elements starting from the first one are *e(1), e(2), e(3), …, e(1108)* respectively.

acm_mag1: There are two such elements working in parallel one receiving *D1* and the other receiving *D2*. I will only discuss the one receiving *D1*. It consists of a buffer having 1024 locations and each location is 31 bit wide. The input *D1* arrives 1 clock pulse after *we* goes high and one clock pulse after *D1* arrives, *D1* starts entering the buffer consecutively at each clock pulse starting from the first location; the location is pointed by the *write_address* arriving in parallel with *D1*. In 1025 clock pulses after *we* is high, writing in the buffer stops. One *clock1* pulse after *enable* goes high, the entered values of *D1* in the buffer are compared to determine the maximum value among them. The maximum is found in the following way: A temporary register of 31 bit is taken initialized to 0 and in the next *clock1* pulse after *enable* is high, its value is compared with the value in the first location of the buffer containing *D1*. If the value of the data in the first location is greater than the contents of the register then the register gets the data from the first location else it keeps its previous value thus updating the register after comparison is done. In the next *clock1* pulse after comparing with first location and updating the register, its value is compared with the value in the second location of the buffer containing *D1*. If the value of the data in the second location is greater than the present contents of the register then the register gets the data from the second location else it keeps its previous value and so on and each time the register content is updated to get the maximum after all 1024 comparisons are finished. Thus the temporary register contains the maximum value of *D1*. Obtaining the maximum in the register takes 1024 *clock1* pulses after *enable* goes high and in 1024th *clock1* pulse after *enable* is high, reading from the buffer is triggered and 2 *clock1* pulses after reading is triggered, the data from the buffer starts appearing in the line *smx2* (31 bit) with the stored maximum appearing in another parallel line *sxmax2* (31 bit). The index line representing the index of *smx2* is kept open. The signals corresponding to *D1, smx2, sxmax2, open* (10 bit index of *smx2*) of this *acm_mag1* in the other *acm_mag1* are *D2, smy2, symax2, sind* (index of *smy2*) respectively. All other signals are common to both.

comparexy: It is started by *e(1025)*. Its inputs *smx2, smy2, sxmax2, symax2* and *sind* start appearing one *clock1* pulse after *e(1025)* goes high. The signal *sxmax2* is compared with *symax2* and whichever is greater is passed to the output line *amax2* (31 bit). The signals *smx2, smy2* and *sind* are transferred in parallel to the output lines *mx2* (31 bit), *my2* (31 bit) and *index* (10 bit) respectively. The outputs start appearing in parallel 2 *clock1* pulses after *e(1025)* goes high. If *smx2* =0 or *smy2* = 0 then a 1 is passed to the output line *mx2* or *my2*.

xfx2fl: This block is generated by Xilinx floating point v4 IP core. There are three such elements working in parallel one receiving *amax2*, another *mx2* and the third receiving *my2*. I will discuss them all as the one for *amax2* (case1), for *mx2* (case 2) and for *my2* (case 3). All three blocks are started by *e(1026)* and the input *amax2* in case 1 or *mx2* in case 2 or *my2* in case 3 arrives one *clock1* pulse after *e(1026)* goes high. Each *amax2*, *mx2* and *my2* is connected to the input terminal *A* of the corresponding IP core. The signal *e(1026)* is connected to the *CE* of the three IP cores and *clock1* is connected to *CLK* of the three IP cores. The function *fixed to float* is selected in the three IP cores. These three IP cores convert the input quantity from fixed point representation to floating point representation. So the floating point outputs are *amax2f* in case 1, *mx2f* in case 2 and *my2f* in case 3 (in all three cases the outputs are 39 bit wide with 8 bit exponent and 31 bit fraction). The input-output latency is 6 *clock1* pulses. For details on the fixed to float operation please refer to Xilinx documentation on floating point v4 IP core.

divg: This block is generated by Xilinx floating point v4 IP core. It is started by *e(1032)*. There are two such elements working in parallel one receiving *amax2f, mx2f* and the other receiving *amax2f, my2f*. I will discuss them both as the one for receiving *mx2f* in case 1 and the other for *my2f* in case 2. The signal *amax2f* is connected in each case to the terminal *A* of the corresponding IP core and is the dividend in both cases. Each of the signals *mx2f* and *my2f* is connected to the input terminal *B* of the corresponding IP core and are the divisors. The signal *e(1032)* is connected to *CE* of the two IP cores and *clock1* is connected to *CLK* of the two IP cores. The function *divide* is selected in the two IP cores. The blocks perform divisions *amax2f* ÷ *mx2f* (case 1) and *amax2f* ÷ *my2f* (case 2) and the outputs are *gx2* and *gy2* (each output is 39 bit wide float with 8 bit exponent and 31 bit fraction) respectively. The input – output latency is 35 *clock1* pulses. The output line DIVIDE_BY_ZERO of the IP core is also generated but not used. For details on the division operation please refer to Xilinx documentation on floating point v4 IP core.

sqrtgxy: This block is generated by Xilinx floating point v4 IP core. It is started by *e(1067)*. There are two such modules working in parallel one receiving *gx2* and the other receiving *gy2*. I will discuss them both as the one for *gx2* in case 1 and the other for *gy2* in case 2. The input *gx2* or *gy2* appears one *clock1* pulse after *e(1067)* goes high. Each of the inputs *gx2* and *gy2* is connected to the input terminal *A* of the corresponding IP core. The signal *e(1067)* is connected to *CE* of both the IP cores and *clock1* is

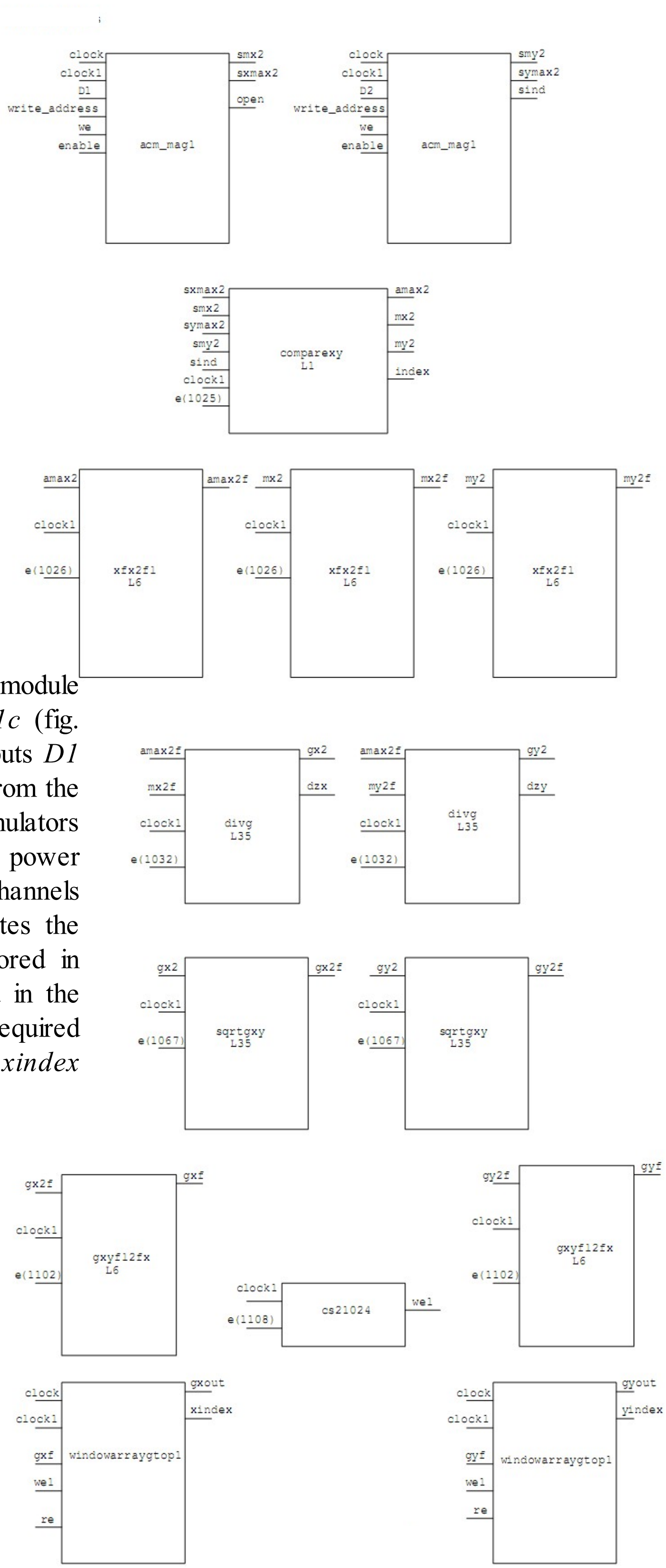


Fig.B.10: The details of the component module *gxy1* of the module *gxytop2n1c* (fig. 4.9). This block takes in the inputs *D1* and *D2* which are the outputs from the last stage power spectra accumulators of *dout2acmf* that are the power spectra of *X* and *Y* channels respectively. This block generates the gain parameters, which are stored in *windowarraygtop1* to be read in the lines *gxout* and *gyout* when required with the indices appearing in *xindex* and *yindex* respectively. All operations shown in the figure are done at 64 MHz clock rate. The number beside L in some blocks in the figure represents the latency of that block in number of clock pulses. The signals to the LHS of the logic blocks in the figure represent the input lines to those blocks and the signals in the RHS of the logic blocks in the figure represent the output lines from those blocks. Ports having same signal names are interconnected

connected to *CLK* of both the IP cores. The function *square root* is selected in the IP cores. The two blocks perform $\sqrt{gx2}$ (case 1) and $\sqrt{gy2}$ (case 2) to obtain outputs *gx2f* and *gy2f* (each output is 39 bit wide float with 8 bit exponent and 31 bit fraction) respectively, each of which is connected to the *RESULT* terminal of the corresponding IP core. The input-output latency is 35 *clock1* pulses. For details on the square root operation please refer to Xilinx documentation on floating point v4 IP core.

gxyfl2fx: This block is generated by Xilinx floating point v4 IP core. There are two such modules working in parallel one receiving *gx2f* and the other receiving *gy2f*. I will discuss both as the one for *gx2f* in case 1 and the other for *gy2f* in case 2. The input clock is *clock1* and is connected to *CLK* of both the IP cores. Each of the inputs *gx2f* and *gy2f* is connected to the input terminal *A* of the corresponding IP core. The signal *e(1102)* is used to start these blocks and is connected to *CE* of both the IP cores. The function *float to fixed* is selected in the two IP cores. The inputs *gx2f* and *gy2f* are converted to fixed point format producing outputs *gxf and gyf* (each output is 22 bit wide with 5 bit integer and 17 bit fraction) respectively. Each of the outputs is connected to the output terminal *RESULT* of the corresponding IP core. The input-output latency is 6 *clock1* pulses.

cs21024: It is started by *e(1108)*. The input clock is *clock1*. It generates a signal *we1* (1 bit), which goes high with *e(1108)* and remains high for 1025 *clock1* pulses after which it goes low. This signal *we1* is connected to the output signal *swe1*.

windowarraygtop1: There are two such modules working in parallel one receiving *gxf* and the other *gyf*. So I will only discuss the one receiving *gxf*. This block consists of a buffer having1024 locations and each location is 22 bit wide. The block is started by *we1*. The input *gxf* starts appearing one *clock1* pulse after *we1* goes high and starts entering the buffer 2 *clock1* pulses after *we1* is high. Writing happens consecutively at each *clock1* pulse starting from the first location and hence in 1025th *clock1* pulse after *we1* is high, the buffer is written completely. The buffer is then ready to be read anytime. Reading will happen at 128 MHz clock rate 2 clock pulses after the read enable signal *re* goes high. The data from the buffer will then appear at the output line *gxout* at each clock pulse consecutively starting from the first location. The other output line *xindex* will receive the index of *gxout* in parallel. The signals corresponding to *gxf, gxout* and *xindex* of this module are *gyf, gyout* and *yindex* respectively in the other module. All other signals are common to both.

5. equalizeparamswin: It is started by the signal *reade*. Two clock pulses after *reade* (which performs read operation from the last stage buffers in the blocks from 2 to 4.of B.1.5.1) is high, *swinfunc, scosn, ssinn, sgxout, sgyout and sindexwin* appear as the inputs to this block. It performs the following multiplications: *sgxout* × *swinfunc* = *sgxf* (22 bit), *sgyout* × *swinfunc* = *sgyf* (22 bit), *scosn* × *swinfunc* = *scosnf* (22 bit), *ssinn* × *swinfunc* = *ssinnf* (22 bit). The results or the outputs of all multiplications are obtained in parallel. The input *sindexwin* representing the indices of the inputs is transferred to the output line *sindexep* (10 bit), which is parallel to the outputs showing their index at any clock pulse. Input-output latency is 2 clock pulses or the output arrives 4 clock pulses after *reade* is high.

6. resetdecaddtest: It is started by *wenable* (obtained in block number 1 of B.1.5.1). 2055 clock pulses after *wenable* is high, *testadd* (1 bit) is generated (goes high).

7. deceqdat8_24: It is started by *reset*. It gets the outputs from the two *decodervar1s* as its inputs: 8 lines for each real and imaginary parts of *X* polarization and 8 lines for each real and imaginary parts of *Y* polarization; There are also 8 lines of indices corresponding to 8 lines of *X* polarization and 8 lines of indices corresponding to 8 lines of *Y* polarization. These inputs arrive one clock pulse after *reset* goes high These inputs are transferred to the following output lines: 8 output lines corresponding to 8 input lines for real parts of *X* polarization: *xre1, xre2, xre3, ..., xre8* (all 22 bit) respectively; 8 output lines corresponding to 8 input lines for real parts of *Y* polarization: *yre1, yre2, yre3, ..., yre8* (all 22 bit) respectively; 8 output lines corresponding to 8 input lines for imaginary parts of *X* polarization: *xim1, xim2, xim3, ..., xim8* (all 22 bit) respectively; 8 output lines corresponding to 8 input lines for imaginary parts of *Y* polarization: *yim1, yim2, yim3, ..., yim8* (all 22 bit) respectively; this block also transfers 8 lines of indices of *X* or *Y* polarization to the output lines *addressdec1, addressdec2, addressdec3, ..., addressdec8* (all 10 bit). The outputs having same numbers beside them are parallel. Input-output latency is one clock pulse.

Important - Now I have 8 lines for each real and imaginary parts of *X* polarization, 8 lines for each real and imaginary parts of *Y* polarization and 8 lines for indices corresponding to the 8 lines for real *X,* or real *Y* or imaginary *X* or imaginary *Y.* I will now deal with only first group of parallel inputs that are *xre1, xim1, yre1, yim1* and *addressdec1* (consecutive groups have a delay of 128 clock pulses between them starting from the first one). The other 7 groups will feed similar stages as the first group with a delay of 128 clock pulses between the consecutive stages starting from the first stage.

8) (i) passdecdat: It is started by the signal *testadd*. It gets the inputs *xre1, xim1, yre1, yim1* and *addressdec1*. It checks when *addressdec1* has all 1s for the first time and in that clock pulse it generates a signal *spassxy1* (1bit). The inputs *xre1, xim1, yre1, yim1* and *addressdec1* corresponding to the next clock pulse after *spassxy1* is high (*addressdec1* then starts with address 0) are transferred to the output lines *sxr1,sxi1,syr1,syi1* and *saddrout1* respectively 2 clock pulses after *spassxy1* goes high.

(ii) eqparramarraytop: There are four such modules working in parallel one receiving *sgxf*, another receiving *sgyf*, the third receiving *scosnf* and the fourth receiving *ssinnf*. So I will only go into the details of the one receiving *sgxf* and the rest will follow from there. This block consists of a buffer of 1024 locations and each location is 22 bit wide. It is started by *wenable*. One clock pulse after *wenable* goes high, the inputs *sgxf* and *sindexep* starts appearing and two clock pulses after *wenable* is high, *sgxf* starts entering the buffer at each clock pulse in the location pointed by *sindexep* that is consecutively starting from the first location. So in 1025th clock pulse after *wenable* is high, writing in the buffer stops. The signal *spassxy1* acts as read enable to the buffer. The data from the buffer starts appearing in the output line *gxoutf1* 2 clock pulses after *spassxy1* goes high. Reading happens per clock pulse consecutively starting from the first location of the buffer. The output line *indexgx1* represents the index of *gxoutf1*. The signals corresponding to *sgxf, gxoutf1* and *indexgx1* in the other three modules are: second module: *sgyf, gyoutf1* and *open* respectively; third module: *scosnf, cosnf1* and *open* respectively; fourth module: *ssinnf, sinnf1* and *open* respectively. All other signals are common to the four modules.

(iii) xycircular: It is started by *spassxy1*. Its inputs *sxr1, sxi1, syr1, syi1, saddrout1, gxoutf1, gyoutf1, cosnf1* and *sinnf1* arrive 2 clock pulses after *spassxy1* goes high. It performs the following operations for phase and gain equalization: *sxr1* × *gxoutf1, sxi1* × *gxoutf1, syr1* × *gyoutf1* × *cosnf1* – *syi1* × *gyoutf1* × *sinnf1, syr1* × *gyoutf1* × *sinnf1* + *syi1* × *gyoutf1* × *cosnf1* yielding new real *X* (*xreal1* say), imaginary *X* (*ximag1* say), real *Y* (*yreal1* say) and imaginary *Y*(*yimag1* say) respectively and in parallel. Next step is formation of circular polarization from these new *X* and *Y,* which is done as follows: *LHCreal1* (real part of LHC) = *xreal1* + *yimag1*; *LHCimag1* (imaginary part of LHC) = *ximag1* – *yreal1; RHCreal1* (real part of RHC) = *xreal1* – *yimag1; RHCimag1* (imaginary part of RHC) = *ximag1* + *yreal1*. The outputs *LHCreal1, LHCimag1, RHCreal1* and *RHCimag1* are parallel and appear 8 clock pulses after *spassxy1* goes high. The output index *indexout1* is also passed in parallel with the corresponding outputs.

Note- The components in (8) are replicated 7 times for the remaining 7 stages with the number 1 beside the signals replaced by 2, 3, 4,..., 8 for the cases from 2 to 8 respectively. The rest of the signals are common to all. So in total there are 15 components with 9 to 15 being replications of 8 as mentioned above.

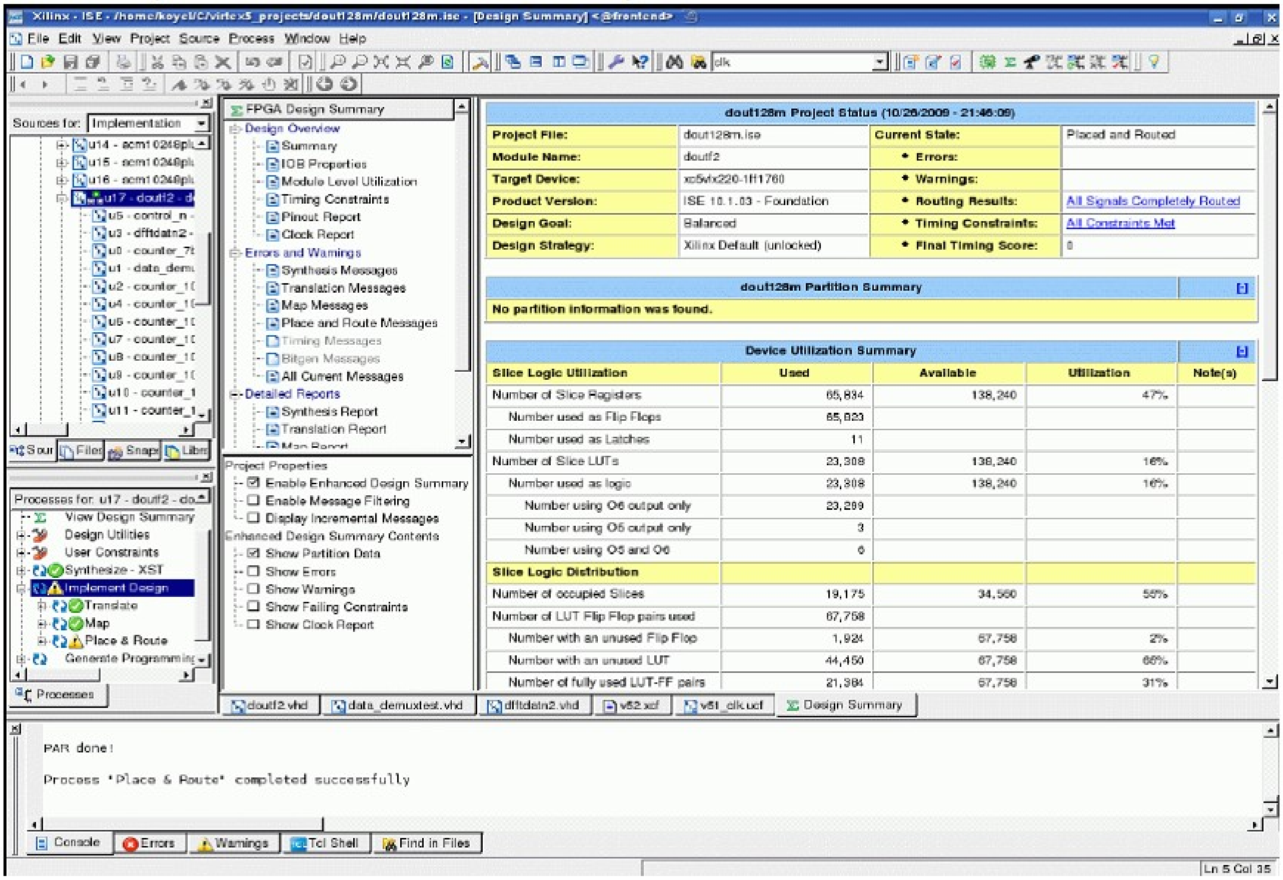
B.2 Implementation information

After discussing the design of the digital circular polarizer in B.1, I will now show the summary pages generated by implementing different parts of the design to be connected later. Each of these parts is able to fit inside a Virtex 5 on a DBBC Core2 board. Fig. B.11 shows the implementation summary page of *doutf2* (B.1.2.2), fig. B.12 shows the implementation summary page of 8 times replication of B.1.3.1 that is the modules from *combunitfft1* to *combunitfft8* are implemented in a single device. Fig. B.13 shows the implementation summary page for the components from 1 to 4 of B.1.5.1. Components from 5 to 7 are implemented in a small device and components from 8 to 15 are implemented in pairs in small devices, which are not shown since they consist of trivial logic only and easy to be implemented. I also carried out formation of circular power spectra and accumulation and it can be found in the design file as well as in the implementation file. Since the accumulation process is similar to the one shown in B.1.3.1 block 4, I have not shown it here.

B.2.1 Implementation summary pages

The window on the top left of the implementation summary pages (fig. B.11 to fig. B.13) has the name of the project highlighted in blue. The window on the bottom left shows the steps completed that are *Synthesize- XST, Implement Design*, which consists of *Translate, Map* and *Place and Route*. The yellow exclamation beside these means successful operation with warnings (can be overlooked) and a green check beside them means successful operation without any warning. The window on the top right shows the status of the project: the *Project File* provides the name of the file containing the project; *Module Name* provides the name of the module implemented; *Target Device* provides the name of the Xilinx device on which the module is implemented; *Product Version* shows the version of the software ISE; *Design Goal* shows the parameter of emphasis, which can be *timing performance* or *area performance* or both (*balanced*); *Current Status* shows the status of the project, which is *Placed and Routed* showing that implementation is complete; *Errors* show number of errors while implementing the module; *Warnings* show the number of warnings while implementing the module (it can be overlooked); *Routing Results* show the routing status of the signals; *Timing Constraints* show whether all timing constraints are met or not; *Final Timing Score* represents a zero if all timing constraints are met else it provides a number. The bottom right window shows the logic usage in the device. The figures from B.11 to B.13 start from the next page.

Note- I enclose a CD with two zip files. One named as circular1.zip having B.1.5.1 and the other named as dout.zip having B.1.4.1. After unzipping the files the project files to be opened are circular24.ise and dout128m.ise respectively.



Project Properties
- ☑ Enable Enhanced Design Summary
- ☐ Enable Message Filtering
- ☐ Display Incremental Messages

Enhanced Design Summary Contents
- ☑ Show Partition Data
- ☐ Show Errors
- ☐ Show Warnings
- ☐ Show Failing Constraints
- ☐ Show Clock Report

dout128m Project Status (10/26/2009 - 21:46:09)			
Project File:	dout128m.ise	**Current State:**	Placed and Routed
Module Name:	doutf2	**• Errors:**	
Target Device:	xc5vlx220-1ff1760	**• Warnings:**	
Product Version:	ISE 10.1.03 - Foundation	**• Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	**• Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	0

dout128m Partition Summary

No partition information was found.

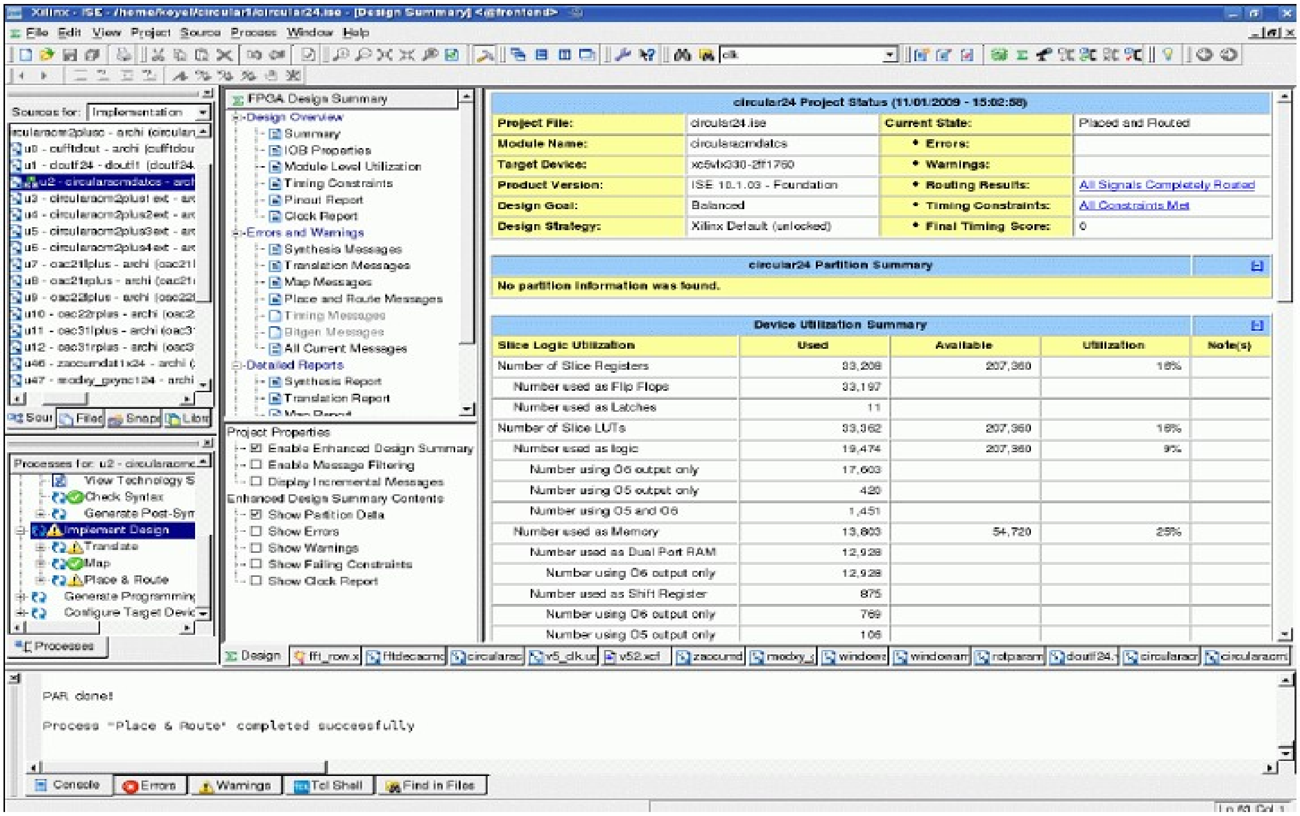
Device Utilization Summary

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	65,834	138,240	47%	
Number used as Flip Flops	65,823			
Number used as Latches	11			
Number of Slice LUTs	23,308	138,240	16%	
Number used as logic	23,308	138,240	16%	
Number using O6 output only	23,299			
Number using O5 output only	3			
Number using O5 and O6	6			
Slice Logic Distribution				
Number of occupied Slices	19,175	34,560	55%	
Number of LUT Flip Flop pairs used	67,758			
Number with an unused Flip Flop	1,924	67,758	2%	
Number with an unused LUT	44,450	67,758	65%	
Number of fully used LUT-FF pairs	21,384	67,758	31%	

```
PAR done!

Process "Place & Route" completed successfully
```

Fig B.11: Implementation summary page of *doutf2* (4.2.2.2)



Project Properties
- ☑ Enable Enhanced Design Summary
- ☐ Enable Message Filtering
- ☐ Display Incremental Messages

Enhanced Design Summary Contents
- ☑ Show Partition Data
- ☐ Show Errors
- ☐ Show Warnings
- ☐ Show Failing Constraints
- ☐ Show Clock Report

circular24 Project Status (11/01/2009 - 15:02:58)			
Project File:	circular24.ise	**Current State:**	Placed and Routed
Module Name:	circularacmdatcs	**• Errors:**	
Target Device:	xc5vlx330-2ff1760	**• Warnings:**	
Product Version:	ISE 10.1.03 - Foundation	**• Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	**• Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	**• Final Timing Score:**	0

circular24 Partition Summary

No partition information was found.

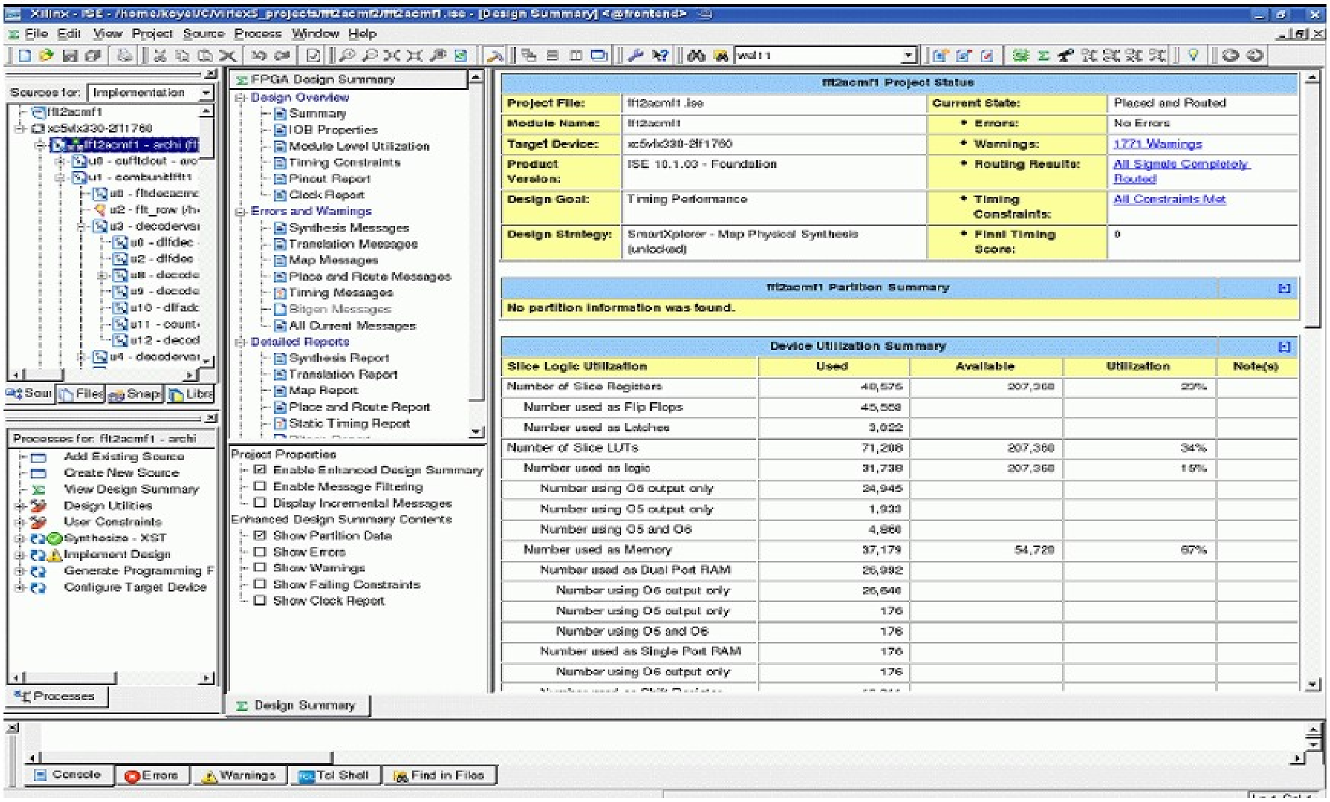
Device Utilization Summary

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	33,208	207,360	16%	
Number used as Flip Flops	33,197			
Number used as Latches	11			
Number of Slice LUTs	33,362	207,360	16%	
Number used as logic	19,474	207,360	9%	
Number using O6 output only	17,603			
Number using O5 output only	420			
Number using O5 and O6	1,451			
Number used as Memory	13,803	54,720	25%	
Number used as Dual Port RAM	12,928			
Number using O6 output only	12,928			
Number used as Shift Register	875			
Number using O6 output only	769			
Number using O5 output only	106			

```
PAR done!

Process "Place & Route" completed successfully
```

Fig. B.12: Implementation summary page for *combunitffft1* to *combunitffft8* that is 8 times replication of 4.2.3.1



Project Properties
- ☑ Enable Enhanced Design Summary
- ☐ Enable Message Filtering
- ☐ Display Incremental Messages

Enhanced Design Summary Contents
- ☑ Show Partition Data
- ☐ Show Errors
- ☐ Show Warnings
- ☐ Show Failing Constraints
- ☐ Show Clock Report

fft2acmf1 Project Status

Project File:	fft2acmf1.ise	**Current State:**	Placed and Routed
Module Name:	fft2acmf1	• **Errors:**	No Errors
Target Device:	xc5vlx330-2ff1760	• **Warnings:**	1771 Warnings
Product Version:	ISE 10.1.03 - Foundation	• **Routing Results:**	All Signals Completely Routed
Design Goal:	Timing Performance	• **Timing Constraints:**	All Constraints Met
Design Strategy:	SmartXplorer - Map Physical Synthesis (unlocked)	• **Final Timing Score:**	0

fft2acmf1 Partition Summary

No partition information was found.

Device Utilization Summary

Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	48,575	207,360	23%	
Number used as Flip Flops	45,553			
Number used as Latches	3,022			
Number of Slice LUTs	71,208	207,360	34%	
Number used as logic	31,738	207,360	15%	
Number using O6 output only	24,945			
Number using O5 output only	1,933			
Number using O5 and O6	4,860			
Number used as Memory	37,179	54,720	67%	
Number used as Dual Port RAM	26,992			
Number using O6 output only	26,640			
Number using O5 output only	176			
Number using O5 and O6	176			
Number used as Single Port RAM	176			
Number using O6 output only	176			

Fig. B.13: Implementation summary page for the components from 1 to 4 of 4.2.5.1.

APPENDIX C: VHDL CODES OF SELECTED MODULES FROM APPENDIX B

In this appendix I have put the codes of those blocks (described in appendix B) in our design, which perform the most crucial operations. It is not possible to keep all codes as per the data flow in the project as that would cover around 140 to 150 pages; All codes are in the ISE (Xilinx) projects in the CD enclosed with the thesis whose details is given in end of section B.2.1 of the thesis. Also the codes generating control signals are not shown as they are easy to understand. Though almost all of the codes are specific to the design and written by me, the codes in this appendix are worth keeping at one place for reference of the corresponding block. I will provide the reference of the blocks in appendix B before their codes and also write the signal names under the block descriptions corresponding to the inputs and outputs of the codes. The signal names of a module in the code is different than that described in appendix B whenever the signal is called from a top module comprising it; this is similar to the case where a routine calls a function and the names of the input or output parameters in the function are different than that used in the routine. Here the top module comprising a block takes the place of the routine or caller function and the block takes the place of the called function.

C.1 Block *data_demuxtest* (p 87, section B.1.2.2, block 5)

All signal names in the code are the same as in the block described under section B.1.2.2, block 5 with input *sdata* in the code replaced by *sdata1* in the block. In this and in the following descriptions the term 'block' refers to the description in appendix B and the code refers to the presented code.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
package registerytst is
subtype WORD8 is STD_LOGIC_VECTOR (9 downto 0);
type mem08 is array (7 downto 0) of WORD8;
type packet128 is array (127 downto 0) of mem08;
type mem1024 is array (1023 downto 0) of WORD8;
end registerytst;
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.registerytst.all;
entity data_demuxtest is
port( clock,re1,re2,re3,re4,re5,re6,re7,re8,we: in std_logic;
    sel : in std_logic_vector(2 downto 0) :="000";
    write_address: in std_logic_vector(6 downto 0);
read_address1,read_address2,read_address3,read_address4,read_address5,read_address6,read_address7
,read_address8 : in std_logic_vector(9 downto 0);
            sdata : in mem08:=(others=>(others=>'0'));
    Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8 : out std_logic_vector(9 downto 0):=(others=>'0')
   );

end data_demuxtest;
architecture archi of data_demuxtest is
signal soutputx1,soutputx2,soutputx3,soutputx4,soutputx5,soutputx6,soutputx7,soutputx8 :
packet128:=(others=>(others=>(others=>'0')));
```

```
signal sinputx1,sinputx2,sinputx3,sinputx4,sinputx5,sinputx6,sinputx7,sinputx8 : mem1024:=
(others=>(others=>'0'));
begin
u0: for i in 1 to 128 generate
u1: for j in 0 to 7 generate
sinputx1((i*j) + ((i-1)*(8-j))) <= soutputx1(i-1)(j);
sinputx2((i*j) + ((i-1)*(8-j))) <= soutputx2(i-1)(j);
sinputx3((i*j) + ((i-1)*(8-j))) <= soutputx3(i-1)(j);
sinputx4((i*j) + ((i-1)*(8-j))) <= soutputx4(i-1)(j);
sinputx5((i*j) + ((i-1)*(8-j))) <= soutputx5(i-1)(j);
sinputx6((i*j) + ((i-1)*(8-j))) <= soutputx6(i-1)(j);
sinputx7((i*j) + ((i-1)*(8-j))) <= soutputx7(i-1)(j);
sinputx8((i*j) + ((i-1)*(8-j))) <= soutputx8(i-1)(j);
end generate;
end generate;
            PROCESS (clock,we,sdata,write_address,sel)
            BEGIN
                        IF (clock'event AND clock = '1') THEN
                                    IF (we = '1') THEN
                                    if sel = "000" then
                                     soutputx1(to_integer(unsigned(write_address)))<= sdata;
         elsif sel = "001" then
          soutputx2(to_integer(unsigned(write_address)))<= sdata;
         elsif sel = "010" then
          soutputx3(to_integer(unsigned(write_address)))<= sdata;
         elsif sel = "011" then
          soutputx4(to_integer(unsigned(write_address)))<= sdata;
         elsif sel = "100" then
          soutputx5(to_integer(unsigned(write_address)))<= sdata;
         elsif sel = "101" then
          soutputx6(to_integer(unsigned(write_address)))<= sdata;
         elsif sel = "110" then
          soutputx7(to_integer(unsigned(write_address)))<= sdata;
         elsif sel = "111" then
          soutputx8(to_integer(unsigned(write_address)))<= sdata;
                                    end if;
                                    END IF;
                                    END IF;
            END PROCESS;
            PROCESS
(clock,re1,re2,re3,re4,re5,re6,re7,re8,read_address1,read_address2,read_address3,read_address4,read_a
ddress5,read_address6,read_address7,read_address8,sinputx1,sinputx2,sinputx3,sinputx4,sinputx5,sinp
utx6,sinputx7,sinputx8)
            BEGIN
                        IF (clock'event AND clock = '1') THEN
                                    IF (re1 = '1') THEN
                          Q1<=sinputx1(to_integer(unsigned(read_address1)));
                                    end if;
                                    if re2 = '1' then
```

```
                    Q2<=sinputx2(to_integer(unsigned(read_address2)));
                         end if;
                         if re3 = '1' then
                    Q3<=sinputx3(to_integer(unsigned(read_address3)));
                         end if;
                         if re4 = '1' then
                    Q4<=sinputx4(to_integer(unsigned(read_address4)));
                         end if;
                         if re5 = '1' then
                    Q5<=sinputx5(to_integer(unsigned(read_address5)));
                         end if;
                         if re6 = '1' then
                    Q6<=sinputx6(to_integer(unsigned(read_address6)));
                         end if;
                         if re7 = '1' then
                    Q7<=sinputx7(to_integer(unsigned(read_address7)));
                         end if;
                         if re8 = '1' then
                    Q8<=sinputx8(to_integer(unsigned(read_address8)));
                         end if;
--                  index <= read_address;
                         END IF;
        END PROCESS;
end archi;
```

C.2 Block *decode_inputs210241* (p 97, section B.1.3.1, block 3's component)

All the signal names in the code are same as that described in the block except *reset1(1023)* in the block replaces *resett* in the code; *s1* and *s2* in the block replace *sel1* and *sel2* respectively in the code; *addressi* in the block replaces *address1* in the code.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;
use work.myram2.all;
ENTITY decode_inputs210241 IS
PORT
        (
                clock,reset,resett,sel1,sel2 : in std_logic;
                datain : in std_logic_vector(DATA_WIDTHd-1 downto 0):=(others=>'0');
                address,address1 : in std_logic_vector(ADDRESS_WIDTHd-1 downto 0);
                dataout1,dataout2 : out std_logic_vector(DATA_WIDTHd -1 downto 0):=(others=>'0')
        );
END decode_inputs210241;

ARCHITECTURE archi OF decode_inputs210241 IS
SIGNAL ram_block1,ram_block2 : RAM1:= (others => (others=>'0'));
signal  address2,address3 :  std_logic_vector(ADDRESS_WIDTHd downto 0);
signal s3 : std_logic :='0';
BEGIN

   address2 <= s3 & address;
   address3<= s3 & address1;
                PROCESS (clock,reset,sel1,datain)
        BEGIN
          if (reset='1') THEN
                                 IF (clock'event AND clock = '1') THEN
                  if (sel1 = '0') then
                         ram_block1(to_integer(unsigned(address2))) <= datain;
                        elsif (sel1 = '1') then
                         ram_block2(to_integer(unsigned(address2))) <= datain;
                        end if;
                        END IF;
                        end if;
        END PROCESS;
        PROCESS (clock,sel2,resett,ram_block1,ram_block2,address3)
        BEGIN
                IF (clock'event AND clock = '1')and(resett='1') THEN
                        if (sel2 = '0') then
      if(address3 ="00000000000") then
                  dataout1<=ram_block1(to_integer(unsigned(address3)));
                  dataout2<=ram_block1(to_integer(unsigned( address3)));
                        else
                        dataout1<=ram_block1(to_integer(unsigned(address3)));
                  dataout2<=ram_block1(to_integer(unsigned("10000000000" - address3)));
                        end if;
                  elsif ( sel2 = '1') then
                        if(address3 ="00000000000") then
                  dataout1<=ram_block2(to_integer(unsigned(address3)));
                  dataout2<=ram_block2(to_integer(unsigned( address3)));
      else
                  dataout1<=ram_block2(to_integer(unsigned(address3)));
                  dataout2<=ram_block2(to_integer(unsigned("10000000000" - address3)));
                        end if;
                end if;
                end if;
        END PROCESS;

        END archi;
```

C.2.1 Packages used in C.2

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
package myram2 is
```

```
constant ADDRESS_WIDTHd         : integer := 10;
constant DATA_WIDTHd     : integer := 22;
TYPE RAM IS ARRAY(0 TO 2 ** ADDRESS_WIDTHd - 1) OF std_logic_vector(DATA_WIDTHd -
1 DOWNTO 0);
TYPE RAM1 IS ARRAY(0 TO 2 ** ADDRESS_WIDTHd) OF std_logic_vector(DATA_WIDTHd - 1
DOWNTO 0);

end myram2;
```

C.3 Block *complex_multiplier1024* (p 99, section B.1.3.1 block 4's component)

All the input and output names in the code are the same as that in the block except *zr, zi, indexor, indexoi* in the code are replaced by *zr1, zi1, indor* and *indoi* in the block description.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
entity complex_multiplier1024 is
port(xr,xi,yr,yi : in std_logic_vector(21 downto 0):=(others=>'0');
    zr,zi : out std_logic_vector(44 downto 0):=(others=>'0');
    clk,reset : in std_logic;
          blank : in std_logic:='0';
    indexi : in std_logic_vector(9 downto 0);
    indexor,indexoi : out std_logic_vector(9 downto 0));
end complex_multiplier1024;
architecture archi of complex_multiplier1024 is
signal sxr,sxi,syr,syi : signed(21 downto 0):=(others=>'0');
signal szr1,szr2,szi1,szi2 : signed(44 downto 0):=(others=>'0');
signal sindr,sindi : std_logic_vector(9 downto 0);
begin
sxr <= signed(xr);
sxi <= signed(xi);
syr <= signed(yr);
syi <= signed(yi);
process(clk,reset,sxr,sxi,syr,syi,blank,szr1,szr2,indexi,sindr)
begin
if reset = '1' then
if (clk'event and clk = '1') then
if blank = '0' then
  szr1 <= resize((sxr * syr),45);
  szr2 <= resize((sxi * syi),45);
elsif blank = '1' then
  szr1 <=(others=>'0');
  szr2 <=(others=>'0');
end if;
  sindr <= indexi;
  zr <= std_logic_vector(szr1 - szr2);
```

```
   indexor <= sindr;
 end if;
 end if;
 end process;
process(clk,reset,sxr,sxi,syr,syi,blank,szi1,szi2,indexi,sindi)
begin
if reset = '1' then
if (clk'event and clk = '1') then
if blank = '0' then
   szi1 <= resize((sxr * syi),45);
   szi2 <= resize((sxi * syr),45);
elsif blank = '1' then
   szi1 <=(others=>'0');
   szi2 <=(others=>'0');
end if;
   sindi <= indexi;
   zi <= std_logic_vector(szi2 + szi1);
   indexoi <= sindi;
 end if;
 end if;
 end process;
 end archi;
```

C.4 Block *demux4ac1024* (p 99, section B.1.3.1 block 4's component)

The inputs and outputs in the code *din, indexin, sel, reset, dout1, dout2, index1, index2* are the same as *zr1, indor, sel2, reset1, zr11, zr12, indor1* and *indor2* respectively in the block description.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use work.myram1024.all;
entity demux4ac1024 is
port(din : in std_logic_vector(44 downto 0):=(others=>'0');
    indexin : in std_logic_vector(ADDRESS_WIDTH - 1 downto 0);
    sel,reset : in std_logic;
    dout1,dout2 : out std_logic_vector(44 downto 0):=(others=>'0');
    index1,index2 : out std_logic_vector(ADDRESS_WIDTH - 1 downto 0));
end demux4ac1024;
architecture archi of demux4ac1024 is
begin
process(sel,din,indexin,reset)
begin
case (reset) is
when '1' =>
case(sel) is
when '0' => dout1 <= din;
        index1 <= indexin;
        dout2<= (others=>'0');
        index2 <= (others=>'0');
when '1' => dout2 <= din;
        index2 <= indexin;
        dout1 <= (others=>'0');
        index1 <= (others=>'0');
when others => null;
end case;
when others => null;
end case;
end process;
end archi;
```

C.5 Block *acm1024* (p 99, section B.1.3.1 block 4's component)

There are four such identical units described in p99, under section B.1.3.1. We will only relate the signals of the first unit as the corresponding signals in the other units with respect to the first unit is known under the block description so they can be related with the signals in the code. The inputs and outputs in the code *clock, D1, write_address, we, re, Q2* and *index* are the same as *clk, zr11, indor1, reset3, reade, oacr1* and *indexr1* respectively of the first *acm1024* described.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use work.myram1024.all;
entity acm1024 is
port(
    clock                       : IN  std_logic;
                D1      : IN  std_logic_vector(44 DOWNTO 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTH - 1 DOWNTO 0);
                ---read_address                 : IN  std_logic_vector(ADDRESS_WIDTH - 1 DOWNTO
0);
                we        : IN  std_logic;
                re                          : IN  std_logic;
                Q2                          : OUT std_logic_vector(DATA_WIDTH - 1 DOWNTO
0):=(others=>'0');
          index      : out std_logic_vector(ADDRESS_WIDTH - 1 downto 0)
        );
end acm1024;
architecture acm1024 of acm1024 is
component counter_10bitac
port(clk,reset:in std_logic;
   q1: out std_logic_vector(9 downto 0));
end component;
component testacm1024
PORT
        (
                clock                       : IN  std_logic;
                D1      : IN  std_logic_vector(44 DOWNTO 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTH - 1 DOWNTO 0);
                read_address,read_address1                  : IN  std_logic_vector(ADDRESS_WIDTH
- 1 DOWNTO 0);
                we          : IN  std_logic;
                re                          : IN  std_logic;
                Q2                          : OUT std_logic_vector(DATA_WIDTH - 1 DOWNTO
0):=(others=>'0');
                index       : out std_logic_vector(ADDRESS_WIDTH - 1 downto 0)
        );
END component;
signal read_address             : std_logic_vector(ADDRESS_WIDTH - 1 DOWNTO 0);
signal s1: std_logic_vector(9 downto 0);
begin
u0:counter_10bitac port map(clock,we,s1);
u1:testacm1024 port map(clock,D1,write_address,read_address,s1,we,re,Q2,index);
u2:counter_10bitac port map(clock,re,read_address);
end acm1024;
---data and write address should arrive 2 clock pulses after we goes high.
```

C.5.1 Component of *acm1024*: *counter_10bitac*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity counter_10bitac is
port(clk,reset:in std_logic;
     q1: out std_logic_vector(9 downto 0));
end counter_10bitac;
architecture counter_10bit of counter_10bitac is
signal count,s1: std_logic_vector(9 downto 0):="0000000000";
signal b: std_logic_vector(9 downto 0) :="0000000001";
begin
process_count: process(clk,reset,count)
begin
   if(reset = '0') then
          count <= "0000000000";
          s1 <= "0000000000";
   elsif(reset ='1') then
     if(clk'event and clk='1') then
        s1<= "0000000000";
        count<= (count+b);
        s1<= count;
     end if;
   end if;
end process;
q1<=s1;
```

```
---q2<= "111" - s1;
end counter_10bit;
```

C.5.2 Component of *acm1024*: *testacm1024*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;
use work.myram1024.all;
ENTITY testacm1024 IS
PORT
        (
                clock                   : IN  std_logic;
                D1      : IN  std_logic_vector(44 DOWNTO 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTH - 1 DOWNTO 0);
                read_address,read_address1                      : IN  std_logic_vector(ADDRESS_WIDTH
- 1 DOWNTO 0);
                we        : IN  std_logic;
                re                              : IN  std_logic;
                Q2                              : OUT std_logic_vector(DATA_WIDTH - 1 DOWNTO
0):=(others=>'0');
                index      : out std_logic_vector(ADDRESS_WIDTH - 1 downto 0)
        );
END testacm1024;

ARCHITECTURE archi OF testacm1024 IS
SIGNAL D1s,D2s : signed(68 downto 0):=(others=>'0');
signal Q1         :std_logic_vector(DATA_WIDTH - 1 DOWNTO 0):=(others=>'0');
SIGNAL ram_temp : RAM:= (others => (others=>'0'));
BEGIN
D1s <= resize(signed(D1),69);
D2s <= resize(signed(Q1),69);
        PROCESS (clock,we,D1s,D2s,write_address,read_address1,ram_temp)
        BEGIN
                IF (clock'event AND clock = '1') THEN
                        IF (we = '1') THEN
                         ram_temp(to_integer(unsigned(write_address)))<= D1s + D2s;
                         Q1 <= std_logic_vector(ram_temp(to_integer(unsigned(read_address1))));

                        END IF;
                        END IF;
        END PROCESS;
        PROCESS (clock,re,read_address,ram_temp)
        BEGIN
                IF (clock'event AND clock = '1') THEN
                        IF (re = '1') THEN
                  Q2<=std_logic_vector(ram_temp(to_integer(unsigned(read_address))));
                  index <= read_address;
```

```
            END IF;
            END IF;
    END PROCESS;

    END archi;
```

C.5.3 Packages used in C.5

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;

package myram1024  is

constant ADDRESS_WIDTH          : integer := 10;
constant DATA_WIDTH       : integer := 69;
TYPE RAM IS ARRAY(0 TO 2 ** ADDRESS_WIDTH - 1) OF signed(DATA_WIDTH - 1
DOWNTO 0);

end myram1024;
```

C.6 Block *mod_xy_square* (p 102, section B.1.3.1 block 5's component)

All the input and output names in the code are same as that described under the block except *modxsq, modysq, indexor, indexoi* in the code are replaced by *zr1, zi1, indor* and *indoi* respectively in the block description.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
entity mod_xy_square is
port(xr,xi,yr,yi : in std_logic_vector(21 downto 0):=(others=>'0');
  modxsq,modysq : out std_logic_vector(44 downto 0):=(others=>'0');
  clk,reset : in std_logic;
      blank : in std_logic:='0';
  indexi : in std_logic_vector(9 downto 0);
  indexor,indexoi : out std_logic_vector(9 downto 0));
end mod_xy_square;
architecture archi of mod_xy_square is
signal sxr,sxi,syr,syi : signed(21 downto 0):=(others=>'0');
signal smodxsq1,smodxsq2,smodysq1,smodysq2 : signed(44 downto 0):=(others=>'0');
signal sindr,sindi : std_logic_vector(9 downto 0);
begin
sxr <= signed(xr);
sxi <= signed(xi);
syr <= signed(yr);
syi <= signed(yi);
```

```
process(clk,reset,sxr,sxi,syr,syi,blank,smodxsq1,smodxsq2,indexi,sindr)
begin
if reset = '1' then
if (clk'event and clk = '1') then
if blank = '0' then
   smodxsq1 <= resize((sxr * sxr),45);
   smodxsq2 <= resize((sxi * sxi),45);
elsif blank = '1' then
   smodxsq1 <=(others=>'0');
   smodxsq2 <=(others=>'0');
end if;
   sindr <= indexi;
   modxsq <= std_logic_vector(smodxsq1 + smodxsq2);
   indexor <= sindr;
 end if;
 end if;
 end process;
process(clk,reset,sxr,sxi,syr,syi,blank,smodysq1,smodysq2,indexi,sindi)
begin
if reset = '1' then
if (clk'event and clk = '1') then
if blank = '0' then
   smodysq1 <= resize((syr * syr),45);
   smodysq2 <= resize((syi * syi),45);
elsif blank = '1' then
   smodysq1 <=(others=>'0');
   smodysq2 <=(others=>'0');
end if;
   sindi <= indexi;
   modysq <= std_logic_vector(smodysq2 + smodysq1);
   indexoi <= sindi;
 end if;
 end if;
 end process;
 end archi;
```

C.7 Block *mux4ac8* (p 104, section B.1.4.1 block 4)

The inputs and outputs in the code *din1- din8, indexin1 - indexin8, sel1 - sel8, selo, dout* and *indexout* are same as *oacr0 – oacr7, indexr0 – indexr7, weoutr0 – weoutr7, selo1, doutr, waddr* respectively in the block description.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use work.myram1024.all;
entity mux4ac8 is
port(din1,din2,din3,din4,din5,din6,din7,din8 : in std_logic_vector(68 downto 0):=(others=>'0');
    indexin1,indexin2,indexin3,indexin4,indexin5,indexin6,indexin7,indexin8 : in
std_logic_vector(ADDRESS_WIDTH - 1 downto 0);
    sel1,sel2,sel3,sel4,sel5,sel6,sel7,sel8 : in std_logic:='0';
          selo : out std_logic:='0';
    dout : out std_logic_vector(68 downto 0):=(others=>'0');
    indexout : out std_logic_vector(ADDRESS_WIDTH - 1 downto 0));
end mux4ac8;
architecture archi of mux4ac8 is
signal sel :  std_logic_vector(7 downto 0):=(others=>'0');
begin
sel(0) <= sel1;
sel(1) <= sel2;
sel(2) <= sel3;
sel(3) <= sel4;
sel(4) <= sel5;
sel(5) <= sel6;
sel(6) <= sel7;
sel(7) <= sel8;
process(sel,din1,din2,din3,din4,din5,din6,din7,din8,indexin1,indexin2,indexin3,indexin4,indexin5,inde
xin6,indexin7,indexin8)
begin
case(sel) is
when "00000000" => dout <= (others=>'0');
              indexout <= (others=>'0');
                                          selo <= '0';
when "00000001" => dout <= din1;
              indexout <= indexin1;
              selo <= '1';
when "00000010" => dout <= din2;
              indexout <= indexin2;
              selo <= '1';
when "00000100" => dout <= din3;
              indexout <= indexin3;
              selo <= '1';
when "00001000" => dout <= din4;
              indexout <= indexin4;
              selo <= '1';
when "00010000" => dout <= din5;
              indexout <= indexin5;
              selo <= '1';
when "00100000" => dout <= din6;
              indexout <= indexin6;
              selo <= '1';
when "01000000" => dout <= din7;
              indexout <= indexin7;
              selo <= '1';
when "10000000" => dout <= din8;
              indexout <= indexin8;
              selo <= '1';
```

```
when others => null;
end case;
end process;
end archi;
```

C.8 Block *acmwin1* (p 112, section B.1.5.1 under component of *zwindow1* (p 110), which is a component of block 2 (p 110))

The inputs and outputs in the code *clock, clock1, D1, write_address, we, enable, reade, Q2* and *index* are same as *clock1, clock, s2, s1, reset1, enable, reade, Q2, index* respectively in the block description.

```
library ieee;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use work.ramwin.all;
entity acmwin1 is
port(
    clock,clock1                        : IN  std_logic;
               D1      : IN  std_logic_vector(63 DOWNTO 0):=(others => '0');
               write_address  : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
               we         : IN  std_logic;
               enable,reade                  : IN  std_logic;
               Q2                          : OUT std_logic_vector(1 DOWNTO 0):=(others=>'0');
    index      : out std_logic_vector(ADDRESS_WIDTHW - 1 downto 0)
        );
end acmwin1;
architecture acmwin of acmwin1 is
component counter_10bitac
port(clk,reset:in std_logic;
   q1: out std_logic_vector(9 downto 0));
end component;
component testwin
PORT(
               clock                    : IN  std_logic;
               D1      : IN  std_logic_vector(63 DOWNTO 0):=(others => '0');
               write_address  : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
               read_address   : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
               we         : IN  std_logic;
               re,enable   : IN  std_logic;
               Q2                          : OUT std_logic_vector(1 DOWNTO 0):=(others=>'0');
               index       : out std_logic_vector(ADDRESS_WIDTHW - 1 downto 0)
        );
END component;
component dff1
port(d: in std_logic;
   clk : in std_logic;
   q : out std_logic:='0');
```

```
end component;
component windowarraytopn1
port(
     clock,clock1                               : IN  std_logic;
                    D1       : IN  std_logic_vector(1 DOWNTO 0):=(others => '0');
                    write_address  : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                    we         : IN  std_logic;
                    re                          : IN  std_logic;
                    Q2                            : OUT std_logic_vector(1 DOWNTO 0):=(others=>'0');
     index       : out std_logic_vector(ADDRESS_WIDTHW - 1 downto 0)
          );
end component;
signal  Q2s     : std_logic_vector(1 DOWNTO 0):=(others=>'0');
signal read_address,index1 : std_logic_vector(ADDRESS_WIDTHW -1 downto 0);
signal re : std_logic;
signal e : std_logic_vector(1025 downto 0):=(others=>'0');
begin
   e(0) <= enable;
   re <= e(1024);
u1:testwin port map(clock,D1,write_address,read_address,we,re,enable,Q2s,index1);
u : for i in 0 to 1024 generate
u2: dff1 port map(e(i),clock,e(i+1));
end generate;
u3: counter_10bitac port map(clock,re,read_address);
u4: windowarraytopn1 port map(clock,clock1,Q2s,index1,e(1025),reade,Q2,index);
end acmwin;
```

C.8.1 Component of *acmwin1*: *testwin*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;
use work.ramwin.all;
ENTITY testwin IS
PORT
         (
                    clock                       : IN  std_logic;
                    D1       : IN  std_logic_vector(63 DOWNTO 0):=(others => '0');
                    write_address  : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                    read_address   : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                    we         : IN  std_logic;
                    re,enable   : IN  std_logic;
                    Q2                            : OUT std_logic_vector(1 DOWNTO 0):=(others=>'0');
                    index       : out std_logic_vector(ADDRESS_WIDTHW - 1 downto 0)
         );
END testwin;

ARCHITECTURE archi OF testwin IS
```

```
SIGNAL D1s: signed(65 downto 0):=(others=>'0');
SIGNAL ram_temp : RAM:= (others => (others=>'0'));
signal result : signed(DATA_WIDTHW - 1 DOWNTO 0):=(others=>'0');
signal i : std_logic_vector(ADDRESS_WIDTHW - 1 downto 0):="0000000001";
signal count : std_logic_vector(ADDRESS_WIDTHW - 1 downto 0):="0000000001";
signal Q3 : std_logic_vector(DATA_WIDTHW - 1 DOWNTO 0):=(others=>'0');

BEGIN
D1s <= resize(signed(D1),66);
        PROCESS (clock,we,D1s,write_address,ram_temp)
        BEGIN
                IF (clock'event AND clock = '1') THEN
                        IF (we = '1') THEN
                         ram_temp(to_integer(unsigned(write_address)))<= D1s;
                        END IF;
                        END IF;
        END PROCESS;
        PROCESS (clock,re,result)
        BEGIN
                IF (clock'event AND clock = '1') THEN
                        IF (re = '1') THEN
                        Q3 <= std_logic_vector(result);
                        if ram_temp(to_integer(unsigned(read_address)))(65 downto 0) >=
resize(signed(Q3(65 downto 4)),66) then
                        Q2 <= std_logic_vector(to_signed(1,2));
                        elsif ram_temp(to_integer(unsigned(read_address)))(65 downto 0) <
resize(signed(Q3(65 downto 4)),66) then
                        Q2 <= (others=>'0');
                        end if;
                  index <= read_address;
                        END IF;
                        END IF;
        END PROCESS;
PROCESS(clock,enable,result,ram_temp,i)
         begin
         if(clock'event and clock = '1') then
         if (enable='1') then
         if ram_temp(to_integer(unsigned(i)))(65 downto 0)> result then
         result <= ram_temp(to_integer(unsigned(i)))(65 downto 0);
         else
         result <= result;
         end if;
         end if;
         end if;
         end process;
process(clock,enable,count)
begin
if (clock'event and clock = '1') then
if (enable = '1') then
```

```
count <= count + "0000000001";
if count = "1111111111" then
   count <= "0000000001";
end if;
i <= count;
end if;
end if;
end process;
          end archi;
```

C.8.2 Component of *acmwin1*: *dff1*

```
library ieee;
use ieee.std_logic_1164.all;
entity dff1 is
port(d: in std_logic;
    clk : in std_logic;
    q : out std_logic:='0');
end dff1;
architecture dff of dff1 is
begin
process(clk)
begin
if(clk='1') and (clk'event) then
q<=d;
end if;
end process;
end dff;
```

C.8.3 Component of *acmwin1*: *windowarraytopn1*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use work.ramwin.all;
entity windowarraytopn1 is
port(
     clock,clock1                          : IN  std_logic;
                  D1      : IN  std_logic_vector(1 DOWNTO 0):=(others => '0');
                  write_address  : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                  we        : IN  std_logic;
                  re                       : IN  std_logic;
                  Q2                         : OUT std_logic_vector(1 DOWNTO 0):=(others=>'0');
     index      : out std_logic_vector(ADDRESS_WIDTHW - 1 downto 0)
          );
end windowarraytopn1;
architecture windowarraytop of windowarraytopn1 is
```

```
component counter_10bitac
port(clk,reset:in std_logic;
    q1: out std_logic_vector(9 downto 0));
end component;
component windowarrayn1
PORT
        (
                clock,clock1                    : IN  std_logic;
                D1      : IN  std_logic_vector(1 DOWNTO 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                read_address   : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                we        : IN  std_logic;
                re                          : IN  std_logic;
                Q2                            : OUT std_logic_vector(1 DOWNTO 0):=(others=>'0');
     index      : out std_logic_vector(ADDRESS_WIDTHW - 1 downto 0)
        );
END component;

--signal          Q3      : std_logic_vector(DATA_WIDTHW - 1 DOWNTO 0):=(others=>'0');
signal read_address : std_logic_vector(ADDRESS_WIDTHW -1 downto 0);
begin

u1:windowarrayn1 port map(clock,clock1,D1,write_address,read_address,we,re,Q2,index);
u2: counter_10bitac port map(clock1,re,read_address);
end windowarraytop;
```

C.8.3.1 Component of ***windowarraytopn1: windowarrayn1***

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;
use work.ramwin.all;
ENTITY windowarrayn1 IS
PORT
        (
                clock,clock1                    : IN  std_logic;
                D1      : IN  std_logic_vector(1 DOWNTO 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                read_address   : IN  std_logic_vector(ADDRESS_WIDTHW - 1 DOWNTO 0);
                we        : IN  std_logic;
                re                          : IN  std_logic;
                Q2                            : OUT std_logic_vector(1 DOWNTO 0):=(others=>'0');
     index      : out std_logic_vector(ADDRESS_WIDTHW - 1 downto 0)
        );
END windowarrayn1;

ARCHITECTURE archi OF windowarrayn1 IS
SIGNAL ram_temp : RAM2:= (others => (others=>'0'));
SIGNAL i : std_logic_vector(9 downto 0):="0000000000";

BEGIN
        PROCESS (clock,we,D1,write_address,ram_temp)
        BEGIN
                IF (clock'event AND clock = '1') THEN
                        IF (we = '1') THEN
                         ram_temp(to_integer(unsigned(write_address)))<= signed(D1);
                        END IF;
                        END IF;
        END PROCESS;
        PROCESS (clock1,re)
        BEGIN
                IF (clock1'event AND clock1 = '1') THEN
                        IF (re = '1') THEN
                  Q2<=std_logic_vector(ram_temp(to_integer(unsigned(read_address))));
                  index <= read_address;
                        END IF;
                        END IF;
        END PROCESS;

        END archi;
```

C.8.4 Packages used in C.8

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;

package ramwin  is

constant ADDRESS_WIDTHW       : integer := 10;
constant DATA_WIDTHW    : integer := 66;
constant DATA_WIDTH1W  : integer := 31;
TYPE RAM IS ARRAY(0 TO 2 ** ADDRESS_WIDTHW - 1) OF signed(DATA_WIDTHW - 1
DOWNTO 0);
TYPE RAM1 IS ARRAY(0 TO 2 ** ADDRESS_WIDTHW) OF signed(DATA_WIDTHW - 1
DOWNTO 0);
TYPE RAM2 IS ARRAY(0 TO 2 ** ADDRESS_WIDTHW) OF signed(1 DOWNTO 0);

end ramwin;
```

Note that the counter_10bitac used in several places under C.8 is already given inC.5.1.

C.9 Block *acm_mag1* (p 119, section B.1.5.1, under component of *gxy1* (p 119), which is a component of block 4 (p 118))

The inputs and outputs in the code *clock, clock1, D1, write_address, we, enable, Q2, Q3* and *index* in the code are same as *clock, clock1, D1, write_address, we, enable, smx2, sxmax2, open* respectively in the block description.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use work.ramacm.all;
entity acm_mag1 is
port(
    clock,clock1                        : IN  std_logic;
                D1      : IN  std_logic_vector(30 downto 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTHac - 1 DOWNTO 0);
                --read_address         : IN  std_logic_vector(ADDRESS_WIDTHac - 1 DOWNTO 0);
                we         : IN  std_logic;
                enable  : IN  std_logic;
                Q2                             : OUT std_logic_vector(DATA_WIDTH1ac - 1 DOWNTO
0):=(others=>'0');
          Q3                            : OUT std_logic_vector(DATA_WIDTH1ac - 1 DOWNTO
0):=(others=>'0');
    index       : out std_logic_vector(ADDRESS_WIDTHac - 1 downto 0)
        );
end acm_mag1;
architecture acm_mag of acm_mag1 is
component counter_10bitac
port(clk,reset:in std_logic;
   q1: out std_logic_vector(9 downto 0));
end component;
component testacm_mag1
PORT
        (
                clock,clock1                       : IN  std_logic;
                D1      : IN  std_logic_vector(30 downto 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTHac - 1 DOWNTO 0);
                read_address,read_address1                       : IN
std_logic_vector(ADDRESS_WIDTHac - 1 DOWNTO 0);
                we         : IN  std_logic;
                re,enable         : IN  std_logic;
                Q2                             : OUT std_logic_vector(DATA_WIDTH1ac - 1 DOWNTO
0):=(others=>'0');
                Q3                             : OUT std_logic_vector(DATA_WIDTH1ac - 1 DOWNTO
0):=(others=>'0');
                index       : out std_logic_vector(ADDRESS_WIDTHac - 1 downto 0)
        );
END component;
component dff1
port(d: in std_logic;
   clk : in std_logic;
   q : out std_logic:='0');
```

```
end component;
signal read_address : std_logic_vector(ADDRESS_WIDTHac -1 downto 0);
signal re : std_logic;
signal s1: std_logic_vector(9 downto 0);
signal e : std_logic_vector(1024 downto 0):=(others=>'0');
begin
e(0) <= enable;
re <= e(1024);
u0:counter_10bitac port map(clock,we,s1);
u1:testacm_mag1 port map(clock,clock1,D1,write_address,read_address,s1,we,re,enable,Q2,Q3,index);
u : for i in 0 to 1023 generate
u2: dff1 port map(e(i),clock1,e(i+1));
end generate;
u3: counter_10bitac port map(clock1,re,read_address);
end acm_mag;
```

C.9.1 Component of ***acm_mag1: testacm_mag1***

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;
use work.ramacm.all;
ENTITY testacm_mag1 IS
PORT
        (
                clock,clock1                    : IN  std_logic;
                D1      : IN  std_logic_vector(30 downto 0):=(others => '0');
                write_address  : IN  std_logic_vector(ADDRESS_WIDTHac - 1 DOWNTO 0);
                read_address,read_address1                      : IN
std_logic_vector(ADDRESS_WIDTHac - 1 DOWNTO 0);
                we        : IN  std_logic;
                re,enable   : IN  std_logic;
                Q2                              : OUT std_logic_vector(DATA_WIDTH1ac - 1 DOWNTO
0):=(others=>'0');
                Q3                              : OUT std_logic_vector(DATA_WIDTH1ac - 1 DOWNTO
0):=(others=>'0');
                index      : out std_logic_vector(ADDRESS_WIDTHac - 1 downto 0)
        );
END testacm_mag1;

ARCHITECTURE archi OF testacm_mag1 IS
SIGNAL D1s : signed(30 downto 0):=(others=>'0');
SIGNAL ram_temp : RAM:= (others => (others=>'0'));
signal result : signed(DATA_WIDTH1ac - 1 DOWNTO 0):=(others=>'0');--(RAM1:= (others =>
(others=>'0'));
signal i,count : std_logic_vector(ADDRESS_WIDTHac - 1 downto 0):="0000000001";
BEGIN
D1s <= resize(signed(D1),31);
```

```
        PROCESS (clock,we,D1s,write_address,read_address1,ram_temp)
        BEGIN
                IF (clock'event AND clock = '1') THEN
                        IF (we = '1') THEN
                         ram_temp(to_integer(unsigned(write_address)))<= D1s;
                        END IF;
                        END IF;
        END PROCESS;
        PROCESS (clock1,re,result,read_address)
        BEGIN
                IF (clock1'event AND clock1 = '1') THEN
                        IF (re = '1') THEN
                    Q2<=std_logic_vector(ram_temp(to_integer(unsigned(read_address))));
                    index <= read_address;
                        Q3 <= std_logic_vector(result);
                        END IF;
                        END IF;
        END PROCESS;
         PROCESS(clock1,enable,result,ram_temp,i)
         begin
         if (enable='1') then
         if(clock1'event and clock1 = '1') then
         if ram_temp(to_integer(unsigned(i)))> result then
         result <= ram_temp(to_integer(unsigned(i)));
         else
         result <= result;
         end if;
         end if;
         end if;
         end process;

         process(clock1,enable,count)
begin
if (enable = '1') then
if (clock1'event and clock1 = '1') then
count <= count + "0000000001";
if count = "1111111111" then
   count <= "0000000001";
end if;
i <= count;
end if;
end if;
end process;

end archi;
```

C.9.2 Packages used in C.9

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
use ieee.std_logic_unsigned.all;

package ramacm  is

constant ADDRESS_WIDTHac        : integer := 10;
constant DATA_WIDTHac    : integer := 31;
constant DATA_WIDTH1ac  : integer := 31;
TYPE RAM IS ARRAY(0 TO 2 ** ADDRESS_WIDTHac - 1) OF signed(DATA_WIDTHac - 1
DOWNTO 0);
TYPE RAM1 IS ARRAY(0 TO 2 ** ADDRESS_WIDTHac) OF signed(DATA_WIDTHac - 1
DOWNTO 0);
TYPE RAM2 IS ARRAY(0 TO 2 ** ADDRESS_WIDTHac) OF signed(1 DOWNTO 0);

end ramacm;
```

Note the other components of C.9 (acm_mag1) are already given in C.5.1 (counter_10bitac) and C.8.2 (dff1).

C.10 Block *xycircular* (p123, block 8 (iii))

The inputs and outputs in the code *clock, reset, xr, xi, yr, yi, indexin, gxw, gyw, coszw, sinzw, LHCreal, LHCimag, RHCreal, RHCimag* and *indexout* in the code are same as *clock, spassxy1, sxr1, sxi1, syr1, syi1, saddrout1, gxoutf1, gyoutf1, cosnf1, sinnf1, LHCreal1, LHCimag1, RHCreal1, RHCimag1, indexout1* respectively in the block description.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity xycircular is
port(clock,reset : in std_logic;
            xr,xi,yr,yi : in std_logic_vector(21 downto 0):=(others=>'0');
    indexin : in std_logic_vector(9 downto 0);
            gxw,gyw,coszw,sinzw : in std_logic_vector(21 downto 0):=(others =>'0');
            LHCreal,LHCimag : out std_logic_vector(47 downto 0):=(others=>'0');
            RHCreal,RHCimag : out std_logic_vector(47 downto 0):=(others=>'0');
            indexout : out std_logic_vector( 9 downto 0));
end xycircular;
architecture archi of xycircular is
component  phasegaineq
port(clock,reset : in std_logic;
    xr,xi,yr,yi : in std_logic_vector(21 downto 0):=(others=>'0');
            indexin : in std_logic_vector(9 downto 0);
    gxw,gyw,coszw,sinzw : in std_logic_vector(21 downto 0):=(others =>'0');
            indexout : out std_logic_vector(9 downto 0);
            xro,xio,yro,yio : out std_logic_vector(47 downto 0):=(others => '0'));
```

```
end component;
component phaseshift90
port(clock,reset : in std_logic;
          indexin : in std_logic_vector(9 downto 0);
   indexout : out std_logic_vector(9 downto 0);
   xr,xi,yr,yi : in std_logic_vector(47 downto 0):=(others=>'0');
          LHCreal,LHCimag : out std_logic_vector(47 downto 0):=(others=>'0');
          RHCreal,RHCimag : out std_logic_vector(47 downto 0):=(others=>'0'));
end component;
signal sindex :  std_logic_vector(9 downto 0);
signal sxr,sxi,syr,syi :  std_logic_vector(47 downto 0):=(others => '0');
begin
u1: phasegaineq port map(clock,reset,xr,xi,yr,yi,indexin,gxw,gyw,coszw,sinzw,sindex,sxr,sxi,syr,syi);
u2: phaseshift90 port
map(clock,reset,sindex,indexout,sxr,sxi,syr,syi,LHCreal,LHCimag,RHCreal,RHCimag);
end archi;
```

C.10.1 Component of *xycircular*: *phasegaineq*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity phasegaineq is
port(clock,reset : in std_logic;
   xr,xi,yr,yi : in std_logic_vector(21 downto 0):=(others=>'0');
          indexin : in std_logic_vector(9 downto 0);
   gxw,gyw,coszw,sinzw : in std_logic_vector(21 downto 0):=(others =>'0');
          indexout : out std_logic_vector(9 downto 0);
          xro,xio,yro,yio : out std_logic_vector(47 downto 0):=(others => '0'));
end phasegaineq;
architecture archi of phasegaineq is
signal syro,syio,sxrodly,sxiodly : signed(63 downto 0):=(others=>'0');
signal sxr,sxi,syr,syi,sgxw,sgyw,scoszw,ssinzw,sphdummy : signed(15 downto 0):=(others=>'0');
signal sxro,sxio,syro1,syro2,syio1,syio2: signed(63 downto 0):=(others=>'0');
signal sxro1,sxro2,sxio1,sxio2,syro11,syro12,syro21,syro22,syio11,syio12,syio21,syio22 : signed(31
downto 0):=(others=>'0');
signal sindexin1,sindexin2,sindexin3,sindexin4 : std_logic_vector(9 downto 0);
constant K : signed(15 downto 0):=(others=>'0');
constant K1 : signed(15 downto 0):= "0000000000000001";
constant phdummy : signed(21 downto 0):="0100000000000000000000";
begin
process(clock,reset,xr,xi,gxw,sxr,sxi,sxro,sxio,sgxw)
begin
if (clock'event and clock = '1') then
if (reset = '1') then
sxr <= resize(signed(xr(21 downto 7)),16);
sxi <= resize(signed(xi(21 downto 7)),16);
sgxw <= resize(signed(gxw(21 downto 7)),16);
```

```
sphdummy <= resize(phdummy(21 downto 7),16);
sxro1 <= sxr * sgxw;
sxro2 <=  K1 * sphdummy;                                ---K1 should be 16 bit unity
sxro <= sxro1 * sxro2;
sxio1 <= sxi * sgxw;
sxio2 <= K1 * sphdummy;
sxio <= sxio1 * sxio2;
sxrodly <= sxro;---(47 downto 0);
sxiodly <= sxio;---(47 downto 0);
end if;
end if;
end process;
process(clock,reset,yr,yi,gyw,coszw,sinzw,syro,syio,sgyw,syro11,syro12,syro21,syro22,syio11,syio12,s
yio21,syio22)
begin
if(clock'event and clock = '1') then
if(reset = '1') then
syr <= resize(signed(yr(21 downto 7)),16);
syi <= K - resize(signed(yi(21 downto 7)),16);
sgyw <= resize(signed(gyw(21 downto 7)),16);
scoszw <= resize(signed(coszw(21 downto 7)),16);
ssinzw <= resize(signed(sinzw(21 downto 7)),16);
syro11 <= syr * sgyw;
syro12 <= K1 * scoszw;
syro1 <= syro11 * syro12;
syro21 <= syi * sgyw;
syro22 <= K1 * ssinzw;
syro2 <= syro21 * syro22;
syio11 <= syr * sgyw;
syio12 <= K1 * ssinzw;
syio1 <= syio11 * syio12;
syio21 <= syi * sgyw;
syio22 <= K1 * scoszw;
syio2 <= syio21 * syio22;
syro <= syro1 - syro2;
syio <= syio1 + syio2;
end if;
end if;
end process;
process(clock,reset,indexin,sindexin1,sindexin2,sindexin3)
begin
if(clock'event and clock = '1') then
if(reset = '1') then
sindexin1 <= indexin;
sindexin2 <= sindexin1;
sindexin3 <= sindexin2;
sindexin4 <= sindexin3;
end if;
end if;
end process;
yro <= std_logic_vector(syro(47 downto 0));
yio <= std_logic_vector(syio(47 downto 0));
xro <= std_logic_vector(sxrodly(47 downto 0));
xio <= std_logic_vector(sxiodly(47 downto 0));
indexout <= sindexin4;
end archi;
```

C.10.2 Component of *xycircular*: *phaseshift90*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity phaseshift90 is
port(clock,reset : in std_logic;
            indexin : in std_logic_vector(9 downto 0);
    indexout : out std_logic_vector(9 downto 0);
    xr,xi,yr,yi : in std_logic_vector(47 downto 0):=(others=>'0');
            LHCreal,LHCimag : out std_logic_vector(47 downto 0):=(others=>'0');
            RHCreal,RHCimag : out std_logic_vector(47 downto 0):=(others=>'0'));
end phaseshift90;
architecture archi of phaseshift90 is
signal syRL,syiL,syrR,syiR,xrLR,xiLR : signed(47 downto 0) :=(others=>'0');
signal sindexout : std_logic_vector(9 downto 0);
constant K : signed(47 downto 0) := (others=>'0');
begin
process(clock,reset,xr,xi)
begin
if (clock'event and clock = '1') then
if reset = '1' then
xrLR <= signed(xr);
xiLR <= signed(xi);
sindexout <= indexin;
end if;
end if;
end process;
process(clock,reset,yr,yi)
begin
if(clock'event and clock = '1') then
if reset = '1' then
syrL <= signed(yi);
syiL <= K - signed(yr);
syrR <= K - signed(yi);
syiR <= signed(yr);
end if;
end if;
end process;
process(xrLR,xiLR,syrL,syiL,syrR,syiR,clock,reset)
```

```
begin
if(clock'event and clock = '1') then
if reset = '1' then
LHCreal <= std_logic_vector(xrLR + syrL);
LHCimag <= std_logic_vector(xiLR + syiL);
RHCreal <= std_logic_vector(xrLR + syrR);
RHCimag <= std_logic_vector(xiLR + syiR);
indexout <= sindexout;
end if;
end if;
end process;

end archi;
```

Important note: All these codes along with the whole design is there in the CD enclosed in this thesis and the description of the contents of the CD is already given in p 124 under Note.

REFERENCES

[1] Macquart, J.P. 2001, “Circular polarization in AGN”, Publications of the Astronomical Society of Australia, 19, 43-48.

[2] Bower, B.C., Wright, Melvyn C.H., Backer, D.C.,Falcke, H., 1999, “The linear polarization of Sagittarius A* II. VLA and BIMA polarimetry at 22, 43, and 86 GHz”, The Astrophysical Journal, 527, 851-855.

[3] Bower, B.C., Wright, Melvyn C.H., Falcke, H., Backer, D.C., 2003, “Interferometric detection of linear polarization from Sagittarius A* at 230 GHz”, The Astrophysical Journal, 588, 331-337.

[4] Bower, B.C., Backer, D.C., Zhao, J.H., Gross, M., Falcke, H., 1999, “The linear polarization of Sagittarius A* I. VLA Spectro-polarimetry at 4.8 and 8.4 GHz”, The Astrophysical Journal, 521, 582-586.

[5] Brentjens, M.A., de Bruyn, A.G., 2005, “Faraday rotation measure synthesis”, Astronomy & Astrophysics, 441, 1217-1228.

[6] de Bruyn, A.G., Brentjens, M.A., 2005, “Diffuse polarized emission associated with the Perseus cluster”, Astronomy & Astrophysics, 441, 931-947.

[7] Beck, R., Gaensler, B.M., 2004, “Observations of magnetic fields in the Milky Way and in nearby galaxies with a Square Kilometre Array”, New Astronomy Reviews, 48, 1289-1304.

[8] Tabatabaei, F. S., Krause, M., Fletcher, A., Beck, R., 2008, “High-resolution radio continuum survey of M 33. III. Magnetic fields”, Astronomy & Astrophysics, 490, 1005-1017.

[9] Arndt, F., Tucholke, U., Wriedt, T., 1984, “Broadband dual-depth E-plane corrugated square waveguide polarizer”, Electronics Letters, 20, 458-459.

[10] Boifot, A. M., Lier, E., Schaug-Pettersen, T., 1990, “Simple and broadband orthomode transducer (antenna feed)”, Microwaves, Antennas and Propagation, IEEE Proceedings, 137, 396-400.

[11] Simmons, A.J., 1955, “Phase shift by periodic loading of waveguide and its application to broad-band circular polarization”, IRE Transactions- Microwave Theory and Techniques, 3, 18-21.

[12] Chen, M.H., Tsandoulas, G.N., 1973, “A wide-band square-waveguide array polarizer”, IEEE Transactions on Antenna and Propagation, 21, 389-391.

[13] Davis, D., Digiondomenico, O.J., Kempic, J.A., 1967, “A new type of circularly polarized antenna element”, Antennas and Propagation Society International Synposium, 5, 26-33.

[14] Srikant, S., 1997, “A wide-band corrugated rectangular waveguide phase shifter for cryogenically cooled receivers”, Microwave and Guided Wave Letters, IEEE, 7, 150-152.

[15] Wade, P., 2003, “Septum polarizers and feeds”, W1GHz; weblink:
http://www.w1ghz.org/antbook/conf/SEPTUM.pdf

[16] Dunning, A., 2002, “Double ridged orthogonal mode transducer for the 16-26 GHz microwave band”, Proceedings of the Workshop on the Applications of Radio Science; weblink:
http://www.ips.gov.au/IPSHosted/NCRS/wars/wars2002/proceedings/comm-b/screen/dunning.pdf

[17] Tuccari, G., 2009, “Interference at a VLBI Station: Analysis and Mitigation”, Future Radio Frequencies and Feeds, Workshop, Wettzell, Germany; weblink:
http://www.fs.wettzell.de/veranstaltungen/vlbi/frff2009/Part6/Interference%20at%20a%20VLBI%20Station.pdf

[18] Cooley, J.W., Tukey, J.W. 1965, “An algorithm for the machine calculation of complex Fourier series”, Mathematics of Computation, 19, 297-301.

[19] Ruiz-Cruz, J.A., Montejo-Garai, J.R., Rebollar, J.M., Montesano, C.E., Martin, M.J., Naranjo-Masi, M., 2006, “Computer Aided Design of Wideband Orthomode Transducers based on the Boslashifot Junction”, IEEE MTT-S International Microwave Symposium Digest, 1173-1176.

[20] Thompson, A. R., Bagri, D. S. 1991, “A pulse calibration system for the VLBA”, Astronomical Society of the Pacific, 19, 55-59.

[21] Tuccari, G. 2008, “DBBC Development Status”, Proc. 5th IVS General Assembly, St Petersberg, .978-5-02-025332-2, 376.

[22] Tuccari, G. 2004, “Development of a Digital Base Band Converter (DBBC): Basic Elements and Preliminary Results In: New Technologies in VLBI”, Astronomical Society of the Pacific Conference Series, 306, 177-192.

[23] Hall, P.J., 2005, “The Square Kilometre Array: An engineering perspective”, Springer Publications, The

Netherlands, 17, 1-430.
[24] Jackson, J. D., 2009 , “Classical electrodynamics”, John Wiley and Sons (Asia) Pvt. Ltd.
[25] Griffiths, D.J., 1997, “Introduction to electrodynamics”, Prentice-Hall, Inc., Englewood Cliffs, N.J..
[26] Proakis, J. G., Manolakis, D., G., 2007, “Digital signal processing- principles, algorithms and applications”, Prentice-Hall, Inc. (now known as Pearson Education Inc.).
[27] Carroll, B., W., Ostlie, D., A., 1996, “Modern astrophysics”, Addison-Wesley Publishing Company, Inc..
[28] Thompson, A., R., Moran, J., M., Swenson, G., W., 2001, “Interferometry and synthesis in radio astronomy”, John Wiley & Sons, Inc..
[29] Lathi, B., P., 1998, “Modern digital and analog communication systems”, Oxford University Press, Inc..
[30] Kildal, P., S., 2000, “Foundations of antennas: A unified approach”, Studentlitteratur, Lund.
[31] Kalman, Dan, 2008, “The most marvelous theorem in mathematics”, The Journal of Online Mathematics and Its Applications, 8. weblink:
http://www.maa.org/joma/volume8/kalman/general.html
[32] Perley, Rick, 2009, “Measurements of C-Band EVLA antenna polarization”, EVLA Memo 131.
[33] Das, Koyel., Roy, A.L., Keller, R., Tuccari, G., 2010, “Conversion from linear to circular polarization in FPGA”, Astronomy and Astrophysics, 509,1-11.

Erklärung

Ich versichere, daß ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –,die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; daß diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; daß sie – abgesehen von unten angegebenen Teilpublikationen – noch nicht veröffentlicht worden ist sowie, daß ich eine solche Veröffentlichung vor Abschluß des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Andreas Eckart betreut worden.

-Koyel Das

Kolkata, den 10.04.2013

Teilpublikationen

Das, Koyel., Roy, A.L., Keller, R., Tuccari, G., 2010, “Conversion from linear to circular polarization in FPGA”, Astronomy and Astrophysics, 509, 1-11.

Declaration

I declare that I have independently written the thesis submitted by me, I have fully cited the sources and aids, I have identified each part of the thesis, including tables, maps, and illustrations, that have been drawn from other works in their wording or sense, as borrowed material. I declare that the thesis has not been presented to any other faculty or university, I declare that, except for the publication identified below, the thesis is not yet published and that I will not publish the thesis prior to the defence. I understand the regulations of the Doctoral Degree Regulations. The thesis submitted by me was supervised by Prof. Dr. Andreas Eckart.

-Koyel Das

Kolkata, 10.04.2013

Journal Publication

Das, Koyel., Roy, A.L., Keller, R., Tuccari, G., 2010, "Conversion from linear to circular polarization in FPGA", Astronomy and Astrophysics, 509, 1-11.

LEBENSLAUF

KOYEL DAS

PERSÖNLICHE DATEN

Email	koyel.aphy@gmail.com
Adresse:	Koyel Das, C/O P.R Das, Ashok Nagar Ichapur, P.O-Ichapur Nawabganj, Dist :24PGS(N), W.B -743144
Telefonnummer	+91- 9831665135
Geburtsdatum	25th September 1981
Nationalität	Indien

HOCHSCHULAUSBILDUNG

2007-2013 -Doktorandin im Bereich Experimentalphysik, Max-Planck-Institut für Radioastronomie-International Max Planck Research School für Astronomie und Astrophysik an der Universität Köln, Deutschland.

08/2005-03/2007 -MSc in Engineering in Advanced Techniques in Radio Astronomy and Space Science, Chalmers University of Technology, Schweden.

1998-2002-BE (Bachelor in Engineering) in Electronics, Nagpur University, Indien.
Spezialisierung in Optical Communications and Switching und Finite Automata Theory

BERUFSERFAHRUNG

September 2003 - März 2004:
Projekt Student, FPGA-System von Telemetrie-Abteilung GMRT, NCRA, Pune, Indien.

Februar - September 2003:
Projekt Student, FPGA Design: Pune University in Zusammenarbeit mit Cirrus Logic Private Limited, Pune, Indien.

FORSCHUNGSSCHWERPUNKTE

Stochastische Prozesse, Instrumentierung, Kosmologie, Interferometrie in der Astronomie, Elektrodynamik, Signale und Systeme.

PROGRAMMIERKENNTNISSE

Sprachen: MATLAB, C, VHDL, Verilog, VISUAL BASIC

Betriebssysteme: DOS, Win XP, UNIX, LINUX

PROJEKTE UND SEMINARE

PhD Projekt: Conversion from linear to circular polarization in real time in FPGA.

Master Projekt: An application of a new exact method of line radiative transfer: Determination of unknown level population.

B.E projekte und seminare:.

Projekte:

1. Supermarket – A software (Netzwerk-Software mit Visual Basic und SQL plus).
2. PCB-Design digitale Logik.

Seminar:

A study of ongoing research on DNA based Computers

PUBLICATION:

Das, Koyel, Roy, A.L., Keller, R. and Tuccari, G., 2010,

Conversion from linear to circular polarization in FPGA (A&A), 509, 1-11.

SOMMERSCHULEN UND KONFERENZEN

European Radio Interferometry School 2007, Deutschland.

Astrophysics at High Angular Resolution, 2008, Deutschland.

Radionet Engineering Workshop 2008, Deutschland.

Future Radio Frequencies and Feeds 2009, Deutschland.

REFERENTEN

Prof. Dr. J.H Black(Chalmers University of Technology, Schweden), email- john.black@chalmers.se

Dr. Alan Roy (Max-Planck-Institut für Radioastronomie, Deutschland), email- aroy@mpifr-bonn.mpg.de

Prof. Dr. Andreas Eckart (Physikalisches Institut, Universitaet zu Koeln, Deutschland), email- eckart@ph1.uni-koeln.de

Prof. Dr. Anton Zensus (Max-Planck-Institut für Radioastronomie, Deutschland), email- azensus@mpifr-bonn.mpg.de

Dr. Gary Smith Jonforsen (Ex-Fakultät, Chalmers University of Technology, Schweden), email-
gary.smith-jonforsen@saabgroup.com

Ich bestätige hiermit, dass ich alle Angaben nach bestem Wissen und Gewissen gemacht habe.

- Koyel Das