A Network Centric Receiver Architecture for Low Frequency Arrays

by

Peeyush Prasad

Thesis submitted to the Jawaharlal Nehru University
for the award of the degree of
Doctor of Philosophy

Raman Research Institute
Bangalore 560 080
India
September 2011

DECLARATION

I hereby declare that the work reported in this thesis has been independently carried out by me at the Raman Research Institute, Bangalore, under the supervision of Dr. C.R. Subrahmanya. The subject matter presented in this thesis has not previously formed the basis of the award of any degree, diploma, associateship, fellowship or any other similar title of any University or Institution.

(Dr. C.R. Subrahmanya) (Peeyush Prasad)
Raman Research Institute
Bangalore 560 080
India.

CERTIFICATE

This is to certify that the thesis entitled **A Network Centric Receiver Architecture for Low Frequency Arrays**, submitted by Peeyush Prasad for the award of the degree of Doctor of Philosphy of Jawaharlal Nehru University, is his original work. This has not been published or submitted to any other University for any other degree or diploma.

Prof. Ravi Subrahmanyan
(Center Chairperson)
Raman Research Institute
Bangalore 560 080
India.

Dr. C.R. Subrahmanya
(Thesis Supervisor)

Acknowledgements

This thesis would not have been possible without the active help and immense support of a gamut of people. Their presence has been extremely important for me to reach a position of writing an acknowledgement page in my thesis.

Firstly, I am deeply indebted to Prof. C.R. Subrahmanya, my thesis advisor. His zeal for technical excellence, dedication towards radio astronomy, overall enthusiasm and a positive attitude is what attracted me to work with him, when he took me in as a fresh engineer. I have learnt a lot from him, not only technically, but also good lessons in life. He has fearlessly exposed me to a wide variety of technical problems, without letting me lose my self-confidence when things wouldn't work, and come up with life-saving ideas just when I would be giving up hope. More so, he has provided just the right amount of guidance, leaving me mostly to my own devices but bringing me back on track when I would (often) deviate. I owe all my technical knowledge to him. It has been a pleasure to work with him during the course of our association, which I hope to continue, and I wish him all the good health and happiness that he deserves.

I would like to express my deep gratitude to colleagues at the Radio Astronomy Laboratory, for having setup many of the analog receiver systems that have been used in this thesis. They have also been around to answer my emergency calls (almost always on weekends!), and helped me with a willingness which is beyond the call of duty. Thanks to Som for being my first mentor, and always make things better with his positive attitude, a quick game of badminton or evening snacks... (Thanks for teaching me how to filch components too!) Thanks to Prabu, who taught me the value of being organized, keeping a thorough lab-record, and NOT making changes to working systems at the last minute (Prabu, I swear I only made cosmetic changes to that Ooty code, and still don't know why it didn't work!). You have been ever-ready for (elaborate) technical discussions, and provided support during those long, late nights spent struggling with concepts and recalcitrant hardware. Thanks, Girish, for all the help and support during the ORT phase, the late, cold misty nights at the

ORT tower, and for being a willing sounding board for my various tactics to tackle system issues... and not to forget, the world class punning! Thanks to KBR, for the imli and the mimmicry sessions, to Vani for her positivity and VHDL/FPGA tips, to Kasturi and Sujatha who taught me how to solder properly and always helped me in difficult times, and to Raghu for the RF tips and ever smiling visage.

I am grateful to my colleagues at RAC, who have always gone out of their way to make my stay there comfortable, both technically and physically. Thanks to Manoharan sir for the whole hearted support to my work, and the warmth and friendliness. Thanks to DNG for the numerous analog related support I have requested (and sometimes demanded) from him. Thanks to Kalyan for the technical discussions and good humour, to Arvind, Rajmohan and the workshop staff for prompt support, to the observers for repeated observations, including escorting me late in the night to the towers, to Mahesh for all the cabling, to VCS and other staff for laying the OFC, and to the VSP and graduate students at RAC.

I would like to especially thank Dwaraka, my co-guide, who always ensured my thesis was coming along on track. Thanks to Chitra for her hot bread and hospitality. I am also grateful to my SAAC Committee, including Madan, and Uday for ensuring I finished the last mile of the Ph.D, and especially Bala, who took a very pragmatic view of things and was always ready with practical advise.

I express my deepest appreciation to all faculty members of the Astronomy and Astrophysics group, as each one of them, in their own distinct way, make this group ideal for pursuing research. Thanks to Desh and Shiv for technical discussions, Anish for Ooty related queries, and Ramesh for hardware related issues.

Many thanks to my friends at the Astro floor, including batchmates Ruta (for her down to earth nature), Wasim and Yogesh for long technical discussions, and support during misery, Kanhaiya and Nishant for some good fun, Chandryee and Nipanjana for soothing vocals, Kshitij for good friendship, and all the others who make the astro floor lively and jovial. Thanks to Vidya, for her immense patience, and prompt and timely help for any administrative issue. Thanks to Mamta Bai for writing out endless declaration forms for our Ooty trips!

My special thanks to Rad for fulfilling all my adventurous, adolescent dreams of flying and sailing, and to Dominique for her hospitality. Thanks to Lakshmi and Ravi, for instilling confidence when most needed... You are an ideal couple!

RRI is very fortunate to have an excellent library, with a wonderful atmosphere. My sincere thanks to the library staff for maintaining it wonderfully, and providing many happy hours of browsing. Thanks to the very supportive administration of RRI, especially Krishna, Marisa, Radha, and Ram for being very proactive in their help, and letting us focus fully on research. My sincere gratitude to the housekeeping staff of RRI, who maintain that special "RRI" feeling thoughout, including when RRI is a riot of colored flowers. You have made an important difference to my life!

Last, but certainly not the least, I wish to thank my family. In fact, I cannot begin to express my gratitude towards my family, and especially my parents, for all the moral and practical support they have provided selflessly, and for many, many years. They allowed me to pursue my dreams, without putting pressure to find a "real" job, or to take "more interest in household affairs". No uncomfortable questions were ever asked about the extremely odd hours I would put in, or whether there was any end to my research at all... Mummy and Daddy, it is only because of your efforts that I didn't just give up and run away! and your consideration in bringing me hot food late in the nights to RRI will stay with me for the rest of my life... Mummy, for her loving and affectionate nature, and Daddy, for being a constant source of inspiration and encouragement. I am especially indebted to my sister Richa, who literally put her life on hold, and took up all my responsibilities towards my parents, including providing financial and moral support to all of us, just so that I could pursue what I wanted. Thanks to my brother-in-law, Gaurav, for his helpful advise and support, ever since he became an integral part of our family. Thanks to my sister Shweta and Anand jiju, for being extremely supportive, and offering valuable help and suggestions at all times. I can proudly say, I love you all, this thesis is as much a product of your efforts, as mine... You all are the best!

A great outcome of my stay at RRI, and which has been a positive turning point in my life, has been my meeting Divya. In fact, I would rate meeting her higher than actually getting my Ph.D! She has been all that I could ever ask for in a partner, and much, much more. Thank you Divya, for firstly believing in me when I was completely unproven, and for skillfully guiding this thesis right from the beginning (remember the coursework assignments, and my first InHouse meeting talk? :-)) My thesis has been really enriched by your critical support, clarity of ideas, and eye for detail. Thanks for (trying) teaching me the about the importance of deadlines, of working with a focus, presenting technical work, and that being simple doesn't mean being technically unsound... in fact, thanks for teaching me about life...

Preface

The study of several astrophysical phenomena related to radio sources and their environment, as well as, a variety of cosmological investigations are enriched by observations at the lowest ranges of the accessible radio window. These include the study of ultra-steep spectrum objects like high red-shift radio galaxies and pulsars, neutral Hydrogen content in the early universe, large scale structures like radio halos, as also sky monitoring for transient events at low frequencies. For many such investigations, existing radio telescopes are limited by factors like their sensitivity, dynamic range and survey speed (constrainted by the instantaneous field of view). These limitations are further aggravated by the increasing occurrence of interference, as well as, atmospheric/ionospheric distortions to observations. However, rapid developments in high speed computing and communication have now provided a means for improving the situation. This, in turn, has led to many recent developments involving new approaches towards instrumenting arrays, as well as the development of new algorithms for calibration and imaging, thereby minimizing systematic distortions in the observational estimates. Further, instrumentation trends for low frequency arrays are moving towards mechanically simple, wide-band systems with large fields of view, and programmable/configurable digital systems replacing the inherently rigidly structured analog processing. This mainly involves direct digitization of the incoming radio signal, and carrying out all further processing digitally.

Along similar trends, an upgrade of the Ooty Radio Telescope (ORT) to a full correlation receiver of 264 elements is being planned, resulting in an order of magnitude increase in the instantaneous field of view. This, combined with an enhanced ($\sim$ 40 MHz) Radio Frequency (RF) bandwidth, a configurable temporal and spectral resolution, and the programmability of combining RF outputs will enable several new classes of observations, and is expected to provide a new lease of life for this mature telescope. However, the above features will require the receiver to handle about $\sim$ 13 GBps of data, with a compute load of almost $\sim$ 5 TFLOPs. Much of this load occurs because of the correlator, which needs to manage >100 times the incoming data rate due to the requirement of all-to-all connectivity. Here, the problem inherently has a low arithmetic intensity, which degrades the performance of a cluster computer.

Thus, the distributed nature of data generators in a low frequency, wide field telescope, coupled with the high I/O requirement, in order to guarantee high sensitivities, requires a careful examination of the communication hierarchy of the receiver design. In addition, spreading of computation hierarchically can result in efficient utilization of the incoming data streams, where effective decisions on data handling can then be taken locally. Both these criteria, as well as the need to keep the instrument configurable, require careful attention to the receiver architecture design. This forms the main focus of a large part of the thesis.

In particular, the following four classes of problems have been addressed:

1. Approaches towards high bandwidth data distribution, distributed real-time processing and analysis of the communication network topology in an interferometric array.
2. New features added to network functionality for handling the large datasets generated by current multi-element arrays, as well as the monitoring and feedback inherently required in a distributed system.
3. Visualization, editing and calibration of the resulting large scale data sets.
4. An interdisciplinary approach for dynamic calibration of observations, using geosynchronous navigation satellite signals.

This thesis comprises of six chapters; their broad contents are stated below. References made in each chapter are listed out in the Bibliography, available at the end of the thesis.

Chapter 1 reviews the observational challenges in low frequency radio astronomy, and also examines the current state of the art of digital receiver design. Further, an analysis of the computational and communication requirements of implementing efficient low frequency array receivers has been carried out, given available technology and trends. An important outcome of this analysis is the highlighting of the role played by the communication and reorganization of large datasets required for the correlation. This motivates our receiver architecture.

Chapter 2 presents a *Networked Signal Processing System* architecture optimized for low frequency radio astronomical applications. In particular, this architecture has a scalable and distributed hardware/software co-design, and is well suited to match requirements of arrays consisting of a large number of spatially distributed antenna elements, with significant requirements of real-time processing and data routing. This architecture explicitly employs techniques of data fusion at multiple levels of a processing hierarchy, while exploiting features available from commodity components for data transport and distributed computing. It also carries out effective load balancing via efficient data routing. In addition, it makes provisions for the implementation of algorithms which may require multiple passes on the incoming, high bandwidth data *in real-time.* Such an approach can help in enhancing the observation's efficiency, e.g., by segregating the real-time data stream along different dimensions, or by carrying out sophisticated pre-processing (which is *descriptive*, rather than *predictive* in nature) on large bandwidth data before the irreversible *data fusion* operation.

Chapter 3 presents an implementation of the NSPS architecture in the form of a programmable, hybrid digital receiver for the upgraded ORT. While the upgrade aims at configuring the ORT as a prototype 40-element array and a wide-field instrument for cosmological investigations, the receiver implementation demonstrates several advantages towards receiver programmability, efficiency, and ease of development offered by the NSPS approach. Further, details of the receiver implementation, its validation and an approach towards its calibration are presented.

Chapter 4 presents a scalable software correlator implemented on commodity processors, and developed as a real-time backend for the ORT prototype array receiver. Such a correlator balances the ease and rapidity of development in a software environment using commodity communication and computing components, while still being a feasible implementation for medium sized arrays. Further, the NSPS allows streaming data to be rearranged to suit the processing element's architecture, making this implementation very efficient.

Chapter 5 discusses novel tools and algorithms developed by us towards visualization, editing and calibration of large-scale low frequency observations. The need for such tools is a natural consequence of configuring a radio telescope as a combination of a large number of elements with large bandwidths, which results in large quantities of data being generated. In addition, some new approaches towards improving the sensitivity of low frequency observations, mainly by carrying out a segregation of observed visibiliities are also presented. Further, the possibility of improving the corrections to ionospheric distortions of low frequency observations by conducting satellite interferometry on geosynchronous navigation satellites has been explored. Furthermore, a scheme for carrying out simultaneous satellite and celestial observations using the GMRT is discussed.

Finally, Chapter 6 discusses the main findings and conclusions of the thesis. An appendix presents a description of the main data structures and hardware entities of the NSPS implementation for the ORT receiver.

Publications

The work presented in the thesis has been presented via the following publications:

1. *A High Speed Networked Signal Processing Platform for Multi element Radio Telescopes*
   Peeyush Prasad, C.R. Subrahmanya
   Vol. 31, 1, 1-22, Experimental Astronomy.

2. *Reconfiguration of the Ooty Radio Telescope as a 40-element programmable telescope*
   C. R. Subrahmanya, **Peeyush Prasad**, B.S. Girish, P.K. Manoharan, D. Nandagopal et. al.
   Manuscript in preparation.

3. *Software Correlator for the 44-element Ooty Radio Telescope*
   Peeyush Prasad, C.R. Subrahmanya
   IEEE International Conference on High Performance Computing (HiPC), 2010
   arXiv:1102.0148.

4. *Dynamic Range Improvement of GMRT Low Frequency Images*
   Peeyush Prasad, C.R. Subrahmanya
   The Low Frequency Radio Universe, Vol. 407, 398-401, ASP Conference series.

5. *Interferometric Observations of Geosynchronous Satellites*
   C.R. Subrahmanya, **Peeyush Prasad**, R. Somashekar
   Proceedings of The 2nd International Conference on Space Technology (ICST), To be published in IEEE Xplore.

Contents

Chapter 1

Introduction

1.1 Low frequency radio astronomy

Although radio astronomy began with observations at low frequencies ($\lesssim$ 408 MHz), the need for high angular resolutions and dynamic ranges for carrying out studies of discrete radio sources and the progress in receiver technology resulted in a shift towards observations at higher frequencies ($\gtrsim$ GHz). Thus, the high frequency domain was explored to a much greater extent, as compared to low frequencies.

However, there has been recent revival of interest in the low frequency radio regime for addressing several problems of astrophysical significance [1], ranging from study of coherent emission processes, e.g., jovian bursts [2], solar [3] and stellar radio bursts [4], pulsar emission [5] to investigation of galaxy mergers, activity in giant radio galaxies, buoyancy in cluster halos, and cluster-cluster interactions [6], and relic radio sources associated with clusters of galaxies [7].

More importantly, a strong scientific motivation for low frequency observations is seen recently for problems which inherently require wide fields of view (or possibly all-sky imaging) and high sensitivities. Such observations are valuable for detecting diffuse features [8], carrying out blind and targeted source variability studies over a range of timescales (from tens of nanoseconds to years) [9],[10], etc., or for studying the large scale structure of the universe. In addition, detection of radio transients [11], which are expected to have the majority of their populations observable only at low frequencies, also require wide fields of view to carry out rapid and continuous sky monitoring.

The above motivating factors have led to a resurgence of interest in low frequency radio astronomy, with an emphasis on large instantaneous fields of view and high temporal and spectral resolutions. This is evident from the fact that recent and upcoming large telescopes are being constructed with a large number of small, mechanically simple antenna elements. Here, each element provides the required large field of view, with the array response being steered in software (and not by mechanical steering of the individual elements). While such a concept would have implied impractical receivers in the past, rapid developments in computing and communication have made them feasible in recent times. However, a new challenge to algorithm development is posed by the need to reconcile with the ever-increasing presence of man-made Radio Frequency Interference (RFI), which is more prevalent at lower, than at higher frequencies.

Further, since the modern trend is of arrays utilizing a very large number of antenna elements, the associated digital signal processing systems (since analog processing is not practical) have to perform large volumes of computing and communication at all levels. The resulting load (which may exceed currently encountered loads by orders of magnitude) is managed by systems ranging from dedicated hardware carrying out repetitive computing, to quasi-realtime computing, (possibly on commodity nodes). In addition, such receiver architectures need configurability and flexibility in processing in order to implement various observing modes, and to allow tapping of data from different processing levels. Furthermore, observing environments at low frequencies can require constant monitoring, e.g., for RFI.

Thus, sensitive low frequency observations from configurable instruments can open up entirely new phase spaces of discovery for radio astronomy. To this end, many of the current efforts are directed towards implementing the Square Kilometer Array (SKA) [12, 13], a telescope of unprecedented sensitivity and configurability.

With this background, this thesis presents a detailed examination of the computing and communication hierarchies for a specific class of multi-element radio telescope, and presents a partitioning of computing and memory across the signal processing hierarchy. In particular, our partitioning of the computing ranges from being dense and suitable for latency critical operations, to being highly configurable and latency tolerant (while being temporally spread, e.g., in commodity processors). Similarly, memory available to the signal processing units ranges from being limited, but available to high bandwidth processing in the early stages (e.g., very early in the data flow hierarchy, in configurable hardware), to being virtually unlimited and accessible in a variety of ways, e.g., via commodity processors. With large number of elements, the major complexity of real time processing in a radio telescope arises from the correlator, which is required to perform cross-correlation of signals arriving at every receiving element in different spectral bands and to cater to dynamically varying differential delays and phases. Thus, the correlator is the fundamental signal processing system examined in this thesis.

We will now discuss some observational challenges at the lowest frequencies of observation.

1.2 Low frequency observational challenges

Low frequency observations have faced considerable challenges to achieve high resolutions, sensitivities and dynamic ranges. Previous attempts at improving the resolution and sensitivity of low frequency observations by using larger arrays, in fact, posed major challenges to their calibration [14]. This, primarily, is due to ionospheric phase fluctuations, which lead to short coherent integration times, resulting in low sensitivities and dynamic ranges. These phase fluctuations, in turn, are caused by the rapidly fluctuating time and position dependent nature of ionospheric structures. In particular, the signal received by one element of an interferometer, experiences an excess path length as compared to another. This excess path length is dependent on the line of sight column density of electrons, and can be modelled [15] as $L_i = 40.3\nu^{-2}N_e$, where, L_i is the ionospheric excess path length (in centimeter) due to the ionosphere, N_e is the electron density (in 10^{16} electrons/m^2), and ν is the observing frequency (in GHz). The introduction of this excess path results in a temporal phase winding due to the added phase which, in turn, leads to random phase distortions on observed visibilities. These visibility distortions at low frequencies of observation are magnified due to the ν^{-2} dependency of the excess path length, as well as due to the larger fields of view. It may be noted that these challenges were difficult to meet using previously available computing and communication infrastructure, but now seem tractable.

Man-made RFI also poses a major challenge to high-fidelity estimations of the radio sky at low frequencies. RFI can be dealt with by either relocating arrays to remote sites (becoming more and more difficult to find), or by implementing adaptive and real time algorithms for tolerating RFI [16, 17], which also require large computing resources.

Furthermore, the operating parameters and mechanisms of even the current instruments lead to the irreversible contamination of visibilities (and subsequent propagation of errors), by the initial visibility estimation process operating on the incoming data. This results in reduced sensitivity. An example of this contamination by errors, is the averaging over RFI affected data, caused due to the time resolution (tens of seconds range) feasible for standard radio observatories. Though such a temporal resolution is sufficient to sample the observed visibility after fringe stopping, it is usually inadequate to cater to a characterization of ambient RFI. Thus, distortions on such visibilities can only be partially corrected by post processing data analysis. This reduces the *reliability* of observations, which in turn, lowers the dynamic range.

In the next section, the signal processing and communication requirements of a correlator are first briefly reviewed. A description of current trends in signal processing in an array radio telescope is given, particularly focusing on the effects of the rapidly advancing computing technology on implementations. This discussion gives a context for a distributed, communication and signal processing platform developed by us for handling the signal processing of a multi-element, low frequency array telescope.

1.3 Correlator-receiver signal processing overview

The main digital signal processing blocks in a traditional spectral correlation (FX, [18]) receiver are: (a) conditioning of the incoming per-antenna time series via companding, filtering, windowing, etc., (b) spectral analysis of the filtered stream via an FFT or filterbank (the F part in the FX), and (c) cross-multiply of each spectral channel of an antenna element with the corresponding spectral channel of every other antenna element (the X part in FX). The FX part of the correlator is represented by the following set of equations:

$$X_p\left(k\right) = \sum_{n=0}^{N} x_p\left(n\right) \exp\left(\frac{-2\pi i k n}{N}\right) \tag{1.1}$$

$$S_{pq}\left(k\right) = < X_p\left(k\right).X_q^*\left(k\right) > \tag{1.2}$$

where, X_p is the Discrete Fourier Transform of the data stream (x_p) from the antenna element p, n is the time index (sample number), k is the spectral channel index, and $S_{pq}\left(k\right)$ is the Cross Power spectrum corresponding to the spectral channel k of the antenna elements p and q.

Here, we see that parts (a) and (b) of the correlator operate on a per antenna basis, and are completely independent of the data from other antenna elements. Part (c), on the other hand, requires data from every antenna element to be brought to the cross multiplying circuit before output corresponding to a certain time-step can be generated. This single operation contributes to the nature of the problem turning out to be I/O bound, as well as to the constraints in I/O decomposition for a Parallel implementation.

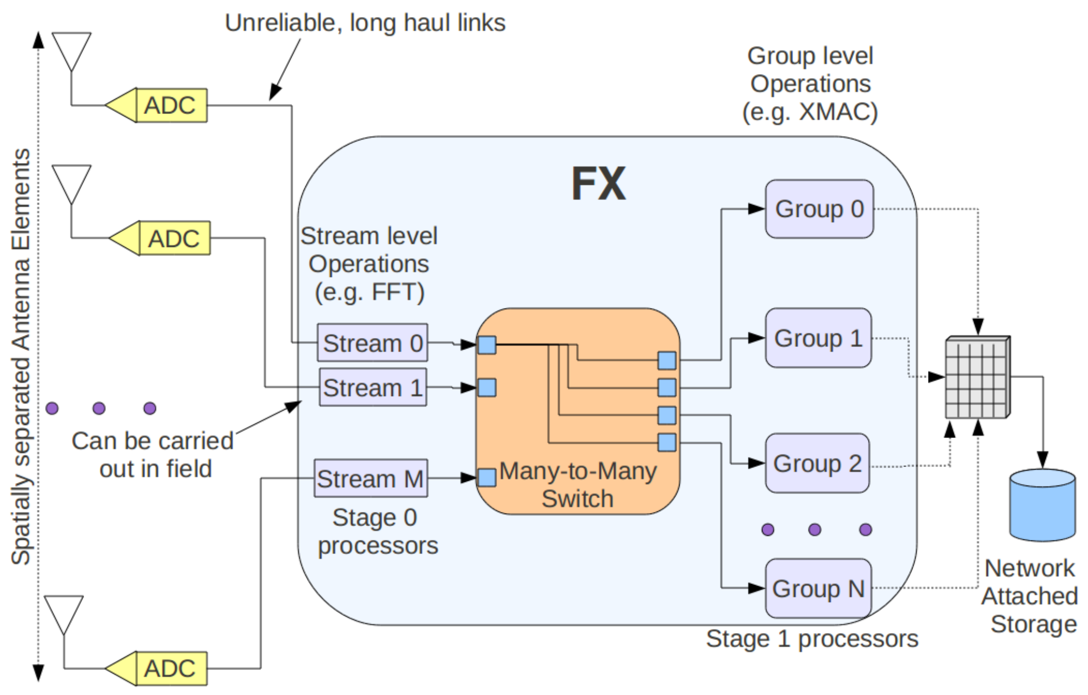

Unreliable, long haul links
Spatially separated Antenna Elements
ADC
ADC
ADC
Can be carried out in field
Group level Operations (e.g. XMAC)
FX
Stream level Operations (e.g. FFT)
Stream 0
Stream 1
Stream M
Stage 0 processors
Many-to-Many Switch
Group 0
Group 1
Group 2
Group N
Stage 1 processors
Network Attached Storage

Figure 1.1: *Signal flow and main components in a typical FX Correlator. Here, the communication mainly consists of the bringing in of data from remote sources, and the formation of data sets amenable to group level operations, while the computing can be categorized as being on a per-datastream, or a per-data group level. Thus, the three main parts of a conventional correlator are the stream level operating entities, the many-to-many data switch, and the group level operating entities.*

This fundamental nature of a correlator engine, namely, the requirement of re-distribution of data at a large scale, is depicted schematically in Figure 1.1. Here, computing on the data is split into two categories corresponding to the two parts mentioned above: a per-datastream part, and the other which requires a set consisting of rigidly related data from all datastreams to be present before any computing can be carried out. These two categories of computing are linked to each other by the many-to-many data disperser (represented in Figure 1.1 by the many-to-many switch), thus forming the three major components of the correlator, namely, *per-stream processors* (referred to as Stage-0 processors), the *data disperser*s, and the *block-level processors* (referred to as Stage-1 processors). The correlations thus formed are then used to synthesize various observation modes via post-processing. To meet the requirements of current observations, a feedback of information from various monitoring processes is required, before the data is irreversibly reduced.

In the next section, we analyze the receiver architectures of contemporary instruments which utilize the current state of computing and communication.

1.4 Current trends in low frequency radio telescopes

In recent times, instrumentation for radio astronomy has been heavily influenced by the rapid technological progress in computing and communication resources. In particular, a variety of processing element architectures, which optimize certain aspects of signal processing or communication are now available in the commodity market, and are being applied to astronomical signal processing. This is in contrast to the earlier trend of developing customized hardware to carry out the more demanding aspects of the processing, with high bandwidth communication being restricted to the analog domain. These factors affect all aspects of array telescope design, ranging from the number and placement of sensors, the correlator architecture, to the possible observing modes. Some aspects of the progress especially relevant to radio astronomy are due to the reducing cost of high bandwidth, long haul I/O resources, as well as increasing I/O capabilities of commodity resources. Another aspect relates to the reducing cost and increasing memory and I/O capabilities of Field Programmable Gate Array (FPGA) platforms, which are reconfigurable hardware used to implement dense computing at high speed and with very low power consumption. Further, rapidly reducing memory volume to cost ratios makes it feasible to equip receiver hardware with large amounts of memory at various levels, resulting in the availability of large I/O buffers for real-time data. This opens up yet another dimension to the receiver implementation configuration.

For instruments commissioned in the past few decades, such advances have given rise to a trend of carrying out the core computing of their correlation receivers (earlier carried out on dense, customized hardware), on commodity hardware with general purpose, but optimized software, thereby increasing their configurability and operating parameters. Prime examples of this trend is the recently commissioned Giant Meterwave Radio Telescope (GMRT) software correlator, which is described below.

The GMRT Software Correlator: The GMRT [19] has been recently equipped with a 32 antenna, 33 MHz, dual polarization, fully real-time software backend [20], using only off-the-shelf components. Here, the received RF analog band from each

antenna is transmitted over optical fibres to a central receiver, where the signal is conditioned and sampled. A correlator and a beam-former (in coherent and incoherent mode) have been implemented in software running on commodity processors. These use Peripheral Component Interface (PCI) bus-based Analog to Digital Converter (ADC) cards and a Linux cluster of 48 nodes with dual gigabit inter-node connectivity for the real-time data transfer requirements. The software backend operates on Intel Xeon clusters, organized into three layers: acquisition, computational and storage layer. The complete software approach with minimum of hardware, allows for many concepts related to distributed and parallel processing to be used, in order to share the computing and communication load among the available resources. Thus, a Message Passing Interface (MPI) based approach has been taken for inter-node communication and synchronization, while OpenMP [21] has been used to utilize the multiple cores on each processor, with library based vector processing. Data dispersion for correlation is carried out using Commercial Off The Shelf (COTS) switches. Here, the receiver architecture is centralized, in that all sensor outputs are brought to the center before being digitized. Finally, the correlator also has a baseband recording mode, where data can be transferred to an offsite HPC cluster over a high bandwidth link, providing close to real-time transfer rates. Such an arrangement makes it possible to closely examine the sampled raw data, for implementing novel algorithms.
However, the above view of enhancing configurability of existing instruments via purely software implementations, turns out to be inadequate for future telescopes, which usually consist of a large number of elements, with a high processing bandwidth. These requirements could earlier be met only by Application Specific Integrated Circuit (ASIC) implementations, with their inherent developmental cost and inflexibility. For such instruments, mainly being proposed in the 21^{st} century, configurable hardware (usually in the form of firmware on FPGAs), as well as software, is extensively used. Such hybrid architectures are essential for the application of computing not only for carrying out the required processing, but also to handle calibration and reliability enhancement of the data, in real time. Newer array receivers based on this approach are increasingly adaptive to a changing observing environment, in real time.

It may be noted that, such a shift in the approach makes it fundamentally different from the earlier view of the receiver architecture as a streaming, linear flow of data, with a fixed series of computation.

The result of these advances can be seen, e.g., in the increasing number of elements in modern arrays and the ability to carry out a variety of observations simultaneously.

We now provide a brief overview of some of the array and receiver designs being currently implemented in order to provide a context for the network centric approach of this thesis.

1.4.1 LOFAR architecture

The Low Frequency Array [22, 23] is a real-time, multi-sensor aperture synthesis array operating between 20 MHz to 240 MHz. It is currently composed of 36 phased-array "stations" spread over an area $\sim$ 100 Km in diameter in The Netherlands. Each station consists of 48 "High band" (120-240 MHz) tiles (each tile being a 4x4 grid of dual-polarization, bowtie elements), and 96 "Low band" (20-80 MHz), dual polarization, drooped dipole elements. The telescope has four main components: (a) sensor field stations (b) high bandwidth Wide Area Network, implemented using 10Gbps Gigabit Ethernet (10GbE) (c) Central Processing Facility (CEP), and (d) software blocks for control and monitoring. Here, we focus only on the CEP.

At each station, an analog beam-former is used at each tile to generate a phased array beam, following which the entire incoming RF band is baseband sampled, and sent to a station digital receiver. Here, a digital beam-former implemented on reconfigurable hardware (FPGAs) creates spectral, dual-polarization phased array beams with configurable bandwidth and number, ranging from a single, 32 MHz beam to eight beams with a 4 MHz spread. The combination is dictated by the total output link bandwidth available from each station. Digitally formed beams from each station are connected to the CEP, which carries out the bulk of the processing in software using a large set of commodity class processors [24].

The LOFAR utilizes several of the aspects discussed above. Since rapid reconfiguration and multiple, concurrent observing capability is a key feature, digitization is carried out early in the signal flow to maximize configurability, with the ability to tap preprocessed data. Large memory at the field station is available in the form of the "Transient Buffer Board", which can store and forward a large ($\sim$ 1 sec) burst of the incoming raw data, to be examined at the central station. The CEP uses several clusters of commodity class processors, with a high bandwidth switch fabric interconnect system, each being optimized for specific tasks.

Software correlator: For the main load of correlation estimation, the particular choice of commodity processor includes a three-dimensional, high bandwidth inter-processor interconnect. A Single Algorithm Multiple Data (SAMD) approach is adopted, with the incoming data streams being split and sent to multiple processing units, each carrying out identical operations on their data streams. Here, beam direction and frequency are used to carry out data splitting, which requires a dispersion of the incoming data (multiple beams from a single station) to the processing elements.

This dispersion is carried out in two stages, the first of which creates group of data containing a smaller time-frequency window from all stations. This stage uses the Infiniband switch fabric with an all-to-all connection scheme. The second stage of fine grained switching is carried out using the processor interconnect fabric. Large cluster memory buffers enable pipelined processing which, in turn, allows calibration solutions to be determined and applied online, as well as to relax hard real-time constraints.

The instrument's agility is demonstrated by the number of simultaneous observing modes currently possible, and also by the ability of the system to accommodate additional special processing blocks within the existing design, for carrying out customized observations. An example of this is the *Transient Detection pipeline*, which operates on bursts of raw data from each sensor, at high time resolution.

1.4.2 MWA architecture

The Murchison Widefield Array (MWA) [25] is an aperture array of 128 tiles situated in a radio quiet region in Western Australia. One of its primary goals is the detection and characterization of the 21cm neutral hydrogen signals from the Epoch of Reionization (EoR) , along with the discovery and monitoring of transient and variable sources, among others. Here, a wide field of view, large fractional bandwidth, high spectral dynamic range and a well defined instantaneous point spread function are necessary. Thus, the telescope operates on any 32 MHz spectral band between 80-300 MHz, with the dense inner core within a 350 m extent giving close to complete UV coverage. Each tile is a 4x4 array of dual polarization elements (referred to as "Bowtie" elements [25]), with an analog beam-former generating a phased array beam of width $15 - 50^o$ (depending on frequency) in a desired direction in the sky, within the $\sim 120^o$ sensor field of view. Thus, the array consists of ~ 2048 sensor elements in all, with the currently installed 32-tile demonstrator system.

The RF signal from each tile is baseband sampled at the tile, where a coarse spectral binning of the entire 80-300 MHz band is carried out. Then, depending on the chosen frequency window, a digital sub-band of 32 MHz is sent over optical fibres to a central digital receiver for correlation. Here, a high resolution spectral binning and data rearrangement for cross multiplication is carried out, before the actual visibilities are formed.

In this system, reconfigurability is achieved by using FPGA hardware at all levels of processing. However, a higher level of flexibility and lowered development time was achieved by implementing the correlator for the demonstrator system on a Graphical Processing Unit (GPU) based system, hosted on a commodity processor system. Here, the commodity processor system carries out aggregation of the incoming data and prepares it for the cross multiply, which is carried out by the GPU subsystem.

For the full system, an FPGA based digital beam-former is planned to be implemented along with the hardware correlator. For maximum sensitivity, this beam-former would require continuous updation of calibration parameters, which are generated by a real-time calibration loop operating on the incoming data. Thus, this array also utilizes digital signal processing via innovative algorithms, in order to efficiently carry out low frequency observations.

1.4.3 APERTIF architecture

The Aperture Tile in Focus (APERTIF) [26] program seeks to enhance the field of view (and thus, the survey speed) of the WSRT by the installation of Phased Array Feeds (PAF) with instantaneous bandwidth of 300 MHz over a 1000-1750 MHz range. PAFs are arrays of electrically small antenna elements ($< \lambda/2$) in the focal plane of the reflector. These allow multiple beams to be formed on the sky, which is the only way of forming closely packed beams on telescopes with small f/D ratios. In the APERTIF, signals from the 121 elements making up the FPA are digitized, spectrally analyzed and processed by a digital beam-former at the antenna base itself, which produces 37 dual polarized beams for every sub-band. This is done by multiplying every sub-band from each sensor with a different complex weight, and adding them together to form the beam. Data from corresponding beams from all antennas is then sent to a central correlator for carrying out aperture synthesis over the whole array. The computing for just the sub-band filter and beam-former is estimated at 100 TMAC/s for 12 dishes, while each beam-former generates 178 Gbps per dish [27], which are sent over 10GbE links to the central processor. In addition, a real-time calibration mechanism exists which turns on a reference noise source on the reflector surface.

In this system, the large computing is carried out using configurable hardware, mainly because a high level of configurability is not needed from the FPA beam-former. However, the processing is carried out digitally, and the large data volumes require an optimized communication hierarchy.

1.4.4 CASPER approach

The CASPER (Collaboration for Astronomy Signal Processing and Electronics Research) project [28] attempts reconfigurability of instrument design via the creation of a library of firmware components with standard interfaces, on configurable hardware like FPGAs. These components can be attached to each other to implement commonly used operations in radio astronomy like beamforming or correlation. This approach is much more efficient than similar attempts in software, and reduces hardware development time. Also, the high compute density and low power consumption of configurable hardware make it the only approach for larger arrays. However, the requirement of almost real-time reconfiguration of the instrumentation for different observing modes is difficult to meet, prompting a hybrid architecture, with the hardware carrying out some fraction of the computing load.

In all the above implementations, analog signals are digitized early in the signal flow, which results in increased configurability as well as the stability of DSP-based subsystems, as compared to analog subsystems. Field based remote preprocessing (due to the ability to distribute dense computing via FPGAs) is required to handle the resulting large computing. Further, high bandwidth data transfer over digital links are essential, as they allow larger processed bandwidths, in addition to increased spatial extents. Finally, a variety of commodity processor based correlator implementations can be seen, with the required data dispersion being carried out using commodity switches. This allows for flexible receiver configurations.

It may be noted that in between pure software receiver implementations (e.g., GMRT) and hybrid approaches needed for arrays with a large number of elements (e.g., LOFAR), there exists a niche for receiver architectures for medium sized arrays with modest ($\sim$few tens of MHz) bandwidths. These do not have a computing requirement heavy enough for extensive hardware offloading, yet it is diverse enough that it cannot be carried out on commodity components without hardware support, e.g., a receiver for the planned upgradation of the Ooty Radio Telescope. Here the existing phased array instrument is being planned to be reconfigured as an array of 264 elements within its 506 m extent (discussed in Chapter 3, Section 3.6). Such a niche area has been thoroughly examined in this thesis, resulting in the design of a Networked Signal Processing System (NSPS) architecture for receiver design, discussed in brief in the next section.

1.5 Network centric design of receiver architectures

The Network centric approach proposed in this thesis leads to a signal processing and data routing architecture tailored to the requirements of adaptive, low frequency instrumentation. Here, the role of the network is elevated from merely a data carrier between processors, to an entity carrying out job scheduling and load balancing in a distributed computing scenario, by way of data routing. Further, this architecture corresponds to the application of computing not only for carrying out the required processing, but also to handle calibration and reliability enhancement of the data, in real time. It utilizes dense and distributed computing, effective load and communication balancing, and high transport bandwidths over large spatial extents, which are feasible due to recent advances in computing and communication, as well as their rapidly falling costs.

Furthermore, an important aspect of our architecture is that it is explicitly equipped with large and fast distributed memories across the signal processing hierarchy for effective load and communication balancing between computing elements. Since achieving high sensitivity can involve unorthodox methods of analysis (which may benefit from accessing pre-processed or even raw data), and may require several passes on the data in order to segregate deviant data, our architecture creates an infrastructure which allows examination and modification of the real-time incoming data stream at high temporal and spectral resolutions. Here, large memories allow for characterization of data using multiple processing passes.

Our approach is, thus, a hardware/software co-design, with ability of offloading a significant fraction of data routing, shuffling and preprocessing onto a customized hardware layer, while the core of the computing can be carried out using commodity processing elements. This allows us to retain rapid processing reconfigurability via commodity processing, while the custom hardware segment contains features which allow special algorithms to be implemented, in order to increase observational sensitivity.

This approach can be highly efficient for medium-sized arrays, where net data rates and computing are matched to commodity hardware and software resources. It may be noted that upcoming array telescopes of the SKA category usually consist of at least two array configuration regimes: a tightly clustered core (medium sized), and outlier stations, spread over a large spatial extent. Our approach is thus directly applicable to the former's signal processing requirements. Further, though medium sized arrays with a large cluster of spatially confined sensor elements did exist in the past (e.g., cylindrical apertures with their dense line feeds), the signal available to the user was always pre-combined, leading to a single, large effective aperture. Thus, such existing instruments can also be interfaced with receivers based on our proposed hybrid architecture, which can lead to much larger effective fields of view.

As a demonstration of our ideas, the Ooty Radio Telescope has been reconfigured with a high sensitivity receiver whose implementation is based on our architecture. We describe the salient details of this telescope next, while pointing out its suitability for carrying out high sensitivity observations.

1.6 Ooty Radio Telescope

The Ooty Radio Telescope (ORT) [29] is a multibeam phased array system, with its reflector consisting of a 506m X 30m equatorially mounted North-South cylinder, operating at a nominal central frequency of 326.5 MHz. Its single polarization line feed consists of an equispaced linear array of 1056 dipoles along the north-south focal line. The entire telescope is arranged to have the cylinder long axis parallel to the rotation axis of the Earth, making it an equatorially mounted telescope, and able to track the same region of the sky continuously for upto 9 hours. Here, each dipole is equipped with a tuned Low Noise Amplifier (LNA) (with about 50 MHz bandwidth), along with an RF phase shifter to facilitate pointing a phased array beam in a declination range of $\delta = \pm 45^o$ [30].

The dipole array is organized into “modules” consisting of 48 dipoles each. There are 22 such modules, with 11 forming the north half (identified as N01 to N11), and 11 modules forming the south half for the telescope (identified as S01 to S11). The telescope is depicted in Figure 1.2, where its line feed can be seen along with the Aluminium channel which houses an analog network for combining the outputs of its dipoles in phase, to form the module output. This is a purely passive network, and consists of a Christmas tree arrangement of combiners, after an appropriate RF phase has been applied to the dipoles. The phased output of each module is down-converted to an Intermediate Frequency (IF) of 30 MHz in the field. This is done using a Local Oscillator (LO) signal at 296.5 MHz which is generated in a Central Receiver Building, and distributed to each of the 22 modules using an equal length cable network. The IF band is then transported over another set of equal length cables to the Central Receiver Building for further processing. In the original beam-forming receiver [31], after further conditioning of the incoming IF signal, inter-module delays are compensated, and the module outputs combined as a phased array, using 12

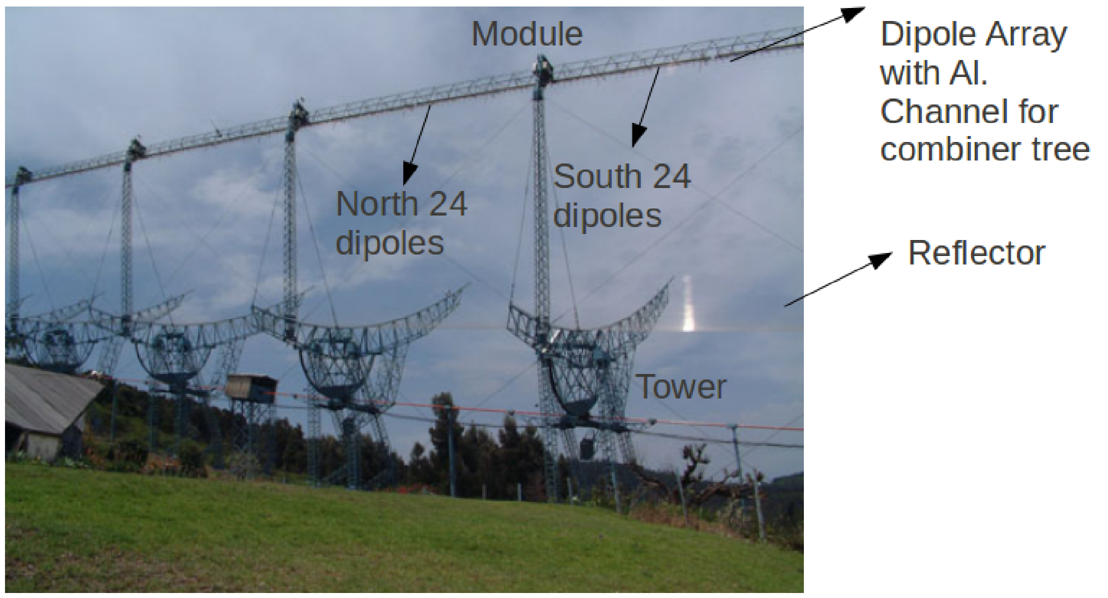


Figure 1.2: *View of the ORT cylinder, with its line feed and organization of dipoles into modules shown.*

beam-formers. To ensure high sensitivity, a per-module phase can be added to the LO signal. Thus, the analog receiver system can generate 12 correlated, phased array beams by correlating corresponding phased array beams from the north and the south half of the telescope. The telescope is steered mechanically in Right Ascension, and electronically via phasing in declination.

Although confusion-limited for observing discrete sources, the high sensitivity ($A_{eff}\sim 8000\,m^2$) and the equatorial mount of the ORT, makes it possible for a modern, programmable receiver to enable it with unique advantages for detection of large scale structures and monitoring the sky for variability with high sensitivity. Keeping this in mind, a programme is under way to reconfigure the ORT as a 264-element radio telescope, for which the first phase targets a reconfiguration of the ORT into a 44-element telescope. In this thesis, we have applied the concepts of NSPS to design a real time signal processing system for this first phase, as described in Chapter 3 and 4.

Further, the need for recognizing RFI in a large synthesis radio telescope and for monitoring rapid phase fluctuations requires the output of the correlators to be examined and edited using a much higher time resolution than necessary for imaging. To benefit from such a high resolution sampling, we have developed some tools for visualization, threshold based flagging, and minimizing systematic biases in the estimates provided to standard deconvolution algorithms. These are discussed in Chapter 5, which also introduces a novel approach for calibrating the non-isoplanatic ionospheric effects in arrays like the GMRT, using geosynchronous navigation satellites which will be available in the near future.

Chapter 2

A High Speed Networked Signal Processing System for Multi-element Radio Telescopes

2.1 Introduction

A Multi-element Radio Telescope is a spatially spread array of antennas (or antenna elements) whose noise-like responses are required to be time aligned, dynamically calibrated and combined or correlated in real time. The resulting estimates of spatio-temporal and spectral correlations between responses of pairs of elements can be used to recover the desired information on the strength and distribution of radio emission within the common field of view [15] using standard post-processing software. Thus, the signal of interest is statistical in nature, resulting from a minute level of mutual coherence arising from weak celestial signals buried in noise. Because of the large number of elements and the high sampling rates necessary for bandwidths exceeding several tens of MHz in recent arrays, real-time statistical estimation is essential to achieve practical data rates and volumes for recording and post processing. For instance, an upgrade of the Ooty Radio Telescope has been recently initiated, and aims at treating the 30m x 506m parabolic cylindrical antenna of the ORT as 264 independent sets of elements, each of which is planned to be sampled at 80 MS/s, leading to a data generation rate of 21 Gigasamples per second, exceeding 80 Terabytes per hour. Till recently, computing requirements of this scale forced a choice of custom hardware to be the most favored platform. However, rapid developments in the fields of digital technology, communication and computing have led to a changing trend towards alternative approaches for upcoming telescopes. Such approaches range between a customized and reusable hardware library of components on an FPGA platform, e.g., the CASPER project [28], and a software-only approach, e.g., at the GMRT [20]. The GMRT is an example of a recent transition from custom hardware to a software-only approach.

In this chapter, we have taken a middle path, where the real-time processing of a multi-element radio telescope is abstracted as a *multi-sensor, data fusion* problem, which has been decomposed broadly into a set of computing and network functionalities. A practical and scalable architecture for implementing such a partitioning is enabled by current technology. Here, we present a new platform/architecture called the Networked Signal Processing System (NSPS), which addresses this multi-sensor, data fusion problem in terms of packetized, heterogeneous, distributed and real-time signal processing.

The NSPS is a co-operative of two kinds of networks, among which one is a custom peer-to-peer network while the other is a part of a commodity processor network. The custom network includes subsystems related to the digitization for high speed data acquisition, and all intermediate routing and preprocessing blocks as the network nodes, in which the emphasis is on traffic shaping, on-the-fly processing and load balancing for effective distributed computing. However, all customized protocols are absorbed while crossing over the last mile to interface to the commodity processor network using a common industry-standard network protocol. The actual estimation of the correlations is carried out by nodes on the commodity network.

In contrast to the traditional use of a packetized network as merely a data transport fabric between processing entities, we use the notion of "a logical packet" based on an application specific "Transaction Unit", which itself may be composed of a large number of physical packets, whose sizes are network-specific. This unit refers to a time stretch long enough to facilitate a dynamical flagging mechanism or/and to relax the constraint on timing, synchronization and scheduling of workloads on the commodity Operating Systems, on which processing is expected to be carried out. Both these requirements necessitate lower level NSPS nodes to be equipped with large memories, which are also used to route traffic selectively (traffic shaping) to higher levels in the NSPS. Two independent considerations have led us towards stretching the transaction unit to a good fraction of a second. One of these, as explained above, is to provide latency tolerance in order to simplify software on standard computing platforms, while the other arises from a desire to make explicit provision for preprocessing using concepts related to Modern Information theory, as discussed in [32].

Another concept in literature which we find useful in the present context is that of "multi-sensor data fusion", defined by [33], as a system model where "spatially and temporally indexed data provided by different sources are combined (fused) in order to improve the processing and interpretation of these data". This model, widely used in applications like military target tracking, weather forecasting etc., has many features relevant for describing the control flow and pre-processing required in a multi-element radio telescope before correlation. In a sense, the NSPS is an adaptation of the data fusion architecture to our domain. Our analysis of the nature of the real-time problem results in a natural partitioning into two broad categories as elaborated in Section 2.3. This is our primary motivation for defining the NSPS architecture, described as a Fusion Tree in Section 2.4. In the next section, we provide a motivation for defining a network centric receiver architecture.

2.2 Network centric approach to array signal processing

The inherent nature of distributed data sources, due to the spatial separation of the antenna elements, suggests that it is important to consider data transportation to a central location and its distribution to various processing elements, to be part of signal processing architecture. However, for the special case of a correlator implementation on commodity hardware, networks can play a bigger role than as mere data transport mediums. In this section, we highlight the diversity and centrality of networks in an array receiver, and the role of effective network management on the efficiency of the commodity component utilization.

2.2.1 Telescope array as a distributed computing environment

The current trend in telescope arrays of deploying a large number of elements with smaller individual collecting areas, to obtain the necessary collecting area (and hence sensitivity), requires large scale computation. However, the computing loads for even moderately sized arrays with software back-ends (e.g., GMRT, or the 40-element ORT) are too large to be sustained on typical commodity computers, unless a high performance cluster out of many such nodes is formed. Thus, an array receiver with a software backed, needs to manage and optimize the usage of several kinds of networks: (a) *long haul network* which brings in data from remote data sources (b)

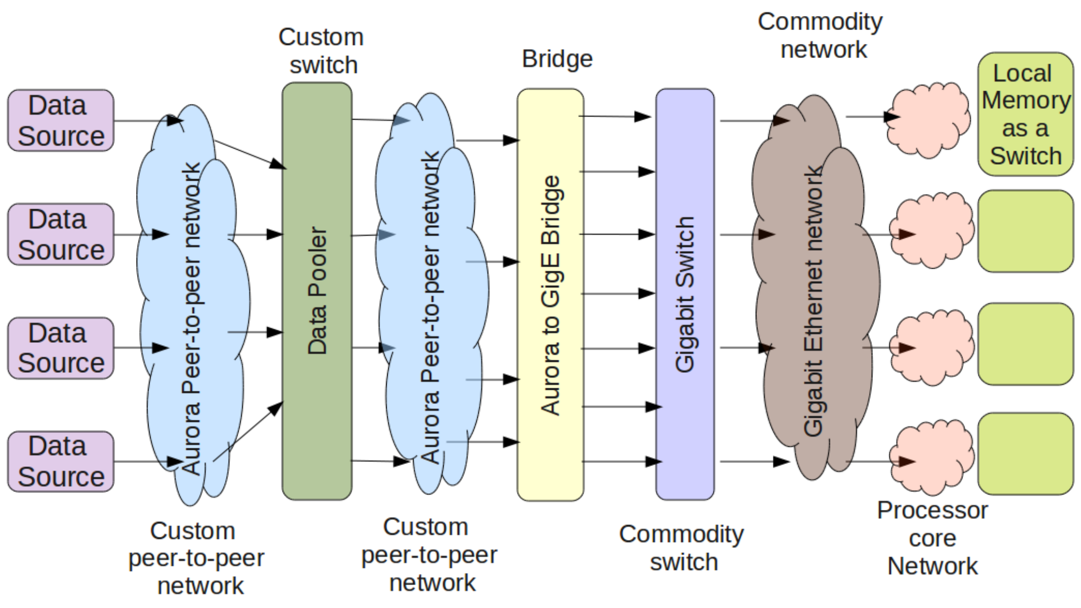


Figure 2.1: *Schematic depicting the various networks in a commodity processor based receiver.*

high bandwidth network between a group of local commodity processing systems, and (c) *intra-processor network* between the many cores making up a modern processor. These are depicted in Figure 2.1, with the shown network technology specific to our implementation.

2.2.2 Load balancing, job scheduling and synchronization in a distributed configuration

As in any distributed computing system, there is an inherent need for partitioning the incoming data, and the subsequent scheduling of such individual partitions, onto the multiple processing elements (i.e., load balancing), in an array receiver. This load balancing needs to be carried out in an optimal fashion, such that a processors' waiting time for data partitions or synchronization events is minimized, leading to individual processing entities remain mostly active. Further, the correlation process requires a fine-grained synchronization between data sources, which also needs to be maintained to maximize coherence.

Further, efficient handling of the increased I/O requires analysis of two aspects: (a) Adequate choice of network protocol stack, which can affect the number (and hence management) of processors, and (b) Proper and efficient routing of all data between processors, which ideally matches their processing bandwidth, with the data rate associated with them. Thus, a high bandwidth network link should terminate into a set of processors via a switch such that the entire processing for all data on the link should be feasible using that processor set.

In our approach, data is delivered by various subsystems to a receiver network by terminating networking links into appropriate switches of various kinds. This is in contrast with conventional approaches, where usually a processor is exclusively required for bringing data into the commodity processors' domain.

The current state-of-the-art systems, consisting of 8-12 processing cores per server, are typically able to process correlations at $\sim$ 40 MS/s per core, for a 40-element array. This corresponds to a $\sim$ 100 MBps rate, which is well suited for a GigE link. Thus, a higher data rate protocol has not been chosen, in order to match the processing and I/O bandwidth per core.

2.2.3 Commodity processors for correlations

Correlator I/O requirements: When implemented with commodity processors, the processing in a correlator turns out to be I/O bound due to several reasons. Firstly, apart from the large volumes of data generated, the correlation operation also has a low compute to I/O ratio, implying that only a few operations are carried out on each byte after its arrival from the I/O subsystem, before it is written back. Further, the estimation of Cross Power Spectra results in an increase in the I/O handling requirement by a factor of N/2 (where N is the number of elements), before data-rate reduction via accumulation can take place. In addition, the drastic increase in I/O due to the all-to-all connectivity required to carry out cross multiplications, ultimately results in processors remaining idle and waiting for data to arrive. Thus, the process is fundamentally I/O bound, with processing elements waiting for appropriately aligned data blocks to reach them, before carrying out the correlation process.

We now examine a conventional software correlator's I/O, computing hierarchy, and its distributed computing environment.

The GMRT software correlator [20] utilizes MPI on a 48-node Xeon cluster, and has a typical approach to the parallel correlator implementation. The total number of Xeon cores available is $\sim$ 112, with the system's total incoming data rate being $\sim$ 2

GBps. In addition, correlations between 64 elements are computed (2 polarizations of 32 antennas). The system has a dedicated *acquisition cluster,* consisting of a set of 16 nodes, each containing a group of four digitizers bundled on a PCI-Express board, on-board a Xeon class machine. It may be noted that this layer is needed purely because of the use of a bus-based interface (PCI-Express), whose bandwidth further restricts the total data streams available to an acquisition layer processor to four. Further, the data dispersion for correlation is realized using commodity GigE switches, which route the dispersed data to a *real-time computing recording cluster*, consisting of 16, quad-core Xeon nodes. Such a dispersion still requires synchronization between different nodes of the compute cluster, over the GigE network.

The GMRT utilizes *software barriers* in order to synchronize processes operating on partitions of the data flow. Such barriers fundamentally cause the processors to idle while waiting for all other processes to reach the same state of computation. In addition, extra communication overhead is incurred on an already congested network between processors, when a barrier implementation ascertains synchronization. This effect is enhanced for the fine-grained synchronization required in streaming correlators, leading to suboptimal processor utilization. Further, scheduling of jobs across processors, or of a temporal decomposition of jobs on the same processor requires a job scheduler, which is an important part of MPI-like implementations. However, these job schedulers are complex, and require periodic status updates of the candidate processors, thus making them non-optimal for correlators, where there is a very strict, and constant relationship between various processors and their data. Thus, correlation processes are best served with very simple and deterministic scheduling.

2.2.4 Network centric approach

We have introduced a peer-to-peer network implemented within the custom hardware domain, to co-operate with a commodity network (e.g., in a High Performance Cluster (HPC) computer) in order to satisfy the overall requirements of real time signal processing. Two independent functionalities are fulfilled by the peer-to-peer network: (a) concentration of data from spatially distributed antenna elements onto a single location, and (b) distribution of this data among the multiple processing nodes of an HPC (i.e., data partitioning and load balancing).

In our approach, we have reduced the need for fine grained synchronization in order to improve processor active periods. This is achieved by making all data pertinent to a compute set, available to a processor locally, and on demand. Further, the all-to-all connectivity between data sources required for correlation is carried out using the local memory of the processing element, which is more efficient than using traditional connectivity. This is implemented via a simple data rerouting within the peer-to-peer network. The re-routed data can then be transmitted to other similar processors in a time multiplexed fashion, which is found to be efficient in reducing overall communication between processors. These rerouting operations can be conveniently carried out by introducing a data concentrator and router in the path, before the data reaches the processing element. Further, the fine grained synchronization constraint on the incoming data can be eased by the introduction of memory buffers on components within the data path, into which high bandwidth data can be written in chronological order of timestamp. This approach also minimizes the effects of missing data on the synchronizing process, as well as allows processors with soft real-time and variable latency responses, to be employed in a strict real-time system.

Thus, our approach is *Network Centric*, where the receiver is seen as a cooperative of networks, due to a large fraction of the above mentioned functionalities being implemented within the custom, peer-to-peer network.

The next section categorizes real-time processing requirements, in order to optimize implementations of each category.

2.3 Real-time processing requirements

Without getting into specific details of the real-time processing required for a large multi-element radio telescope, we abstract them into a combination of three broad categories:

- *Embarrassingly parallel processing*, e.g., spectral decomposition of the incoming time series (say via FFT or polyphase filter banks) and the recognition and management of path-induced distortions/interference on timescales significantly smaller than the integration time chosen for the correlations.
- *Pipelined processing*, e.g., multi-beam formation (say K beams) by phasing N elements requires $Klog_2N$ pipelined operations for each spectral band.
- *Data Fusion* operations, in which the data originating from different antenna elements are hierarchically fused (combined via routing and application-dependent processing) along a chosen set of dimensions which include time, frequency and spatial spread.

The most important fusion operation for an antenna array is the real-time correlation of signals from every possible pair of antenna elements in different frequency sub-bands. Apart from being an $O(N^2)$ process (for an N element array) from a computational point of view, this brings in the additional complication of routing large volumes of distributed data to appropriate data processing elements to provide a complete-graph connectivity between the sources of data and processing elements. Significantly, cross-correlation between all possible pairs of signals is also essential for using self-calibration techniques to enable dynamic calibration of instrumental and atmospheric contributions to the data corresponding to different elements, before they are subjected to an irreversible fusing operation in a phased array. This makes a spectral correlator an implicit requirement, even for a phased array, for minimizing the irreversible loss of information resulting from distortions induced in the the path or the local environment.

In our approach, we bifurcate the requirements of a real-time system into *Commodity* and *Custom* segments. In the current state of technology, the commodity segment can be fulfilled by subsystems available in the market while the custom segment can generally be realized on the basis of customized hardware and/or firmware layers based on COTS technology. Such a bifurcation is explained below for different functional categories of the NSPS:

2.3.1 Computation

- *COTS Segment*: Computationally complex and/or latency tolerant processing, typically realized on a programmable platform ranging from workstations to a high performance cluster.
- *Custom Segment*: Latency critical, logic intensive and repetitive pattern of deterministic processing, well suited for a configurable platform, typically FPGA-based.

For efficient computation, we pay special attention to reducing coupling between data in order to target explicit parallelism at all levels of processing. Multiple parallel circuits are implemented in the FPGA-based custom segment, while the current trend of multicore processors with access to shared memory is exploited in high level software. Further, the desired high signal bandwidth and large number of antenna elements make the processing complex and compute intensive. This aspect, and the advantage of quickly implementing exploratory algorithms, make a commodity compute cluster an attractive choice for the central computing. This is recognized by explicitly including the cluster in the COTS segment mentioned above.

2.3.2 Data routing

- *COTS Segment*: Commercial switches with all-to-all connectivity are used for data routing to commodity processors and broadcasting in the last mile, as well as for load balancing. The routing is controlled by manipulating the destination addresses on data packets. Connection-less protocols like User Datagram Protocol (UDP) are adequate for high-speed, streaming applications where a small fraction of lost packets does not affect performance adversely. Packet collisions are minimized in full duplex, point-to-point connections between network partners, and also because data flow is extremely asymmetric. Further, the criteria for load partitioning discussed in Section 2.4.8 very often result in under-utilization of link speeds to match them to sustainable processing bandwidths.
- *Custom Segment*: Customized switches with static routes for traffic shaping are relevant when only a subset of the network data needs to flow to a subset of the nodes based on certain conditions. They are generally implemented in configurable logic.

Since many FPGAs support Gigabit Ethernet MAC as a hard (or publicly accessible soft) IP, this feature is useful while introducing a bridge between the peer-to-peer network and the commodity network in the last stage of the custom segment. In addition, some or all the major subsystems may have management support from an embedded or explicit on-board processor.

2.3.3 Protocol and network topology

- *COTS Segment*: A commodity network compatible with a typical high performance compute cluster, which includes Gigabit Ethernet as a *de facto* standard for interfacing with external systems.
- *Custom Segment*: A peer-to-peer network which may include significant on-the-fly application specific operations, suitable for implementing on a standard FPGA platform.

An implementation of the actual processing on a dedicated set of identical hardware circuits or parallel processors can take advantage of an intelligent network capable of elementary on-the-fly operations to achieve a balance on the dataflow and

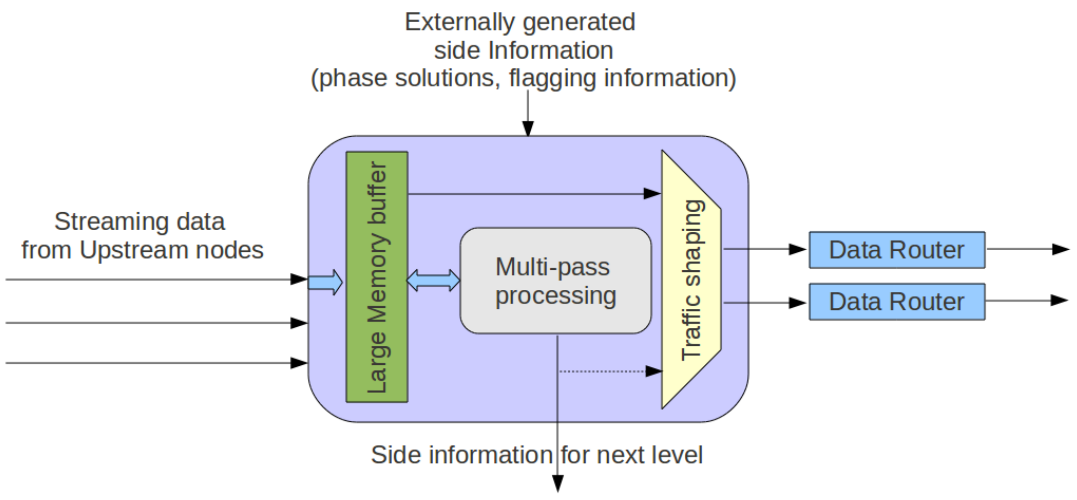


Figure 2.2: *Data Pooler node of the NSPS shown carrying out data fusion and traffic shaping.*

computing requirements. This is depicted in Figure 2.2, where a "Data Pooler" node, as defined in Section 2.4, is shown carrying out real-time fusion of the streaming data by using the side information made available by external sources. At the same time, the pooler is seen generating side information out of the fused data set by way of multiple processing passes on the stored data. The pooler can then segregate the data, and route the segregated components to different data sinks using the routing information available with the peer-to-peer link nodes. We use "traffic shaping" here in a more general sense than in Internet traffic shaping (which delays lower priority packets in favour of better network performance of higher priority packets) to refer to both segregation of the incoming stream, as well as the specific routing of segregated data to different sinks. Further, the efficiency of hierarchical computation can be significantly improved by accommodating some degree of pre-processing and/or partitioning of data in each level of the custom segment to facilitate the next level.

2.3.4 Process scheduling

- *COTS Segment:* The overall task supervision, command and monitor, user interface and the dynamic system monitoring are tasks whose complexity is best left to the commodity segment to handle, where a variety of tools ranging from MPI, compiler resources and advanced operating systems like Linux or VxWorks are available.
- *Custom Segment*: Event driven scheduling with periodic or quasi-periodic events

generated conveniently in a low latency logic implementation suitable for an FPGA. The interval between the events is stretched to handle an application-specific transaction unit to the extent permissible within the available resources.

2.4 NSPS as a Fusion Tree

In this Section, we present the NSPS architecture as a *Multi-sensor Data Fusion Tree,* in which both conventional and "virtual" sensors play a role. While entities like antenna elements, round-trip phase/delay monitors, noise calibration etc. can be treated as "conventional" sensors, "virtual" sensors result from processing blocks at various levels. For instance, pre-processing can result in a flagging mechanism to improve the reliability of fusion systems like correlators, in which the original data is erased while compressing its information into a statistical estimate, to be passed to the next level.

The signal processing system proposed in this paper is a set of spatially separated nodes of varying communication and processing capabilities, which are interconnected by a customized high speed tree-like packet switched network interfaced to master commodity nodes. This is equivalent to a *Data Fusion Tree*, with the nodes of the tree performing operations like traffic shaping, packet routing, or pre-processing before data fusion. Accordingly, we have described the overall architecture of NSPS in the form of a *fusion tree*, schematically represented in Figure 2.3. However, each level provides a different mixture of functional capabilities. This has resulted from our recognition of the following features of the functional requirements:

- Distributed computing across many nodes of different processing capabilities, with local parameters guided by a processor capable of seeing a subset of all data.
- Data routing nodes with configurable routes for routing preprocessed data subsets.
- Nodes with large memory buffers to enable multiple passes on data, also for enabling memory based data transposition.
- A High speed network interconnecting all nodes, with ability of COTS nodes to tap into the network.
- Interface to a master commodity node through a standard interface like Gigabit Ethernet (GigE).

Thus, from a functional point of view, we classify the nodes in NSPS into the following three categories:

1. *Data Poolers/Fusers:* Nodes with sufficient on-board memory to allow packetizing and multiple processing passes on incoming data. These break the need for many-to-many connectivity in the correlation process by transposing data via memory based switches. The transposition, based on different parameters, ultimately serves a packet of data suitable for processing by a single element of a distributed system in an embarrassingly parallel manner.
2. *Data Routers:* These elements are endowed with high speed links to either peers or more powerful processors to whom incoming packets are routed based on statically configured routes. These form an integral part of our architecture, helping in the load balancing by directing appropriate subsets of preprocessed data to different processing elements. We use commodity switches for routing data to multiple external sinks by forming many-to-many connections between data sources and data processors.
3. *Data Processors:* These elements have high compute density and can be used for preprocessing, as well as for data rate reduction. We classify processors into two groups as mentioned earlier:
   (a) Those catering to computationally complex and/or latency tolerant processing, an example of which is the estimation of system calibration parameters based on a long (few minutes) history of data, and its dynamic updation. This processing is generally carried out by sending a subset of the data to a central processor.
   (b) Latency critical, logic intensive and repetitive pattern of deterministic processing. An example of this class is the real-time block-level data encoding process requiring the estimation of block level statistics. This can be carried out by multiple passes on small segments of data.

Thus, we visualize the NSPS as a *restricted* distributed system, depending primarily on stripped down lightweight networking protocols and the static routes set up during system configuration. Data routers, both customized and commodity, play an important role in reorganizing data to be computationally palatable to processing nodes in this scheme. A master node is in charge of command, configuration and control, and is almost always a commodity node like a PC. It may be noted that custom processing is spread across the NSPS tree by explicitly advocating local intelligence in every node. As an illustration of the inherent facilitation of distributed processing in the NSPS, some important aspects of the interconnection mechanism are elaborated in

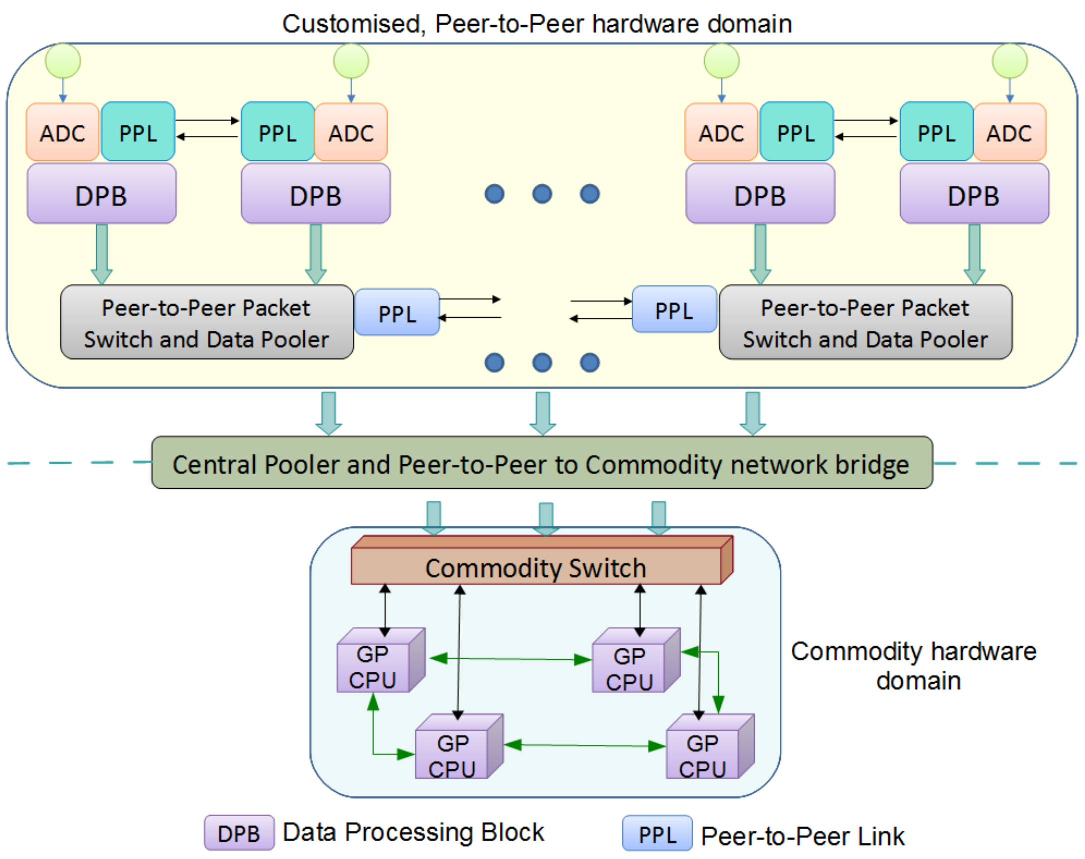

Customised, Peer-to-Peer hardware domain
ADC
PPL
PPL
ADC
ADC
PPL
PPL
ADC
DPB
DPB
DPB
DPB
Peer-to-Peer Packet Switch and Data Pooler
PPL
PPL
Peer-to-Peer Packet Switch and Data Pooler
Central Pooler and Peer-to-Peer to Commodity network bridge
Commodity Switch
GP CPU
GP CPU
GP CPU
GP CPU
Commodity hardware domain
DPB Data Processing Block
PPL Peer-to-Peer Link

Figure 2.3: *Conceptual layout of a Networked Signal Processing System architecture showing the principal participating entities.*

the following subsections.

2.4.1 Fusion tree interconnection model

Interconnects in the Data Fusion Tree consist of the following four essential graphs, which can either be logical or explicitly physical manifestations.

1. *Data* network is a simplex, high bandwidth net connecting the leaf nodes of the Fusion Tree to a central processor, possibly passing through several collation levels of the signal processing tree.

2. *Control and Monitor* network is a full duplex, low bandwidth network and interconnects all nodes hierarchically through management processors to a central monitoring station.

3. *Calib* network is a full duplex, low bandwidth network. This allows calibration information to reach the data fusing nodes before the sequence of irreversible fusion operations take place.

4. *Clock* network is a full duplex network, providing the distribution of clocking signals to the various nodes, as well as allowing a round trip clock phase measurement.

In actual implementations, it may be simpler to realize these in terms of a set of simplex networks, among which clock, control, monitor and calib are directed towards the leaf nodes while the data and status (including response to monitor queries) belong to simplex networks which flow from different levels of the fusion tree into the master node.

Our network implementation can be bifurcated into the following sets:

1. Customized high speed serial peer-to-peer links terminating into peer-to-peer switches which implement a subset of the complete graph connectivity.
2. Commodity high speed serial links terminating into commodity networking equipment, with ability to interface to standard processing nodes.

In typical implementations, we expect custom links to be on a Passive Optical Network (PON) based communication stack. Gigabit Ethernet is the preferred choice for the backbone of commodity segment to connect to the custom network.

We exploit the high speed serializing ability of modern FPGAs to dispatch data on high bandwidth copper links and use PON components to meet the spatial spread required to reach remote nodes over fibre links. As long as the bandwidth requirements are met, no specific preference is implied for a choice among different networking technologies. Thus, some implementations may utilize the embedded multi-gigabit serializers in FPGAs for peer-to-peer links while others may refer the cross dispersion of data to a compute cluster's high bandwidth infiniband network, or use commodity Gigabit Ethernet. This leads to the need for a flexible bridging mechanism which can be exploited by implementations. For instance, data can be conveniently transmitted over a peer-to-peer or a commodity link due to the maintained commonality of their interfaces.

2.4.2 NSPS tree network characteristics

The high speed network internal to the NSPS tree has features which are restricted and stripped down versions of those found in commodity networks. This is an optimization due to the highly controlled network which exists within the telescope receiver environment. Our network differs from regular networks in several aspects:

- A controller node is assigned for every sub-tree at a given level. This entity forwards command and status information between controller (level-0) and upstream levels. Thus, it is not a typical peer-to-peer network which does not have such a hierarchical control structure. Broadcast and multicast domains available in commodity networks are used to implement control and monitor mechanisms, while explicit “pull” mechanisms are implemented on the custom network. Here, the “pull” refers to the explicit request for data made by a downstream node to an upstream node. The advantage of a “pull” mechanism is that data is made available to a downstream node only when it is ready to handle the data, as inferred from the downstream node’s request. Also, if a downstream node is busy, then the upstream node loses data in integral packets, thus maintaining timing information.

- The network configuration and routes are fixed statically in an application dependent manner at configure time. There is no node discovery, and data routing does not have an explicit mechanism for handling node failure. Since all data flows towards a logical sink, there is also no destination address, although source addresses can be preserved. This simplifies network management to a large extent at the cost of non-redundancy of NSPS entities.
- Communication protocols: All elements in our network, including master nodes, generate similar kinds of packets which contain an 8 byte header with fixed fields. These are typically indicative of the nature of accompanying data, as well as its timestamp, source and other meta information. System state can also be propagated through these packets, or by forming special status packets. The restricted meta information processing makes it simpler to realize packet formation in hardware with simple state machines.
- High speed serial interconnects: All entities in the internal network communicate via high speed serial interconnects with a clock recoverable from the encoded data. This approach allows us to transmit data long-haul over fibre, or short-haul over copper without any changes. In particular, we discourage bus-based interconnection between physically separated nodes.

2.4.3 Interface to external network

Once preprocessed data is ready within the NSPS, it needs to be transferred onto a commodity network for reaching commodity nodes for post processing or archiving. Local intelligence in the peer network can be used to partition the data such that the interfaces to commodity nodes use link speeds commensurate with their processing ability. For simplicity, we have used Gigabit Ethernet as a typical standard external interface. This is a popular high speed serial interconnect with a vast amount of infrastructure available in the commodity market. It also allows transmission over copper (UTP) to interface directly with commodity servers, or fibre (via conversion to 1000BaseX) for long-haul transmission. Commodity servers of moderate ability can then be used as data sinks with minimal customization. This is also motivated by the fact that many modern FPGAs have embedded high speed serial interconnects on chip, with complete Gigabit Ethernet support in the form of on-chip Gigabit MACs or as publicly available libraries.

For the last mile connectivity, UDP can be used since it is a simple connectionless protocol with minimal overheads on top of IP. It is also possible to fill the relevant UDP fields during system configuration, and hold them static for the duration of an observation. Each of the internal network types carrying data (*Data, Calib* and *Control*) can then be easily made available over a different UDP port as part of the design. This allows an application program to associate independent threads to service these streams.

2.4.4 Control and Monitoring network

The distributed nature of our architecture requires status monitoring of all nodes and links, which can be handled by the individual sub-tree roots and communicated to the master controller. This is implemented by a status "pull" scheme by which controlling entities periodically query the status of all nodes in the NSPS tree rooted with them by way of a special AYA ("AreYouAlive") packet. The nodes respond with an IAA ("IAmAlive") packet containing selected status information. Similarly, control packets contain command and configuration data. Each command packet typically results in a status reply from the targeted entity, which confirms the receipt of the command, and regenerates a control packet for the entities controlled by it. The master node can use this in an appropriately scheduled housekeeping operation to discover failure of nodes.

2.4.5 Calibration network

This network is meant to carry data from the NSPS tree which is relevant to forming calibration solutions for the array. The calibration mechanism is to be applied differently for the two main modes of observation with the NSPS:

- In the interferometric mode, the correlator can work independently of the actual gains and phases of the sensor elements, since the observations include calibration scans at reasonable time intervals. Off-line processing can infer intermediate variations by supplementing interpolation between calibration scans with dynamic calibration schemes like self-calibration based on the partial, low bandwidth dataset available over the calibration network.

- On the other hand, real-time beam formation includes an irreversible fusing operation which requires dynamic calibration to be part of data fusion. Fortunately, it is often possible to use a relatively small subset of the data (non-contiguous timeslices or a chosen frequency sub-band) for this purpose to enable short term predictions of gain variations. These can be fed to the fusing nodes well in time before irreversible fusing operations are performed. Since the complexity of the actual algorithm used for calibration makes it better suited for a general purpose computer, the *calibration network* can be used to route the relevant subset of data to a commodity switch and deliver the calibration parameters to the appropriate NSPS tree level.

2.4.6 Clock network

In the spatially distributed, direct RF sampling NSPS architecture, clocks passed to samplers have very stringent signal quality constraints in terms of net jitter and stability. The alignment of the multiple data sources before fusion requires high relative stability of the sampling clocks with random jitters much smaller than the reciprocal of the highest frequency in the sampled signal. Clock distribution should also include a mechanism for ensuring the traceability of timekeeping at all digitizing blocks to a centrally maintained time standard to a very high accuracy. The implementation can benefit from commercial clock distributors which have embedded phase-lock loop clock synthesizer with on-chip Voltage Controlled Oscillator (VCO) and a per port delay tuning for the distributed clocks.

2.4.7 Data network

All data flowing in the NSPS is packetized with a custom, low overhead header. All subsystems accept and generate data in a packetized fashion. This reflects the inherent asynchronous nature of our system. Packets traversing our platform are atomic and capable of independent existence. The packetizing of data means that data loss due to network congestion or buffer over-runs is never arbitrary, but always in units of packets. At any instant, our network can have different kinds of packets traversing it, corresponding to different stages in the processing. The basic unit of packet size is maintained as 8 bytes, which is a natural unit or sub-unit for different memory and processing hierarchies. Adequate padding is used if necessary to maintain this condition. The header is mandated to have a few fixed fields which are common in size and layout across packet types, allowing processing entities to examine packets which can be processed by them, while discarding the others. In a broadcast network, this approach can waste bandwidth when packets are discarded, but the wastage can be minimised by setting up static routes between partner nodes. This is possible in both the custom and commodity peer-to-peer link nodes. The Command network, on the other hand, is a broadcast network, with nodes passing on commands not addressed to them to all other nodes downstream of themselves. Data sources can include packet specific extensions to the packet headers generated by them. The following fields are suggested as a mandatory part of the packet header:

- Source identifier: At every level of the tree, nodes are endowed with a unique identifier which supplants the existing upstream source id, if any processing is carried out on the packet.
- Datatype: This field allows processing entities to recognize which packets are palatable to them and to reject others.
- Data pixel descriptor: This field lays out the size and description of the smallest unit of data transfer to be one of an allowed set, which is implementation dependent.
- Streams: This field records the number of independent signal sources present in each packet.
- Packet size: The size of a packet is expressed in units of words as specified by the datatype field.
- Timestamp: This field is populated as early as possible in the data generation path and maintained across data processing. This field is generally populated by a timestamp counter running on either a reference clock or on the sampling clock itself and is traceable to the centrally maintained time standard.

This specification is efficient for real-time streaming data description with minimum overhead. For archival of processed data, a standard format which allows multiple binary streams to maintain their identity, like the FITS or VLBI Data Interchange Format (as proposed by the VDIF Task Force (2009)) can be used.

2.4.8 Load partitioning and scheduling

No subsystem in our scheme is source synchronous, be it at the hardware or the software level. All subsystems have enough memory for a store-and-forward of several packets. This allows the processing to happen at the packet level, on a faster clock than the sampling clock of front end ADCs. It also eases the timing requirements of designs implemented in FPGAs and makes them more tolerant of clocking errors. Sequencers play an important role in our architecture, generating events on which the processing progresses. The sequencers generate necessary globally aligned events to which any action taken on the basis of commands from commodity network will get aligned. A simple example is an implementation where all processes in the peer network operate at a block level, with a periodic event signifying the need for scheduling a new process as a result of the arrival of a new block of data.

It is important to match the communication bandwidth to processing abilities at every level in the signal processing tree. More specifically, event markers generated by the sequencers should facilitate a partitioning of processing in each level into abstract transactions, where each transaction deals with the entire data collected over a convenient timeslice **and** the relevant data are locally available *on demand*. In particular, it is desirable that processing at the central commodity segment is facilitated at a cadence suited for general purpose operating system scheduling to achieve latency tolerance. For instance, to be commensurate with a housekeeping tick of 10 milliseconds in typical Linux configurations, the transaction timeslices should be several times longer.

2.4.9 Choice of fusing dimension

Among the available axes in processing space along which the data can be partitioned and distributed to parallel processors, the time axis is often the most convenient for slicing, as individual timeslices can be considered independent. For a large network, we realize this from a hierarchical set of Data Poolers populating different levels of the processing tree, which can collate data from different sources and partition them along the time dimension at each level.

2.5 Discussion

While asynchronous, packetized processing over standard networks is a relatively new concept in radio telescopes, it is being embraced enthusiastically due to the many benefits it offers to the system designer. Even among this class of telescope data processors, contemporary architectures usually have a direct link from the samplers to the central processor. Some operations like a digital filter bank or FFT are carried out remotely, while others like cross-correlation is done centrally. The memory-rich architectures of modern FPGAs help in distributing computing to remote nodes and enables buffering to allow multiple passes on the streaming data. As the data volume grows, e.g., in the central pooling stations, the processing can be supported by large off-chip memory using commercial memory modules, routinely supported by modern FPGAs. This provides substantial enhancement to buffering for transaction level operations and data partitioning. The use of standard software stacks also allows us to leverage the various high performance modes being worked upon by system optimizers, e.g., the zero copy mechanism on Linux.

Another challenging problem with large arrays is the so-called *corner turning* problem, which refers to the transposition of the input signal matrix needed to achieve the all-to-all communication necessary for correlation. Earlier approaches have looked at either commercial switches or entirely customized switches for routing data[34]. We break this problem down into levels, and apply a hybrid of commercial as well as custom routing. The Data Pooler element is utilized to implement a memory based switch, while the COTS (GigE) network controller manages another level of redirection by manipulating UDP destination addresses.

2.6 Conclusions

We have presented a packetized, heterogeneous and distributed signal processing architecture for radio interferometric signal processing which elevates the network to a core system component. The architecture addresses some of the core issues pertaining to interferometric signal processing. We visualize this problem as that of an appropriate workload creation and scheduled dispatch to matched processors over a data flow tree. Here, the leaf nodes are sources of data, with data processors handling a managed slice of the processing at the intermediate nodes of this tree. We emphasize the use of COTS components, both hardware and software, for rapid deployment, ease of maintenance, and lowering the cost of the architecture implementation.

The goal of realizing a programmable telescope with NSPS is facilitated by defining rigid interfaces between both hardware and software components. This can allow exchange of a variety of data with varying communication and computing requirements between levels in the network. Most of the individual nodes in the NSPS can change the nature of their processing within the limits specified by their designed personality and available resources at the node. This allows offloading of computing requirements in a hierarchical manner up the NSPS tree, trading off implementation time with hardware capability of an application mode. Due to the rigidity of interfacing protocols as well as the standardized networks making up the system, we can comfortably add nodes which can tap into the NSPS in order to carry out a different processing chain. Data duplication, if required, can be carried out by COTS components (e.g., by switches operating in broadcast mode) thus reducing development load.

We have presented an outline of the NSPS implementation being planned for configuring the ORT as a programmable 264 element telescope. Our architecture is optimally tuned to service the needs of medium sized arrays. We advocate full software processing for smaller arrays, with an increasing factor of hardware offload as the array size grows. This approach has being taken by us in building a 40 element demonstrator as a precursor to the receiver for the full 264 element ORT array. This receiver exploits all NSPS aspects we have dwelt on, and is in an advanced stage of completion.

Chapter 3

NSPS Implementation for Ooty Radio Telescope

3.1 Introduction

The Ooty Radio Telescope (ORT) is currently undergoing a major upgrade as part of a plan for its reconfiguration into a programmable array, with enhanced sensitivity and an order of magnitude increase in its field of view. This will be achieved by effectively digitizing every 1.92 meter section of the 506 m line focus of the ORT cylinder, leading to an equispaced, linear array of 264 antenna elements. In the first phase of this upgrade, the ORT has been reconfigured into a 44 antenna element array, by tapping the 327 MHz RF output of every half-module (which is itself constituted of a phased array of 24 dipoles) of the array.

In this chapter, a real-time processing system for this array is presented. This system, based on concepts of the Networked Signal Processing System (NSPS) architecture, consists of prototype custom hardware implemented as a test-bed for the major NSPS architectural concepts, as well as a full software backend. Thus, the focus of the current work was on validating the new RF sub-systems and digitizers for the upgrade and establishing efficient protocols for the custom and commodity network sections of the NSPS. In addition, critical aspects of the sampling clock distribution and synchronization between the distributed data sources, as well as data acquisition into the commodity hardware levels of the NSPS, have been addressed. A part of the processing and large memory management required by the NSPS architecture has thus been simulated in software running on commodity hardware. Further, the data I/O and processing throughputs sustainable on the target hardware were found to be adequate for implementing a real-time receiver for the 44-element array, although the full 264-element receiver implementation would require some aspects of pre-processing and data routing to be shifted to configurable hardware. It may be noted that although the full upgrade implementation is complicated due to the larger number of communicating entities, many of the main concepts of the NSPS have been demonstrated in the presented digital receiver implementation.

Thus, in this chapter, we present technical details related to reconfiguring the ORT as a wide-field instrument equipped with a sensitive, interferometric receiver, from its current role as a phased array telescope. Such an upgrade leads to increased sensitivity, field of view, high temporal and spectral resolutions, in addition to the ability to rapidly reconfigure the instrument. These features enable several new classes of observations, notably, faster surveys, searches for variability in the sky via rapid sky monitoring and high sensitivity observations of large scale structures.

It may be noted that out of the 44 half modules initially released with the upgraded RF signal conditioning subsystem, 4 half modules at either end of the array (S11 and N11, north and south) are being used to carry out RF validation of various subsystems, and are not considered in the description presented here. Thus, a 40-element RF system has been commissioned, and these 40 elements alone are interfaced to the upgraded digital receiver, although the software backend is flexible enough to incorporate the additional elements. Here, the interferometric receiver system is an implementation of the NSPS architecture, which deals with enabling a high speed, scalable, communication and signal processing network, as well as providing the ability to distribute the data dispersion (necessary to bring aligned data from different antanna elements for correlation) across various NSPS hierarchical levels. Further, since the NSPS architecture maximizes the use of commodity computing and communication, the implementation utilizes off the shelf components to the extent possible.

However, for many applications, the typical cache hierarchies and synchronous, burst oriented RAMs of commodity processing systems limits their efficiency, e.g., with random accesses of data in memory. They are also bottlenecked during inter-processor or inter-core I/O, for data bandwidths of the scale required by streaming correlators. Further, repeated queries on the streaming data and multiple data exchanges are generated between the distributed data sources, in order to establish the high degree of synchronization needed before the spatial correlation function can be reliably estimated. Commodity hardware is inefficient for such operations. Thus, we split the receiver implementation between commodity processors and custom hardware, wherein, the former caters to the core processing, while the latter carries out data shuffling and routing.

Here, we describe the role of the custom hardware section of the NSPS in our hybrid approach to the correlation estimation problem, as well as the crucial aspects of the system software architecture. Further, a system design for the full 264-element processing system, while establishing links to the 40-element system implementation, is presented. The core computing of the correlator, along with the post-processing required for calibrating the array, and generating the various modes of operation of the receiver, is detailed in Chapter 4.

3.2 ORT infrastructure: The legacy system

The ORT (introduced in Chapter 1) usually operates in phased-array mode, and is currently equipped with several heterodyning receivers for its various modes of operation. The upgraded receiver system described here has been implemented using several existing sub-systems, reviewed in the next sections. Following this, a description of newer infrastructure for RF conditioning is given, which was developed as a first phase of the ORT upgrade program.

3.2.1 Frontend

RF Frontend

This subsystem is composed of an array of 24 dipoles (termed a “half-module”), along with a passive, christmas tree arrangement of 4, 2 and 3-way combiners, in order to form a single combined RF output from the dipole array. Every dipole, with a separation of 0.478 m ($0.5208\,\lambda$, where $\lambda = 0.9182$ m, corresponding to the center frequency of 326.5 MHz) from the next, is equipped with an LNA and a stripline phase shifter capable of applying a phase to the incoming RF in $\frac{\lambda}{16}$ units. This allows the combined array beam from 24 dipoles to be steered within an effective range of $\pm 30^o$ in declination. It may be noted that this applied phase is the so called “proper” phase, corresponding to the residual phase after integer multiples of 2π have been removed. This leads to bandwidth decorrelation in the RF frontend output, for large bandwidths. The RF frontend [30] is depicted in Figure 3.1, and is used by all the available receivers. It is implemented in the Aluminium channel along the line feed of the ORT. It may be noted that the RF Frontend output (corresponding to an 11

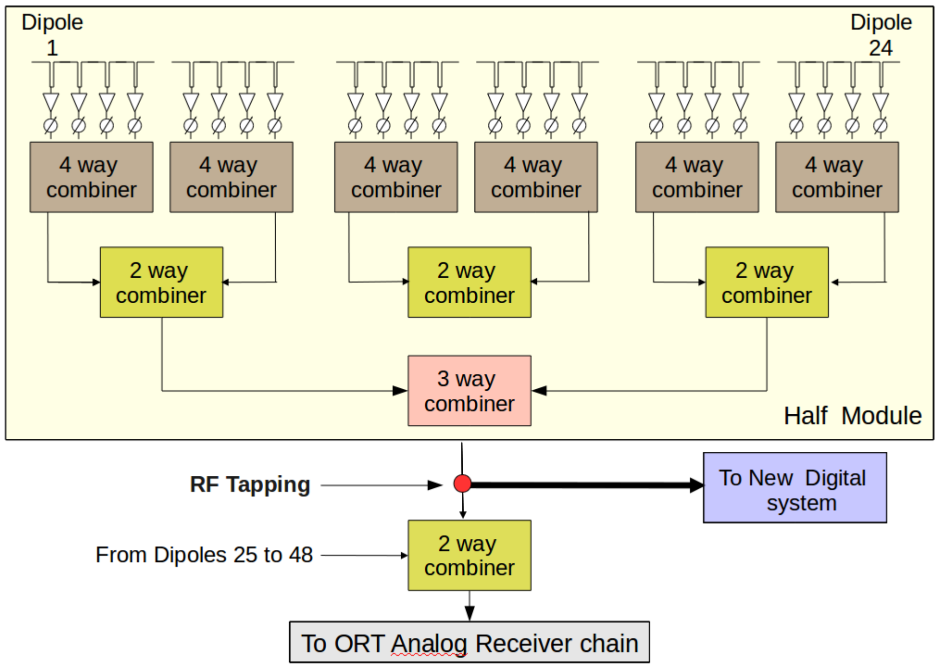


Figure 3.1: *ORT RF Frontend for an array of 24 dipoles (Half module). The RF tapping location for the RF digital receiver is shown. The RF frontend is implemented within the line feed of the telescope.*

m section of the ORT array) does not reach the central receiver room, and is thus not available to any of the existing backend receiver systems.

The Frontend IF conversion subsystem

This subsystem carries out the heterodyning of the combined RF output from the RF Frontend, to an Intermediate Frequency (IF) of 30 MHz. It operates on the output of two half-modules, and consists of a 2-way combiner to create a single module output, which is amplified, passed through an image rejection filter, and then mixed with an incoming Local Oscillator (LO) at 296.5 MHz. This sub-system is depicted in Figure 3.2, and is implemented within the per-module feed support structure of the ORT. The resulting IF output is then transmitted to a central receiver room over equal length cables, where the rest of the IF processing is carried out. This subsystem has the ability of tuning the per-module LO phases for applying a broadband phase correction, in order to enhance the sensitivity of the phased array outputs from the complete ORT.

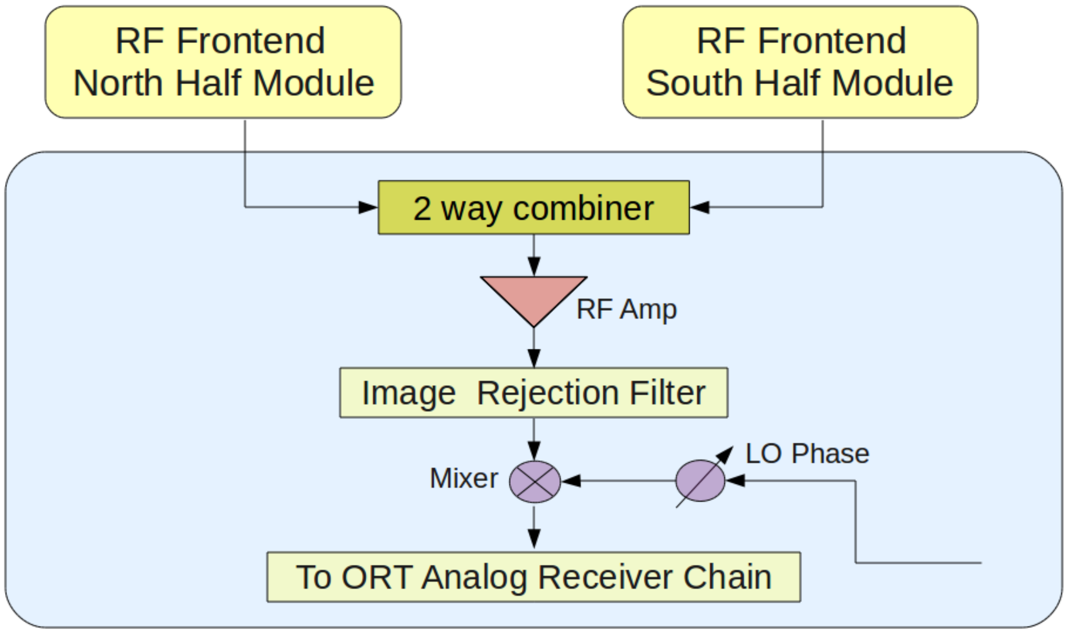


Figure 3.2: *Frontend IF conversion subsystem, which generates a single IF output by combining the RF of 48 phased dipoles. The generated IF is then sent to the central receiver room for further processing.*

3.2.2 Backend receiver systems

The ORT is equipped with an analog IF receiver [31] which operates with a $\sim$ 4 MHz bandwidth. It has 12 identical analog beam-formers, separately for the north and the south half of the telescope, which generate 12 phased array beams separated by $3'$. Corresponding beams from each half of the telescope can then be correlated using an analog correlator, resulting in 12 correlated beams with the full resolution of $6'$x2^o. Thus, the total field of view is $\sim 72'$x2^o. The sensitivity of the system is typically about 25σ per Jy, for a 1 sec integration, and the system temperature T_{sys} is measured to be $\sim$ 150 K [35]. The schematic of the signal flow of this receiver is shown in Figure 3.3.

Recently, a new digital IF receiver [36] has been implemented as an enhancement to the existing analog receiver. It has an increased bandwidth of $\sim$ 10 MHz, while the beam forming is carried out via software running on commodity hardware. The receiver includes LO phase control and a multiphase clock for compensating fractional sampling clock delays, in order to achieve high sensitivity. The functional block diagram of this system is shown in Figure 3.4.

It may be noted that the above receivers operate from the central receiver room, and only on the received IF signals. Thus, they have limited configurability, and a

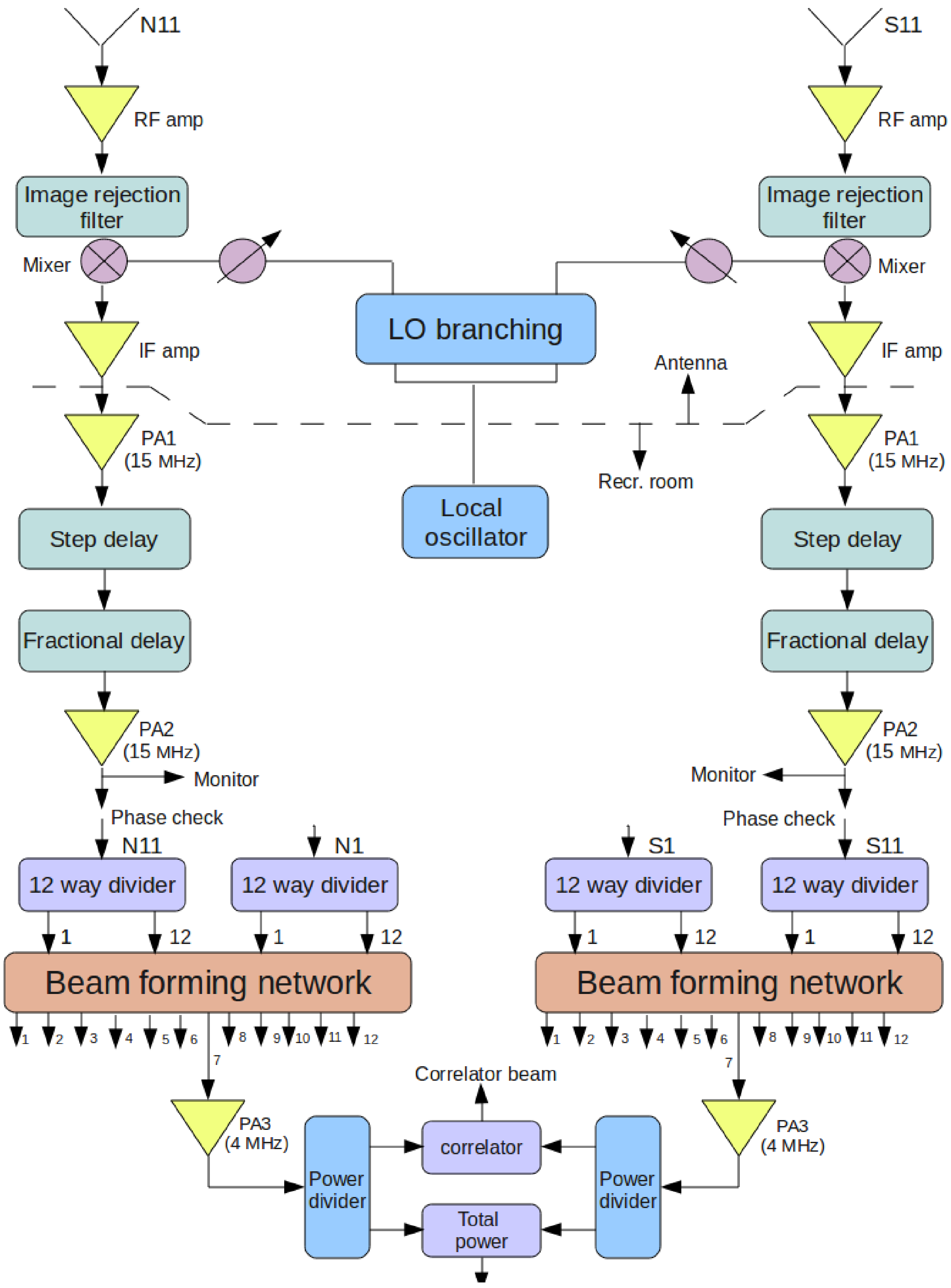


Figure 3.3: *ORT analog receiver signal flow and main components. The system is capable of generating 12 phased array beams from the north and south half of the telescope, each separated by 3', and correlating the corresponding north and south beams. Delays are implemented using switched cable lengths.*

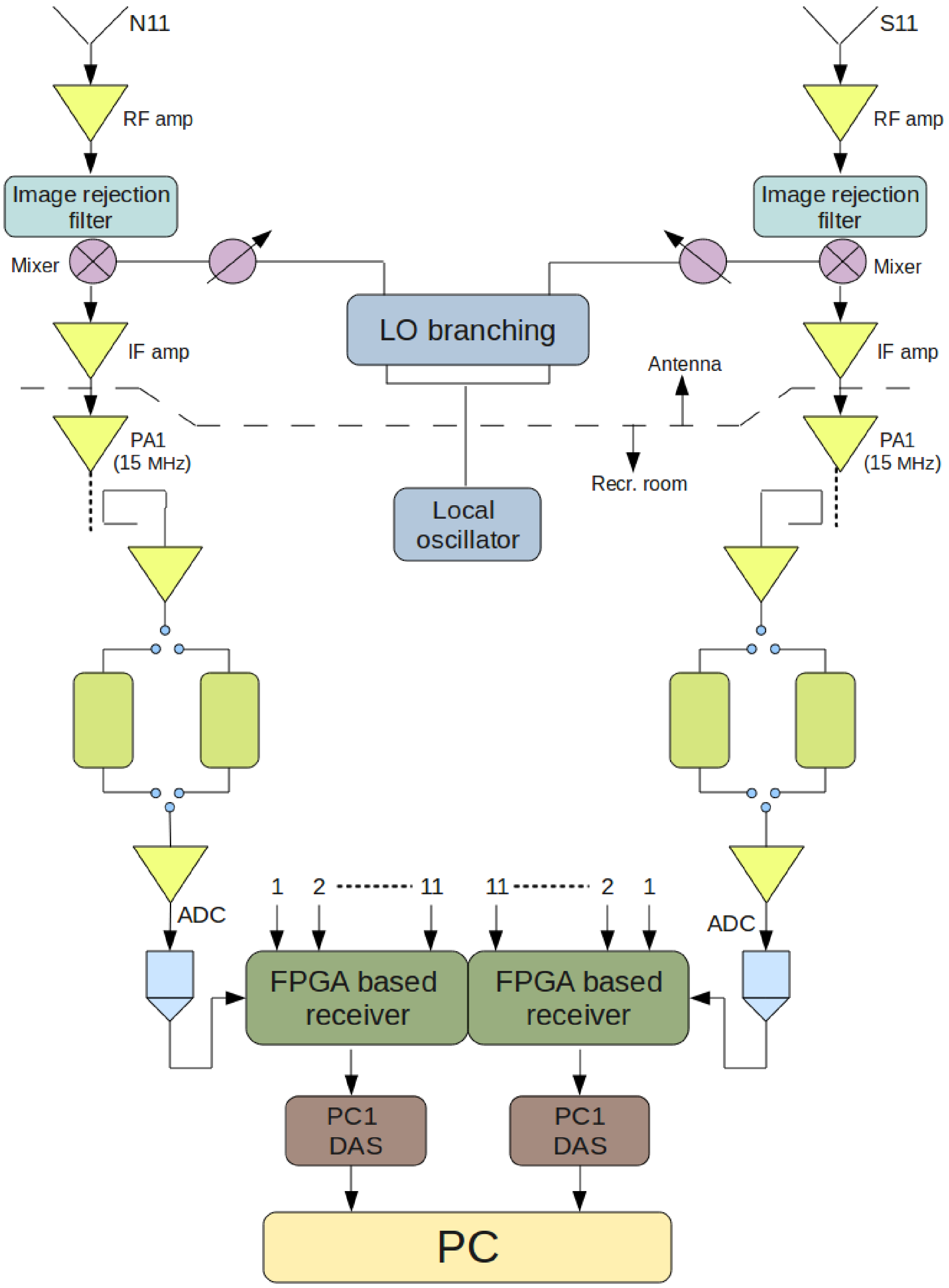


Figure 3.4: *ORT digital receiver signal flow and its main components, along with the signal conditioning block. The IF from each module is digitized, and the rest of the processing is carried out on commodity processors.*

restricted field of view which is determined by the combined beam width of 48 dipoles being phased within the two half modules.

Further, the existing ORT receiver configurations allow for carrying out observations, which include, e.g., Inter Planetary Scintillation studies [37], Pulsar observations [38], Radio Recombination line observations [39] etc. However, certain newer classes of observations, which can be highly benefitted by the ORT instrument design, require higher sensitivity and a larger field of view, in addition to a higher degree of configurability than currently possible. Thus, the instrument was required to be coupled with a configurable and sensitive receiver to carry out such observations. A further decision of allowing the co-existence of the existing heterodyned receivers, along with minimizing the downtime of the telescope was taken, which had implications on the design of the upgraded systems.

It may be noted that apart from the above, several dedicated receivers for different types of observations have been built in the past, including a pulsar receiver [40], an autocorrelator and spectral line receiver [41]. These have subsequently been phased out, highlighting the need for a flexible receiver capable of taking on multiple roles.

3.2.3 Telescope control and receiver interface

The ORT array is steered mechanically in Right Ascension (RA), and electronically in declination.

RA (East-West) control: The RA steering is controlled manually, with the current antenna position available to the telescope operator via a read-out of the antenna encoders. This position information is also made available to the upgraded ORT receiver system over a system network, where it is recorded periodically into the headers of the generated correlations by control threads within the upgraded receiver.

Declination control: The declination is electronically set, and traditionally consists of two stages. In the first stage, the RF phase of each dipole is set (in units of 22.5^o), such that the combined half module (24 dipole) beam is pointing towards the chosen declination. This control is carried out by a declination control PC which, in turn, sends the 4-bit phase setting data over an HDLC link to every dipole. The declination control PC, in turn, can be commanded by a control thread within the upgraded receiver. Further, while the earlier IF receivers required another level of IF delay setting before beamforming, a phased array response in all directions within the beam is created with the upgraded system, due to which only the RF declination setting mechanism from the existing control path needs to be maintained. It may be noted that certain declinations can have a pointing error of upto $8'$, due to the quantization of the applied phase. The RF declination control system is described in [42].

3.2.4 Distribution of synchronizing signal

The ORT has an available mechanism for carrying out a coherent replication and distribution of a centrally generated LO signal to the IF conversion frontend in every module. This mechanism, used by earlier IF receivers, consists of a christmas tree arrangement of power splitters over equal length RF cables. In implementing the upgrade, we have utilized this available network in order to transmit a synchronizing signal at $\sim$ 50 MHz (described later in Section 3.5.7) from the center to the remote digitizers. It may be noted that while the synchronizing signal is combined at the center with the LO before being transmitted to remote elements, it is effectively filtered out before the LO is used in the mixers to generate the IF. This was found to have no noticeable effect on existing systems.

3.3 ORT upgradation

Keeping in mind the constraints placed by existing receivers, and the suitability of the ORT for certain classes of observations, an upgrade of the ORT is in progress. Here, we present details of a reconfiguration of the ORT as a 40-element array, with a new interferometric receiver. To this end, the ORT array analog frontend was modified, and the received RF from each of the 40 half modules was suitably tapped and conditioned, as explained in Section 3.3.1. Signals with a bandwidth of $\sim$ 20 MHz from each of the 40 array sensor elements were then fed into a 40-element complex spectral correlator receiver, which was designed and implemented using a simplified NSPS architecture.

3.3.1 RF signal conditioning subsystem

This subsystem caters to tapping the combined RF from the existing ORT RF frontend (Figure 3.1), conditioning the tapped RF for direct RF sampling, and feeding it into the new NSPS based digital receiver. Further, in order to allow co-existence of the IF receivers, this subsystem creates an appropriate coupled signal to feed to the existing frontend IF conversion subsystems. Thus, the RF signal conditioning subsystem effectively bypasses the frontend IF conversion subsystem for the RF digital receiver. In addition, it amplifies the RF such that the signal noise is dominant over the quantization noise of the ADC. It also minimizes out of band spurious signals over the analog bandwidth of the chosen ADC, which would otherwise alias into the sampled band due to harmonic sampling.

Here, commercially available Surface Acoustic Wave (SAW) filters were used to benefit from their much sharper roll-off and higher stop-band attenuation. Consequently, the high power 296.5 MHz LO transmitted from the center to every module for heterodyning (needed by the existing receivers), was effectively attenuated by ~ 30 dB, which otherwise would have aliased into the sampled pass band. Further, the out of band signals within the analog bandwidth of the ADCs were also substantially attenuated.

This subsystem was also required to have a high dynamic range due to the designed absence of an Automatic Gain Control (AGC) system, in order to maintain linearity in heavy RFI situations, as well as with the strongest sources like the sun.

Thus, the RF signal conditioning subsystem was composed of two stages. In stage-1, the first stage of amplification (pre-amplifier) of the combined RF received from the RF frontend is carried out to counter cable losses during transmission of the RF from the line feed to the reflector base. In addition, a coupled port is provided to interface the RF frontend output to the next system in the existing analog chain, the IF conversion subsystem. These modules are located along the telescope line feed, and in close proximity to the RF Frontend output. In stage-2, the incoming RF is filtered to ~ 18 MHz via a two stage SAW filter combination, and further post-amplification is carried out to condition the gain for the ADC requirements. The modules making up this stage are located below the reflector, at the antenna base. The output gains are equalized to the extent possible, with the actual gain control carried out digitally.

The RF bands before and after the analog conditioning are shown in Figure 3.5. Here, the suppression of several out of band signals, in addition to the LO signal, is clearly seen.

A schematic of the RF signal conditioning subsystem configuration for the 40-element array is depicted in Figure 3.6. Here, following the stage-1, each half module from a group of 10 (240 dipoles) is brought to a common location (the "pillar") over an equal length LMR195 cable, for RF digitization. This set of 10 signals is then referred to as a "signal cluster". The stage-2 modules are located within the pillar, and also provide the bias for each of the stage-1 amplifier modules.

Initial prototypes of these modules were developed in-house at RRI by colleagues in the Radio Astronomy Laboratory (RAL), with subsequent mass production in col-

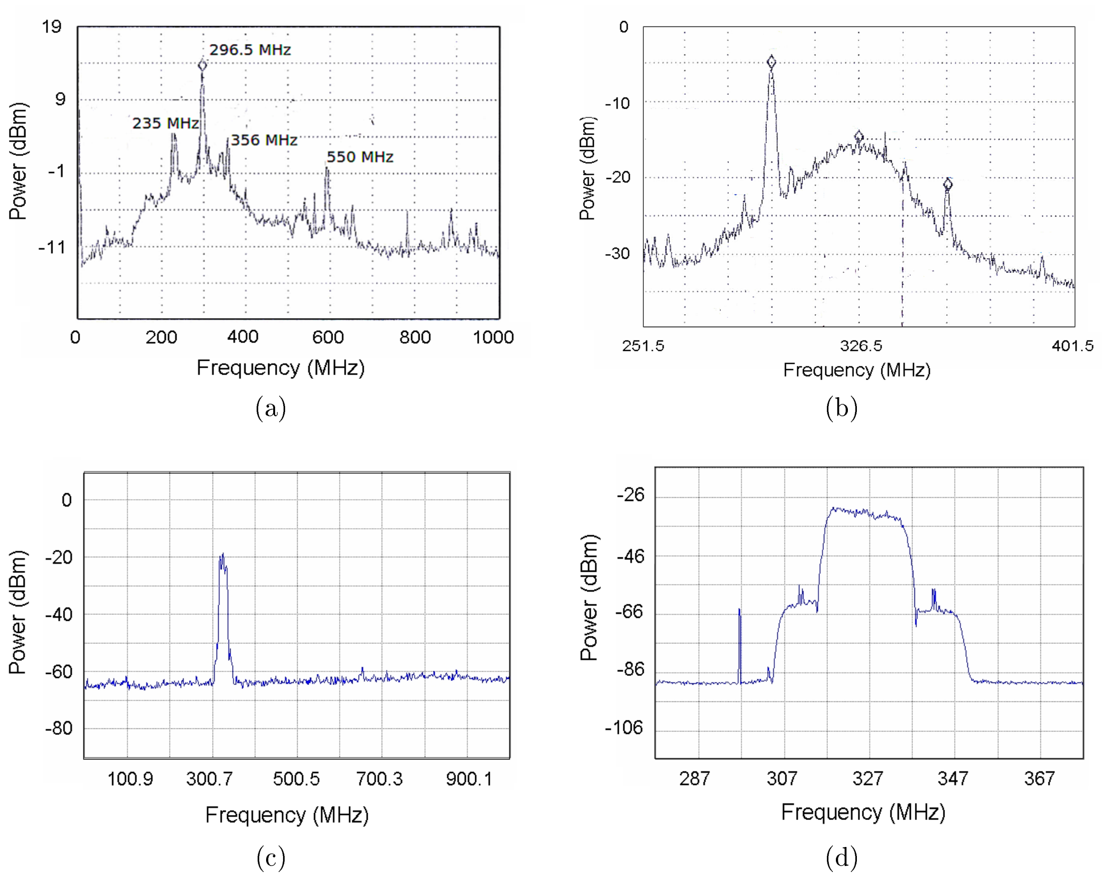


Figure 3.5: *Comparison of the RF band, before (a, b) and after (c, d) signal conditioning. (c) show the attenuation of out of band components within the ADC analog bandwidth, while (d) shows the leaking LO signal being attenuated, in addition to band-shaping for sampling.*

laboration with a local industry. The modules were installed by the observatory staff, and their effect on the existing analog system was first minimized, before continuing with the mass production and deployment.

The incoming RF from the antenna is directly sampled as close to the reception point as possible, by the ADCs in the field. In order to carry out such a direct RF sampling, ADCs with large analog bandwidths are required. Further, the ORT RF frequency is $\sim$ 327 MHz, which falls within the specifications of a large number of ADC parts, with the one chosen having an analog bandwidth of $\sim$ 600 MHz. The required signal transfer from remote, spatially distributed data sources, to a centralized location is carried out by data transmission over high speed, serial, packet-switched networks. These are implemented using optical fibres, which provide the advantage of low cost and efficiency of digital networks for long haul data transport. All further processing is carried out in software using general purpose processors,

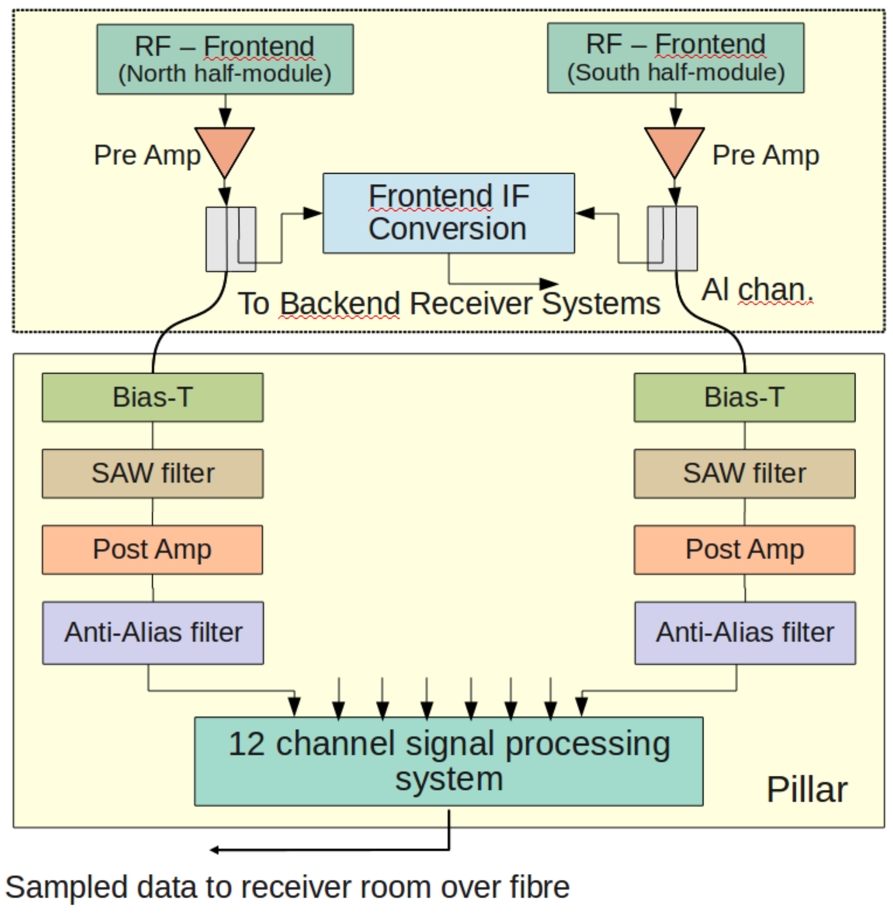

RF – Frontend (North half-module)
RF – Frontend (South half-module)
Pre Amp
Pre Amp
Frontend IF Conversion
To Backend Receiver Systems
Al chan.
Bias-T
Bias-T
SAW filter
SAW filter
Post Amp
Post Amp
Anti-Alias filter
Anti-Alias filter
12 channel signal processing system
Pillar
Sampled data to receiver room over fibre

Figure 3.6: *Schematic showing the major components of the analog subsystem for conditioning the tapped RF from 24 combined dipoles.*

which is feasible for such medium sized arrays using the current state-of-the-art. Such an approach allows for a considerable reduction in complexity of the instrument, while enhancing its reconfigurability. This reduction in complexity is seen in Figure 3.7. Figure 3.8 provides a broad level schematic of the entire system.

The overall signal flow from the telescope is shown schematically in Figure 3.9. Here, data generated by spatially distributed sensors (represented by the ADCs) is shown to be dispersed and rearranged in a distributed manner, over the different levels of the NSPS custom hardware hierarchy. This is in contrast with a conventional correlator flow, which typically uses a centralized switching (Figure 1.1) for carrying out the data dispersion. Following the dispersion, packets containing data from all sensors are routed to individual cores of a commodity processor, where the actual estimation of the spatio-temporal cross correlation function takes place. This is represented by the FX block in Figure 3.9, and constitutes the commodity processing

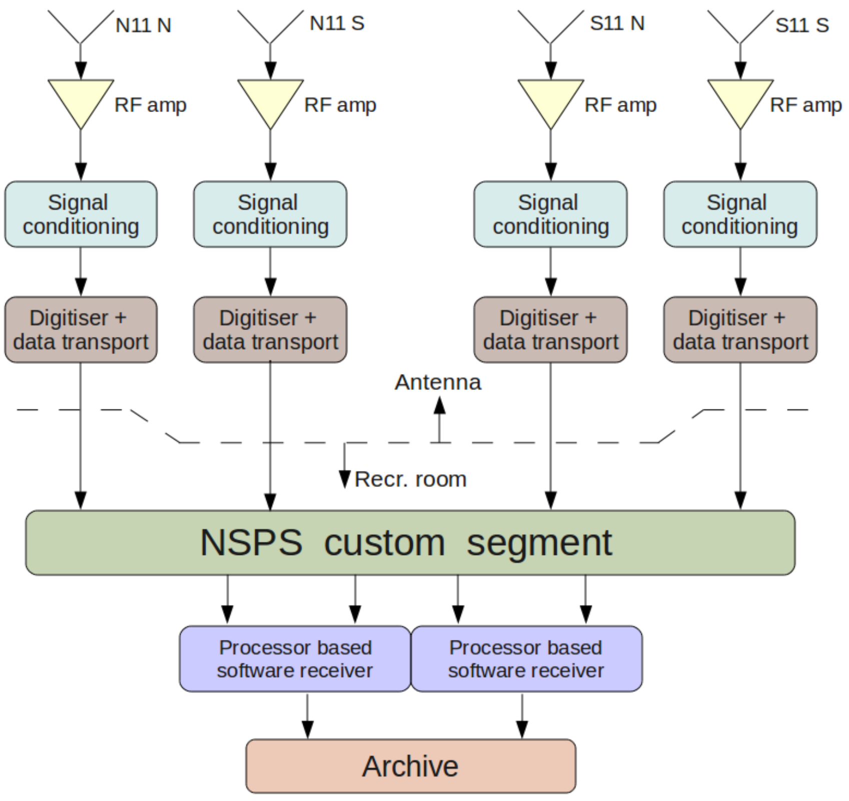


Figure 3.7: *Schematic depicting the main components of the NSPS based receiver for ORT, viz., the analog signal conditioning, the remote digitizer blocks, the NSPS custom segment, and the NSPS commodity segment.*

section of the hybrid correlator.

3.3.2 NSPS based hybrid array receiver

The communication and computing requirements of the 40-element array interferometric receiver naturally devolved into four levels of the NSPS hierarchy, with functionalities as elaborated below:

1. *ADC Modules (Remote digitizers):* These constitute the distributed sources of data, and are themselves composed of the digitizers. They carry out the first level of data pooling, and correspond to Level-3 in the NSPS hierarchy.

2. *Central Data Pooler:* This component constitutes the Level-2 of the NSPS, and implements its Data Pooler entity, which pools the data from various allotted upstream Level-3 entities.

3. *Bridge:* This component consists of another stage of data buffering, and implements the bridge between the customized segment of the NSPS and the

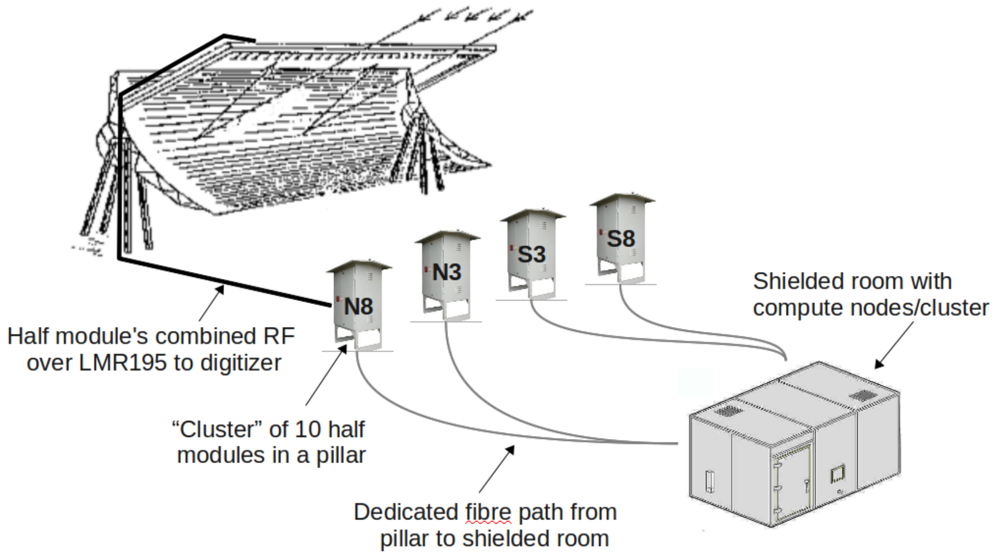

N8
N3
S3
S8
Shielded room with compute nodes/cluster
Half module's combined RF over LMR195 to digitizer
"Cluster" of 10 half modules in a pillar
Dedicated fibre path from pillar to shielded room

Figure 3.8: *Broad level schematic view of the ORT 40 element interferometric array, and its receiver. Each of the shown pillars handles 10 half module antenna elements, performing digitization, packetization, data transposition and timestamping of the data, before transmitting them over an optical fibre network to a central processor.*

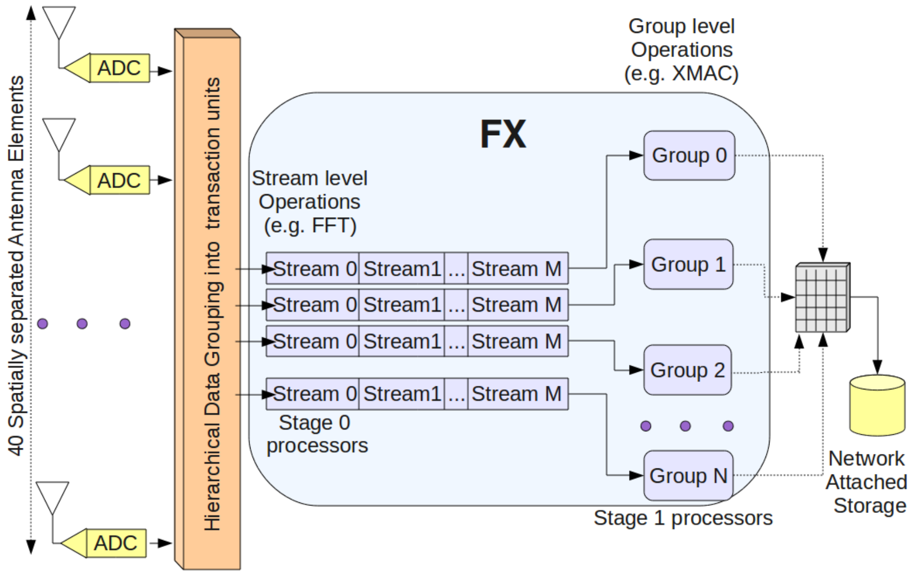

40 Spatially separated Antenna Elements
ADC
ADC
ADC
Hierarchical Data Grouping into transaction units
FX
Stream level Operations (e.g. FFT)
Stream 0
Stream1
...
Stream M
Stage 0 processors
Group level Operations (e.g. XMAC)
Group 0
Group 1
Group 2
Group N
Stage 1 processors
Network Attached Storage

Figure 3.9: *Data flow in the NSPS based hybrid correlator implementation shown in the NSPS custom segment (where data is grouped hierarchically into transaction units) and in the NSPS commodity segment (where transposed data is routed to individual processors for carrying out the FX correlation).*

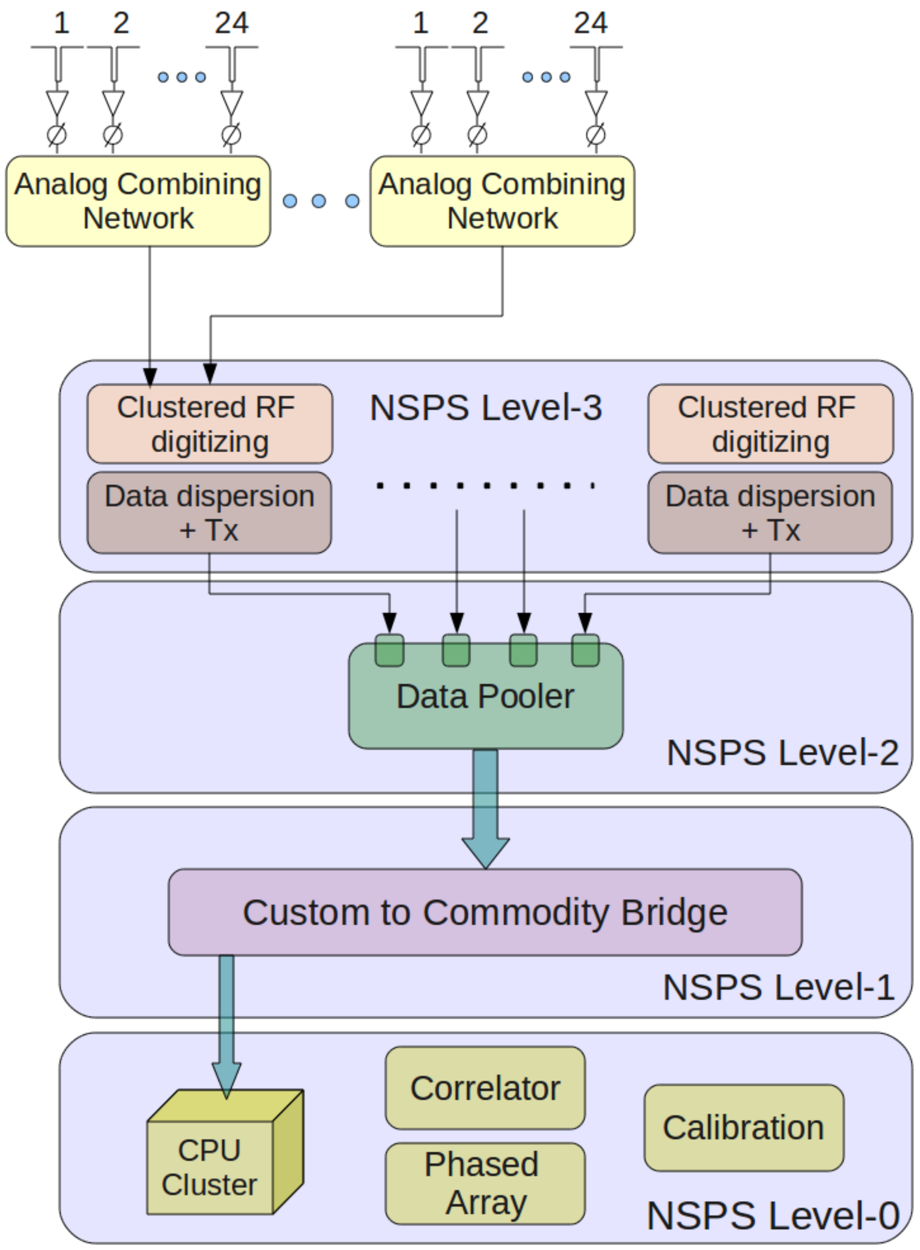


Figure 3.10: *Schematic view of the components of the implemented ORT 40-element receiver system, mapped onto an NSPS architecture based implementation.*

commodity segment. It constitutes the Level-1 of the NSPS.

4. *HPC cluster:* This component constitutes the root, or Level-0 of the NSPS hierarchy, and implements the central compute engine. It is made up of a cluster of commodity class processors.

A schematic view of the receiver, with each of these NSPS hierarchical levels mapped to functional blocks, is depicted in Figure 3.10. It may be noted that the coherent sampling of distributed RF signals required by the correlator is carried out by transmitting a high stability reference via the fibre based *clock network* of the NSPS.

The "Data Pooler" and the "Data Router" entities of the NSPS architecture are implemented on a custom FPGA based board, using the onchip BlockRAM resources of the FPGA. The pooler allows the correlator to operate in convenient transaction units, where data continuity is guaranteed. The correlator forms one of the multiple possible backends for the *fusion tree* of the NSPS.

For setting delay compensation values at the digitizer boards, the *control and monitor network* of the NSPS is utilized. This allows the generated data streams to be aligned to sampling clock precision (in time) at NSPS Level-3. This is crucial, since carrying out such an alignment in the lower levels of the NSPS (or in software) is extremely inefficient. Further, the *custom to commodity bridge* of the *data network* of the NSPS is implemented to provide data over standard GigE ports and UNIX sockets, resulting in easy interfacing with the Level-0 nodes.

Load balancing by the NSPS data pooler

Large I/O rates are generated by the distributed samplers, due to a combination of high sampling rates, large number of sensor elements, and higher number of data bits per sample. For instance, for the ORT upgrade discussed in this thesis, each of the 40 antenna elements generate $\sim$ 20 MBps, when sampled with a 4-bit resolution, at $\sim$ 40 MS/s. This leads to a total data rate of $\sim$ 800 MBps from the system, leading to a distributed approach to handle these rates.

In particular, since the correlation process requires the local availability of data from all sensor elements on-demand, the routing of data from every sensor increases the data bandwidth to a level which cannot be handled by a single commodity processing element. Thus, to simplify load balancing in a distributed computing environment, we utilize the process of "data pooling". This reorganizes data into "Transaction

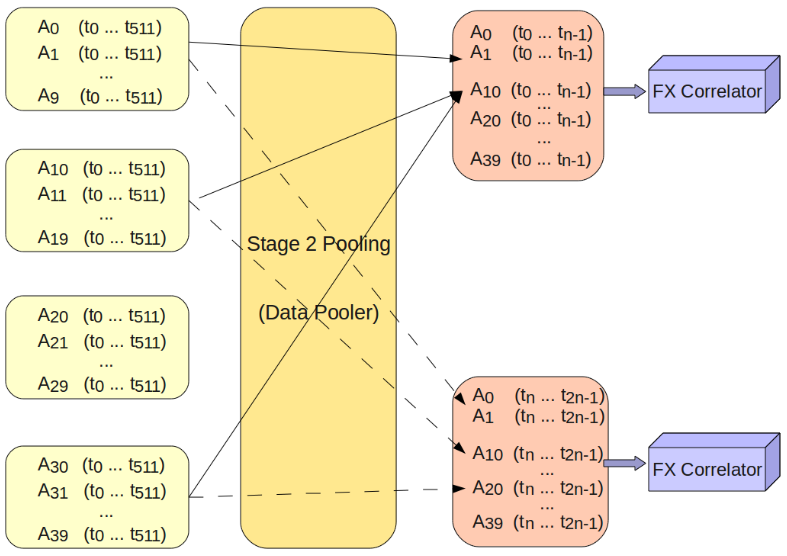


Figure 3.11: *Schematic view of the time domain data transposition carried out by the Data Pooler Entity of the NSPS. Here, $A_i\,(t_p...t_q)$ refers to $(p-q)$ consecutive samples from antenna element A_i.*

Units", and routes consecutive such units selectively to multiple processors at the lower NSPS levels, thus carrying out a shaping of the incoming traffic.

Such a regrouping of the data can be carried out using onboard memory buffers, with the data being split along either the time or frequency axis (if an FFT is being carried out within the NSPS hardware), before being dispatched. It may be noted that since no replication of data takes place at this level, the incoming and outgoing data rates remain the same. On the other hand, each parcel of data sent to a processor is self contained for correlation, i.e., it contains the same timeslice of data from every sensor element. Such a re-grouping is depicted schematically in Figure 3.11 for the 40-element array.

Another aspect which determines the load balancing in the NSPS fusion tree is to control parcelling of data to processors, based on their processing bandwidth. Thus, each processing element is fed data only at a rate which it can comfortably process. This is efficiently done by the Data Pooler due to its selective data routing. The Data Pooler, thus, handles both the large I/O rates, and the load balancing of processing on this data.

Transaction oriented processing model

In a conventional data network, the unit of communication is a data packet, which is an ordered set of data grouped along with metadata into a unit convenient for transfer over the chosen network technology.

We have extended this concept to make the basic unit of computation a "transaction unit", which deals with a timeslice of data convenient from the astronomical signal processing perspective. For instance, a transaction unit may comprise of data acquired over a timeslice of a few milliseconds. In the context of correlation, a transaction unit precludes the requirement of high bandwidth communication between the processing and data collecting elements in the distributed system, while large scale data transposition can be carried out using memory buffers. Such an approach also allows us to introduce a deliberate pipeline delay in the incoming data stream. This, is turn, facilitates synchronization between data from different streams, in addition to allowing multiple passes to be carried out on the incoming streaming data, in real time. The generation of a transaction unit requires the collation, and *holding* of data which, in turn, needs large memories at the highest levels in the data hierarchy. This is carried out in the memory of the Data Pooler entities of the NSPS.

3.4 System architecture

In this section, we describe the system architecture of the 40-element ORT array receiver.

The system carries out direct digitization of the incoming RF, with the data then being sent over a tree-like *packet switched* network. Here, all dataflows use a store and forward, packetizing interface, with no streaming flows. Thus, the entire routing and processing following packetization is carried out at the packet level only. The overall system is distributed in nature, and thus requires management of the hierarchy.

3.4.1 Packet configuration

Packets are atomic, and capable of independent existence, in order to distribute the system state over the NSPS tree, instead of centralizing it. This also allows the same software infrastructure to tap packets from different levels. To implement this approach, each packet of data preserves relevant meta-data about a subsystem from where it is generated in a per-packet header. This metadata can include information like the type of data, the source id of the data generator, the number of data streams making up the packet, the bits per sample, etc. which in turn, allows processing routines to query the meta data and proceed further, without relying on an external database of subsystem information.

It may be noted that the sizes of packets, which can be different for various types, are also encoded within the packet header. All packets begin with a common 16-byte header, and are mandated to be multiples of 16 bytes in size. This restriction allows for efficient memory access on commodity vectored processors (by ensuring appropriate memory alignment). Further, sequence numbers from the different layers of the NSPS can also be stored, which allows for traceback and data grouping.

Packet flow configuration: All packets flowing within the signal processing tree can be segregated into the following types:

1. *Real-time, high bandwidth, streaming flow,* which is composed of raw or preprocessed data within the transport/processing tree.
2. *Processed data flow*, which can be real-time, (e.g., correlation output, beam formation) or quasi real-time (e.g., offline correlation followed by beam forming). This is treated as part of the data network.
3. *Low bandwidth status flow*, which is a part of network and system management.
4. *Low bandwidth calibration/monitoring flow*, which can contain small subsets of the full data stream for monitoring, or for dynamic estimation of calibration parameters.

These flows should be logically separable in the system, thereby allowing separate processes to individually tap onto them. To this end, packets belonging to different flows are identified by a unique “Datatype”, which is maintained across the signal processing tree, till a block level operation on the packet changes its type. For enabling debugging, the system design also allows for the collection of data streams from different levels in the NSPS hierarchy.

Addressing: Since the data flow is heavily asymmetric, the addressing from the data sources at the higher levels of the NSPS to the lower levels is controlled by static routing. However, each source of distributed data maintains its identification via a source id field, which helps in identifying them at the lower NSPS levels. The control packets, which need to be addressed to specific nodes within the NSPS, are transmitted in a broadcast fashion with the destination address. Intermediate nodes then examine these control packets, and broadcast them forward if the destination does not match their own address.

Packet structure: The packet structure, apart from fulfilling the above requirements of packet independence, should also be easy to query and implement in configurable hardware, without needing excessive protocol management. Hardware choice restricts raw data from the digitizers to appear in groups of 12 or less. For the 40-element array, each group contains 10 data streams. With a grouping of 512 samples per packet, a basic raw data packet is 2576 bytes (including the 16 byte hdr), assuming 4-bit quantization per sample. These packets are segmented on reaching the last mile GigE interface, in order to comply with the protocol's 1500 byte Maximum Transaction Unit (MTU).

The raw data packet consists of a 16-byte header, followed by actual data, which can be a timeseries, or a slice of a spectrum. The header includes space for meta information from intermediate NSPS levels, which is initially empty, and is filled up as the packet traverses the hierarchy. Such an approach gives a common packet size across the hierarchy, yet allows different layers to record relevant information. If extra information needs to be added, an extension header can be created. In our system, the main packet types relate to a per-sensor raw stream, and correlated data streams, whose structures are described below. Apart from these, details of other kinds of packets, as elaborated in the packet flow configuration, are listed in Appendix A. A data packet can store different kinds of data, differentiated by the data type field of the structure.

For raw data flows, this structure is given in Listing 3.1:

```
typedef struct
{ unsigned int seq :12;     // Sequence of this pkt at current level
  unsigned int id  : 4;     // The id of the NSPS node at this level
} __attribute__((packed)) LayerHdrType;

typedef struct
{ unsigned int srcid   :4; // The unique source id of the data source
  unsigned int datatype:4; // The kind of data carried by this packet
  unsigned int bits2pix:4; // The sample quantization or datatype
  unsigned int chans   :4; // Number of datastreams within the packet
  unsigned short words;     // The size of this packet in 8-byte units
  unsigned int tick;        // The timestamp of this packet
  LayerHdrType lay[3];
  unsigned char grpid;
  unsigned char unused;
} __attribute__((aligned (16), packed)) DataHdrType;
```

```
typedef struct
{ DataHdrType hdr;
  unsigned char data[0];
} DataPktType;
```

Listing 3.1: Memory layout of a raw data packet traversing the NSPS hierarchy.

Correlated data from all baselines, for a particular time instant flows across the NSPS network in packets with a structure as shown below:

```
typedef struct
{ unsigned char srcid;
  unsigned int datatype: 4;
  unsigned int bits2pix: 4;
  unsigned int chans    : 4; // Interpreted as number of blines in pkt
  unsigned int words    :12;
  unsigned int tick;
  unsigned short taps;
  unsigned short blines;
  float f_samp;              // Sampling rate
} CorrHdrType;

typedef struct               // Represents cross spectrum of 1 bline
{ char a0;                   // The first sensor element(-1=>flagged)
  char astar1;               // The second sensor element
} PackBlineHdrType;

// In this layout, all baseline data is clumped together, while
// baseline information is at the packet beginning.
typedef struct
{ CorrHdrType hdr;
  PackBlineHdrType *bline;
  _Complex float cspect[0]; // The CPS follows this
} PackCorrPktType;
```

Listing 3.2: Memory layout of a correlated data packet traversing the NSPS hierarchy.

3.4.2 Design control flow

For any given observation, a configuration file specifies observational parameters, e.g., field location, observation mode (transit/tracking) etc. System parameters like

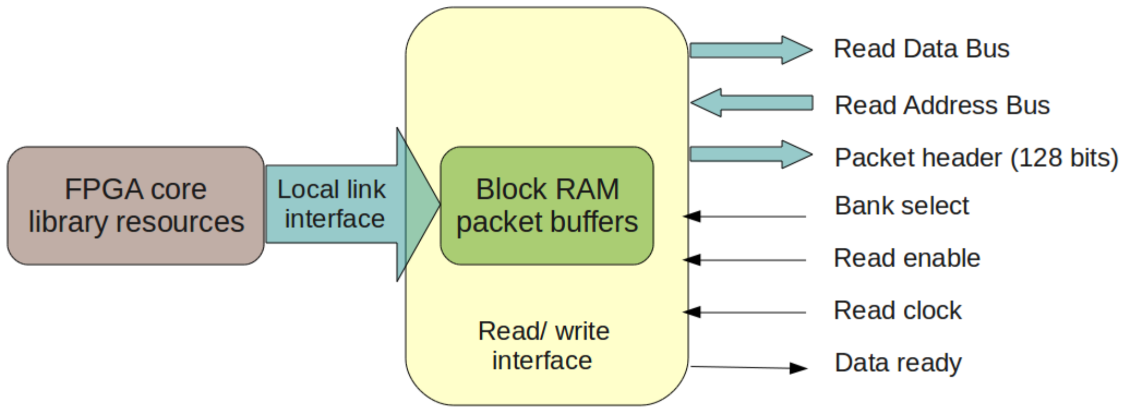


Figure 3.12: *Schematic of the packet level Reader-Writer interface used in the custom hardware segment of the NSPS.*

the sampling clock, the NSPS tree connection hierarchy (including the commodity cluster mapping to the NSPS bridge entities), the sensor to ADC channel mapping, etc. are recorded in another system configuration file, which only a privileged user can modify. Further, each NSPS entity (like Data Pooler, Bridge etc.) maintains its internal status, e.g., its capabilities, current firmware code version, etc., within its local memory, which can be queried from the control processor.

System control is implemented by the root NSPS node via control packets, which can be addressed to different levels, or to individual entities within levels. Further, system status can be determined by sending queries to each NSPS node, which generate status packets in response.

Control involves setting parameters like the Look Up Table (LUT) entries and the delay settings in the hardware, or generating the timestamp reset signal, the start signal, the stop signal etc., from the root node to each remote data source. Since the root level cluster sees upto 8 GigE links, we designate one link to be the master controller having the authority to issue command packets. These can then be forwarded by intermediate entities till the destination is reached.

3.4.3 Reader-Writer interface

The reader-writer process at the interface of each NSPS custom hardware entity carries out packet store-and-forwarding to the next level. In case the write rate is faster than the read (which can be possible in a configurable system), the design ensures that data is lost in units of integral packets only, thereby maintaining timing synchronization between the distributed streams.

Each interface has an associated simple dual ported buffer composed of two banks, with one bank for writing, and the other for reading. Further, the reader and writer clocks are assumed to be different in rate and phase, and unrelated to each other. The initial read and write addresses are generated by an external control logic, based on the current status of the reading process, and on packet boundaries. The writer process generates a signal (data_rdy) for the control logic (a few clocks before it completes writing one packet/buffer) which, in turn, examines the current read address. In case of a busy reader, the currently written bank is overwritten. Otherwise, the reader is invoked to empty the ready bank. At the next signal, the reader will automatically see the latest data, while the writer will go to the next bank, after any number of cycles on the current bank. This scheme is depicted in Figure 3.12, and is used extensively in each NSPS custom hardware entity.

3.5 Digital subsystem

This section discusses an implementation of the NSPS architecture with four hierarchical levels, as described in Section 3.3.2. This implementation was used to preprocess, route and correlate samples of the appropriately conditioned RF data, thus allowing for various modes of the receiver to be implemented. Some of the primary requirements of the digital subsystem (implemented in the custom hardware segment of the NSPS) for the array are:

1. Appropriate sampling of the conditioned RF.
2. Preprocessing, grouping and routing of streaming data.
3. Appropriate and synchronized timestamping of data, on the basis of which combining of data from the different data sources can be carried out.
4. A flexible data transport mechanism for aggregating spatially distributed data to a central repository or correlator.

3.5.1 Remote digitizer and pooler entities

The remote digitizer module, shown schematically in Figure 3.13, carries out harmonic sampling of the incoming RF at 327 MHz from a group of 10 antennas, followed by a first level of pooling on the sampled data. This subsystem corresponds to the Level-3 in the NSPS hierarchy. It consists of an analog and a digital sub-module, implemented on different PCB boards and interconnected using a high-speed connector.

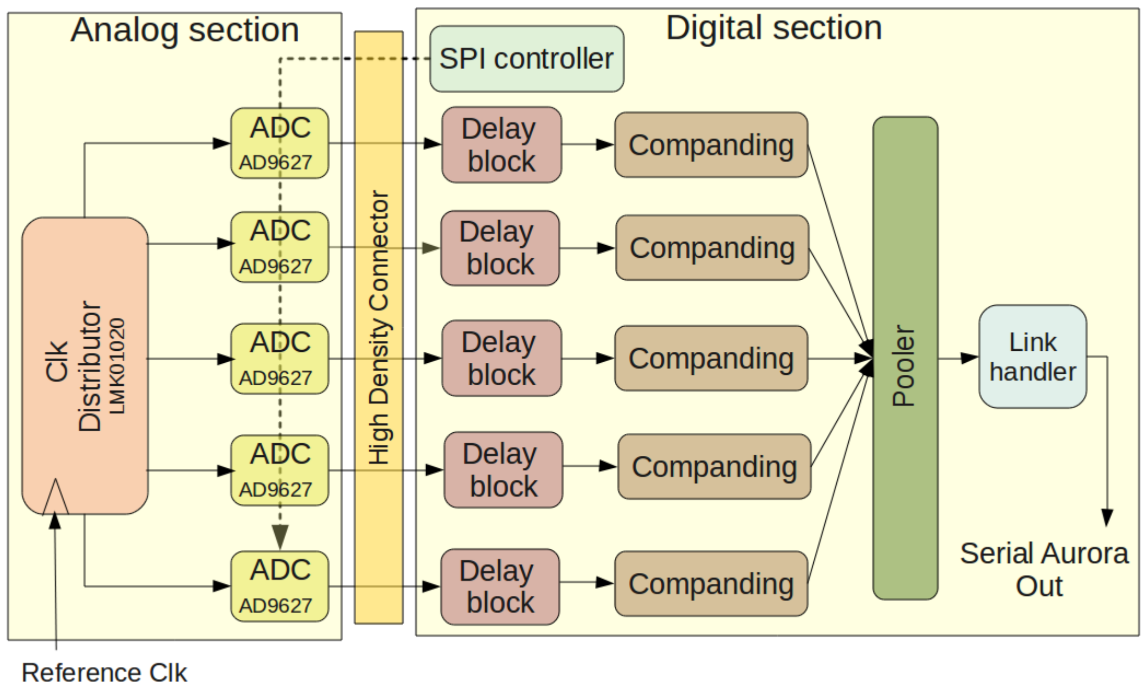


Figure 3.13: *Digitizer section and its main components: clock distributor, ADCs, delay block, companding block, first level data pooler and the serial link handler.*

The analog section (with an isolated power supply to minimize leakage from the digital section) consists of a set of 6 AD9627, dual-channel, 110 MS/s, 10-bit ADCs which are capable of sampling RF of upto 600 MHz. The choice of sampling clock results in the RF band falling in the 8th harmonic. Its generation and distribution are described later in Section 3.5.6. The particular choice of ADC accepts a sinusoidal clock input, and is equipped with an on-chip sampling clock conditioner, clock divider and duty cycle stabilizer. This makes it possible to use the on-chip features to convert a sine wave input clock to a square-wave, enable duty-cycle stabilizer, and divide by suitable integer to generate the sampling clock from the input. Thus, an input clock, whose frequency is 2-8 times the actual sampling rate needed, can be used. This puts the clock and its harmonics out of the received 327 MHz band.

The ADC boards are coupled over high speed LVDS board-to-board links to the pooler and communication handler, which reorganizes, frames and timestamps the data at the remote source itself. It also contains an implementation of the companding LUTs, which convert the incoming 8-bit samples to 4-bits/sample, leading to an output data rate of ∼20 MBps per ADC channel, or ∼200 MBps per ADC module. This data is sent over a single high speed (2.5 Gbps/250 MBps) serial Aurora link from each ADC module, to a central Pooler card, over an optical fibre link.

Adequate shielding of the digital section has been provided by packing the submodules in milled boxes, while the ADCs are shielded with soldered cages, in order to minimize pickup by the sensitive ORT antenna.

Level 3 pooler:

The “Transaction Unit” at this level corresponds to a fixed number (usually 512) of samples from an ADC pair. Thus, it can be created using the internal BlockRAM based memory of the FPGA itself. The buffering scheme consists of a double buffer per ADC (channel pair), whose reader can implement various data layouts by appropriately accessing the channel buffers, in order to allow optimal access from processors downstream of this level. The formatted buffer outputs are then collated into a single packet for transmission. Apart from the ADC input, all other logic at this level runs at a 125 MHz core clock, with a 64 bit bus. Thus, the maximum data rate internal to the FPGA is 1 GBps.

3.5.2 Central data pooler

At the central location, a single Virtex5 board is used to implement a second level of data pooling on the outputs of each pillar via the central Data Pooler entity of the NSPS architecture. This forms the Level-2 of the NSPS hierarchy. Here, the card is equipped with eight high speed serial links, which can be used to implement both the custom and the commodity network data link layers. As a Data Pooler entity, all eight links are used as part of the custom network infrastructure. The necessary data transposition is carried out by placing incoming packets in locations within a local, high speed buffer implemented using the FPGA Block RAMs, based on their 32-bit timestamps. This approach caters to missing packets from the same timeslice on links from individual pillars, which can otherwise lead to loss of synchronization. Thus, the Level-2 data pooler carries out a temporal traffic shaping within the NSPS custom hardware domain.

The onboard buffer also allows for the creation of a reasonable transaction unit, on which other processing passes can be carried out as a block, (if necessary). This approach also provides a guarantee of mostly contiguous data over the entire transaction unit which, in turn, allows us to implement a burst mode, with a higher net incoming data rate than that of the outgoing links. Further, this feature can be used in observing modes which require continuous data sampled at very high temporal resolutions. It may be noted that the current pooler implementation limits the trans-

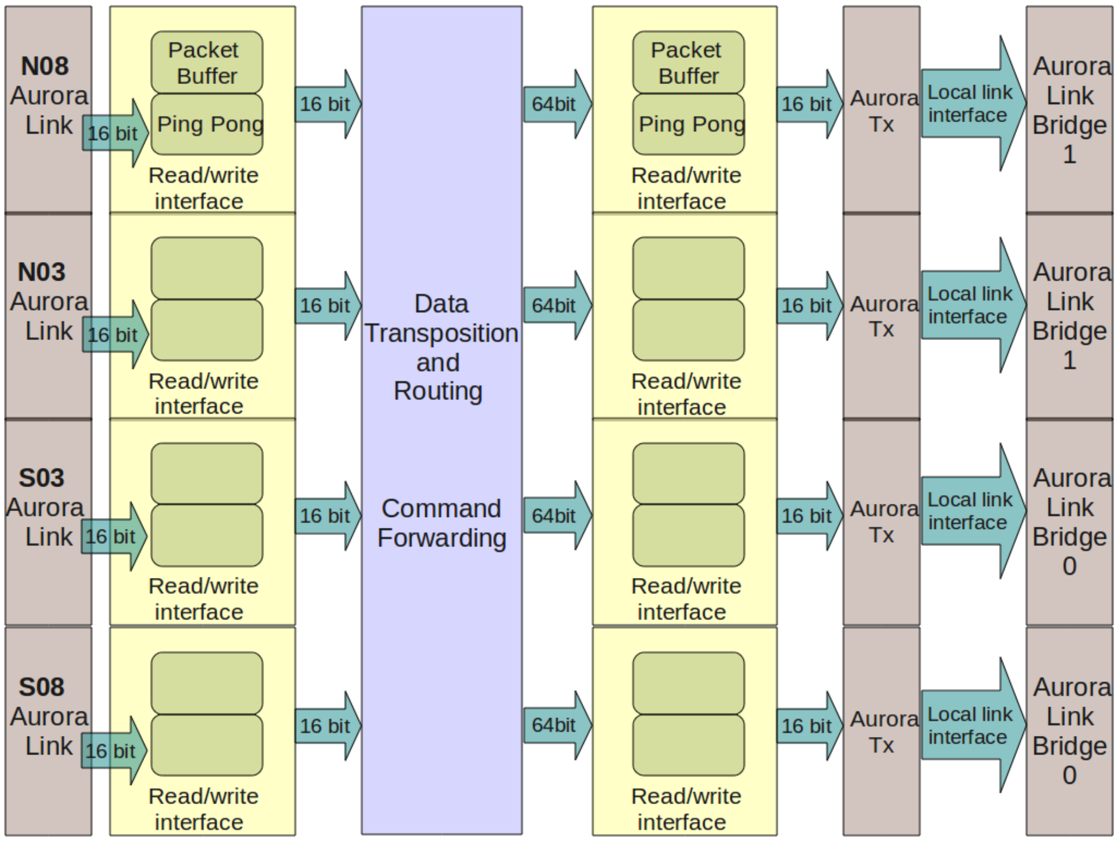


Figure 3.14: *Block diagram of the Layer 2 data pooler card showing the various levels of implemented packet buffering, as well as the data transposition block.*

action unit to a few tens of multiples of the Short Term Accumulation (STA) cycle, within the constraints of the dynamic range of the integer logic used for the correlator implementation. A newer version of the pooler will incorporate DDR memory to increase the transaction size substantially.

Firmware code layout in the data pooler node

While the incoming data is buffered at the packet level in FPGA BlockRAM, the transaction unit is created using a large, per output link, BlockRAM based memory buffer. The layout of the data written into this buffer can be tuned to match the Level-0 node's requirement (for most efficient reads and computing) by an appropriate reading order from the packet buffers. Here, a single entity, i.e., central data transposition and routing entity, present between the packet and pooling buffers (shown in of Figure 3.14) deals with these reads. This gives the flexibility to generate different data layouts, due to the well defined interface to buffers on both input and

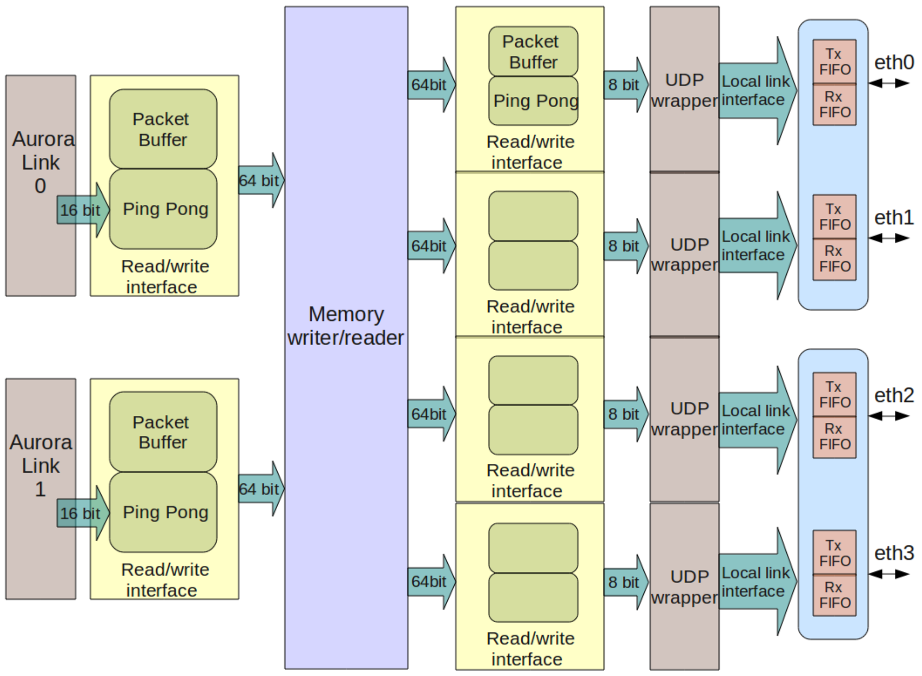


Figure 3.15: *Block diagram showing the connectivity within the Peer-to-Peer to commodity bridge card.*

output sides. The block level schematic of the Level-2 Data Pooler is illustrated in Figure 3.14.

3.5.3 Aurora to GigE bridge

The NSPS Level-1 nodes act as bridges between the custom and the commodity network of the NSPS architecture. Thus, the bridge manages not only the protocol translation, latency differences between the more latency prone commodity network and the rigidly synchronous custom network, but also any bandwidth differences between the two domains.

The hardware at this level is identical to that at Level-2, except that out of the eight available high speed serial links per card, four are configured as part of the commodity network, while two are configured as part of the custom network. Here, for the 40-element system with currently supported bandwidths, two such cards are sufficient to manage the full incoming data rate, with each card handling $\sim$ 400 MBps on two Aurora links. The data received on each link (the collated output packets of the Data Pooler card) is buffered, and routed in a time multiplexed manner to the commodity processors, over two commodity (GigE) links. Thus, the four available GigE links can serve the full incoming bandwidth. In case the system configuration

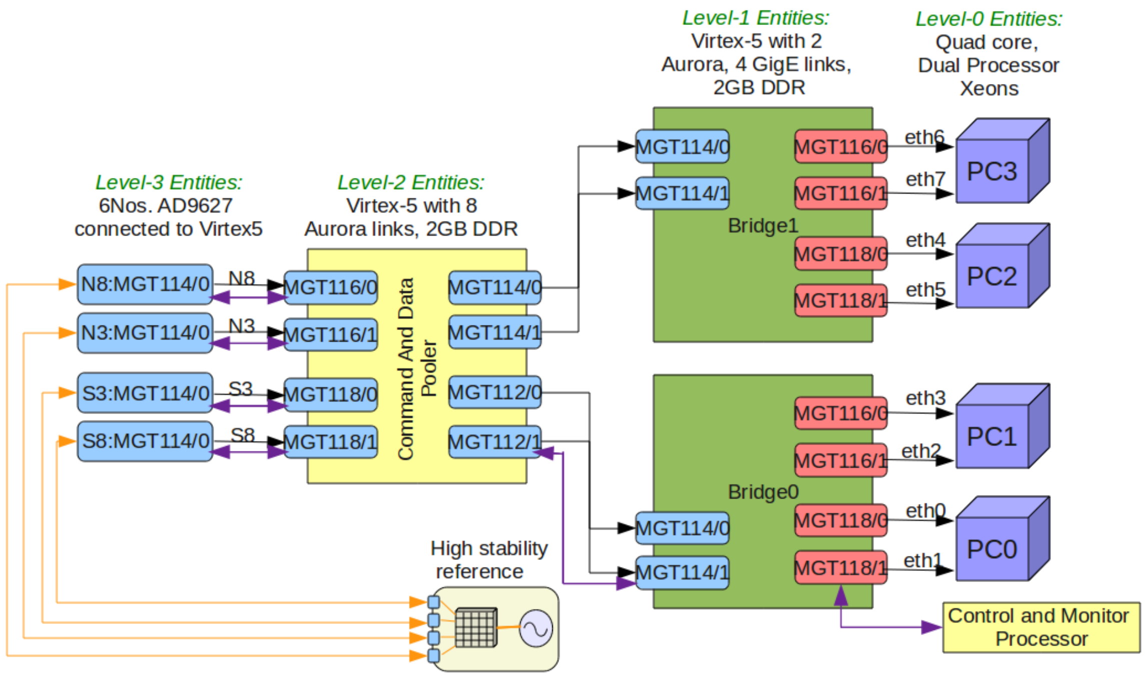


Figure 3.16: *Block level connection scheme of the NSPS based receiver, depicting the mapping of the serial link infrastructure onto each NSPS component.*

results in a higher overall data rate, more bridge cards can be added to the NSPS, such that the output bandwidth to a commodity processor matches its processing bandwidth.

The overall layout of logic is depicted in Figure 3.15, while the block level connection scheme of the system is depicted in Figure 3.16.

3.5.4 Commodity class processor cluster

At the root of the NSPS signal processing tree is a cluster of commodity server class machines. These form the Level-0 node of the NSPS implementation. For the 40-element receiver, the root node is currently a small cluster of four server class machines with dual Intel Xeon class processors. This cluster represents the commodity segment of the NSPS, with nodes being interconnected by a commodity networking stack. Data from the remote digitizers, after traversing the NSPS signal processing tree, terminate into this Level-0 node. All processing on the streaming data (to implement the various modes of the receiver) is carried out on the commodity, multi-core processors of the Level-0 cluster. A description of this processing is the subject of Chapter 4. The cluster is also equipped with a large data archive, capable of storing streaming raw data for several hours of observations.

3.5.5 Data transport mechanism

In this section, we describe the NSPS data transport network implementation. Transportation of raw or preprocessed data from its source to a compute engine requires careful consideration due to the distributed nature of the signal sources. Also, the large bandwidth of the streaming data requires a high bandwidth I/O stack. The streaming nature of data precludes any implementation for reliability between communicating peers, (e.g., via re-transmission and acknowledgements) as these require memory for buffering. Further, due to the guarantee of internal consistency of timing provided for every packet, data loss can only translate to a loss in sensitivity. Thus, we have chosen a high speed serial I/O mechanism to transport data both within the customized as well as commodity segments of the NSPS design. Such I/O implementations are currently routinely available as part of modern FPGA platforms, which are also equipped for dealing with commodity protocols like Gigabit Ethernet. This enables them to directly communicate with commodity PC hardware as well.

NSPS custom network

Based on the choice of FPGA part (Xilinx Virtex-5), we have chosen the Aurora serial protocol stack for board-to-board communication requirements within the NSPS custom domain. This high speed, light weight, point-to-point serial protocol is layered on top of the Xilinx RocketIO interface, which has native hardware support on the FPGA. The protocol operates as a packetized, connection oriented data stream with explicit setup and tear-down of connections. It incorporates an 8bit/10bit [43] data encoding scheme for reliable transfer of data. Further, a 16-bit packet level First In First Out (FIFO) interface is made available to the local side for transmission and reception of data. Handshaking signals are used both at the local side, as well as between remote entities, to ensure proper data transfer. The protocol does not support retransmission in case of error. We operate this protocol at a wire-speed of 2.5 Gbps within the peer-to-peer NSPS custom network.

NSPS commodity network

For communication between the customized segment and the commodity hardware forming the backend of our system architecture, we have chosen Gigabit Ethernet (GigE) as the Data Link Layer. This is a popular, point-to-point commodity protocol stack for high-speed serial transport of data over Unshielded Twisted Pair (UTP) cables, with a wide support base. Also, our chosen FPGA platform offers library

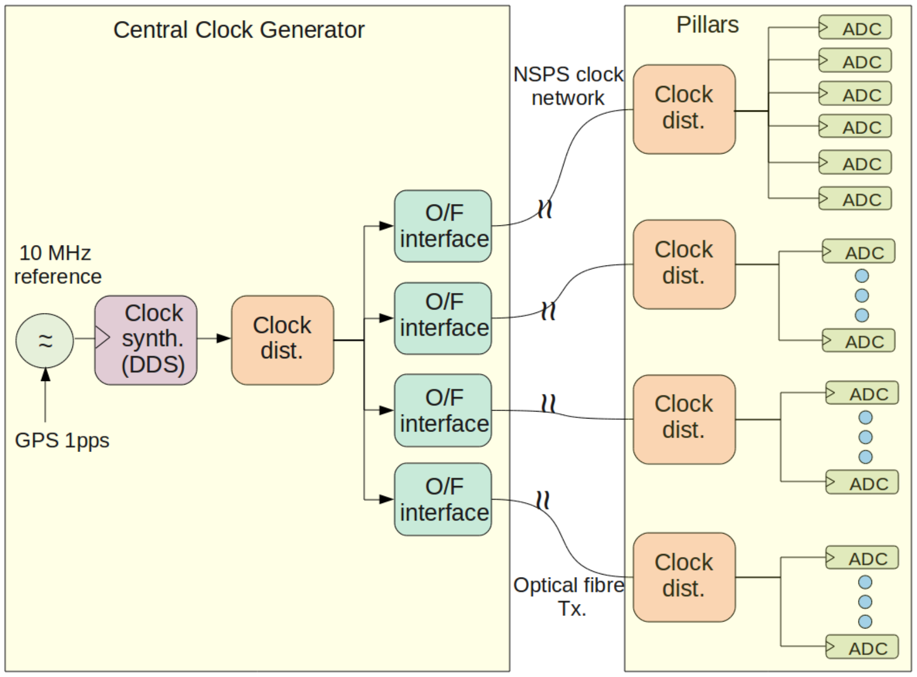


Figure 3.17: *Coherent distribution of a high stability sampling clock to the remote nodes of the NSPS implementation.*

support for GigE implementations in a variety of forms. The User Datagram Protocol (UDP) has been chosen as the transport layer protocol between the custom and the commodity network, due to its ease of implementation within the custom side bridge card FPGAs, in addition to its efficiency due to software buffering by the Operating System. This lightweight protocol also enables host side data stream discrimination within the commodity segment via UDP ports, to which different service threads of processing can be attached.

Further, packet sizes have been restricted to a few KiloBytes throughout the NSPS tree, with segmentation and reassembly carried out, if required. This restriction arises due to several factors, like the packet buffering within FPGA BlockRAMS, the custom segment Aurora links requiring to send “Clock compensation” symbols periodically to maintain synchronization with the remote peer, and the MTU of GigE being 1500 bytes.

3.5.6 Sampling clock generation and distribution

The sampling of the incoming RF from all elements in the array requires the availability of a high stability sampling clock, with its coherency maintained across the sampled elements, in order to combine the element outputs in a phase coherent manner. Here, the jitter specification on the sampling clock is related to the maximum frequency content of the incoming RF, even though the maximum sampling rate can be restricted to greater than twice the incoming available bandwidth.

Further, using the standard harmonic sampling formula [44], a sampling clock of 39.5 MHz was found to be adequate for sampling the $\sim$ 17 MHz anti-aliasing filter output (with a 0.5 MHz guard band on either side). This sinusoidal sampling clock was generated centrally using a Direct Digital Synthesizer (DDS), whose reference was a GPS disciplined Rubidium Oscillator available at the observatory. The reference has an inherent clock stability of $\sim 10^{-12}$. Once generated, four copies of this clock are made using a clock distributor board which utilizes a low relative jitter and skew clock distributor chip, (the LMK01020, 30 fs additional jitter, 30 ps skew), and are directly available for transmission over optical fibre modules to the four remote ADC modules housing the distributed digitizers. We use the NSPS *clock network* (Figure 3.17) for the sampling clock distribution to these digitizers. Here, this clock network is a digital, optical fibre based network, consisting of point-to-point links from the center to every pillar. Each ADC module then coherently distributes the received centrally generated clock to every digitizer within the digitizer group, using an on-board LMK device. Thus, the relative alignment of sampling clocks within the ADC module is assured to a small fraction of the RF wavelength. A provision for monitoring the phase of the distributed clock to each pillar has been made by enabling a round-trip calibration for the propagation over optical fibre.

3.5.7 Alignment of distributed data

In a real-time correlator implementation, data generated by coherently sampling the distributed sources needs to be aligned to the level of a few sampling clock ticks in order to preserve correlation. This is because the error in the correlation coefficient estimate between broad band signals is reduced, when such signals are aligned to a small fraction of their reciprocal bandwidths. Further, since the remote digitizers of the ORT receiver are organized into 4 digitizer groups, each containing a set of 10 digitizers as a single module, this alignment is required to be maintained within, and also across the ADC groups. Consequently, the data streams are aligned in two stages. The first stage of data alignment between an ADC module's 10 digitizers is carried out by setting delay registers associated with each digitizer to the appropriate delays between the data streams. These delays can be loaded by the central controller as part of system initialization.

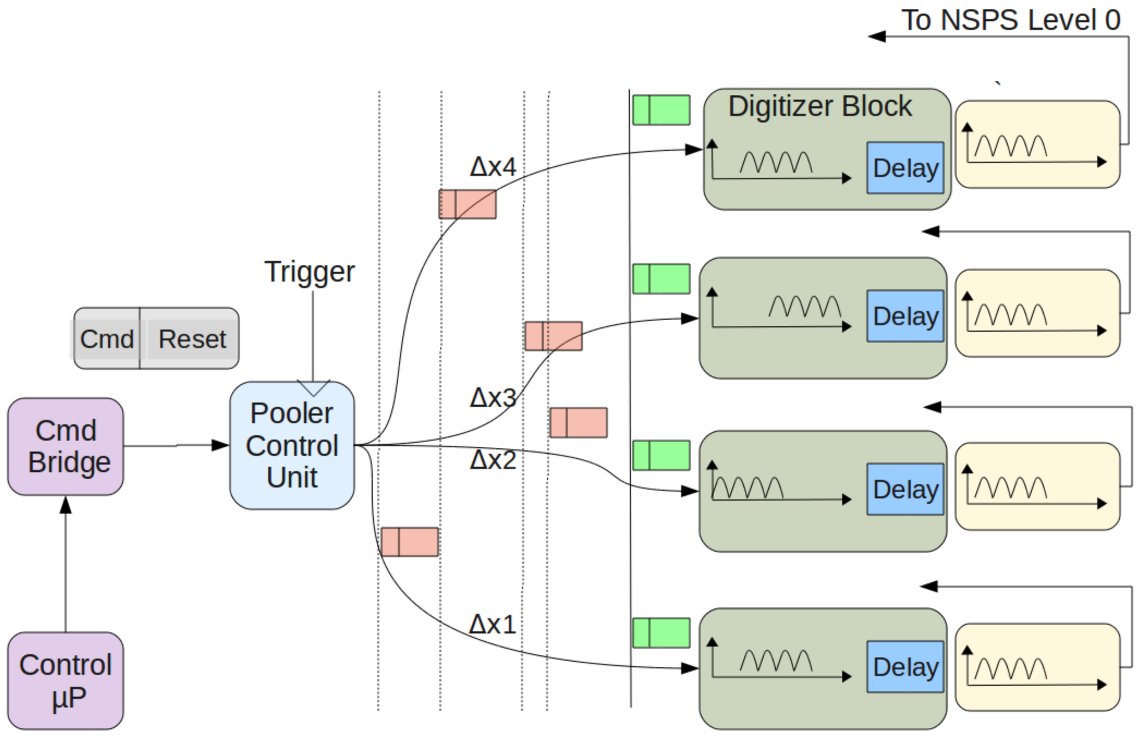


Figure 3.18: *Schematic showing the propagation of a reset control packet from a central command processor, in order to affect a time aligned reset of the remote entities.*

In order to carry out the second stage of data alignment data across signal clusters, a common reset is propagated from the center to each signal cluster, as depicted in Figure 3.18. This reset is initiated by the control processor in the form of a reset command packet to the Data Pooler card, which is buffered by it till an external trigger (e.g., a 1 pulse per second (1 pps) from a GPS receiver) is received. Then, the reset command packet is simultaneously transmitted by the Data Pooler to all the ADC modules, where its reception, and the release of the soft resets within the ADC module are expected to vary minimally, due to differential hardware or protocol latencies. These latencies remain constant with time, and are estimated using a common, explicit synchronizing signal broadcast throughout the system. Inter-comparison of the timing of data from different ADC modules can be carried out using the presence of an included 32-bit timestamp in the header of every packet generated from an ADC module. These timestamps are generated within the FPGA logic of the remote ADC modules via a 32-bit timestamp counter updated on the incoming sampling clock, and are propagated across different levels of processing and datatypes, till the final

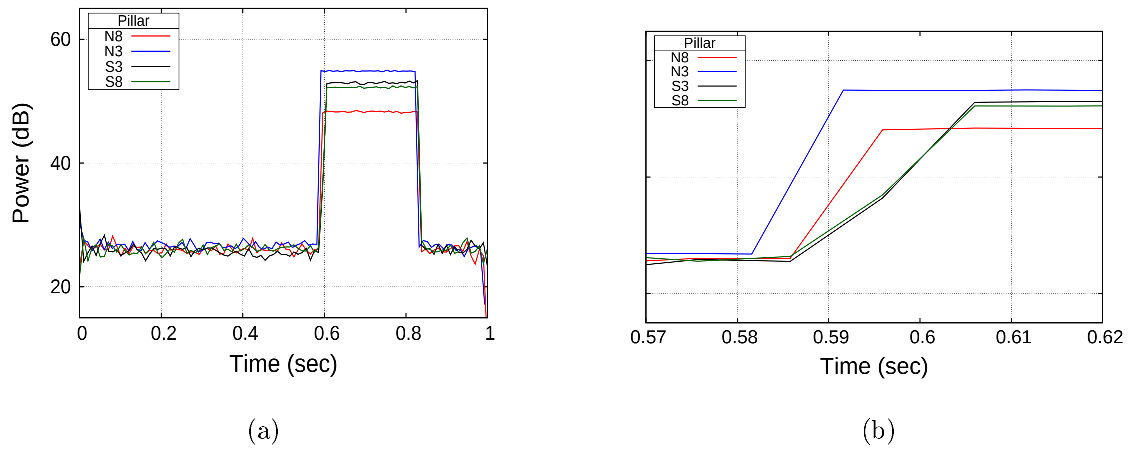


Figure 3.19: *Plot showing the scheme for estimating inter-pillar delays, using RF transmission of an FSK encoded 1pps to each pillar, on equal length cable. This signal is subsequently digitized and sent back for digital demodulation. The plots show total power within 1 sec, after carrying out digital demodulation. (b) is a zoomed version of the plot (a).*

archiving. Thus, the inter-ADC module latencies are compensated by initializing the timestamp counter of each ADC module to the measured latencies, in steps of sampling clock period. Further, the actual data transmission from the remote digitizer modules is activated with offsets of the inter-ADC module delays (again in sampling clock units), resulting in data packets with identical timestamps being close in absolute time, to the level of a few samples. It may be noted that the high stability of the sampling clock and its coherent distribution enforces strict relative timing between all the generated datastreams within the NSPS.

A higher (better than one sample) level of synchronization between ADC modules is implemented by injecting a common synchronizing signal from the central receiver to every signal cluster. This mechanism uses an existing, equal path length RF cable network used to distribute the local oscillator to the IF frontend (as described in Section 3.2.4). This scheme [45], essentially distributes a trigger to the remote digitizers, by modulating it as a phase-continuous, Frequency Shift Keying (FSK) between two synthesized tones. Here, the switching is aligned with an external signal (a GPS 1 pps in the current scheme), and the switching signal is generated using a DDS module. At every pillar, this signal is combined with one of the existing incoming analog signals, and both are sampled. The data is then made available to both the remote side FPGAs (to demodulate and initialize themselves, if required), as well as to the Level-0 processors in order to carry out a finer level of centralized synchronization.

Such an arrangement allows us to calibrate the path length differences to each ADC module by carrying out phase difference measurements on the synchronizing signal, which can be directly converted to delays. The estimation accuracy is limited by the SNR on the phase difference estimations, which, due to the periodic nature of the signal, can be improved by coherently folding the acquired synchronizing signal over several seconds. In addition, the phase stability of the digitized path can also be monitored continuously, and applied as part of continuous calibration. The latter can be done by monitoring the phase variation on the injected, high SNR FSK signal as a function of time. The injected tones have a separation of ∼300 KHz, which resolves microsecond level delays, while their frequency is bin-centered for our choice of sampling clock and FFT size, allowing for accurate phase comparisons.

As seen in Figure 3.19, a synchronization to within $10\,\mu s$ is possible by using just the power of one of the FSK tones, which results in a pulse-like transition at the 1 pps. The figure also shows the FSK switching, signalling the occurrence of a 1 pps pulse. A closer look at the transition shows relative delays of a few milliseconds between the different clusters of data sources. These are determined during system commissioning, and corrected during observations.

The traceability of absolute time in the system is maintained by recording the time when the system was reset. This can be added as a constant offset to the 32-bit timestamp available in the header of every data packet. Thus, the set of commodity processors which carry out the correlation depend on this timing, which is maintained throughout the system by the strict time-tagging carried out by the FPGA. Further, while integrating the processed data, an average time computed from the time-tags of the packets making up the integration time is used to represent the processed data packet. The collation of such processed data into chronological sets also uses the FPGA time-tag to reorder packets.

3.6 Reconfiguration plan for the ORT

In this section, we provide an example of an NSPS implementation by giving an outline of the system being planned for modernizing the ORT [29]. A collaborative program for carrying out this upgrade has been undertaken jointly by the Raman Research Institute (RRI) and the National Centre for Radio Astrophysics (NCRA), which operates the ORT. It may be noted that this reconfiguration plan is currently under implementation, and the hardware described in this section has already been procured.

The ORT is a 506m X 30m equatorially mounted cylindrical telescope with an equispaced linear array of 1056 dipoles along the focal line. Each dipole has a tuned LNA [30] with about 40 MHz bandwidth centered at 327 MHz. As part of the planned upgrade, the feed array is logically divided into 264 identical segments, where each segment represents an independent antenna element of size 1.93m x 30m. The aim of this upgrade is to reconfigure the ORT into a programmable 264-element array. When completed, the reconfigured ORT will have an instantaneous field of view of $\sim 27^o$, bandwidth of $\sim$ 35 MHz and will be equipped with an NSPS-based digital receiver. Currently, prototypes have been tested for a large fraction of the custom segment of the NSPS and the analog signal conditioning subsystems. The final production and integration is expected to be completed in 2012. The digitizers are organized in 22 digitization blocks located below the reflector at a spacing of about 23 m, where each digitization block includes a 12-channel digitizer capable of operating at 100 MS/s. All the 22 digitization blocks are connected in a star topology with a central system using a peer-to-peer network on optical fibres with multiple links operating at speeds of 2.5 Gigabits/sec from each block.

3.6.1 ORT NSPS layout

The proposed system consists of four hierarchical levels as illustrated in Figure 3.20, where Level-0 is the root node and is realized by a high performance cluster with Infiniband for inter-node communication and GigE for communication with the NSPS. The high speed peer network uses the light weight Aurora protocol simplex links for data uplink, with last mile via GigE. Currently, bandwidths of upto 100 MBps per GigE link into Level-0 memory have been sustained.

- *Level 3:* This level is composed of the distributed digitization infrastructure and is installed at the antenna base. The prime components of this level are the digitizers, the sampling clock reception and conditioning circuitry, the first level data organizer and the peer-to-peer link handler. All these components required for handling 12 sensor elements from a 23 m section of the ORT are implemented as a single board.

  - The ADCs (dual channel AD9600) are capable of a 100 MS/s sampling for signals with frequencies upto $\sim$ 600 MHz. The $\sim$ 60 dB dynamic range at 10 bit resolution allows us to implement an AGC in software. More importantly, the implementation can benefit from the on-chip sampling clock conditioner, divider and duty cycle stabilizer. For instance, it is

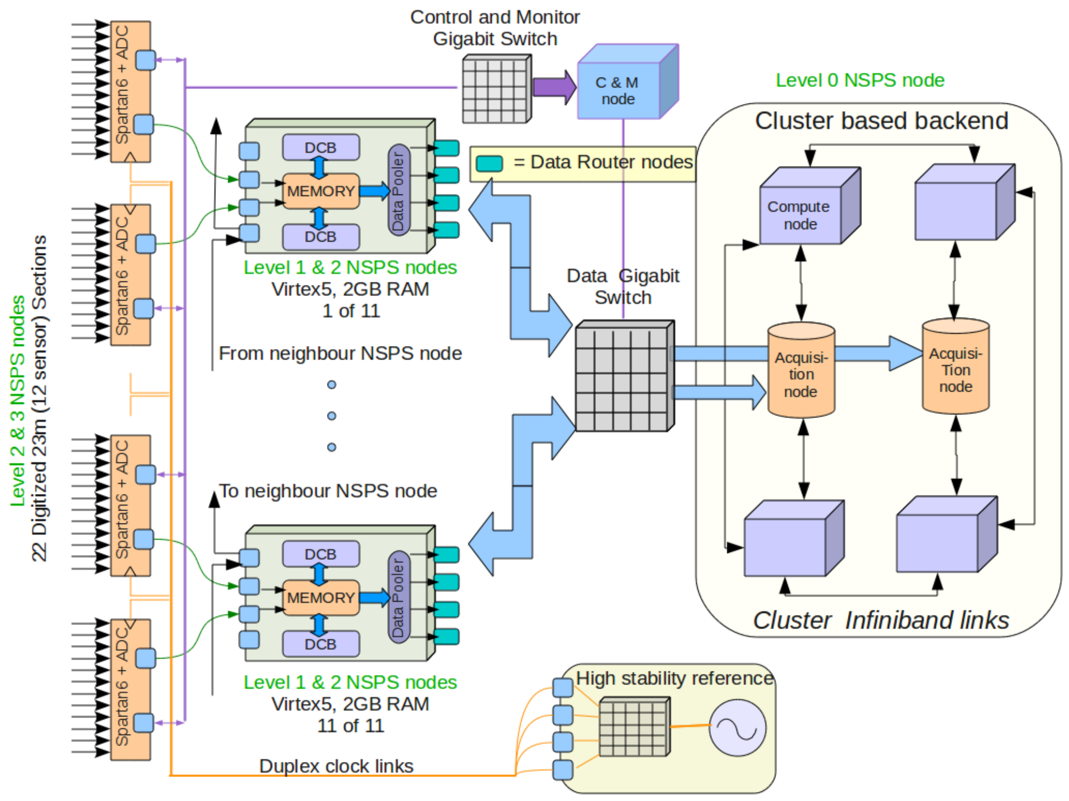


Figure 3.20: *Proposed NSPS implementation for configuring ORT as a 264 element programmable telescope.*

possible to provide a sine-wave with frequency 2-8 times the sampling clock, and use the on-chip features to convert to square-wave, enable duty-cycle stabilizer and divide by suitable integer to get the sampling clock, thus reducing the overall sampling clock jitter and hence the phase noise in the sampled data. This feature is useful in direct (harmonic) sampling of the incoming 327 MHz RF since the Nyquist sampling interval in a band-limited RF is decided only by the bandwidth while jitter tolerance depends on the highest frequency content.

- This level has an embedded clock synthesizer and distributor for the on-board 12-channel ADC based on a reference distributed on fibre by the central high stability clock distributor. It is received using a digital fibre optic receiver. The embedded clock distributor is based on a clock buffer and distributor (LMK03020) which has an on-chip Voltage Controlled Oscillator (VCO) and per-port delay tuning.
- The first data processing block is implemented in a Xilinx Spartan6 (LX45T)

FPGA to perform a conversion of the ADC 10 bit resolution data to ~ 4 bit via configurable, table-driven logic, where the look-up-table (LUT) is dynamically updated to accommodate innovative schemes of compression and segregation. For instance, we can assume that a choice among a predetermined set of LUTs is best suited for coding/compressing the data in a set of physical packets associated with a Transaction Unit. Here, each table implements a different encoding of the input word to an output with lesser number of bits per word. Further, we assume that every word of each physical packet is encoded by a choice between two LUTs out of the set, to represent normal and segregated(flagged) data. For decompression by downstream nodes, the tags of these two LUTs can be accommodated in the packet header, while a one-bit selection between the two can be associated with each data word - thus achieving the dual purpose of flagging and scaling at the word level. Such a scheme can accommodate a wide range of scale factors and hence a large dynamic range within a logical packet. Suitable thresholds for packet-level choice of LUTs can be generated on the basis of integrated power over a reasonable time stretch as part of the pre-processing.

  - The Data Router node buffers data from all 12 sensors in internal memory and reorganizes them to form packets containing identical time-stretches from all antenna elements. The peer-to-peer link out of each 23 m section which connects Level-3 to Level-2 is implemented using the 4 available RocketIO multi-gigabit onboard serializers on the Spartan6. Our choice of SFP for the current implementation can sustain link speeds of upto 2.6Gbps on single mode fibre, while we use the light-weight Aurora protocol at a wire speed of 2.5Gbps for communication between Level 3 and Level 2.

- *Level 2:* This Level is implemented using an FPGA (Virtex5 LX50T) board whose on-board resources include 2GB of memory and 8 multi-gigabit transceivers and expansion connectors. This board can sustain the following Level-2 functionalities:

  - FFT block: Here, the data processor block first decodes data from a pair of sensors, packs them into the real and imaginary parts of a 32-bit complex integer word, and implements a pipeline stage (e.g., radix-64) of a split radix FFT for all pairs of incoming channels. The processing resources are enough to handle upto 24 channels (2 Level-3 entities) at the maximum sampling rate.

  - The Data Pooler block operates using the large local DDR2 RAM and the interconnection with other Level-2 cards to pool subsets of both local and remote Level-2 data into transaction oriented packets by partitioning data along the time axis. Here, each transaction refers to the processing of a specific timeslice for the entire array.
  - The Data router block collects data from the local RAM in units appropriate for transfer to each outgoing port, packetizes them and sends out selected time slices to Level-1 entities. Provision for computation offloading is provided in the form of spare peer-to-peer links which can add on more Level-2 cards.

- *Level 1:* At this level, a memory-based, NSPS to commodity network bridge is implemented. Large bursts of continuous time slices are first buffered in RAM, and then sent out over GigE as properly timestamped packets. The level implements load partitioning as configured by the root level by manipulating ethernet destination addresses of streams going into the data GigE switch. There is possibility of implementing a data processor block for the remaining 16 point FFT operation pending from the split radix FFT.
- *Level 0:* At the root of the NSPS tree, a medium level cluster is proposed to handle both the communication and processing requirements for forming correlations between all sensors. It is important to note that the cluster internode traffic is significantly reduced due to the data routing and transposition carried out using the Data Pooling nodes at the various levels. The formation of actual correlations and the calibration parameter estimation is carried out by this level. The above mentioned partitioning of the load into the 3 levels can be used to bring a subset of data from all 264 elements into one node via a quad-GigE card.

3.6.2 System control

The control network is a simplex, one-way channel from a master with unique id (in the commodity segment) to the peer network through the bridge node. Thus, while data can be routed to arbitrary nodes in the cluster depending on the UDP destination addresses (set during configuration), commands are accepted by the bridge node only through a privileged link from the master. This simplifies assigning privileges to operations related to starting, stopping or resetting the acquisition state machines, configuring the network routes on customized hardware switches, changing ethernet destination addresses, or changing the contents of the LUTs used in higher level nodes like the digitizers.

3.7 Discussion

The implementation of 40-element array receiver establishes many ideas which are directly applicable to the full 264-element implementation. Some differences between the current 40-element, and the planned 264-element implementation relate to the sizes of packets traversing the custom and commodity networks. In the full system, each digitizer node will be generating packets on the Aurora links which will have a larger number of samples from all digitizers. This allows for a lower metadata overhead, while encapsulating a larger timeslice. The commodity segment implementation will employ Jumbo frames, using which an Aurora packet can be directly translated into a GigE packet without segmentation, as is being currently done. This avoids the reassembly load in the commodity segment, while also improving reliability, which is reduced when fragments of the segmented packets are lost.

For a generalization of our approach to the 264-element system, the massive centralization of data pooling can be simplified using large DDR2/DDR3 memory buffers in the custom hardware hierarchy, as is already planned for the final system. A peak data rate of 100 Ms/s x 4bits for 264 elements of the ORT would correspond to about 13.2 GB/s, for which the Level-1 buffering of 22 GB in 11 Virtex-5 cards (shown in Figure 3.20) is comfortable to sustain transactions of upto a fraction of a second duration. Further, due to the unilateral nature of the data transfer, the Aurora links are planned to be operated in simplex mode, since only a single, low bandwidth reverse path is adequate per digitizer node for command reception. This allows the usage of spare Aurora links in peer cards at a given NSPS level, to be used for transmitting data relevant for pooling. The large buffers planned for the hardware are also expected to ease constraints on synchronization.

3.8 Conclusions

We have presented details of reconfiguring the ORT into a 40-element array, and equipping it with a hybrid digital, packetized, FX spectral correlation receiver, which will lead to a larger field of view and higher overall sensitivity of the instrument. This, in turn, will enable several new classes of observations to be carried out by the ORT. Here, the receiver is based on a scaled down version of the NSPS architecture, suitable for the requirements of such an array. Further, a majority of the processing is carried out in software, while the minor changes made to the analog frontend section do not affect the ORT's earlier receivers. The software based signal processing, in addition to providing the ability to tap data at various levels of processing required for system validation, increases the configurability of the instrument in terms of allowing the formation of various sub-arrays, with a variety of backend processing. This implementation has served as a test-bed for refining the prototype cards, the networking protocols and helped in defining the processor side software architecture. Most importantly, it has helped in testing the response of the ORT in the RF domain with digital processing, which has not been attempted earlier. Further, many of the infrastructural aspects of the full system, e.g., fibre topology, gain budgets, calibration aspects etc. have been resolved using the experience with the prototype 40-element array receiver.

Chapter 4

A 40-Element Hybrid Software Spectral Correlator

4.1 Introduction

In a typical array, a Correlator Receiver is used [46] to: (a) align data streams recorded at different spatially separated stations, (b) correct for dynamically varying direction dependent, geometric delays and phases, and static instrumental delays/phases of individual antennas, and (c) estimate the correlation coefficient between data from different antennas. The recorded correlations are then used in post-processing, e.g., for forming a multi-beam phased array, or a synthesis imaging mode, etc. Hence, correlators are an important component of a multi-element Radio Telescope configured as an interferometric array. In particular, a *software* based correlator offers significant flexibility of interfacing and reconfiguration, as well as the advantage of high level programming tool chains and low development times.

In the recent past, the pace of development of commodity compute and I/O components has made it possible to implement software correlators for arrays with medium I/O and computing requirements, on reasonably sized High Performance Computing (HPC) clusters. This is partly due to the availability of optimized vector libraries and increasing DSP support on modern general purpose CPU architectures, as well as due to the rise of high bandwidth I/O interfaces to commodity processors, like Multi-lane PCI-Express and Gigabit Ethernet. Several such implementations, for both connected element interferometry (LOFAR [24], and GMRT [20]), and VLBI (DiFX [47]) have been functional for some time now.

However, one of the major limitations of a software-only approach arises due to the required cross-dispersion of data from every antenna element in the array, to the logic which carries out the correlation. Further, although commodity processors have very efficient compute units, their abilities of receiving and shuffling a large amount of high bandwidth data between multiple processing elements, is inherently limited by their commodity architectures. This makes most of the software correlators I/O bound, rather than compute bound. It may be noted that, even when their I/O capabilities are enhanced via high bandwidth networking add-ons (like Infiniband), the inherently asynchronous and latency prone software layer makes it difficult to implement a distributed, streaming application with strict real-time synchronization requirements, on commodity processors and Operating Systems (OSes).

Thus, for the upgraded Ooty Radio Telescope (ORT), we have proposed a *hybrid approach* for a correlator implementation, which is a heterogeneous, hardware/software co-design. In particular, high bandwidth data routing and shuffling (which are pre-requisites for correlation), but are inefficiently handled in software, are offloaded partially to customized hardware, while the core computing of the correlator is carried out on commodity processors. Our hybrid approach thus relies on hardware offloading of the minimum of processing and routing necessary to make the core computing of the correlator efficient on commodity hardware.

This approach is, in fact, a direct implementation of the NSPS architecture proposed by us (described in Chapter 2). Such an NSPS based implementation allows the use of the highly efficient compute engines of commodity CPUs for integer based correlation. It may be noted that the dynamic range of these integer operations is adequate for a streaming correlator application [48]. Further, operations which are inefficient on commodity processors, like data transposition, synchronization and re-arrangement, are carried out in configurable hardware (in the form of FPGAs) of the NSPS custom hardware segment. In addition, effective load balancing is carried out by the Data Pooler entity of the NSPS custom hardware (described in Chapter 3). This, in turn, allows a set of commodity class processors to handle the high data rates from the 40 antenna elements. The implementation, thus, involves logic resources via FPGAs on custom boards, for latency critical, but deterministic processing, and interconnected, general purpose processors, for latency tolerant, but complex computing.

It is important to note that conventionally, hybrid correlators refer to those which use different kinds of processing elements, each suited for a specific stage of processing, e.g., carrying out FFT in FPGA and cross multiplication in commodity processors.

In this chapter, we present a hybrid, software based spectral correlator for multi-element radio telescopes. In particular, the design and implementation of a heterogeneous, parallelized, packetized, FX spectral correlator is presented. A core feature of this design is the data communication between subsystems terminating in large memories, which has implications on synchronization, compression and reliability of the transferred high bandwidth data. Further, details of the implemented software correlator architecture, an optimized integer FFT and an integer cross multiply and accumulate (XMAC) implementation are also discussed.

In the next section, we describe the role of commodity class processors in the hybrid approach adapted by us towards the correlation estimation problem.

4.2 Commodity processors for streaming correlation

The overall signal flow from the telescope is shown schematically in Figure 3.9 of Chapter 3. As is shown, data generated by spatially distributed sensors (represented by the ADCs) is dispersed and rearranged in a distributed manner, over the different levels of the NSPS custom hardware hierarchy, before being sent to multiple commodity processors for correlation.

We rely on using modern enterprise class processors (coupled with matched motherboards having multiple independent I/O pipelines), as efficient platforms for implementing streaming applications, like correlators. This approach benefits from the multiple, highly optimized Arithmetic Logical Units (ALUs), (usually with several pipelines, wide execution units and high clock rates) provided by modern commodity processors. In addition, larger volume RAM, faster and larger cache memories, increasing number of cores, as well as larger vectored register sizes, also increase the efficiency of this approach.

Further, certain features of the data and computation flow in a streaming correlator application help in optimizing its implementation on commodity processors. These are briefly discussed below:

1. **Low arithmetic intensity:** The basic operation of spectral correlation, which consists of fetching two complex operands, forming their product, and writing the result to an accumulator, results in a low compute instruction/byte ratio of 1. Thus, the implementation has to handle a large amount of data shuffling and data access over broad memory ranges, while carrying out relatively few compute operations on this data.
2. **Streaming data flow:** The data flow in a correlator is extremely smooth, with

no random accesses or runtime dependent jumps. This allows the processors' caching mechanism, as well as frontend data pipelines to be used very efficiently, while hardware and software data prefetching on these flows can be effectively carried out due to the linear flow of the incoming data.

3. **Predictable and regular instruction flow:** The core computing in a correlator is simple, deterministic and repetitive, and thus does not contain sudden and random jumps in control flow. The branching is extremely regular, leading to a lower level of branch misprediction in a commodity processor, and thus, a lower level of pipeline bubbles and flushing, which increases efficiency. Further, the repetitive nature of the instructions prevents pipeline stalls due to under or overflow of pipeline queue buffers.

4. **Integer operations:** The receiver's 4-bit sampling quantization [48] results in a sufficient spectral dynamic range (40-50 dB) needed to handle the RFI environment at the ORT, while leading to acceptable I/O bandwidth within the various networks in the implementation. Due to this, the correlator is entirely implemented via integer instructions, which efficiently use the integer ALU pipelines of a processor. It may be noted that integer ALUs on commodity processors routinely have faster throughputs, in addition to being larger in number than the more complex floating point ALUs. This speedup with integer only instructions is due to:

   (a) Denser vector computing using instruction sets like the Streaming Second Extensions (SSE) on the target processors. These specialized instructions operate on a group (vector) of data at a time, leading to larger number of operations per processor clock cycle.

   (b) Denser data packing, leading to lower number of cache-line misses while shuffling data, as well as while loading from I/O buffers.

   (c) Lower latency and higher throughputs of integer arithmetic instructions as compared to floating point arithmetic.

5. **Multiply and Accumulate (MAC):** The MAC process, which forms a majority of the computing needed for the generation of the spectral cross correlation, can be efficiently implemented on a commodity processor. However, this requires the incoming data to be laid out optimally in memory. This important aspect is elaborated upon, in Section 4.3.4.

Memory as a switch: Estimation of the spectral correlation between data streams from different antennas consists of a per-stream processing segment (which operates independently on each sensor's data stream, and thus is embarrassingly parallel), as well as a group-level processing segment, which requires a many-to-one connectivity between sensor data streams and the group processor. We have chosen to realize this many-to-one connectivity using the memory associated with commodity processing systems as a "switch", thus enabling the efficient utilization of the multiple compute cores available on commodity processors. Such an approach was found to be more efficient as compared to switches implemented using explicit I/O, or data replication via broadcasting mechanisms.

However, implementing many-to-one "switches" in the memory of a processor requires the availability of data from all sensor streams for a given timeslice, on demand. This important function is carried out in the NSPS's Data Pooler entity, which utilizes its on-board memory to carry out a traffic shaping. This shaping refers to the grouping of incoming distributed data into packets containing data from the same timeslice, but for all 40 streams. These packet groups are then routed to individual commodity processors, thereby allowing the all-to-all memory based switch to be implemented in the processor's memory. This aggregation of data from distributed data sources into a single data packet can be extremely expensive if carried out in a conventional manner using a centralized switch (shown in Figure 1.1 of Chapter 1).

4.3 Correlator implementation for ORT

We now present an implementation of our hybrid FX spectral correlator approach (based on the NSPS architecture) for the 40 antenna element ORT array.

4.3.1 NSPS Level-0 node hardware

In this section, we present the specific technical details of the Level-0 nodes for the hybrid correlator implementation.

Here, these Level-0 nodes are implemented on commodity server class machines. In particular, the compute nodes are Intel Xeon E5430 (Harpertown) class, quad core, dual processor systems, which offer 128 bit Single Instruction Multiple Data (SIMD) registers as part of the SSE and later Instruction Set Architectures (ISAs). These instruction sets are rich in integer operations on a vector of data organized as 16 bytes, 8 short ints or 4 integers. Each of the four cores has 3 different vector ALUs,, which are independent of the Floating Point Units (FPUs), scalar integer ALU and the vectored FPU. The vector ALUs are placed on 3 ports of the Re-Order Buffer, thus allowing a high theoretical throughput of almost 1 vectored instruction every clock cycle. The memory subsystem of the Level-0 node used in our implementation was found to have memory bandwidths of $\sim$ 40 GBps for 128-bit sequential reads and $\sim$ 34 GBps for writes from cache memory (using the STREAM benchmark). Each node is equipped with two GigE links, and 16 GB of DDR2 RAM. The software environment on these nodes is Intel's icc compiler based, on a 64-bit GNU/Linux installation.

A block level schematic view of the NSPS Level-0 nodes interfaced with the NSPS custom segment depicted in Figure 4.1. This view shows the incoming transaction units (composed of multiple physical frames holding data transposed by the NSPS Data Pooler) traversing multiple, parallel I/O paths commonly found on typical commodity hardware, before being written into large memory buffers. These buffers, in turn, replicate and house the transaction units in contiguous regions in memory.

Most importantly, the individual transaction units are then assigned to individual processing cores for their correlation, with a minimum of communication expected between different cores. This is due to the transposed transaction unit having available, all the data required for correlation by a processing core. Such an approach is benefited by the current trend of increasing number of processing cores per physical processor.

4.3.2 Software correlator architecture

The schematic of the correlator architecture is depicted in Figure 4.2. As shown, the software correlator implementation consists of the following main software components:

1. *Network handler modules,* which receive raw data from the NSPS hardware.
2. *Memory buffer modules*, which allow for coarse level synchronization, and block operations over longer time stretches using large memory buffers.
3. *Core processing modules*, which consist of an optimized integer FFT, and an integer XMAC implementation.
4. *Collator module*s, which collate processed data from all the core processing modules, to form the output of the correlator.

In addition, the *sequencer* module manages the temporal sequence of the generated output. For a multicore architecture, each of the modules can be assigned to an

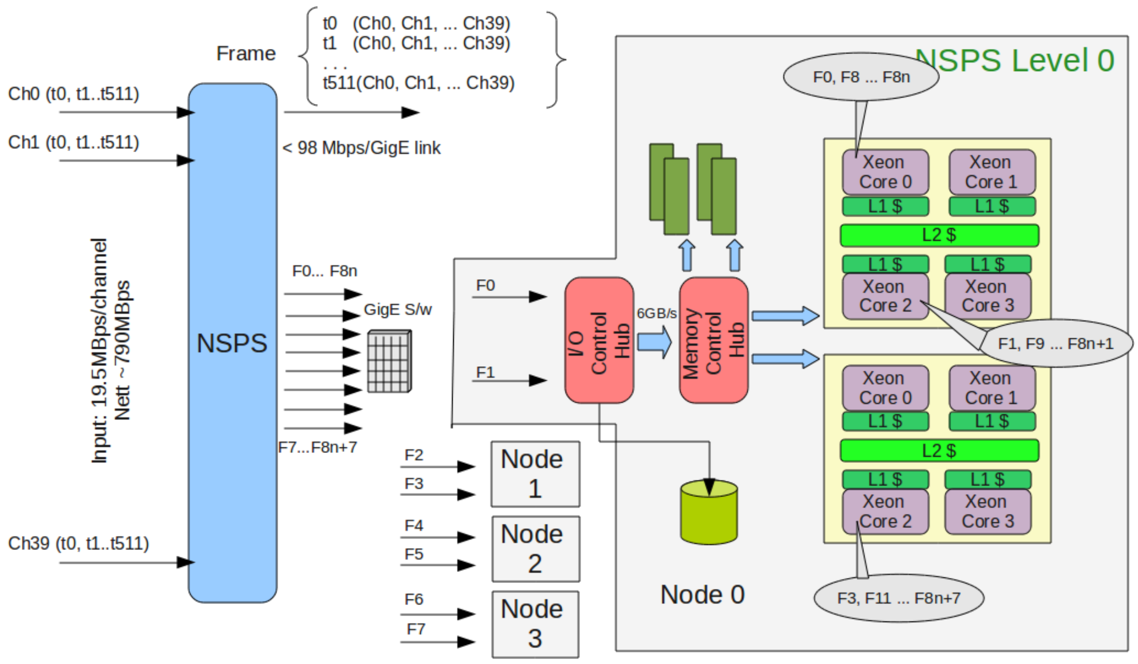


Figure 4.1: *Schematic depicting the NSPS custom segment interfaced with the NSPS Level-0 commodity hardware. The multiple cores of execution are matched to the transaction unit based output of the NSPS's custom section.*

individual computing core for the duration of the processing, thus minimizing jumps of the running processes across cores, and maximizing cache locality. Here, the NSPS distributes the total I/O over multiple links in order to match the per link I/O to the processing bandwidth of each processor being used in the Level-0 NSPS nodes.

In particular, the time domain data decomposition and the synchronization across transaction units carried out by the NSPS custom hardware allows us to adopt a Single Algorithm, Multiple Data (SAMD) approach to parallelize the correlator implementation. A popular framework for such parallel implementations is OpenMP [21], which realizes automatic parallelization of code, based on programmer issued compiler directives. However, an explicit thread based approach, where the threads have minimal communication between each other, was followed, which allowed for a better mapping of the logical subdivisions of the problem to the resources, than that using an OpenMP based approach.

The NSPS custom hardware serves out atomic and independent data packets on multiple GigE links, to the dual processor compute nodes. Here, each GigE link is served by an I/O thread which creates a pool of the incoming data packets in a large buffer in thread local memory. These threads then serve packets in a strict sequence to each of the four available cores in a processor, in a Round-Robin fashion.

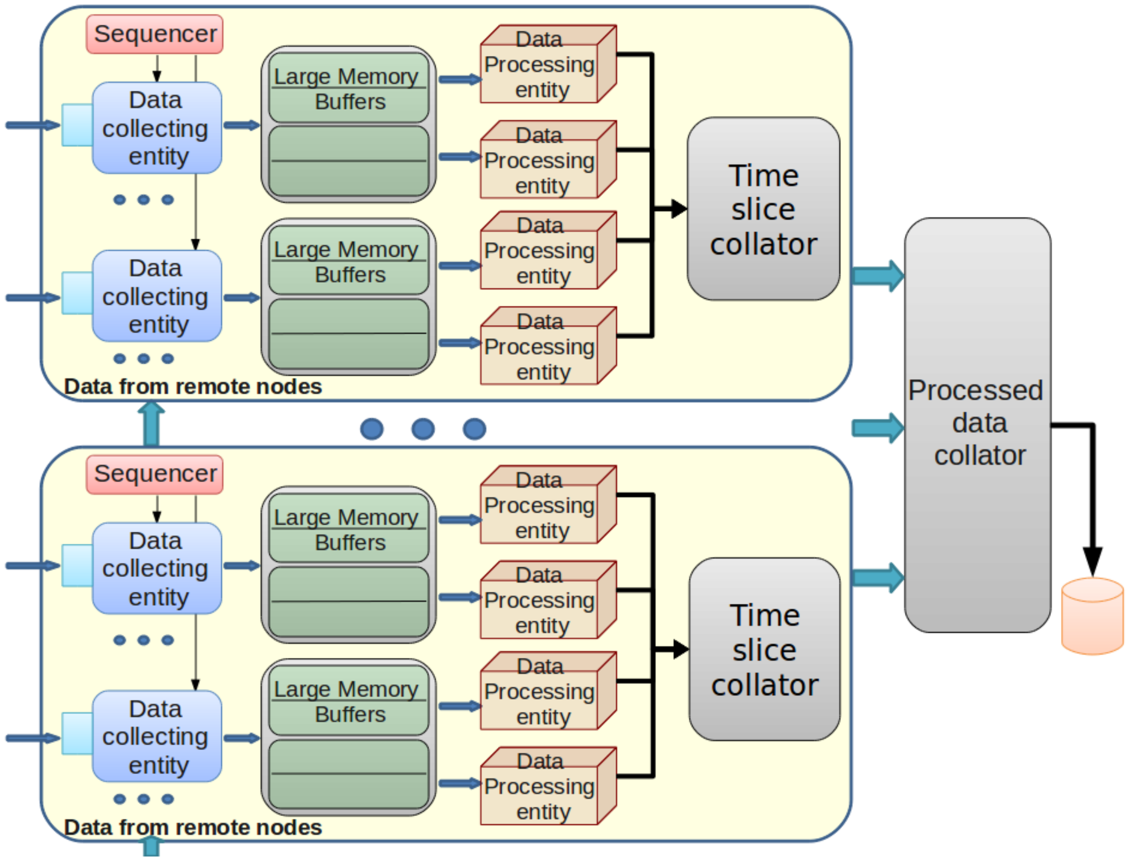


Figure 4.2: *Block diagram of the software correlator architecture. Here, data is collected from each GigE link by a data collecting entity, which organizes the incoming transaction units into large memory buffers. A sequencer indicates their arrival and schedules their processing onto individual data processing entities. The processed output is then collated in two stages: time slice collation, followed by collation of data from multiple nodes.*

Each correlator thread utilizes a set of two incoming packets, (which correspond to 16 timeslices of 64 samples each) for carrying out the correlation.

At this level of time integration ($\sim 25\,\mu s$), the data rate from the correlator, in fact, increases by about 10x due to the large number of baselines for the 40 element array. This output data is then sent to the Long Term Accumulation (LTA) process, which maintains its own cache for carrying out long integrations. After accumulating $\sim$ 400 incoming data blocks, a compression of $\sim$ 40x is achieved, with the data rate of the generated correlations being $\sim$ 20 MBps. This is the net output data rate, which, in practice, is shared by 4 nodes, leading to a very manageable 5 MBps data write rate to disk.

4.3.3 FFT implementation

We have designed and implemented a vectorized, pair FFT module using only integer operations. Here, a single complex FFT on two packed, unrelated data streams (either a pair of antennas or different timeslices of the same antenna) is carried out, followed by the extraction of the individual streams.

This implementation is optimized for a 4-bit input precision, and generates outputs in a manner optimal for the XMAC, computed in the next stage. This FFT module operates on all input streams, and carries out a 64-point, Radix-8 Decimation-In-Time FFT, in two stages (i.e., stage-0 and stage-1) of Radix-8 FFT. Such a two-level decomposition can be compactly summarized by the following equation [49]:

$$X(k) = X(8r+s) = \sum_{m=0}^{7} W_8^{mr}\ W_{64}^{ms} \sum_{l=0}^{7} x(8l+m)\, W_8^{sl}, \tag{4.1}$$

where, $X(k)$ is the k^{th} Fourier component, $x(n)$ is the incoming time-series from an antenna, $W_{64}\,\&\,W_8$ are the twiddle factors, and $r\,\&\,s$ vary from 0 to 7. Each Radix-8 stage of the FFT is, in turn, implemented as two Radix-4 stages followed by a Radix-2 stage. This results in a spectral resolution of $\sim$ 625 KHz for the $\sim$ 40 MHz sampling.

Some of the additional advantages of our FFT implementation are:

1. Stage-0 data shuffling occurs at the byte level, reducing memory move operations.
2. Radix-8 FFT involves only the usage of complex add and subtract operations, or the swapping of the real and imaginary components (as against full complex multiplies), for implementing the FFT butterfly.
3. A scaling factor of $\frac{1}{\sqrt{2}}$ needs to be applied as part of the Radix-8 butterfly implementation. Since this is a common factor for both real and imaginary components of an intermediate complex number, we can use a scaled, 8-bit representation for this factor in stage-0, and a 16-bit representation in stage-1 of the FFT. This scaling is also applicable to the twiddle factors involved. Since these representations are more accurate than the incoming 4-bit data, the error in the integer FFT is limited by the precision of the incoming data, rather than by the integer FFT itself.
4. A fully vectorized FFT was implemented, with all memory moves and accesses being optimized for 128 bit SSE instructions.

5. The final stage of the FFT reorganizes data in the most optimal manner for the XMAC operation (see section 4.3.4). Also, the extraction of individual streams from the pair FFT output is combined with this data shuffling/reorganization. This absorbs the extraction load to a large extent.
6. The FFT can be easily extended to 512 points by the addition of another Radix-8 stage, for achieving a higher spectral resolution (if required).

In the following subsections, we describe the sequence of steps involved in implementing the FFT. Here, the incoming raw data streams are first decomposed into appropriately strided subsets, with the real and imaginary components accommodating different timeslices of a pair of streams. Stage-0 of the FFT is then carried out, whose output is shuffled for stage-1, which in turn, results in the 64-point, 16-bit complex FFT output. Subsequently, the individual streams' output is extracted, and simultaneously rearranged in order to carry out the XMAC optimally.

4.3.3.1 Data decomposition for stage-0 of the FFT

The NSPS implementation for the ORT organizes data into groups of 512 time-contiguous samples, referred to as *frame*s. Incoming frames are decomposed into 8 *timeslices* of 64 contiguous samples each for the next stage (stage-0) processing. The 4-bit data samples are then picked from every 64 sample data set, with a stride of 8, resulting in 8 samples per data set. These samples are then unpacked from 4-bits into 8-bits. Corresponding samples from 16 such timeslices (from two consecutive frames) are then put into a single SSE register (128-bit wide), hence each sample is separated from the next by 64 samples. Thus, 8 such SSE registers are used to accommodate all the samples for carrying out the first stage, 8-point FFT on 16 samples (organized as 8 complex numbers). Such a decomposition results in the optimal usage of the SSE registers.

4.3.3.2 Stage-0 Radix-8 FFT

In this stage, a pair of Radix-4 FFTs is carried out separately, on the even and odd samples in the 8-sample group. These are implemented using native packed arithmetic instructions, as the Radix-4 butterfly implementation consists of direct addition or subtraction of complex numbers. Further, the multiplication by $\pm i$ in the Radix-4 butterfly, corresponds to a swap between the real and imaginary components of each complex number. This swap operation was optimally carried out using a packed swap instruction, available from the chosen processor's instruction set.

The Radix-2 at the end of stage-0 requires a multiplication with factors like $(\pm 1 \pm i)$, with a scaling by $\frac{1}{\sqrt{2}}$. These multiply operations are implemented by first forming all the sums and differences of the real and imaginary components of the complex Radix-4 output, followed by a swapping between the resultant components, which depends upon the multiplicative factor.

Further, all stage-0 operations are in-place, and carried out in 8-bit mode. Thus, 4 spare bits are generated on expanding the input data (which has a precision of 4 bits) to bytes. These 4 spare bits allow accumulation over 16 incoming data values, while in a stage-0 Radix-8, an accumulation is carried out only on 8 data samples. Thus, the bit-growth in our Radix-8 implementation is well within the constraints set by the available word size. It may be noted that all scaling and normalizations are carried out after the accumulations, in order to preserve precision. Once the stage-0 operation is carried out for all 8 timeslices, the output is multiplied with the appropriate twiddle factor (W_{64}^{ms} in Eq. 1), rearranged in a bit-reversed order, and then passed on for the stage-1 FFT implementation.

4.3.3.3 Stage-1 Radix-8 FFT

Stage-1 carries out another 8-point FFT on a set of 8 complex numbers, formed by taking the correspondingspectral channels of each stage-0 group output. Since this stage is implemented with 16-bit integer precision, an intermediate process handles converting the 8-bit complex outputs of stage-0, to 16-bit complex. Thus, stage-1 generates 16-bit complex outputs, corresponding to the 64-point FFT of the incoming raw datastreams. Since the output is in-order, a separate code block can be used to extract individual, transformed data streams for carrying out the XMAC. This output extraction is merged with the task of rearranging the data in the most optimal manner for the XMAC.

Data scaling: In each of the two Radix-8 stages, the Radix-2 implementation (following the Radix-4) requires multiplication by a $\frac{1}{\sqrt{2}}$ factor, which is represented as an 8-bit scaled integer. The Radix-4 outputs, (fed as inputs to Radix-2), and the $\frac{1}{\sqrt{2}}$ factor are converted to 16-bit before this multiplication. Since the Radix-2 output is also restricted to 16 bits, the `pmulhw` instruction is used, which discards the lower 16 bits of the 32-bit output of the multiplier. At this stage, proper scaling of the input is required to ensure that data bits are not discarded during the `pmulhw` instruction. This scale factor depends on the range of the incoming data, and must be precomputed and set. A similar operation is carried out during the stage-1 implementation, where once again, such scale factors need to be tuned. Note that if floating point operations were to be carried out, such a data dependent scaling would not be needed, thereby avoiding overheads. However, the integer operations are much more efficient, as the actual float operations have a lower throughput (See Table 4.3).

Arithmetic intensity: The arithmetic intensity (defined as the number of operations performed per byte of data loaded from memory) is extremely low for the correlation algorithm. This implies that, for actual implementations, the memory shuffling and move operations can form a significant fraction of the computing load. Hence, for the FFT implementation, we have used the most efficient way of moving data, i.e., using 128 bit SSE moves. Here, cache misses due to moves (which result in stalling of the instruction) were minimized. Significantly, the current implementation results in memory moves accounting for close to 50% of the total instructions. This is essentially due to the shuffling of data being carried out while it is compressed at 4-bits/sample.

Further, the dominant arithmetic instruction in the FFT is packed adds and subtracts on bytes and words, which are available in the SSE ISA in a “signed saturated mode”. This allowed operations on data to be carried out without scaling to prevent overflow, thereby reducing the otherwise required checks.

The frequency of occurrence of the instructions dominating our implementation, and their estimated latencies and throughputs are depicted in Table 4.8.

4.3.4 Cross multiply & accumulate implementation (XMAC)

The XMAC implementation requires a much higher processing bandwidth than the FFT itself, and thus accounts for a large fraction of the processing time in a correlator. For the XMAC implementation in our approach, the important task of collation of data from all distributed sources into a single packet is carried out by the custom hardware (Data Pooler) of the NSPS architecture. This allows correlation within such a packet to be carried out by a single processor core, without communicating with any other core. Further, the time alignment of the distributed data (needed for high correlator efficiency) is also carried out by the NSPS hardware. In comparison, carrying out these tasks of data collation and alignment in software using commodity networks is highly inefficient.

However, in spite of eliminating inter-processor communication at a fine level via data collation, the layout of this collated data can still lead to inefficient memory access. The large I/O resulting from the low arithmetic intensity of the correlation

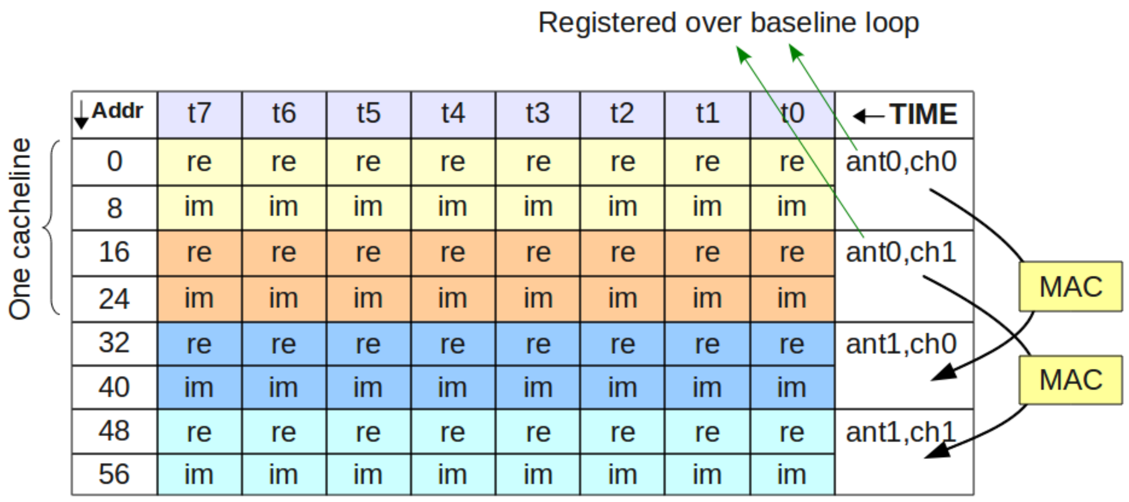


Figure 4.3: *Schematic representation of the optimal data layout for XMAC. The FFT output is 16-bit integer complex, with the XMAC output being 32-bit integer complex. This can be converted to 32-bit floating point complex during LTA.*

process further increases this inefficiency. Thus, there is a requirement of designing an optimal data layout for carrying out the XMAC.

We now discuss some aspects related to designing such an optimal layout. In the spectral cross multiplication process, the data from the same sensor needs to be used several times on an average, in multiplication operations with data from all other sensors. Thus, an optimal layout should allow registering (in CPU registers) such an often used data object. Further, the output data layout of popular libraries used for the previous FFT operation can result in the required data objects spanning several cache-lines. This leads to the loading of an entire cache-line into L1 cache, for every object access. Thus, an optimal layout should aim at increasing spatial locality, leading to efficient cache-line access and utilization. Furthermore, the accumulators required for every spectral channel of each baseline (following the cross-multiplication), can quickly overflow the limited L1 cache of commodity processors, even for small arrays with limited spectral resolution. This, in turn, leads to a lowered throughput of the correlator. In our case, for a 64-point FFT, the 40 antenna elements lead to 780 baselines being formed, and an integer-complex representation for each complex accumulator results in their occupying , leads to just these accumulators occupying about $\sim$ 200 KB, which overflows L1 cache by 6x. Thus, an optimal data layout should minimize the number of accumulators needed, thereby reducing cache misses.

4.3.4.1 Optimal XMAC layout

Based on the aspects discussed above, we have carried out an efficient arrangement of the FFT outputs, which results in an optimal XMAC implementation. This is possible due to the available flexibility of reorganizing the FFT output as per computing convenience. The optimal layout is depicted in Figure 4.3, and has the following features:

1. Multiple timeslices of the same data object are placed in a single SSE register. Notably, the primary data compression in a correlator occurs in the *temporal accumulation* of the formed correlation coefficients, (which takes place after the complex multiplication). This is most efficiently done using instructions like `phaddd` and `pmadd`, which require all timeslices to be available in the same SSE register. Thus, such a choice of integer-only ISA instructions further supports the chosen placement.
2. Data from different antennas are closely placed in memory, in order to minimize cache misses.
3. The real and imaginary 16-bit components of each spectral channel are separated, such that they reside in different SSE registers. This is optimal due to the presence of instructions like `pmulwd`, which can carry out vector multiplications on the separated data, resulting in efficient generation of products of the kind `(Re (Ant0)*Re (Ant1))`, `(Im(Ant0)*Im(Ant1))`, `(Re(Ant0)*Im(Ant1))` and `(Im(Ant0)*Re(Ant1))`. These are then used to form the complex cross products.
4. Data from a single sensor is held in an SSE register, while all 39 baselines with this sensor are being formed. This minimizes memory access for such frequently used data.

Further, SSE instructions with the ability to saturate each component of the vector were used. This feature avoids possible error in the final correlations, if such a saturation occurs during the various temporal accumulations.

4.3.4.2 Hierarchical accumulation

There are several contexts which require different integration times over which correlations are to be accumulated. While the requirement of high temporal resolution (low integration time) is dictated by observational requirements, e.g., the local RFI environment, or search for high time-resolution variability, post correlation processing requirements dictate lower temporal resolutions (higher integration time), which result in higher sensitivity and a lower I/O load to manage. As discussed earlier, the particular choice of data layout and the SIMD instructions used, also dictates a minimal integration period. Further, observational parameters also need to be accounted for, e.g., ionospheric scintillations in low frequency observations, which have a typical bandwidth of ∼few KHz, thus requiring low (millisecond) level integrations.

Thus, the accumulation was implemented in a hierarchical manner. At the lowest level of this hierarchy, the *pre-integration* corresponds to a time span required for an efficient SIMD implementation, and was fixed to 8 timeslices. This allows us to carry out this pre-integration using only 16-bit precision, making the implementation more efficient. The next hierarchical level consists of a programmable *Short Term Accumulation* (STA), which was carried out using 32-bit arithmetic. This results in high data rates and temporal resolutions, with the accumulated correlations being typically routed in-memory to online processes, which carry out multiple passes on this data. This is required to generate *side-information,* which is later used to segregate unreliable data before the highest level of the hierarchy, i.e., the *Long Term Accumulation* (LTA). Here, segregated data is allowed to be separately accumulated over a longer time (typically tens of milliseconds).

Such a hierarchical scheme not only allows for better estimates of the cross correlation coefficients, but also creates a pool of so-called unreliable data, which can then be sent to a separate processor for further analysis.

4.3.5 Software data pooling

In the implementation of the 40-element array receiver interfaced with the described software correlator, the "Data Pooler" element has been implemented via software, on commodity computing systems. This system, whose configuration is depicted in Figure 4.4, carries out a real-time software based data pooling, followed by real-time correlation. Here, distributed data generated in the custom segment of the NSPS is connected to the commodity segment via a bridge layer. These data are acquired by a *recorder* sub-level, which can either store bursts of the incoming, high bandwidth data to disk for offline processing, or route them to output ports for transport to the next sub-level.

The pooler element utilizes the large memories available on such systems in order to carry out a second level of data transposition on the incoming, high bandwidth data from different recorders, creating transaction units of upto 1 minute with the target hardware. This transposition is carried out by a data *playback and pooling* layer, which places incoming packets in locations in the system RAM, based on their 32-bit timestamps. Such an approach caters to missing packets from individual pillars,

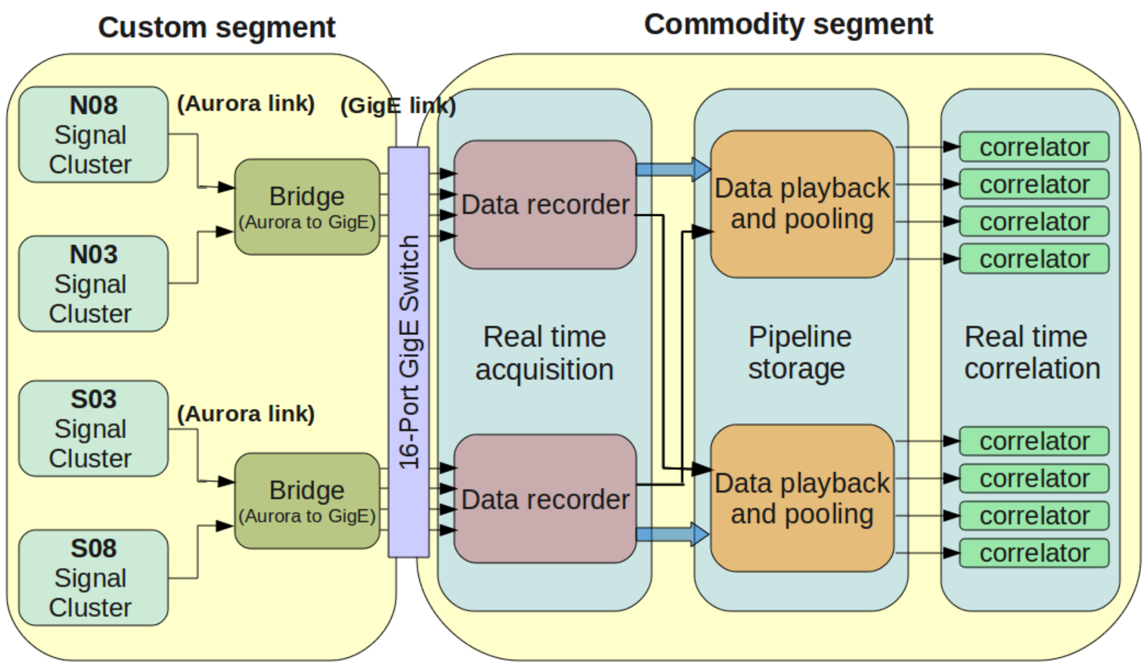


Figure 4.4: *Scheme for software based data pooling. Here, the NSPS custom segment acts as a streaming Data Acquisition System. The data transposition of the incoming data is carried out using commodity switches, while the data pooling and creation of transaction units is carried out by a pipeline stage in the commodity segment of the NSPS.*

which can otherwise lead to loss of synchronization. These pooled data packets (each consisting of time-aligned data from all distributed sensors) are then forwarded to a cluster of commodity processors over the commodity segment of the NSPS network, which carry out the bulk of the processing in order to form an estimate of the spatial correlation function. Thus, the Level-3 Data Pooler carries out a temporal traffic shaping within the NSPS commodity hardware and network domain.

This software system facilitates the validation of several subsystems relevant to the NSPS architecture, including link and data integrity, checks for synchronized clock distribution etc., before their ultimate integration into the hardware.

4.4 Results

This section presents comparisons of our correlator against a “control” correlator implementation, having identical computational and correlator parameters (in both floating point and 16-bit integer versions). The control correlators were implemented using the most optimal standard library functions available, and operate using standard data layouts. In particular, Intel's Integrated Performance Primitive (IPP) [50] library was used for the main computational load of the FFT and XMAC. This library offers a vectorized FFT, generic `AddProduct()` functions for float complex vectors, and several other vectorized arithmetic functions. The float `AddProduct` function ostensibly uses the floating point Fused-Multiply-Add (FMA) instructions available on modern Intel architectures [51]. For the integer comparison, we used the vector multiply and vector add functions separately to generate the complex products. Further, standard code and data optimization procedures were followed where ever possible, like allocating memory on cache-line size aligned addresses, avoiding split cache-lines by operating within a cache-line if possible, etc. Intel's Vtune profiler was used to obtain processor and memory subsystem specific comparison results.

The results can be grouped into three categories: (a) correlator subsystem throughput rates excluding the external I/O requirements (but including the memory subsystem) (b) compute and memory access efficiency related, and (c) full system throughputs. We first present comparisons of individual sub-components of the correlator, and subsequently compare the full correlator implementations.

A breakup of the overall load among the major components of the correlator is tabulated in Table 4.5, to bring out the effect of the optimization of each functional block.

4.4.1 Communication protocols

Since the correlation process is expected to be I/O bound, and a standard protocol (UDP) interfaced to a commodity switch and motherboard is utilized, the incoming rates sustainable on our target hardware were measured. The data was generated by the NSPS Level-1 entities, which in the current implementation, is a custom Virtex-5 FPGA based board with a GigE implementation.

In our implementation, we were able to sustain a total bandwidth of $\sim$105 MBps for long durations into Level-0 node memory, with a loss fraction of $<1\%$. The loss fraction for multiple data rates is tabulated in Table 4.1. These numbers were obtained by inter-comparing the timestamps from the incoming streaming raw data. Thus, we found that commodity hardware with stock networking stacks can operate with high efficiency, for multiple such data links (upto 4 on our hardware) terminating in a single Level-0 hardware node.

4.4.2 FFT

The major components of our FFT implementation, their load contribution to program execution and their main functionalities are tabulated in Table 4.2. As can be

Samp clock	Per GigE Rate (MBps)	% data received	Longest drop (MB)
4 Active GigE links, (3GB/burst)			
20 MHz	50	[99.94, 99.77, 99.99, 99.97]	[0.96, 1.57, 0.02, 0.04]
42 MHz	105	[95.58, 98.75, 99.67, 96.04]	[1.01, 1.12, 1.38, 1.68]

Table 4.1: *PC based burst mode link reliability of a Bridge card. The packet structure allows embedding of sequence numbers in order to verify link reliability. The set of numbers in col. 3 and 4 correspond to each of the four active links.*

Functional block	% load	CPI	Major functionality
`mergeBlockPair ()`	26	1.083	Timeslice reordering, nibble to byte conversion
`radix8Byte()`	9	0.566	Stage 0 FFT
`byte2WordPairTwid()`	31	0.620	Byte to word conversion, twiddle multiplication and reordering for stage 1
`radix8WordStride8()`	15	0.572	Stage 1 FFT
`retrieve_pair_fft()`	12	0.629	Pair FFT extraction

Table 4.2: *Functional load distribution of the integer FFT implementation.*

seen, the shuffling of data within the FFT kernel (implemented in the `mergeBlockPair` and `byte2WordPairTwid` functions) takes a large fraction of the compute time, compared to the actual multiply and accumulates. This can be attributed to a much higher fraction of `mov`, or I/O, than compute instructions, seen in the histogram of instructions in our implementation (Table 4.8).

As a measure of compute efficiency, the CPI (Clocks Per Instruction retired) ratio was used. This should ideally be close to 0.25 for a 4-way superscalar processor architecture such as ours. We obtain an overall CPI value of 0.483 for the complex FFT engine with pair retrieval. The IPP floating point FFT generated the best CPI of 0.441.

Next, we compared the performance of our FFT implementation, termed `NSPS_int`, with that of standard optimized libraries. In our tests, we found Intel's IPP FFT to be marginally faster than that available from fftw3 [52], for FFT sizes ranging between 64 and 512. For the purpose of illustration, we present measurements for a 64-point implementation, which are tabulated in Table 4.3. In order to carry out these bench tests, a large data pool of several GigaBytes was first assembled in memory, and subsequently, only memory reads were used to feed the FFT engines. Thus, we involved the memory and cache sub-systems, but not the I/O subsystem. Also, the memory access is unidirectional due to the read only access. Comparison between code generated by gcc and Intel icc compilers were made, and the latter was found

64 point FFT Category	Single complex (MFFT/s)	20 stream complex (MFFT/s)	Speedup (NSPS_int Vs Others)	CPI
NSPS int	8.825	1.31	1	0.483
IPP 16-bit int	0.292	0.014	30.2	0.479
IPP float	0.411	0.020	21.4	0.441
FFTW3 strided float	0.200	-NA-	44	-NA-

Table 4.3: *Comparison of the speed up and FFT throughputs between various reference implementations and our integer FFT implementation. These rates are for complex FFT without pair FFT extraction.*

to be a few percent faster than gcc. Hence, the icc compiler was used for all further tests.

Further, all implementations operate on the packed 4-bits/sample data layout, i.e., the cost of the raw nibble extraction is also included in the measured numbers. All tests were carried out on a single core of the target system, with the most optimal variant of the FFT used.

It is interesting to note that our integer FFT implementation was faster than the library implementations by several factors. Further, an almost 2 orders of magnitude speedup was obtained while operating on a packed data stream. This demonstrates efficient use of memory access for the larger dimension complex vectors. Note that the IPP float FFT is faster than the short-int FFT, contrary to expectations.

4.4.3 Cross multiply and accumulate (XMAC)

In Table 4.4, we can see an average speedup of $\sim$ 4x for our optimized XMAC implementation (using the data layout described above), against those using IPP library functions (`ippsMul` and `ippsAdd, ippsAddProd`). It may be noted that these numbers only measure compute performance, and have been generated by in-memory, static test data buffers, thus eliminating external I/O.

This table also depicts the higher efficiency of the integer units by $\sim$2x against the floating point units of the processor. Further, it is clearly seen that the IPP float implementation using the addProduct () or FMA instructions is actually faster than the IPP integer implementation. It is important to note that the XMAC engine forms a major part of the correlator load, as borne out by the load distribution presented in Table 4.5, with speedup gains affecting the full correlator performance.

4.4.4 Correlator

The per-core load contribution of the functional blocks of the correlator is tabulated in Table 4.5, for the IPP implementation, and in Table 4.6 for the NSPS based

Acc.	xL1D	IPP 16-bit GMAC/s	NSPS 16-bit GMAC/s	Speed Up	IPP float (Addprod)	NSPS float GMAC/s	Speed Up
44x32	3.7	0.304	1.21	3.98	0.342	0.670	1.96
44x64	7	0.352	1.29	3.66	0.378	0.670	1.77
44x128	14	0.385	1.29	3.35	0.388	0.672	1.73
44x256	29.5	0.404	1.29	3.19	0.385	-NA-	-NA-

Table 4.4: *Tabulated speedup of complex XMAC for correlation using optimal data layouts and hand-coded SSE assembly, against standard functions from the Intel IPP vectorized library. The first column gives the number of antenna elements x spectral channels. The second column gives the L1D overflow factor of the accumulators. The speedup is of the NSPS implementation against IPP's FFT output layout.*

Function	IPP 16bit int % Load
XMAC	76
Data Extraction	7.0
Complex 64-point FFT	5.8
ipps_BitRev	1.9
extract_fftpair	2.4
ippsConvert_8u32f	1.5
ippsRealToCplx_32f	1.7
ipps_Mul	3.5

Table 4.5: *Profiled compute load distribution of the major code blocks of the reference correlator implementation.*

implementation. The measured load distribution re-iterates the importance of the data layout for optimization.

The net measured throughput of the various correlator implementations, tabulated in Table 4.7, clearly shows the effects of our optimization. We achieve a total speedup of a factor ∼3.5x. These throughputs are in units of actual input network data rate handled, which is related to the total allowed system bandwidth. For our system, with a sampling clock of ∼ 40 MS/s, 4-bit sampling and 40 antenna elements, this translates to ∼ 800 MBps of total data to be handled. The throughputs were measured by first processing only in-memory data with no I/O load (mem. reads), and then with the network I/O handlers feeding incoming streaming data to memory (net. reads). It may be noted that the net. reads correspond to a single GigE link's data bandwidth, as against the eight links needed to serve the full incoming data rate.

Function	% Load
XMAC (STA)	57.4
Cnvt to 32-bit+LTA	15.6
byte2wordpairTwid	6.7
mergeBlockpair	3.5
radix8WordStride8	3.9
retrieve_pair_fft	2.4
radix8Byte	1.9

Table 4.6: *NSPS correlator load distribution among the main software blocks. The correlator operates on the incoming, high bandwidth data-streams, in quasi-real-time.*

Correlator Type	Mem. reads (per core)	Net. reads (per core)
IPP float (ippAddProduct)	26.4	24.8
IPP 16-bit int	15.8	15.1
NSPS int	91.8	88.7

Table 4.7: *Measured correlator throughputs, in MBps of data read from a memory pool (mem. reads), and using live packets over a single network link (net. reads). The rates correspond to a correlator time constant of 1600 packets.*

Figure 4.5 shows the bandpasses of typical antenna elements from the telescope, after an integration of ∼100 ms. The ripples and the droop seen in the bandpass are due to the anti-aliasing SAW filter response. Typical baseline outputs from some baselines for a calibrator source (Hercules -A, $S_{327} = 175$ Jy) are also shown in Figure 4.6. Further, Figure 4.7 shows the closure phase errors in the dataset. These are used to identify antennas causing the visibilities to deviate from their expected values.

4.5 Calibration of the 40-element prototype array

While carrying out an observation, the signals from each antenna sensor element of the ORT can have an unknown and random delay and phase with respect to each other. These delays and phases can have two parts: one due to the geometry of the array in relation to the source position (which changes in a calculable manner for different sources, but not with time), and the other due to the instrumental delays, i.e., differential delays between the electrical paths taken by the signals before they are correlated. These instrumenatl delays are expected to remain static for a typical observing session, and for different declinations, but can slowly change with time. Both these delays need to corrected before the data from different elements in the array can be combined.

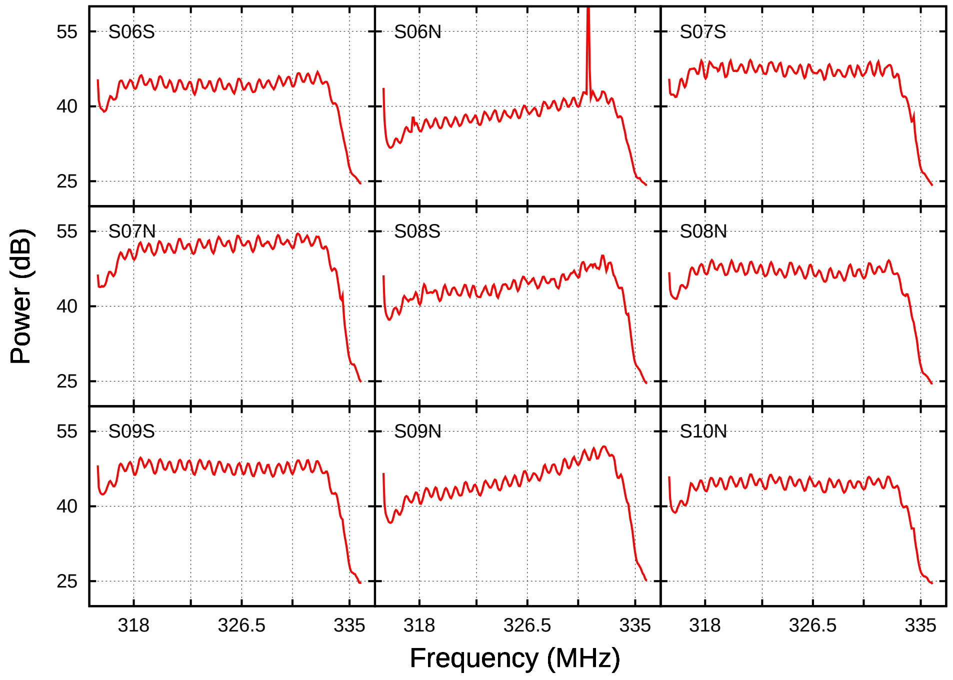


Figure 4.5: *Typical bandpasses of antenna elements from the upgraded ORT. The ripples seen are inherent to the anti-aliasing bandpass SAW filter. These effects can be calibrated to a large extent due to their temporal stability. The spectral line in the S06N antenna bandpass is due to the combining of the synchronizing FSK signal with the sky signal.*

Thus, the calibration of an interferometric array recovers true visibilities from the observed ones, which can be corrupted by both instrumental and environmental effects, and refers to estimating the per sensor element complex gain as a function of time, sky position or frequency. These estimations are subsequently applied to the observed data as corrections, in order to increase the sensitivity of the instrument. Thus, the calibration consists of obtaining a per-sensor element broadband phase and delay contribution due to the instrumentation. The errors due to the latter are expected to start dominating, once wider fields of view are imaged. This can be summarized by the following equation [53]:

$$V_{ij}^{obs} = V_{ij}^{true} G_i G_j^* G_{ij} + c_{ij} + e_{ij} \tag{4.2}$$

where, for an array with N sensors, V_{ij} represents the complex output of the interferometer formed by sensors i & j, G_i is the total complex gain for telescope

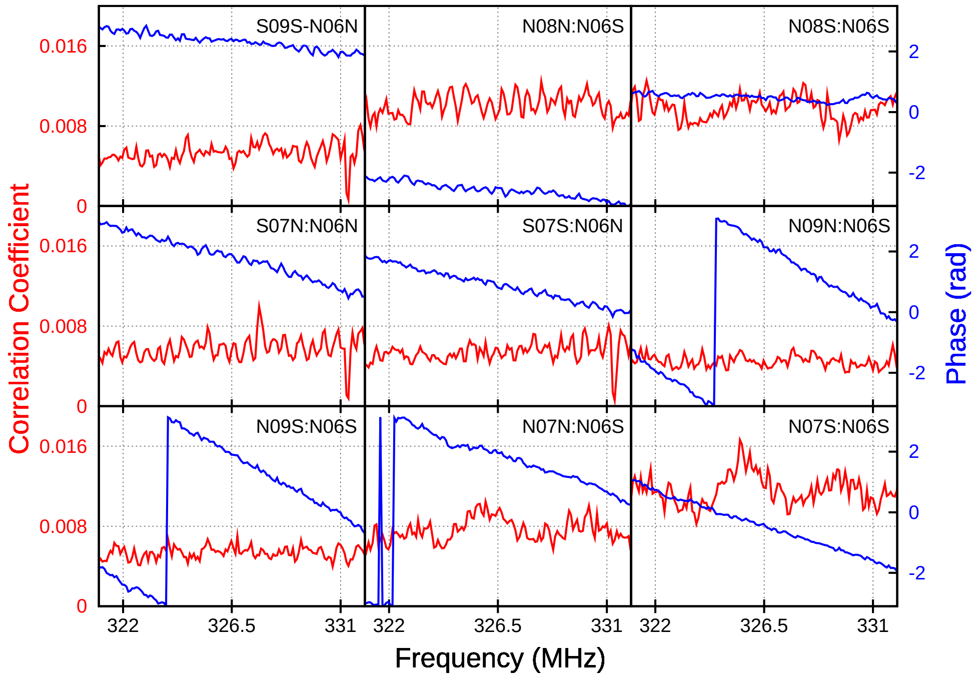


Figure 4.6: *Cross power spectrum of baselines ranging in length from 23 m to 345 m. The correlation coefficients on the source Her. A from different baselines, after an integration of* ~ 100 *ms are seen to vary due to the absence of an AGC, which is implemented in software.*

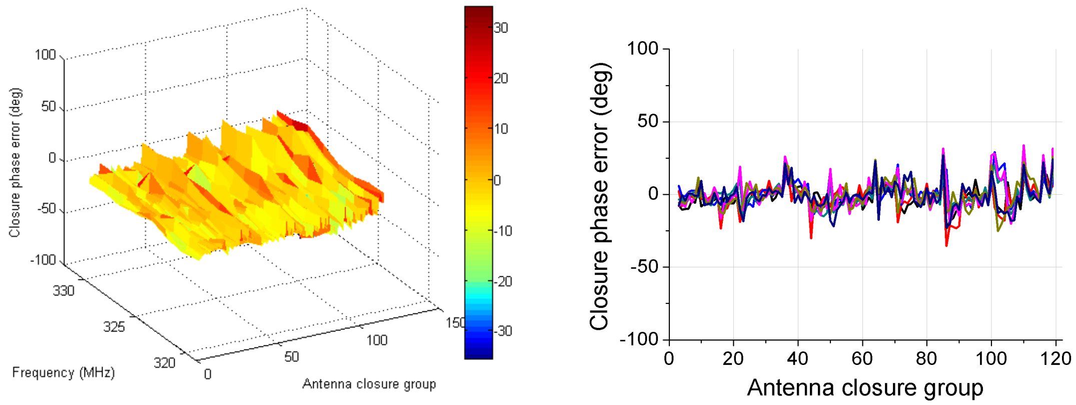


Figure 4.7: *Closure phase errors being used to identify deviant antennas within the recorded dataset. Such antennas have a closure error significantly larger than that of other groups, and can thus be isolated.*

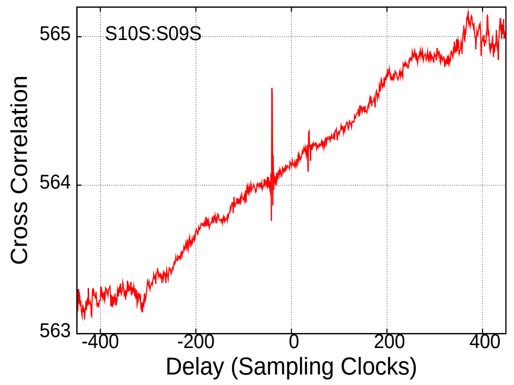


Figure 4.8: *Plot showing the estimation of sample level delay between the antennas from a representative baseline. The cross correlation between the two antennas is seen to peak at the delay offset corresponding to the actual delay between them.*

i, incorporating amplitude and phase errors due to the sensor, G_{ij} represents the baseline dependent, complex gain, after all known corrections are applied, c_{ij} is an additive error, and e_{ij} is the thermal noise.

Calibration can be achieved by carrying out dedicated observations of calibrator sources (which have known and invariable position, spectrum and flux) close to the field under study, with the per-sensor complex gains usually determined before and after carrying out the observation. This method assumes that the gains G_i, G_j and G_{ij} are fairly constant with time, and over different (nearby) parts of the sky (standard calibration).

Since correlation maintains the phase difference between a pair of elements, the phased array response can be created post-facto in a correlation receiver. This is possible, provided the differential delays are compensated before correlation, to a temporal resolution better than the inverse bandwidth of the spectral resolution of each data stream, and the sensor phases are calibrated before forming a phased array map.

Before elaborating on the details of the calibration implementation, we briefly discuss some aspects of calibration specific to the ORT upgrade.

4.5.1 Delay and phase

In the upgraded system, which has minimal analog components, the variable instrumental delays are expected to be caused only due to the analog conditioning subsystem, and possibly due to variable drifts in differential sampling clock phase between elements. A μs level delay can arise in each path due to the SAW filters installed, but the differential delays are expected to be minimal, as all filters were obtained from the same production batch. In addition, the fibre paths from the signal clusters to the center are not of equal length, and can vary by upto 100 m between the ends of the array. This can add a delay of upto hundreds of nanoseconds (or several tens of clock cycles) between signal clusters. Once the signals arrive at the center, no further relative delays are expected to be added.

As per the Fourier Transform delay to phase relationship [49], any uncompensated delay between two datastreams gives rise to a frequency dependent phase, leading to the generation of a phase-ramp across the sampled band. Further, a constant phase offset (broadband phase) to the band is expected, due to various effects like harmonic sampling.

Thus, initial delays between antennas, to the level of a few sampling clock ticks, are extracted by first carrying out observations of strong calibrator sources at low declinations, and then detecting the peak in the full-band time correlation between all antennas and the reference antenna (as shown in Figure 4.8). These are corrected by delaying each antenna's data stream in integer sampling clock units, which increases correlation, and reduces the phase winding per spectral channel in the cross power spectrum. It may be noted that carrying out delay corrections in base-band on a harmonically sampled signal leads to the generation of a constant cumulative phase accumulated by every unit of delay introduced at base-band, which itself is given by the ratio of phase center frequency (326.5 MHz) and the sampling frequency. Since this phase is delay dependent, it varies for different baselines. Once the coarse delays were estimated, they were accommodated in a system level table maintained on the software correlator platform, from which the table driven fixed delay application process operates.

Delays finer than a sampling clock unit per baseline, can be estimated using the slopes on the phase ramps observed. These are then added to the integer delays already estimated, to obtain a per-baseline delay. This delay, for every baseline, is then collapsed into per antenna delays using ANTSOL, and recorded into a system file, from where they can be applied as haedware based corrections for subsequent observations. These fractional sampling clock delay corrections are shown for a representative baseline in Figure 4.9.

The calibrated visibilities for a single sensor cluster are depicted in Figure 4.10.

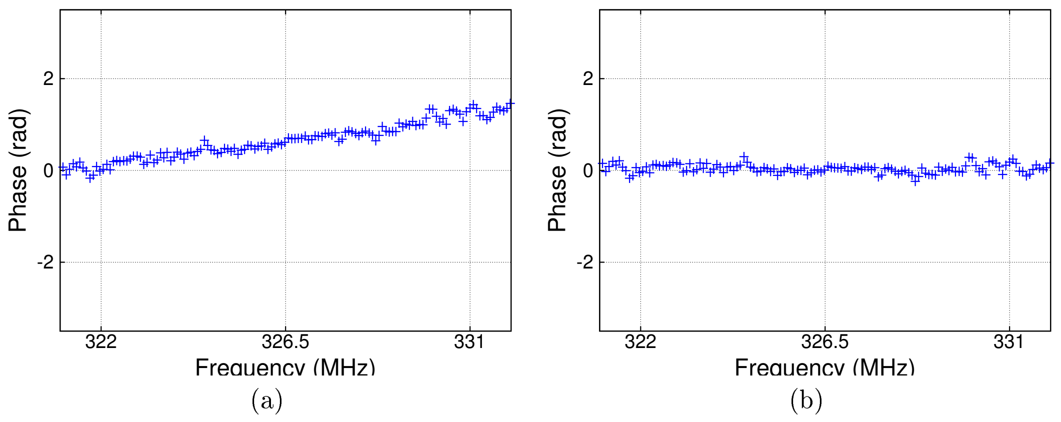


Figure 4.9: *Plots showing the phase of a representative baseline, (a) before and (b) after a fractional sampling clock delay has been applied.*

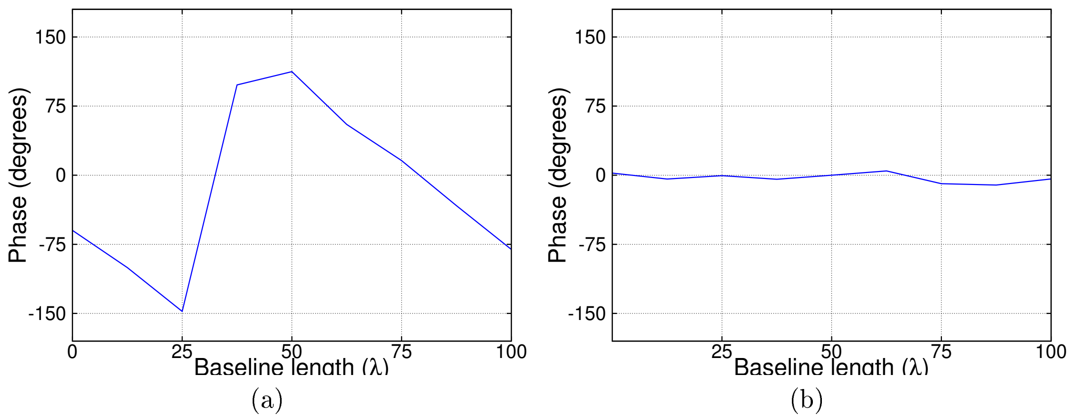


Figure 4.10: *Plots showing the phase of visibilities across the telescope array, (a) before and (b) after calibrating with Her. A. The spatial extent is ∼115 m of the ORT array, corresponding to 1 sensor cluster (10 half modules).*

4.5.2 Redundancy based calibration

The redundancy baseline calibration technique determines antenna based complex corrections directly from the data, without making assumptions about the sky brightness distribution. This is possible when an array configuration results in the same baseline vector (i.e, any pair of sensors having the same orientation and spacing to a source) occurring more than once. Further, an equispaced linear array (such as the ORT) results in a large number of such redundant baselines, which can be used to estimate the true visibilities more accurately. By taking the difference between the estimation of true and observed visibilities, the per-sensor complex correction factors can be determined, and subsequently applied to observations. For this approach, no separate calibrator observations need to be carried out, due to the simultaneous redundant sky observations available, provided enough SNR for comparing baselines is available. It may be noted that redundancy calibration will result in a loss of the actual absolute flux, as well as the absolute position (due to an arbitrary, undetermined, linear phase slope over the array). These can usually be countered by referencing the observations to other apriori information. Further, each redundant baseline is weighed by the square of the SNR, in order to suppress the effect of bad baselines. Redundancy calibration has been implemented using a linear least squares approximation, which has an analytical solution [53]. Here, the amplitudes and phases of the per sensor complex gain are separately solved for, by first expressing them as complex exponentials, followed by linearizing the gain equations, by taking their logrithmic form. The gains and phases of the calibration equation, can then be separately solved. Thus, if $V \equiv \exp(\upsilon + i\psi)$and $G \equiv \exp(g + i\phi)$, the separated gain and phase equations are:

$$\upsilon_{ij}^{obs} = \upsilon_{k(i,j)}^{true} + g_i + g_j + a_{ij} \tag{4.3}$$

$$\psi_{ij}^{obs} = \psi_{k(i,j)}^{true} + \phi_i - \phi_j + b_{ij} \tag{4.4}$$

where, $k(i,j)$ is a correspondence which maps a baseline formed by the antennas i and j, to the corresponding redundant baseline.

For the ORT 40-element prototype array, 38 redundant baselines are available in the full correlation output, out of the full set of 780 baselines. The gain equation matrix formulation consists of the unknown vector of 40 sensor gains and 38 true visibility gains, while an additional constraint on the absolute flux level is specified via the condition $\sum g_i = 0$. The phase equation matrix is similar, and the absolute telescope phase is constrained by specifying that the average phase for all sensors is zero, $\sum \phi_i = 0$. The input to the calibration routine is the normalized cross power spectra from a clean observation stretch, with the full $\sim$ 18 MHz band available as four averaged channels of 4.5 MHz, and a temporal averaging of $\sim$ 1 sec. A separate calibration solution for every channel is generated, and written into a calibration solution file, which can then be used to apply the corrections onto the observed data.

4.6 Formation of a 1-D map

Once the array has been calibrated, the complete "v" coverage of the observed field from the ORT enables the creation of any number of phased array beams in different directions, resulting in a 1-D map. For an equispaced, linear array like the ORT 40-element array, this can be carried out by taking a spatial Fourier Transform of the visibilities across different array spacings. These spacings are available to us in the form of the visibility function for every non-redundant baseline from the array. Thus, once calibrated complex visibilities are available, they are arranged into non-redundant groups. Subsequently, visibilities from corresponding spectral channels from each group are arranged along the spacing axis. The negative spacing axis is populated with the complex conjugates of the visibilities, while the zero-spacing is filled with a complex 0. In order to create an oversampling of beams, the visibilities are then zero-padded to a number convenient for Fourier Transforming.

The FFT of spectral visibilities from every non-redundant baseline thus arranged, results in the formation of an estimate of the sky brightness in N different directions, where N is the input visibility vector size. This results in the creation of a 1-D map with a resolution corresponding to the full array extent (0.1^o), and covering the available field of view of $\sim 4^o$. The created beams can then be averaged spectrally, in order to increase the sensitivity of the observation.

4.7 Discussion

Though our correlator has been implemented for 40 ORT elements, scaling it up to a higher number of elements is possible essentially due to the data transposition carried out by the NSPS data pooler element. A replication of such data pooler elements can then cater to the larger number of elements. Further, the required scaling of the computing can be carried out by the addition of Level-0 nodes, each of which cater to a smaller timeslice.

In our approach, an optimized, integer, pair FFT was implemented, instead of using popular vectorized FFT libraries, e.g., fftw3 and Intel's IPP FFT. Fourier Transforming the incoming streaming data (which is quantized to a 4-bit level), using such libraries exceeds the precision requirements of our streaming correlator application. In addition, their large float/double word-size makes them inefficient from the memory perspective for our use. This is because these libraries usually operate on single precision floating point input words, (8x larger than our input word-size of 4 bits), and generate output in floating point format. Though the IPP FFT can operate on 16-bit integer input data and generate 16-bit integer outputs, the internal processing occurs in the single precision float format, thereby reducing its efficiency. Further, carrying out a pair FFT using library implementations makes the following stage of

XMAC highly inefficient, due to their standard output formats.

In addition, for the XMAC implementation, we have used vectorized multiply and add instructions, instead of *Fused-Multiply-Add* (FMA) instructions, increasingly available on commodity processors, (e.g., MMX `pmaddwd`, or the SSE `dpps`). These FMA instructions can be used to efficiently implement vectored complex multiplications (for spectral correlations) in a single instruction, and thus are more efficient than carrying out multiplication and addition operations individually. However, in the initial stages of the correlation, the 4-bit quantization of the incoming data does not warrant the precision available from floats, or even 32-bit integers. This precludes the use of these FMA instructions, which currently are available only for floats or 32-bit integers. Hence, in our approach, we carry out the accumulation using 16-bit complex integers over an STA. Further, this approach has a lower memory footprint for the data intensive part of the XMAC.

The dynamic range of our correlator can be increased with companding by a non-linear mapping of data from the ADCs to 4-bits, via Look-Up-Tables (LUTs). This reduces the I/O bandwidth requirements of the NSPS, while maintaining the full dynamic range available to the ADC. Further, the decoding required due to companding can be accommodated in our implementation as a first preprocessing step, which can be merged with the nibble to byte expansion in stage-0 of the FFT.

Current processors have an increasing number of cores with larger vector units, and richer ISAs, e.g., the Intel SandyBridge processor micro-architecture (which has upto 6 cores/CPU, 256-bit vectored registers and a new ISA called the Advanced Vector Extensions). Also, the increased bandwidth requirements of these processors are being matched by newer I/O architectures. This trend is directly relevant to our correlator design due to the ability of parceling independent jobs to multiple cores, and reducing the data dependency (hence communication) between them.

Table 4.8 depicts the frequency of occurrence of the various I/O and compute instructions in our correlator implementation. It highlights the importance of the approach adopted by us for the software correlator implementation, namely, minimizing memory accesses via optimally laying out of the streaming data.

4.8 Conclusions

In this chapter, we have proposed a hybrid software/hardware approach for carrying out streaming correlation, based on the NSPS architecture. Here, while as-

Vector Instruction	Extract +Shuffle	FFT St.0	Shuffle +Twid. mult.	FFT St.1	Pair Extract +Shuffle	XMAC	Total
`movdqa`[1]	9920	4000	5120	7680	5120	7028	38868
`movdqa`[2]	3760	1440	3840	1600	1280	6240	18160
`pmul`	0	640	0	0	0	12480	13120
`pmulhrsw`	0	0	5120	640	0	0	5760
`paddw/psubw`	0	4240	0	8480	2560	6240	21520
`phaddw`	0	0	0	0	0	5460	5460
`pmovsxwd`	0	0	0	0	0	0	0
`pand`	5120	0	0	0	0	0	5120
`psllw/psrlw`	2560	320	1280	0	0	0	4160
`punpck`	2560	640	0	0	0	0	3200
`packsswb`	0	320	0	0	0	0	320
Total	23920	11600	15360	18400	8960	37448	

Table 4.8: *Frequency of occurrence of I/O and compute instructions in the optimized FFT and XMAC implementations for the major functional components of the NSPS based software correlator. The numbers refer to those needed for correlating two packets containing all 40 data streams, with 512 samples of each antenna element within.*

pects of data routing, preprocessing and synchronization have been moved to custom hardware, the estimation of the correlation coefficients is carried out in commodity processors. This approach results in decrease of development time of the correlator, and an increase in its efficiency, flexibility and reconfigurability. An implementation of this approach for a 40-element hybrid software spectral correlator was also presented. Here, we have demonstrated that this combination of custom and commodity hardware results in an efficient realization, which retains the best qualities of both software and hardware based correlators.

In our approach, the NSPS custom hardware utilizes memory based switches distributed over the signal processing tree for carrying out the data dispersion, and generates packets containing regrouped data from all sensor elements. This approach makes our correlator scalable.

We show that our vectorized, integer FFT implementation is a factor of ∼20x faster than that of an implementation using Intel's optimized vector FFT library. Further, our chosen data layout, carefully optimized for the target processor architecture, gives us a speedup of another factor of ∼4x in the cross multiply. Thus, our full software correlator obtains a speedup of a factor ∼3.5x, allowing a pair of cores in the target hardware to correlate data from 40 sensor elements with an analog bandwidth of ∼20 MHz. Thus, our hybrid approach for software correlation shows that it is possible to implement medium sized correlators in optimized software on commodity processors, by offloading data routing and shuffling onto hardware.

Chapter 5

New approaches for Visualization and Editing of Interferometric Data

5.1 Introduction

In this chapter, we focus on some aspects of data visualization, editing and calibration which we consider to be particularly important for observations at low frequencies. Their importance stems from the ever increasing presence of Radio Frequency Interference (RFI) at low frequencies and the rapid phase fluctuations suffered by incoming signals due to the ionosphere. Any attempt to minimize the biases in measured visibilities because of these factors, requires a sampling of visibilities much faster than the rate needed by the changing geometrical effects during an observation. Though preliminary, the results presented in this chapter already suggest the need for over-sampling the visibilities by a factor of 100 or more, in typical stages of pre-processing and calibration before they are further averaged to suit the deconvolution/image processing packages. In particular, we elaborate upon some new empirical approaches and software tools developed by us, which are useful for enhancing the *reliability* and sensitivity of low frequency observations. The primary criterion employed by us depends on the effective segregation of visibilities into deviant and concordant groups, possibly in real time.

We first demonstrate that an attempt to reduce the systematic biases resulting from low level RFI on the observed dataset requires high temporal and spectral resolutions which, in turn, allow us to obtain handles on different kinds of distortions appearing in the data. Such distortions can occur at very different characteristic time and frequency scales, each requiring possibly a different approach to minimize its effects on images derived from the observations. However, the data rates resulting from such a high resolution requirement can be very large and often complicate analysis due to the inherent problems associated with handling large data volumes. Towards this end, we present some software tools developed by us which allow rapid visualization and editing of high bandwidth data, and demonstrate them on test observations carried out using the GMRT. It may be noted that the architecture of these tools not only allows for their easy interfacing to the existing data collection stages of other telescopes, but also makes them easily adaptable for real time applications. Further, the tool implements some of the observing strategies developed by us, which allow for the segregation of recorded data in an automated fashion.

In addition, we also examine the feasibility of setting up co-located facilities for estimating phase distortions caused by non-isoplanaticity of the ionosphere using signals from geosynchronous navigation satellites (GEO). Towards this end, we present some preliminary results from an L-band interferometer set up at the GMRT site, using the fibre-optic network of the GMRT. Although the setup was initially planned as part of a low frequency transit survey to be carried out with the GMRT, the project was given up subsequently when the operationalizing of the space segment for routine availability of such satellite signals was delayed beyond the time frame of the present thesis project. Details of the survey criteria are given in [54].

5.2 High temporal and spectral resolution observations

The visibilities generated by the GMRT after fringe stopping need to be sampled only at intervals of tens of seconds, to ensure that changes in baseline due to earth-rotation are within the Nyquist sampling criteria for image formation. However, the results of test observations carried out by us at 245 MHz suggest that the nature of RFI in contemporary operating environments at GMRT can call for high temporal and spectral resolutions to avoid otherwise undetectable systematic biases in visibilities.

In our observations, the GMRT hardware correlator was used with the highest temporal and spectral resolution feasible with the hardware. This corresponds to a temporal resolution of 0.131 sec (corresponding to the rigidly fixed Short Term Accumulation in the hardware), and spectral resolution (62.5 KHz) corresponding to 256 channels within the observed bandwidth of 16 MHz. Since these are not part of standard observing procedures, several modifications were made by us to the

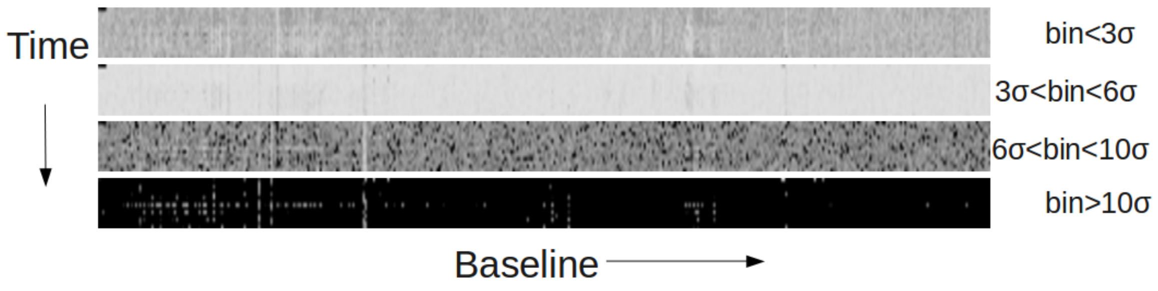


Figure 5.1: *Time evolution of threshold based visibility flagging information in the form of a histogram. The larger than expected bin counts in the* 6σ *to* 10σ *range reveal low level distortions to visibility amplitudes.*

real-time data acquisition chain of the GMRT in order to support the high data rates of $\sim$ 50 GB/hr generated by the high resolution requirements. Interestingly, one such modification made by us relates to the explicit management of occasional racing conditions between the various data writers and readers, created due to the presence of several simultaneous threads of execution on the multiple cores of the modern CPUs in use by the GMRT. This might not have been a problem with earlier data acquisition computers, where the process scheduler of the operating system was required to manage only a single core of processing.

An interesting perspective of the effect of RFI in the recorded data is provided by Figure 5.1, which depicts the time evolution of the histogram of threshold based flagging, pictorially. Here, flagging information from all 435 baselines of the GMRT for a chosen time stretch, and a chosen spectral channel, are depicted along the columns, or x-axis, while their time evolution is depicted along the rows, or y-axis of the figure. Individual bins of the histogram, visible as sub-panels, correspond to the thresholds applied on the amplitudes of the observed visibilities in units of σ (standard deviation) of the visibility amplitudes, estimated using data corresponding to a time stretch of $\sim$ 8 secs. If a particular visibility amplitude within the chosen time stretch exceeds the populations' σ by a certain amount, the corresponding histogram bin count is increased. A separate histogram is generated for every time stretch of $\sim$ 8 sec. It may be noted that the standard GMRT visibility temporal resolution is $\sim$ 16 secs. The entire information is then presented as a grey-scale image in the standard pgm ascii image format, with a smaller bin count corresponding to darker greys.

An analysis of this data reveals that for 70-80% of the 8-sec blocks used for generating the histograms, errors between 6σ and 10σ (depicted in the third sub-panel from the top, in Figure 5.1) are present for 10-20% of the time, corresponding to $<$

2 secs (not necessarily contiguous) within an 8-second block. It is important to note that such deviations remain undetected by conventional integration/flagging mechanisms operating on the data integrated along either the time or frequency dimension, although they affect the reliability of the visibilities significantly. Such deviations should be discarded while estimating mean visibility (over larger time intervals of tens of seconds) needed to be passed on to image processing software. These low-level and random sources of corruption can appear due to the local RFI environment, or because of the effect of the intervening media (especially the ionosphere) on the observed visibilities.

Thus, the above analysis emphasises the need to carry out high temporal and spectral resolution observations, and apply an initial level of threshold based flagging in order to segregate deviant visibilities before fusing the data via integrations. However, such an operation on large volumes of data needs to be automated in order to be feasible. The preprocessed data can then be passed on to regular data analysis tools. To this end, we have developed a large data volume visualizing tool, which incorporates automated flagging using a variety of approaches, and is described next.

5.3 Tools for large volume visibility visualization and editing

Our observational approach generates large volumes of data (amounting to several hundreds of GB for typical observations), due to high temporal and spectral resolutions of these observations. These, in turn, are required for effective visibility characterization and preprocessing, before calibration or imaging techniques available in standard data analysis packages can be applied. However, the conventional observing schemes and the analysis tools available put serious constraints on the efficiency with which analysis of this high volume data can be carried out. Hence, we have developed new software tools for editing, calibration and preprocessing of visibilities, before standard image processing software can be used.

5.3.1 Visibility visualizer

A QT class-library based tool was developed as part of a fast, large volume visibility visualization tool set to assist the display and editing of data, and to meet several requirements of flagging and calibration. Here, visibilities and their attributes were treated as a time series of a vector and presented as images. It acts as a top level tool for both visualizing and editing incoming visibilities, as well as for carrying out novel pre-processing techniques on large data sets.

The developed tool operates using a simplified version of the native GMRT data recording format, in order to conserve run-time memory. Due to the architecture of the GMRT real-time acquisition software, which provides shared memory based public interfaces and UNIX socket based parallel data streams, the tool can directly interface with the GMRT real-time data pipeline, allowing for real-time visualization of GMRT visibilities. After carrying out appropriate visualization and pre-processing, it can generate FITS format files, in order to allow standard data analysis tools like AIPS to be used for further processing.

The tool primarily utilizes the fast image rendering capabilities of QT (using the bitBLT procedure) in order to display visibility amplitudes and phases as separate images, which can be updated rapidly.

Figure 5.2 shows an example of the display of visibility amplitudes. In these figures, each panel represents an antenna or a baseline. Each row of pixels corresponds to one time slice of the available spectral channels (256 in the case of the depicted dataset), with each pixel intensity representing the desired visibility attribute. Each frame corresponds to $\sim$ 1 minute of visibility data at 0.13 sec resolution, and can be rapidly refreshed for all 435 GMRT baselines. These are then shown together in a scrollable window. The time evolution of different aspects of the data can be visualized using our tool, e.g., visibility amplitudes and phases etc.

Basic operations supported by our tool include display of visibilities and processed outputs as images, reformatting to various file formats, extraction/reordering and recording of subsets etc., with a baseline being the operational unit. A unified callback interface exists which simplifies the plugging-in of modules for different operations like correlation between different days, FFT along any axis etc. In our implementation, any data selected by the user is extracted by a file reader object, pooled by a data formatter object and processed by a centralized processor object, which interfaces with a per baseline display object. A central controller manages the graphical user interface (GUI) and sequences all the desired operations. It is worth noting that this tool is ideal for the display of large volumes of data - each frame in the Figure 5.2 corresponds to about a Megabyte of visibility data which can be refreshed simultaneously for upto a dozen frames at 100 Hz. Each of the frames in Figure 5.2, which display visibilities, includes frames of 256 x 400 pixels.

Our tool includes a facility for multi-level data flagging based on user defined thresholds, specified in rms units. The threshold limits are determined across spectral channels, as well as over a fixed time interval. The flagging information is embedded

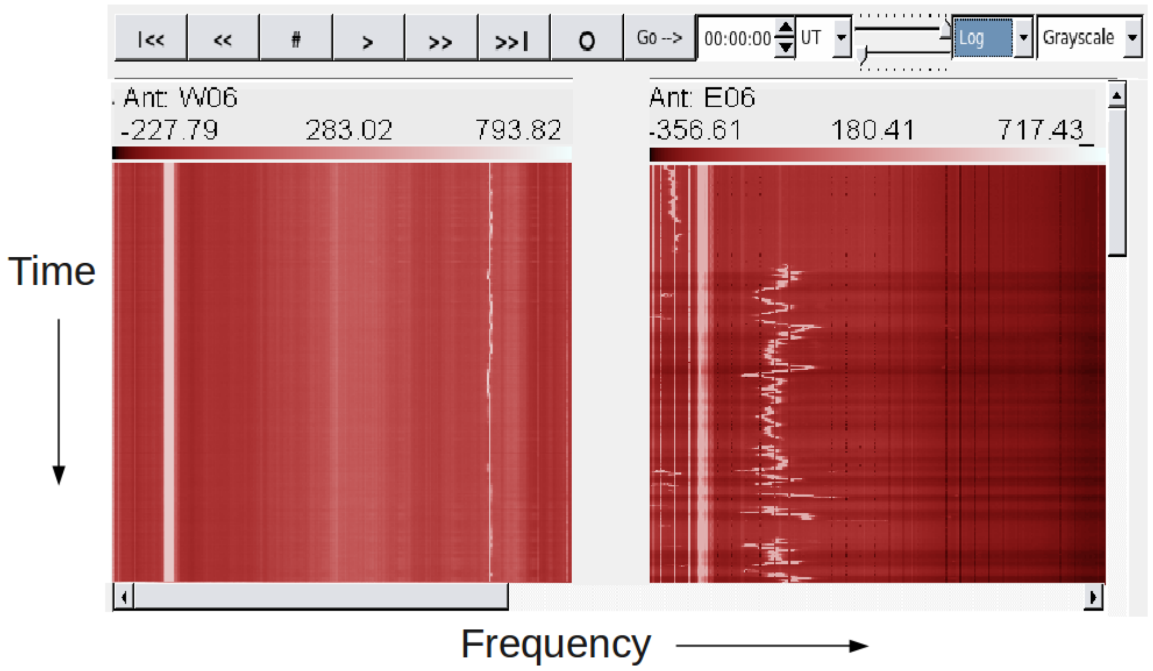


Figure 5.2: *Self correlation counts for two GMRT antennas displayed using the visibility visualizer. Evidence of intermittent RFI in the panel on the right, in addition to continuous narrow band RFI in both frames can be clearly seen. Each panel depicts about a minute of data at 0.131 sec resolution per pixel.*

within the dataset, with the visualizer allowing the displayed pixels to either represent the flagged data, or the flagging levels chosen. Flagged data are stored in the FITS format with their weight set to -1, which implies ignoring the data. An example of the tool carrying out such a flagging is depicted in Figure 5.3, where the different flagging levels are shown in different colours. For instance, pixels coloured pink correspond to visibilities with amplitude deviations between $6 - 10\sigma$, while pixels in red denote deviations above 10σ, with the visibility information being replaced by the chosen color, in case the visibility needs to be flagged. As explained above, the tool can then generate a flagged dataset from the incoming data.

The tool also allows for a user-defined mapping between the observed visibilities and the displayed pixels' color. Such an approach allows for an intuitive interface between the data and the user, as the manipulation of the color index table helps in quickly segregating data corresponding to different power ranges.

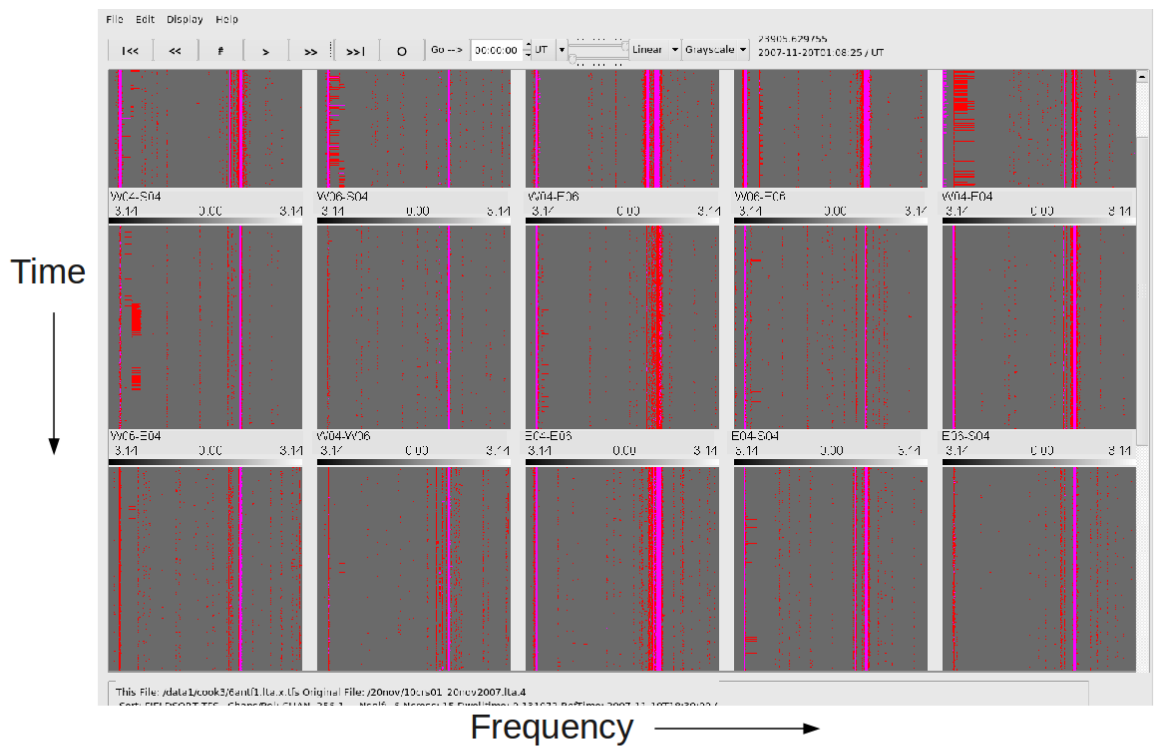


Figure 5.3: *Real-time display of automatically flagged visibilities from different baselines, based on amplitude thresholds. The flagged data are shown as pixels coloured with the threshold bin into which the amplitude falls.*

5.4 Collation of multiple observations for high sensitivity

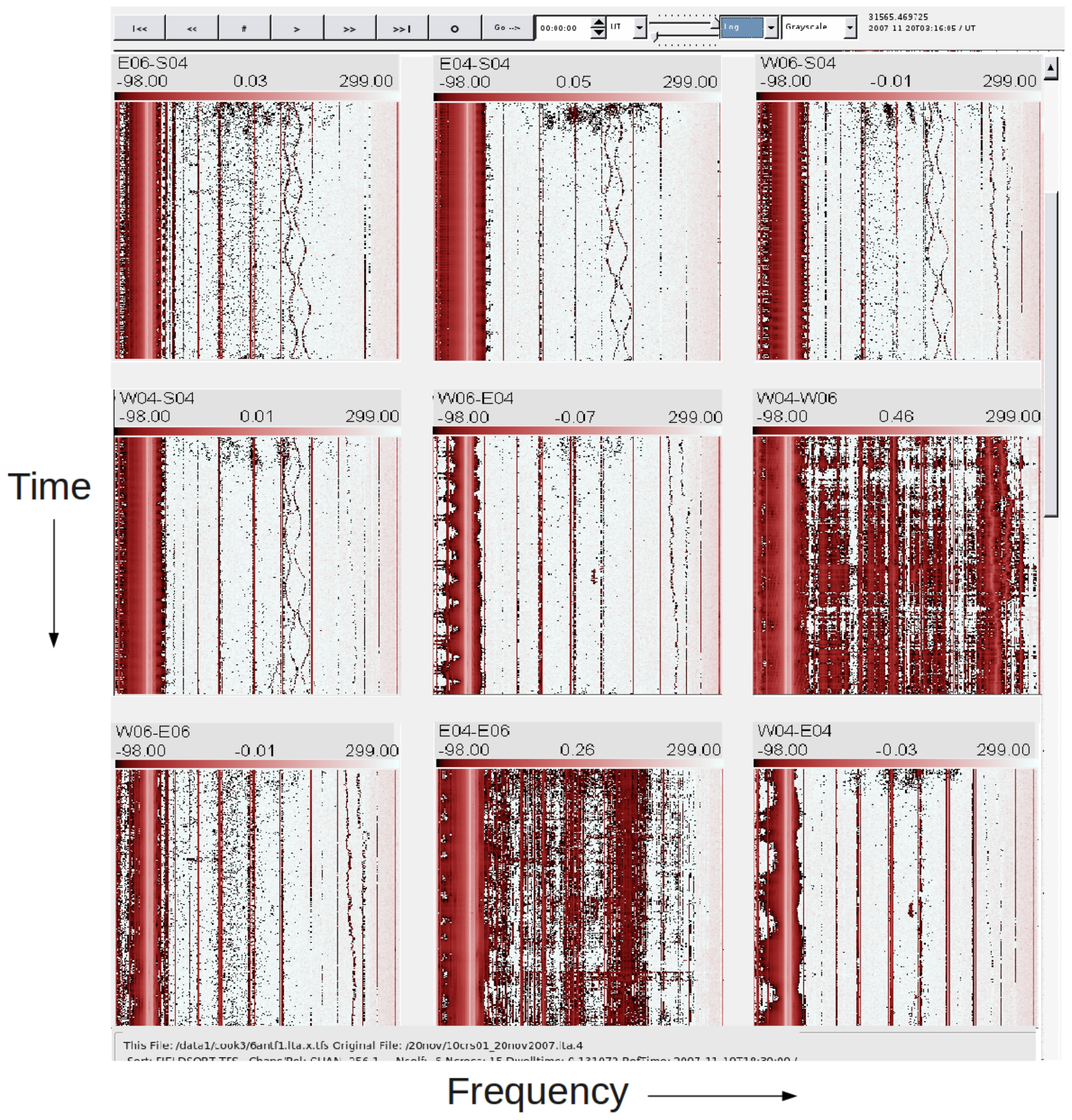
Requirements of high sensitivity and/or studies of variability at many timescales often call for multiple observing sessions under nominally identical instrumental set up. For instance, one of the primary motivators of the upgrade being planned for the ORT is to investigate large scale structures and HI mass fluctuations at redshift close to 3.2 as well as monitoring the sky for transients or variability at different time scales. While ORT is an equatorially mounted radio telescope offering identical components of visibility to be estimated by tracking a given region of sky for upto 9 hours in a day, a telescope like the GMRT restricts integration for a given visibility component to a few instances of about a minute on a given day, depending on the baseline.

Deep observations often stretch to multiple sessions with identical (u,v) coverage. In some extreme cases like the cosmological investigations with the upgraded ORT, the estimated observing time may span many months - to get an equivalent of hundreds of hours of integration to look for any signature of Baryonic Acoustic oscillations (BAO) [8] at redshift close to 3. In fact, the BAO study is one of major motivators of the upgrade of ORT to a 264-element programmable telescope. In all these cases, observations within a single observing session are expected to be edited using the criteria similar to those mentioned, before improving estimates by reasonable integration (typically a few seconds). It is then necessary to ensure that multiple observing sessions are normalized to correspond to common instrumental gain and edited to minimize systematic deviations between observations, before combining them to enhance the precision of estimate.

Our approach for collapsing data from multiple observing sessions is based on the cross-correlation of the time series representing visibilities corresponding to identical baselines on different days. Such an approach allows us to quickly identify low level deviations to observed visibilities by carrying out an inter-comparison of the supposedly identical visibilities from different sessions. The visibilities themselves are expected to remain identical from day to day, due to the unchanging nature of the field under observation. Thus, these inter-comparisons can be made by cross-correlating the observations from different sessions. Since the different observing sessions can have a gain, or time offset due to variations in the instrument response over a period of time, such a visibility-visibility cross-correlation can also determine the clock and gain offset between the two observing sessions. Since interference is unlikely to replicate on different days, this approach will be very effective to detect an event which is temporally or spectrally localized to one or a small subset of the observing sessions. This can be taken into account to make better estimates using longer integrations permitted by multiple observing sessions, and/or to detect localized events related to transient phenomena.

In order to examine the feasibility of such an approach, we carried out transit mode observations with the GMRT of identical LST ranges. As before, this observation was also carried out at the highest time and frequency resolution available from the GMRT hardware correlator. The visibility visualizer tool was then used to analyze the inter-day cross-correlations between visibilities corresponding to identical baselines. A view of the resulting correlations is depicted in Figure 5.4, where a higher correlation results in a color closer to white, while low correlations are mapped to darker colors.

In such a dataset, visibilities which are consistent across days show up with a higher correlation coefficient, while those affected by local distortions appear with lower correlations. Further, flagged visibilities are not considered for correlation. The output of such an analysis can be used to automatically flag visibilities from the multiple observing sessions, and the remaining, consistent visibilities can then be



This File: /data1/cook3/6antf1.lta.x.tfs Original File: /20nov/10crs01_20nov2007.lta.4

Figure 5.4: *Per channel, complex cross correlations of 235 MHz visibility time series over identical LST range on two different days. Regions of constant brightness over time show consistency between two days, while inconsistencies, due to low correlation, appear as dark spots. The upper panel shows this affect as a wavy pattern in all baselines with antenna S04. Baselines W04-W06 and E04-E06 show large inconsistencies from day to day.*

integrated in order to achieve the increased sensitivity.

5.5 Interdisciplinary approach towards ionospheric calibration

In a typical interferometric observation, a session can include periodic observations of calibration sources (which are sources with well modelled structures,close to the field of observation) to help calibrate slowly varying instrumental gains and phases. This is then improved during deconvolution using self-calibration techniques at minute level intervals to track more rapid variations due to atmospheric conditions [15]. An integration of about a minute is a balance between a minimal sensitivity to detect in-field departures from the model, and the timescale of phase variations. Such techniques get complicated at low frequencies because of the enhancement of path-length fluctuations, leading to contamination from fluctuations on scales much smaller than the primary beam at the ionospheric heights or the array extent. This requires the phase calibrator source to be within the same patch as the field of observation (i.e., in close proximity), or a separate calibration to be done for each patch within the primary beam. Thus, non-isoplanaticity of the ionosphere is a major limiting factor to the dynamic range of low frequency synthesis telescopes, resulting in the highest dynamic ranges achieved in telescopes like the GMRT at metre wavelengths falling far short of sensitivity limits. One approach to handling the calibration in such cases was suggested by [55]. For such a technique to be practical, it is necessary to get a reasonable initial estimate of ionospheric delays in the primary beams of antennas located in the array. Further, though self calibration can cater to rapid ionospheric fluctuations, it assumes isoplanaticity, and requires a field dominated by a compact, strong source. Further, it removes ionospheric phase errors only in the source's direction [56]. Carrying out simultaneous observations of the same field at two different frequencies (where higher frequency observational phases are referenced to the lower frequency visibility phases), requires instrumental support, which may not be available easily.

In addition, the L1 and L5 coherent navigation signals radiated by GPS satellites have also been used to model the phase screen between the observer and the satellite [57], by estimating the arrival time delay between these signals. However, such measurements of the phase fluctuations require the modelling of both small and large scale ionospheric TEC variations, which are not afforded by the $\sim$500 Km Ionospheric Pierce Points due to the GPS satellites [58].

The advent of Wide Area Augmentation Service (WAAS) in satellite navigation brings an interesting possibility of obtaining such estimates by a co-located interferometric set up, for observing navigation signals from such satellites. In India, the first such satellite is likely to become operational in October 2011 as part of GAGAN (GPS Aided Geo Augmentation to Navigation) [59] project to assist future aircraft landing. Several such satellites have been announced by the Indian Space Research Organization (ISRO) for this project, as well as related to the Indian Regional Navigation Satellite System [60], based on a set of geo-synchronous satellites. As a result of these projects, one expects a set of 10 geo-synchronous satellites by 2015, carrying WAAS signals visible throughout the Indian subcontinent. This will result in a unique opportunity for continuous estimation of arrival time differences of signals from these satellites at a set of antennas located, e.g., within the extent of a synthesis telescope like the GMRT. Such observations can result in direct estimates of differential ionospheric delays over the array extent, to be made throughout an observing session. These can then be collated to form reliable estimates of differential delays at the observing frequency, using the λ^2 [15] dependence of ionospheric delays. This makes a strong case for setting up an inter-disciplinary facility in a location like the GMRT to benefit both the satellite and the radio astronomy communities.

In the following paragraphs, we present some preliminary results from such a facility which was available in 2007. Both the described experiments related to L band signals from geostationary satellites. Among them, one had a payload carrying digital audio broadcast by Worldspace on Asiastar and the other (INMARSAT 4F1) had a navigation payload leased temporarily by ISRO for their tests related to the GAGAN project. In one experiment, dark fibres in the GMRT fibre optic network were used to carry the received RF directly into the receiver room, which housed the signal processing units commissioned by Raman Research Institute. The other experiment uses a unique feature of the GMRT feed arrangement, which points the L-band feed towards the sky when the 235 MHz feed is pointed towards the reflector.

The schematic of the setup used for the two experiments outlined in this section is depicted in Figure 5.5, where a satellite signal interferometer is shown co-located with a radio telescope.

Such a setup allows simultaneous observations to be carried out on the sky as well as the satellite, along its line of sight. Moreover, the arrangement requires a purely passive receiver setup. Further, the high SNR on satellite down-links makes it possible to estimate their visibility phase with high accuracy, and within a short integration period.

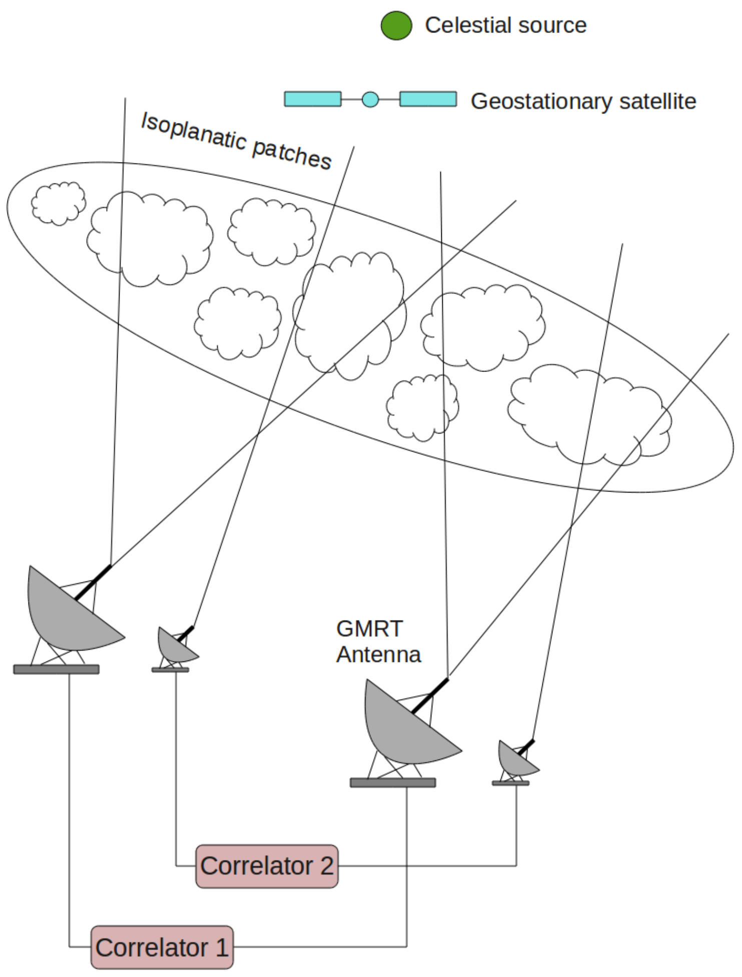


Figure 5.5: *Schematic showing a co-located satellite interferometer with an interferometric array.*

The estimates of the arrival time differences from multiple satellites can then be used to model variations of ionosphere in terms of the estimated delays suffered by the satellite signal (at its frequency of transmission) along various lines of sight. The resulting ionospheric model, created simultaneously and for the duration of the sky observation, can then be translated to the lower frequency of celestial observation, using the standard thin screen model of ionospheric delay variation with frequency [15], and applied to the observed visibilities as corrections.

It may be noted that this approach requires the ability to estimate the arrival time differences from satellites with high accuracy, due to the λ^2 dependency of the model delays. Thus, if the GEO satellites provide an L-band down-link, the frequency of celestial observations may be ∼10-20 times lower, leading to a requirement of a phase estimation $\sim$ 100 times more accurate. Further, multiple lines of sight through the atmosphere need to be available, in order to accurately model the ionosphere. An important aspect of this approach is the ability to continuously estimate the arrival time differences, which removes the ambiguity of multiple cycles of 2π in the phase measurements. Further, the thin screen ionospheric delay model allows for the translation of estimated *delays* from the higher frequency of observation, to the lower celestial frequency. Thus, the primary observable, (the instantaneous Cross Power phase on the satellite interferometer baselines), needs to be converted into delays, by accounting for every turnover of 2π in the phase measurements.

The perturbation of geosynchronous orbits due to geopotential and other effects like luni-solar attraction and solar radiation pressure, causes secular and periodic changes to their orbit elements. This translates to a relative motion of the satellite along the line of sight with respect to the earth, giving rise to a slowly varying (roughly diurnal sinusoid) differential Doppler component in the observed interferometric phase. However, since this systematic phase variation of the received signal due to the Doppler on the GEO satellite signal is an extremely smooth, diurnal sinusoid, it can be modelled to a very high degree of accuracy. The residual effect on the satellite phase (once external contributors have been removed by modelling), can be reliably attributed to the relatively rapidly varying effects due to ionosphere and a somewhat slower variation in the instrumental phases.

In this section, we describe the receiver infrastructure developed by us to carry out satellite interferometry, and results from two separate observations.

5.5.1 Satellite interferometry of WorldSpace satellite

The Asiastar satellite had a Worldspace digital audio transmission at 1492 MHz, which was used to carry out interferometry. The advantage of using signals from this Geosynchronous satellite was the inexpensive analog receiver system, consisting of a commercially available WorldSpace Yagi antenna with a built-in LNA. In view of the inherent signal strength coming from such satellites, the required bandwidths and integration times are well within simple processing capabilities of a normal workstation. However, the angular resolution which can be achieved by this method (which improves with decreasing wavelength) depends on the array extent measured in units of wavelength.

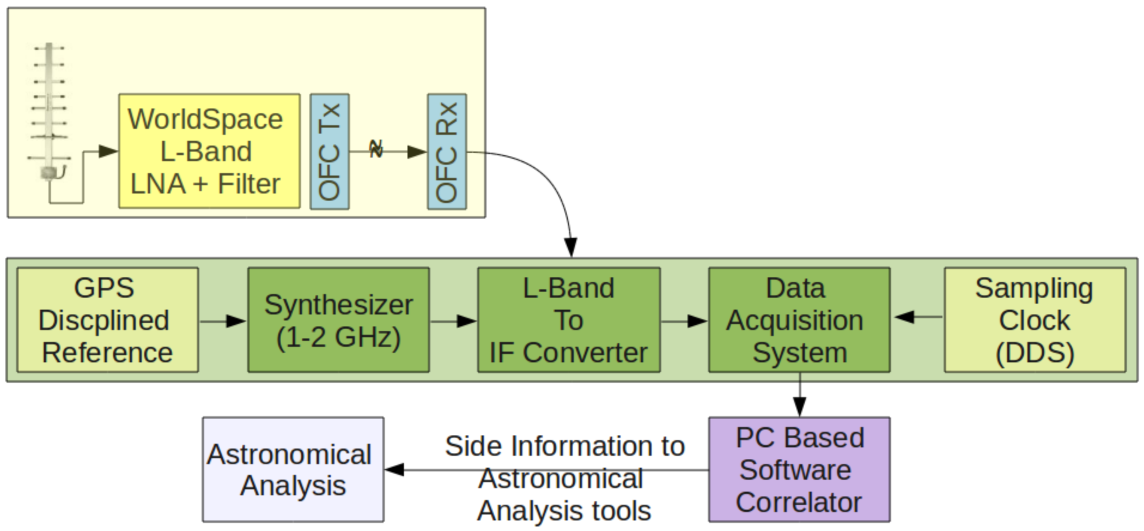


Figure 5.6: *Schematic view of the receiver infrastructure developed for carrying out satellite interferometry. The limits set by reference clock stabilities can be reached by cross-correlating a very small part of the received band, due to the SNR on the satellite signal.*

5.5.2 Experimental setup

In this experiment, we exploited the spare capacity of the GMRT fibre-optic network to obtain a long baseline L-band interferometer. The specific sites chosen, were those in the original plan of the GMRT (and hence with proper termination of fibre-optic cables) but where no GMRT antenna were installed. These two locations were used to position a pair of low-cost antennas (meant for receiving digital audio signals from Worldspace), effectively separated by about 11.2 Km.

The receiver setup is depicted in Figure 5.6. The received RF from every node of the interferometer was directly transmitted over the existing unused GMRT optical fibre network to a central location for further processing.

Once the signals were received at a central location, they were coherently down-converted to an Intermediate Frequency (IF) of 70 MHz, as shown in Figure 5.6. These signals were then digitized using custom made ADC boards, and the data acquired onto a commodity PC using custom made PCI Data Acquisition Cards. Subsequently, a software spectral correlator was implemented in the PC, and the arrival time differences were continuously estimated.

The slow relative motion of the satellite with respect to the Earth, results in fringes visible in the estimated cross-correlation, and an example of such an estimation is depicted in Figure 5.7. Here, the x-axis refers to the time offset from start of

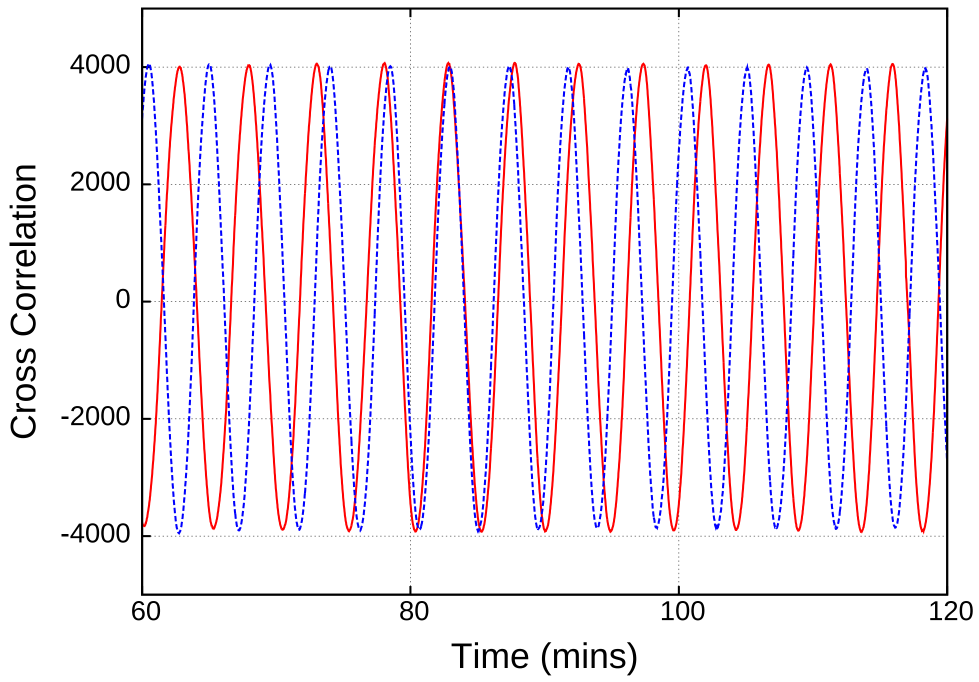


Figure 5.7: *Observed fringes from the WorldSpace satellite's L-band signal. The slow phase variation evident in the phase offset between the two sets of fringes observed* ~ 1*Hour apart is due to the Doppler on the satellite signal, due to its motion relative to the interferometer.*

observation, while the y-axis is the real part of the cross correlation. The figure shows two sets of fringes (in red and blue) from a one-hour stretch of data, with a one-hour gap between the two sets. These are typical interferometric fringes, in which a linear variation of path length difference is translated into a sinusoidal variation of the cross-correlation. By counting cycles and measuring the residual phase offset, one can infer that the time taken for a path length variation of 25 cycles is about 7220 secs, implying that the mean time for path length to vary by one wavelength (20.17 cm) was $7220/25 = 288.8$ secs. The SNR was adequate for achieving an accuracy of a fraction of degree in phase estimation, with a one-second integration. Thus, one can conveniently detect a tiny relative displacement (by a fraction of a millimetre) of the satellite along a direction parallel to the line joining the two antennas. The output demonstrates the high SNR on the phase variation of the satellite signal. More importantly, it can be seen that, due to a continuous measurement of the cross-correlation, the phase variation can be tracked over multiple cycles of 2π, thus leaving no ambiguity while converting the instantaneous phase to a delay.

Such a technique will be better explored by setting up a campus-wide network of satellite receivers within the Raman Research Institute [61].

5.5.3 Satellite interferometry of INMARSAT 4F1 satellite

In order to model a phase screen over an array the size of the GMRT, several satellites are required to be within the field of view. Further, to estimate arrival time differences from each satellite, phase variations of the Cross Power from each of the multiple satellites need to be tracked. In addition, since Geosynchronous satellites are fixed relative to an observer's local coordinates, the GMRT needs to be operated in *Transit* mode, in order to carry out such observations.

For the special case of 245 MHz observations with the GMRT, such a scenario of tracking the phase of multiple satellites is feasible, without the requirement of setting up an additional co-located interferometer. This is possible with the trading off of one polarization of the incoming sky signal from a few antennas involved in recording data from the satellite. Further, both the sky and the satellite signals can be simultaneously received and correlated using the existing GMRT receiver setup. This is explained in the following sections.

5.5.4 Simultaneous satellite and sky observations with GMRT

For carrying out observations with the GMRT in transit mode, one antenna in each of its three arms was dedicated to recording signals from a GEO satellite at L band. Further, the specific choice of the declination ensured that this satellite always remained within the field of view. This allowed us to have a continuous measurement of the arrival time differences of the satellite signals at these three dedicated antennas. Hence, high SNR data for estimates of instantaneous phase distortions (due to the sky above the array along three lines of sight), was obtained. The navigation signals pertaining to our observations were part of trial runs for a WAAS implementation, involving a broadcast of signals at 1.176 GHz (L5 band) from INMARSAT 4F1 – a geosynchronous satellite in an orbit inclined at 2.4^o, with the signal within the received band of the L-band feed of the GMRT.

5.5.4.1 Array configuration

The GMRT was operated as two sub-arrays, one corresponding to the antennas used for the dual satellite-sky observations, and the other corresponding to the rest

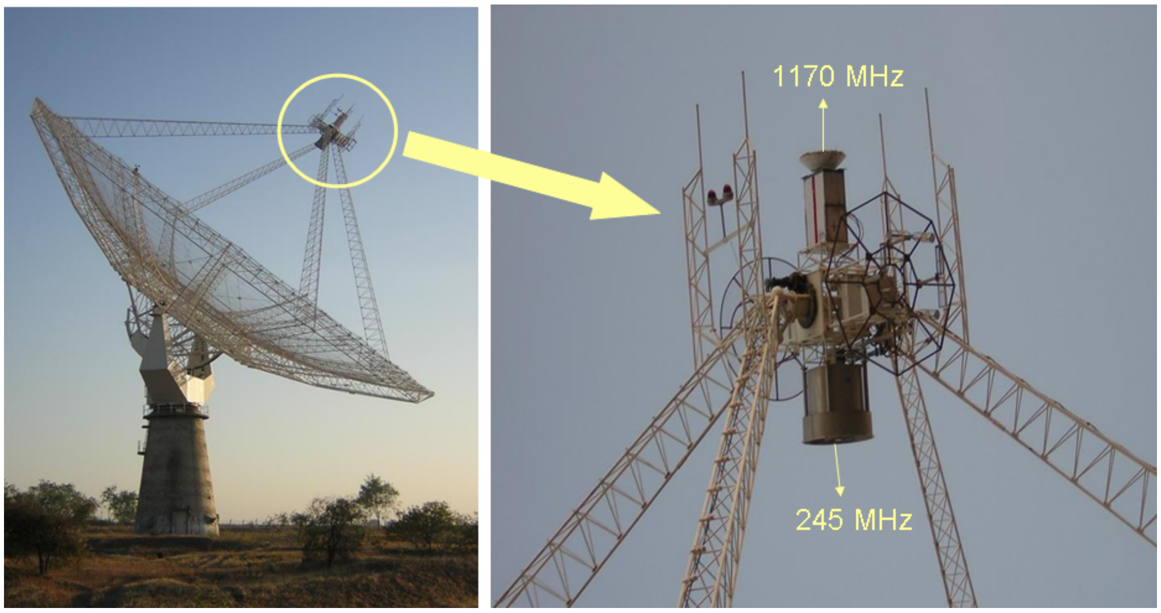


Figure 5.8: *Simultaneous satellite and sky observations with the GMRT, made possible due to the L-band feed looking into the sky (and towards the satellite) while the 245 MHz feed looks into the dish for regular observations. The receiver configuration of the GMRT allows for the received band from different feeds to be routed through the IF system, to the central receiver.*

of the antennas forming the array. While the navigation satellite signal at 1.17 GHz was received using the L-band feed of the GMRT antenna looking at the sky, the low frequency sky signal (235 MHz) was received from the feed looking into the dish, as depicted in Figure 5.8. This was possible since the GMRT antenna's gain was not required to be sensitive to the inherently strong satellite signals. Further, the loss of gain due to the circularly polarized WAAS signal being received from the linearly polarized GMRT L-band feed was not significant enough to affect the accuracy obtainable from the method, due to the high relative power of the received satellite signal.

Once the RF from both sky and satellite was received, the GMRT frontend was operated in a special dual frequency mode (enabled by the GMRT receiver) to simultaneously transmit the 235 MHz and 1.17 GHz signals in the two IF bands available for transmission of the IF from the GMRT antenna to the central receiver. These signals were then sampled and transported to the GMRT correlator system, where correlations between the satellite signals were formed.

It may be noted that all the dishes were pointed towards a nominal direction corresponding to the declination of the satellite in the middle of a one-hour observing session. This setup is displayed graphically in Figure 5.9.

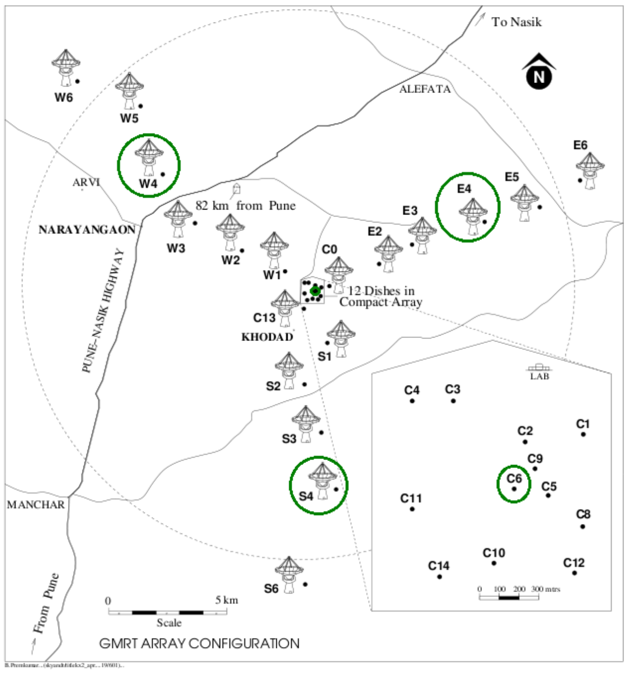


Figure 5.9: *GMRT array configuration during simultaneous satellite and sky test observations. Here, antennas used in the test observations, namely, W04, E04 and S04 are circled.*

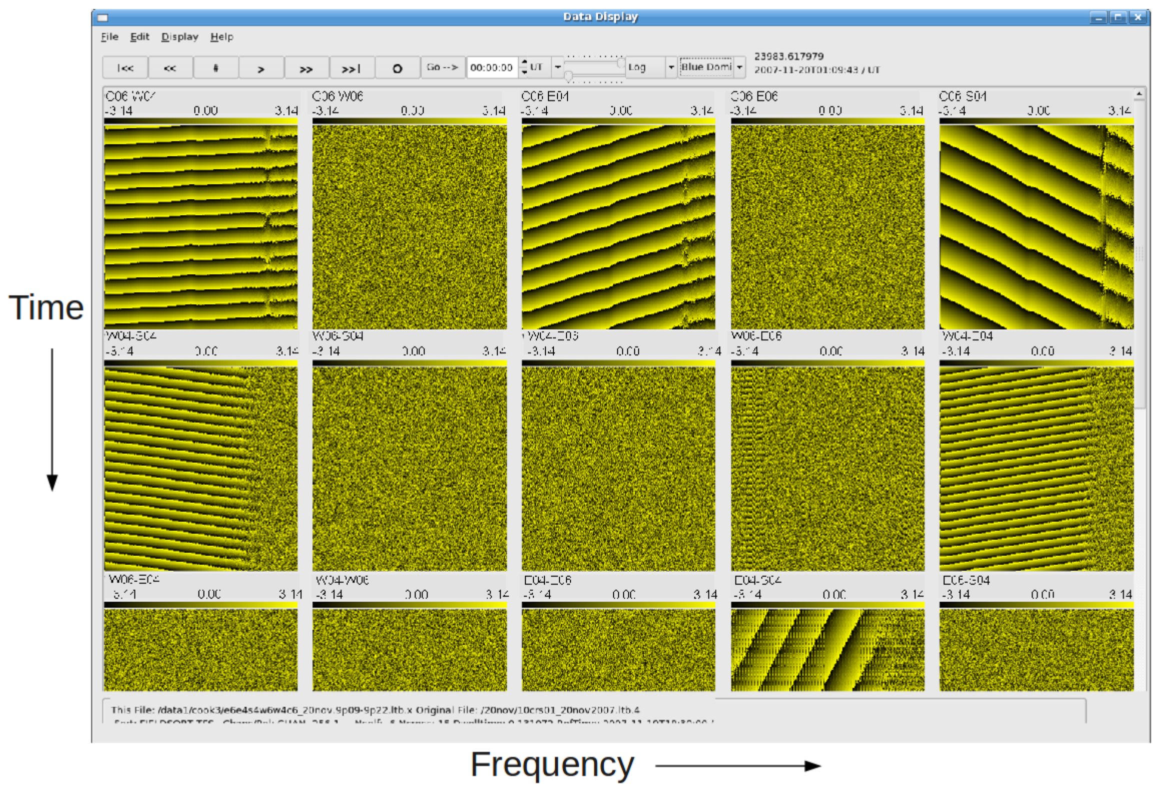
5.5.4.2 Receiver system configuration

The two 16 MHz bands from the GMRT front end available in the USB HiRes mode of the GMRT correlator, typically contain one polarization each from the dual polarized GMRT feed. In our observations, one sideband of the chosen polarization on each of the mid-arm satellite antennas was used to bring the satellite signal to the correlator, while the other sideband was dedicated to celestial observations. This is possible as all RF bands received by a GMRT antenna are down-converted to IF at all times, with a final level switching routing a polarization of the selected band into the analog transmitters.

5.5.4.3 Hardware correlator configuration

The data were recorded in the Upper Side Band (USB) High Resolution mode of the hardware correlator. IT may be noted that the high spectral resolution of 62.5 KHz (corresponding to 256 channels across 16 MHz) in this correlator mode leads to a better control on the RFI.

A significant aspect of our observing strategy was to take advantage of the short time integration of 0.131 sec permitted by the GMRT hardware correlator. This



This File: /data1/cook3/e6e4s4w6w4c6_20nov.9p09-9p22.ltb.x Original File: /20nov/10crs01_20nov2007.ltb.4

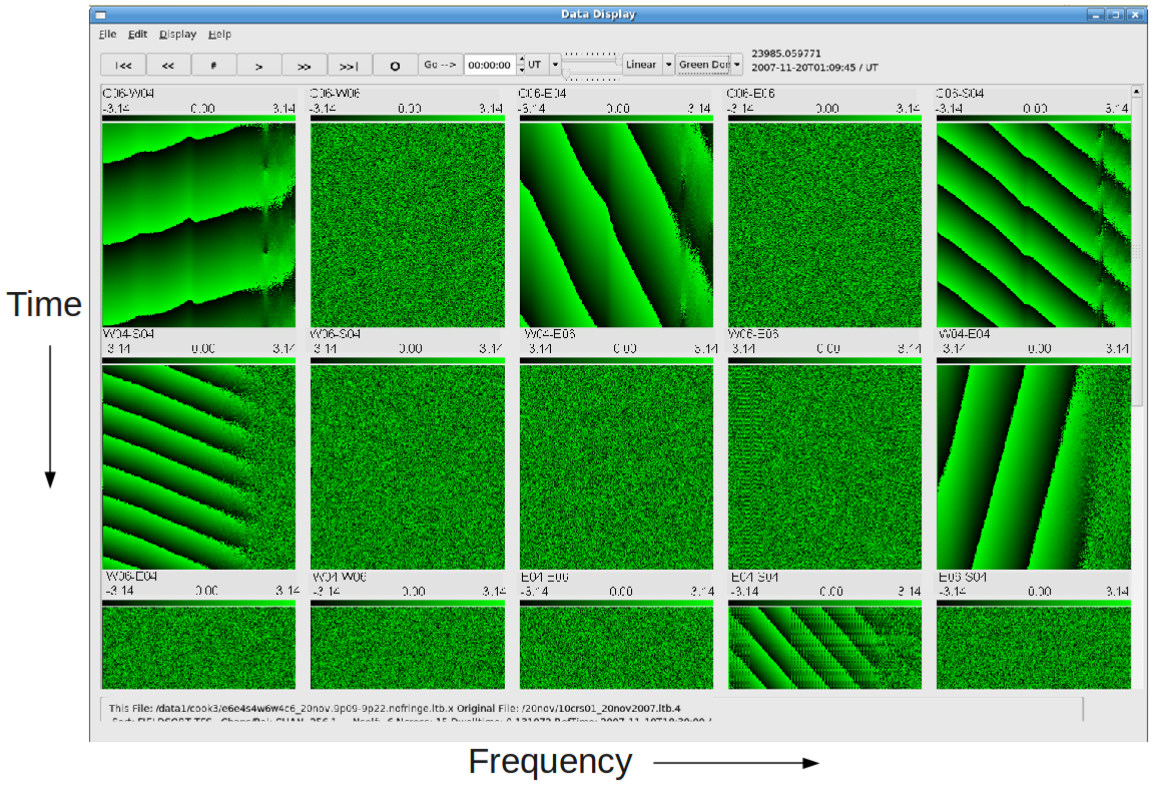
Figure 5.10: *Visualizer plates showing the artificial fringe on the satellite visibilities, induced by the fringe stopping carried out for the moving sky by the GMRT correlator.*

minimizes data required to be flagged during bursty interference and provides a means to incorporate better data filters for managing phase distortions due to the ionosphere, although our approach requires a data rate of about 50 GB/hr to be sustained.

5.5.4.4 Partial fringe stopping

The field of view at the chosen three arm antennas consisted of two components: the sky moving at sidereal rate, and the GEO satellite (which is stationary within a typical observing session of $\sim$ 3 hrs and has a diurnal sinusoidal period). Even though in transit observations geometrical delay (τ_{geo}) is time independent, it does change enough within the accumulation time of 1 STA (0.131 sec), thereby necessitating fringe stopping.

While it was necessary to incorporate fringe rotation to preserve correlation for the celestial sources, this would result in a loss of correlation for the satellite signal. This, in turn, is due to the correlator applying identical fringe stopping parameters on both the sky and satellite components, even though the τ_{geo} for the satellite signal is time independent. Thus, fringe rotation for the sky was applied only partially to the



This File: /data1/cook3/e6e4s4w6w4c6_20nov.9p09-9p22.nofringe.ltb.x Original File: /20nov/10crs01_20nov2007.ltb.4

Figure 5.11: *Visualizer plates showing the reduction in the artificial fringe rate on the satellite visibilities, after offline fringe correction.*

incoming data. This slowed down the phase winding rate for celestial baselines, while increasing them for the satellite baselines. To note, the fringe rate was restricted to less than a rad/sec for both the sky as well as the satellite baselines for the fastest varying baselines. This was done by specifying a value of 180 MHz for the sky RF to the fringe stopping subsystem, instead of the actual 235 MHz.

The residual uncorrected fringe on both the sky as well as the satellite signal would still lead to loss of sensitivity due to the STA over some part of the sinusoidally varying visibilities. Thus, this residual fringe has to be unwound offline in order to observe any of the 'sources' from the sky signals.

A similar fringe correction needs to be applied to the satellite data as well. This was carried out as part of the pre-processing on the recorded data from the satellite, before meaningful parameters could be extracted from the satellite phase. Figures 5.10 and 5.11 depict the phase of the visibilities from the baselines dedicated to observing the satellite signal, before and after the fringe stopping. The high SNR on the satellite phase is noticeable.

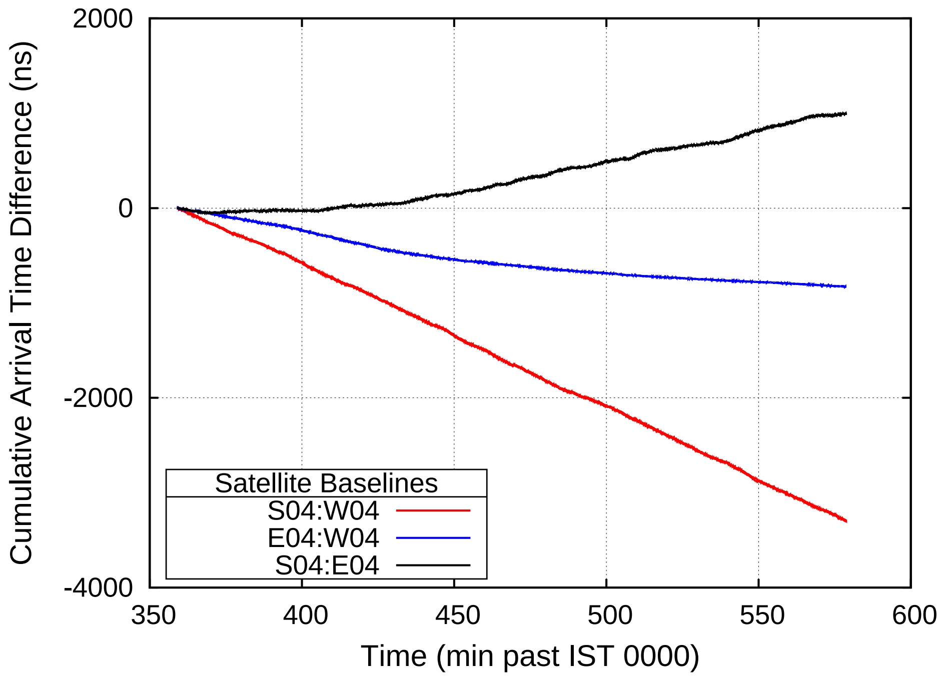
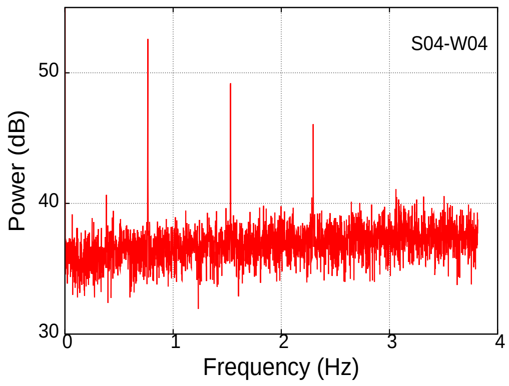


Figure 5.12: *Observed cumulative arrival time difference evolution from three different baselines observing the INMARSAT satellite during Nov, 2007.*

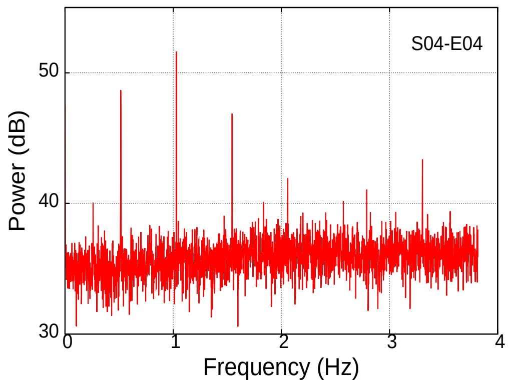
5.5.5 Results

The phase variations on the satellite signal visibilities, observed from three antennas were extracted as a continuous cumulative value, and converted to arrival time differences. This was carried out after the partial artificial fringe (due to the correlator settings) was corrected. These are depicted in Figure 5.12, where the cumulative variation of the estimated arrival time differences can be seen over a time duration of ~ 500 mins. Here, the arrival time difference shows a smooth variation because of the Doppler on the satellite signal, arising due to its motion relative to the interferometer. It is important to note the random variations in the arrival time difference, over the smooth component due to satellite Doppler. These are seen over timescales of a few minutes, and can be attributed to ionospheric effects.

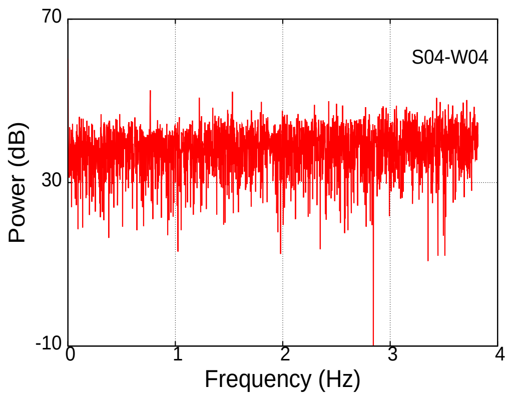
The dominant smooth variation of the satellite phase due to the diurnal motion of the satellite was compensated, and a power spectrum of the residuals over a timescale typical for ionospheric fluctuations was taken. This is depicted in Figure 5.13. Here, the SNR of the extracted phase variations is significant, in spite of the particular



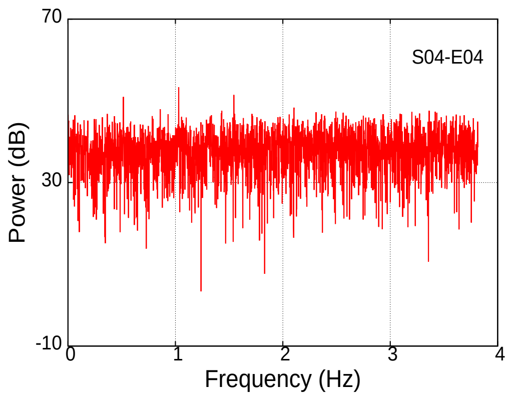
(a)



(b)



(c)



(d)

Figure 5.13: *Power spectrum of random phase variations observed on the residuals after contribution due to Doppler of INMARSAT satellite had been removed. Plots (a) and (b) show the power spectrum from two baselines for the same time instant during the continuous recording, while plots (c) and (d) show the typical power spectrum recorded from the same baselines at identical times, when no ionospheric random path length differences are present. The observed lines in (a) and (b) are consistent with signatures of travelling ionospheric clouds adding random extra path lengths to the received signal.*

observation being carried out pre-dawn during November, 2007, when the sun was expected to be inactive.

It may be noted that this method can enable multiple lines of sight to be sampled, by carrying out imaging on the satellite signal sources (which are treated as point sources). This would be readily possible due to the presence of correlated data from all satellite antenna elements as part of the GMRT correlator output.

This technique thus allows us to devise a simple scheme for carrying out a rapid survey of the sky with the GMRT, benefitting from the advantage offered by keeping a GEO satellite signal within the field of view [54].

5.6 Conclusions

In this chapter, we have presented novel approaches and tools useful for enhancing the sensitivity of low frequency observations. This, in turn, is achieved by increasing the reliability of the incoming dataset from a radio telescope, before carrying out further analysis. An important conclusion of our analysis, is the requirement of high temporal and spectral resolutions of observations, in order to better characterize the RFI environment. Our visibility visualizer tool allows rapid visualization of large datasets, which in turn, enables rough limits to be set on the parameters for the next stage of analysis. Further, it can carry out simple editing of the recorded large volume visibilities to carry out automated flagging on the incoming data.

A useful approach to carrying out observations requiring long integrations is to spread the necessary observing time over multiple sessions, preferably carried out on different days. This allows a better data editing of deviant visibilities by carrying out cross correlations between the multiple days' datasets. Our tool is flexible enough to accommodate this and other approaches within its framework, in order to be able to apply them on large databases. Another important aspect considered is the applicability of simultaneously recorded satellite signals to enhance the quality of observed visibilities. In particular, the possibility of using a high SNR GEO satellite signal as a phase reference for reducing distortions caused to visibility phases due to the ionosphere has been examined. We have also presented some key results from experiments conducted on GEO satellites using an interferometer developed by us, and co-located with the GMRT. In addition, software tools developed by us to cater to the preprocessing of high volume data, resulting due to sampling of visibilities at a very high temporal and spectral resolutions, have been presented.

Chapter 6

Conclusions

In recent times, there is a resurgence of interest in low frequency radio astronomy, driven by many new observational goals enabled by increasing receiver capabilities. These fundamentally relate to the modern receiver's ability to cope with large-scale real-time computing on high volumes of incoming data, as well as its increasing configurability. However, the relatively low arithmetic intensity, and the distributed nature of data sources pose new challenges, which require a careful examination of the communication and computing aspects from the implementation perspective of the receiver architecture design. This is partly due to the availability of a large variety of components, allowing receiver implementations to span a range of design parameters. One aspect which heavily influences design choices are the abilities of currently available commodity computing and communication components, due to their ease of development and lower cost. In addition, low frequency observational aspects which influence the receiver's design also need to be considered.

In this thesis, we have evaluated the design criteria of receivers tuned towards carrying out high sensitivity low frequency observations, while being flexible in their configuration, for a specific class of multi-element telescopes. We have generally concluded that the best options involve a combination of Commercial Off The Shelf (COTS), and custom segments.

Some important conclusions reached by us in this context are:

1. Large volumes of data need to flow across all levels of the communication hierarchy of a distributed receiver. This, in turn, is due to the fine temporal sampling of the observables required in order to effectively segregate deviant data due to the observing environment.
2. The network in a distributed receiver architecture plays a central role, which is

larger than merely that of a data transporter, and can be used to affect load balancing and job scheduling.

3. The ability to buffer large volumes of data early in the dataflow hierarchy, and subsequently operate in "Transaction Units" can influence the implementation of a distributed receiver, with aspects ranging from synchronization between the distributed data sources, to load balancing via data reordering.
4. Multiple processing passes may be required to be carried out on streaming data for effective segregation. Further, the computing needs to be distributed across the hierarchy, in order to carry out pre-processing or segregation of data at the various levels.
5. The computing requirements of the receiver can be partitioned into categories, and matched appropriate commodity components.
6. The development of algorithms to cater to low frequency observations require continuous real-time feedback based on a descriptive model of the incoming data.

Based on these conclusions, we have designed a Network centric receiver architecture, termed the Networked Signal Processing System (NSPS). This is a packetized, heterogeneous and distributed signal processing architecture, with its various networks as core system components. In this approach, the communication and processing is visualized as a problem of an appropriate workload creation and scheduled dispatch to matched processors over a data flow tree.

Further, we have implemented a programmable receiver system for the Ooty Radio Telescope (ORT), based on the NSPS architecture. This part of the work demonstrates the viability of the NSPS approach. Here, the custom segment of the NSPS was implemented in FPGA, and equipped with a high-speed peer-to-peer network. This segment carries out the main task of load balancing via data reorganizing. It also ensures synchronization between the distributed data sources.

In addition, we have equipped the implemented receiver system with a software based correlator, operating on commodity class processors. This approach gives maximum flexibility of accepting different data formats and number of elements, while leading to lowered development times. Here, we conclude that commodity class processors, while being extremely flexible, can be competitive in their processing efficiencies in comparison with other approaches. This is possible, provided sources of their inefficiency, e.g., frequent communication to other processors, inefficient utilization of their processing architecture and memory hierarchy, are minimized. Towards this end, the custom hardware segment of the NSPS provides explicit data partitioning to achieve a high degree of load balancing, while eliminating the barrier synchronization requirement to a large extent.

The distinguishing feature of our approach for medium sized arrays is the explicit dependency on carrying out data processing via software on commodity hardware, as well as a custom hardware section with the ability to route and regroup data at any level of preprocessing. This is possible only with the data transposition ability of the Data Pooler element, along with the ability to operate on discrete Transaction Units.

Finally, we have presented tools for visualization, editing and calibration of large volume data sets, which are mainly necessitated by the requirement of observations with high temporal and spectral resolutions. Further, the feasibility of certain observing strategies for carrying out high sensitivity observations have been explored. In particular, the high SNR signal from Geosynchronous navigation satellites, observed using facilities co-located with a radio telescope within the field of view of of the celestial observations, has been utilized. Such a strategy was used to carry out transit mode observations from the Giant Meterwave Radio Telescope (GMRT).

We would like to point out the possibilities of future extension to this work. The upcoming trend of multiple independent cores in commodity processors (upto 6 per processor in the just released Intel SandyBridge microachitecture), as well as larger vector units (256 bits in SandyBridge) benefits the NSPS approach, which places a majority of the computing load on such processors. Porting our software correlator to these architectures should allow direct gains in processed bandwidth, or in the number of elements handled.

Further, the stage-0 of the FFT implementation (which has been designed to allow offloading to hardware), can be implemented in the Level-3 hardware of the NSPS implementation. This, in turn, will reduce a large fraction of the first level data shuffles. In addition, once the entire FFT is offloaded to hardware, the possibility of using the spectral dimension for carrying out the Data Pooling is feasible. The output can then utilize the 16-bit "half-floating point" support now available from libraries on the target FPGA platform. The software correlator might also benefit from the usage of half-floating point support, which is beginning to be available in commodity processors. It may be noted that currently, only memory access and datatype conversion instructions are available with the AVX instruction set implemented for the SandyBridge, although once arithmetic instructions are available, the load of computing scaling factors based on the incoming data can be eliminated, while also increasing the dynamic range of the receiver.

Bibliography

[1] CL Carilli and S. Rawlings. Motivation, key science projects, standards and assumptions. *New Astronomy Reviews*, 48(11-12):979–984, 2004.

[2] A. Nigl, P. Zarka, J. Kuijpers, H. Falcke, L. Bahren, and L. Denis. Vlbi observations of jupiter with the initial test station of lofar and the nancay decametric array. *Astronomy and Astrophysics*, 471:1099–1104, 2007.

[3] C. Mercier and G. Chambe. High dynamic range images of the solar corona between 150 and 450 mhz. *The Astrophysical Journal Letters*, 700:L137, 2009.

[4] R.M. Hjellming. Radio stars. *Galactic and Extragalactic Radio Astronomy*, 1:381–438, 1988.

[5] AG Lyne, M. Burgay, M. Kramer, A. Possenti, RN Manchester, F. Camilo, MA McLaughlin, DR Lorimer, N. D'Amico, BC Joshi, et al. A double-pulsar system: A rare laboratory for relativistic gravity and plasma physics. *Science*, 303(5661):1153, 2004.

[6] JJ Condon. Extragalactic astronomy at low frequencies. In *From Clark Lake to the Long Wavelength Array: Bill Erickson's Radio Science*, volume 345, page 237, 2005.

[7] DE Harris, CP Stern, AG Willis, and PE Dewdney. The megaparsec radio relic in supercluster, rood no. 27. *The Astronomical Journal*, 105:769–777, 1993.

[8] C. Blake and K. Glazebrook. Probing dark energy using baryonic oscillations in the galaxy power spectrum as a cosmological ruler. *The Astrophysical Journal*, 594:665, 2003.

[9] TH Hankins, JS Kern, JC Weatherall, and JA Eilek. Nanosecond radio bursts from strong plasma turbulence in the crab pulsar. *Nature*, 422(6928):141–143, 2003.

[10] J. M. Cordes, T. J. Lazio, and M.A. Mclaughlin. The dynamic radio sky. *New Astronomy Review*, 48:1459–1472, 2004.

[11] D.R. Lorimer, M. Bailes, M.A. McLaughlin, D.J. Narkevic, and F. Crawford. A bright millisecond radio burst of extragalactic origin. *Science*, 318(5851):777, 2007.

[12] P.E. Dewdney, P.J. Hall, R.T. Schilizzi, and T.J.L.W. Lazio. The square kilometre array. *Proceedings of the IEEE*, 97(8):1482–1496, 2009.

[13] AR Taylor. The square kilometre array. In *IAU Symposium*, volume 248, pages 164–169, 2008.

[14] S. Bhatnagar. Calibration and imaging challenges at low radio frequencies: A reivew of the state of the art. In *ASP Conf. Series*, volume 407, pages 375–383, 2008.

[15] A.R. Thompson, J.M. Moran, and G.W. Swenson. *Interferometry and synthesis in radio astronomy.* Wiley-VCH, 2001.

[16] S.J. Wijnholds, J.D. Bregman, and A.J. Boonstra. Sky noise limited snapshot imaging in the presence of rfi with lofars initial test station. *Experimental Astronomy*, 17(1):35–42, 2004.

[17] S.W. Ellingson. Rfi mitigation and the ska. *The Square Kilometre Array: An Engineering Perspective*, pages 261–267, 2005.

[18] Y. Chikada, M. Ishiguro, H. Hirabayashi, M. Morimoto, KI Morita, K. Miyazawa, K. Nagane, K. Murata, A. Tojo, S. Inoue, et al. A digital fft spectro-correlator for radio astronomy. In *Indirect Imaging. Measurement and Processing for Indirect Imaging*, volume 1, page 387, 1984.

[19] G. Swarup, S. Ananthakrishnan, VK Kapahi, AP Rao, CR Subrahmanya, and VK Kulkarni. The giant metre-wave radio telescope. *Current Science*, 60:95, 1991.

[20] J. Roy, Y. Gupta, U.L. Pen, J.B. Peterson, S. Kudale, and J. Kodilkar. A real-time software backend for the gmrt. *Experimental Astronomy*, 28(17):26–60, 2010.

[21] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[22] H.D. Falcke, M.P. van Haarlem, A.G. de Bruyn, R. Braun, H.J.A. Rottgering, B. Stappers, W.H.W.M. Boland, H.R. Butcher, E.J. de Geus, L.V. Koopmans, et al. A very brief description of lofar–the low frequency array. *Proceedings of the International Astronomical Union*, 2(14):386–387, 2006.

[23] M. de Vos. Lofar: the first of a new generation of radio telescopes. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 5, pages v–865. IEEE, 2005.

[24] K. Schaaf, C. Broekema, G. Diepen, and E. Meijeren. The lofar central processing facility architecture. *The Square Kilometre Array: An Engineering Perspective*, pages 43–58, 2005.

[25] C.J. Lonsdale, R.J. Cappallo, M.F. Morales, F.H. Briggs, L. Benkevitch, J.D. Bowman, J.D. Bunton, S. Burns, B.E. Corey, L. DeSouza, et al. The murchison widefield array: Design overview. *Proceedings of the IEEE*, 97(8):1497–1506, 2009.

[26] T. Oosterloo, MAW Verheijen, W. van Cappellen, L. Bakker, G. Heald, and M. Ivashina. Apertif-the focal-plane array system for the wsrt. In *Proceedings of Wide Field Astronomy & Technology for the Square Kilometre Array (SKADS 2009). 4-6 November 2009. Chateau de Limelette, Belgium. Published online at http://pos. sissa. it/cgi-bin/reader/conf. cgi? confid= 132, id. 70*, volume 1, page 70, 2009.

[27] WA van Cappellen and L. Bakker. Apertif: Phased array feeds for the westerbork synthesis radio telescope. In *Phased Array Systems and Technology (ARRAY), 2010 IEEE International Symposium on*, pages 640–647. IEEE, 2010.

[28] A. Parsons, D. Werthimer, D. Backer, T. Bastian, G. Bower, W. Brisken, H. Chen, A. Deller, T. Filiba, D. Gary, et al. Digital instrumentation for the radio astronomy community. *Arxiv preprint arXiv:0904.1181*, 2009.

[29] G. Swarup, NVG Sarma, MN Joshi, VK Kapahi, DS Bagri, SH Damle, S. Ananthakrishnan, V. Balasubramanian, SS Bhave, and RP Sinha. Large steerable radio telescope at ootacamund, india. *Nature*, 230(17):185–188, 1971.

[30] AJ Selvanayagam, A. Praveenkumar, D. Nandagopal, and T. Velusamy. Sensitivity boost to the ooty radio telescope: A new phased array of 1056 dipoles with 1056 low noise amplifiers. *IETE Technical Review*, 10:333–339, 1993.

[31] NVG Sarma, MN Joshi, DS Bagri, and S. Ananthakrishnan. Receiver system of the ooty radio telescope. *Journal of the Institution of Electronics and Telecommunication Engineers*, 21:110–116, 1975.

[32] Peeyush Prasad and C.R. Subrahmanya. A high speed networked signal processing platform for multi-element radio telescopes. *Experimental Astronomy*, 31(1):1–22, 2011.

[33] L.A. Klein. *Sensor and data fusion: a tool for information assessment and decision making*, volume 138. SPIE Press, 2004.

[34] A. Lutomirski, M. Tegmark, N.J. Sanchez, L.C. Stein, W.L. Urry, and M. Zaldarriaga. Solving the corner-turning problem for large interferometers. *Monthly Notices of the Royal Astronomical Society*, 2011.

[35] D. Anish Roshi. A study of recombination lines from the galactic centre region due to transitions at very high rydberg states. Master's thesis, University of Poona, Pune, 1995.

[36] T. Prabu. *A New Digital Receiver for the Ooty Radio Telescope*. PhD thesis, Center for Electronic Design and Technology, Indian Institute of Science (IISc), Raman Research Institute, Bangalore, 2011.

[37] PK Manoharan. Three-dimensional structure of the solar wind: Variation of density with the solar cycle. *Solar physics*, 148(1):153–167, 1993.

[38] P. Gothoskar and Y. Gupta. Scintillation velocities of five millisecond pulsars. *The Astrophysical Journal*, 531:345, 2000.

[39] NG Kantharia and KR Anantharamaiah. Carbon recombination lines from the galactic plane at 34.5 & 328 mhz. *Journal of Astrophysics and Astronomy*, 22(1):51–80, 2001.

[40] PS Ramkumar, T. Prabu, M. Girimaji, and G. Marker leyulu. A digital signal pre-processor for pulsar search. *Journal of Astrophysics and Astronomy*, 15(3):343–353, 1994.

[41] D. Anish Roshi. *A Study of the Ionized Gas in the Galactic Plane using Radio Recombination Lines near 327 MHz*. PhD thesis, National Center for Radio Astrophysics, Tata Institute of Fundamental Research, Pune, 1999.

[42] Ashok Singal. *Lunar Occultation Observations and a Study of Cosmic Size Evolution of Extragalactic Radio Sources*. PhD thesis, University of Bombay, Bombay, 1988.

[43] A.X. Widmer and P.A. Franaszek. A dc-balanced, partitioned-block, 8b/10b transmission code. *IBM Journal of research and development*, 27(5):440–451, 1983.

[44] R.G. Lyons. *Understanding digital signal processing*. Prentice Hall PTR, 2004.

[45] C.R. Subrahmanya. Satellite astrometry: A status update. *Private Communication*, 2006.

[46] L. D'Addario. Cross correlators. In *Proceedings of the Astronoical Society of the Pacific*, volume 6, page 59, 1989.

[47] AT Deller, SJ Tingay, M. Bailes, and C. West. Difx: A software correlator for very long baseline interferometry using multiprocessor computing environments. *Publications of the Astronomical Society of the Pacific*, 119:318–336, 2007.

[48] J. Max. Quantizing for minimum distortion. *Information Theory, IRE Transactions on*, 6(1):7–12, 1960.

[49] R.N. Bracewell. *The Fourier transform and its applications*. McGraw-Hill, 2000.

[50] *Intel Integrated Performance Primitives for Intel Architecture: Reference Manual*. Intel Corporation, 2010.

[51] *Intel 64 and IA-32 Architectures Software Developer's manual, Vol. 2A/2B: Instruction Set Reference*. Intel Corporation, 2010.

[52] M. Frigo and S.G. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[53] M.H. Wieringa. An investigation of the telescope based calibration methods redundancy and self-cal. *Experimental astronomy*, 2(4):203–225, 1992.

[54] Peeyush Prasad and C. R. Subrahmanya. Dynamic range improvement of gmrt low frequency images. In *Proceedings of The Low Frequency Radio Universe, ASP conference series*, volume 407, pages 398–401, 2009.

[55] CR Subrahmanya. Low frequency imaging and the non-isoplanatic atmosphere. In *IAU Colloq. 131: Radio Interferometry. Theory, Techniques, and Applications*, volume 19, pages 218–222, 1991.

[56] TJ Pearson and ACS Readhead. Image formation by self-calibration in radio astronomy. *Annual review of astronomy and astrophysics*, 22:97–130, 1984.

[57] CM Ho, AT Mannucci, UJ Lindqwister, and XQ Pi. Global ionospheric perturbations monitored by the worldwide gps network. *Geophysical Research Letters*, 23:3219–3222, 1996.

[58] WC Erickson, RA Perley, C. Flatters, and NE Kassim. Ionospheric corrections for vla observations using local gps data. *Astronomy and Astrophysics*, 366(3):1071–1080, 2001.

[59] S.V. Kibe. Indian plan for satellite-based navigation system for civil aviation. *Current Science*, 84:1405–1411, 2008.

[60] A. Bhaskaranarayana. Indian irnss and gagan. In *COSPAR Meeting, Montreal*,

2008.

[61] C. R. Subrahmanya, Peeyush Prasad, and C.R. Somashekar. Interferometric observations of geosynchronous satellites. In *ICST conference proceedings*, 2011.

Appendix A

Packet Structures for the ORT NSPS implementation

This appendix lists the memory layouts of various kinds of packets flowing in the ORT NSPS hierarchy, along with data flow discriminators in the form of UDP ports.

```
/* FPGA can send status and data packets only, while PC can send
 * status, data and command packets. All pkts have the same header,
 * and are distinguished by the datatype/srcid field.
 * NOTE: These entries also define the various port numbers to wait
 * on. These are hardcoded in the FPGA code as well.
*/
enum {DataPkt=0xd0, StatPkt=0xf0, CmdPkt=0xc0};
enum {DataPort=(DataPkt<<8)+255, StatPort=(StatPkt<<8)+255,
      CmdPort=(CmdPkt<<8)+255};
/* Types of data packets:
* These flow only from the FPGA to the PC.
* A 4 bit field; giving 16 different types of handlable data.
*/
enum
{ DataUserDef ,          // User defined data type
  DataCnt     ,          // Contents are a 32bit counter, for debug
  DataRaw     ,          // Contents are ADC raw data
  DataRawZip  ,          // Contents are ADC raw compressed data
  DataFFTRaw  ,          // Contents are FFT Complex output
  DataFFTZip  ,          // Contents are FFT Compressed output
  DataFFTPow  ,          // Contents are FFT Power spectra only
  DataSpectCorr,         // Contents are Spectral Correlations
  DataRawStat ,          // Contents hold data statistics only
  DataMeta,              // Contents hold metadata (history/scanhdr)
  DataSpectCorrPack,     // Contents are Spectral Correlations with
// minimal header. Size is in 8-byte word units
  DataTypes};

/* Types of status packets:
 * These flow only from the FPGA to the PC.
 */
enum
{ SSubIAA = 1,       // I am alive status packet
  SSubStart  ,       // Tx in start state
  SSubStop   ,       // Tx in stop state
  SSubInvalOp,       // Command invalid
  SSubCmdDone,       // Command successfully executed
  SSubLUT    ,       // Contains current LUT entries as data
  SSubMon    ,       // This is a monitor packet
  StatusTypes};

/* Types of Command packets:
 * These flow only from PC to FPGA
 */
enum
{ CSubReset    ,    // Assert internal reset. All counters reset
  CSubSendAYA  ,    // Send an Are you alive packet to FPGA
  CSubStart    ,    // Start FPGA application
  CSubStop     ,    // Stop FPGA application
  CSubUpdateHdr,    // Update internal IP hdrs
  CSubUpdateLUT,    // payload has LUT updation data, see pktsub
  CSubSendLUT  ,    // Send current LUT entries to PC
  CSubDoDbg    ,    // Ask FPGA for a 32bit cntr at specified rate
  CSubFlushFIFO,    // Flush all FIFOs/buffers. TSCs are unaffected.
  CSubGPIO     ,    // Manipulate the 32 bit GPIO lines on FPGA
  CSubChanSel  ,    // Choose from upto 16 channel inputs
  CSubInit     ,    // Send an init command to data sources
  CSubSetDelay ,    // Set a delay for given channel
  CSubSetTS    ,    // Set the TimeStamp counter to the given value
  CSub2ch8bpp  ,    // Choose a 2ch, 8bpp streaming mode from eth0
  CSubUserDef  ,    // Look at pktsub for decode
  CmdTypes};

typedef struct
{ HdrType hdr;
  unsigned char data[0];
} CmdPktType;
```

```
/* Layout of a status packet. This has fields which are manipulatable
 * by the App for app specific status. Also has common status fields.
 */
typedef struct
{ unsigned char mac[6];
  unsigned char ip [4];
  unsigned short csum;
} FPGADstType;


/* Every status packet has at the very least this much information*/
typedef struct
{ unsigned char FPGAVER;    // Indicates version of the FPGA code
  unsigned char ability_state;// advertised ability of FPGA code.
                            // Should be one among DSub* above (4bit).
                            // LSB 4 bits are current state
  unsigned short appdef;    // FPGA app specific field.
  unsigned int GPIOlines;   // Current value of FPGA internal GPIO
  FPGADstType destaddr[4];  // Internal current destinations.
  unsigned char data[0];    // Contains things like LUTs, current
                            // destinations etc. Size should be
                            // calculated based on type of status
                            // packet as well as size reported in
                            // UDP hdr.
} StatPktType;

typedef struct
{ unsigned char srcid;
  unsigned int  datatype: 4;
  unsigned int  bits2pix: 4;
  unsigned int  chans   : 4;
  unsigned int  words   :12; // Sizeof pkt(including hdr) in
  unsigned int  tick;        // 8Byte units.
  unsigned int  taps;
  float         f_samp;      // Sampling frequency in Hz
} SpectHdrType;

typedef struct
{ SpectHdrType hdr;
  float pspect[0];
} PowPktType;

typedef struct
```

```
{ unsigned char srcid;
  unsigned int  datatype: 4;
  unsigned int  bits2pix: 4;
  unsigned int  chans   : 4;
  unsigned int  words   :12;  // sizeof pkt (with hdr) in 8byte units
  unsigned int  tick;
  LayerHdrType lay[3];
  unsigned char grpid;
  unsigned char dummy;
  unsigned int  samps;
  float f_samp;               // All headers should be 8 byte aligned
} __attribute__((aligned (16), packed)) StatisticHdrType;

enum {HistBins=32};
typedef struct
{ float mean, var;
  unsigned int hist[HistBins];
} StatisticWordType;

typedef struct
{ StatisticHdrType hdr;
  StatisticWordType word[0];
} StatisticPktType;


// Timing related: NOTE: ORDER OF hw/sw matters!!
typedef struct
{ unsigned int hw, sw; } Tick;
typedef union
{ Tick t;
  unsigned long long abscnt;  // Software and Hardware absolute cntr.
} LongTick;

// Information specific to the field being observed
typedef struct
{ char name[32];
  float ra, dec;             // src coordinates in J2000
} SrcInfoType;

// Information specific to the observer for this scan
typedef struct
{ char name[32];
  char projcode[16];
  struct timeval time;      // Timestamp at beginning of scan
} ObsInfoType;
```

Appendix B

Photographs

This appendix consists of a few photographs of the implemented NSPS system for the ORT.

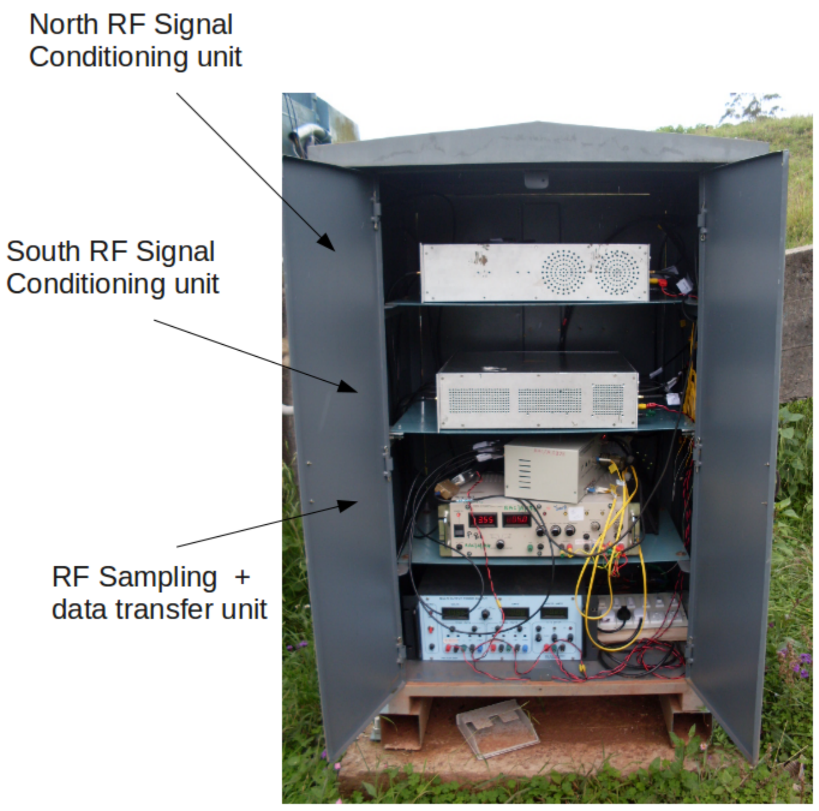

North RF Signal Conditioning unit
South RF Signal Conditioning unit
RF Sampling + data transfer unit

Figure B.1: *Populated pillar, with analog signal conditioning systems and Level-3 remote digitizers visible.*

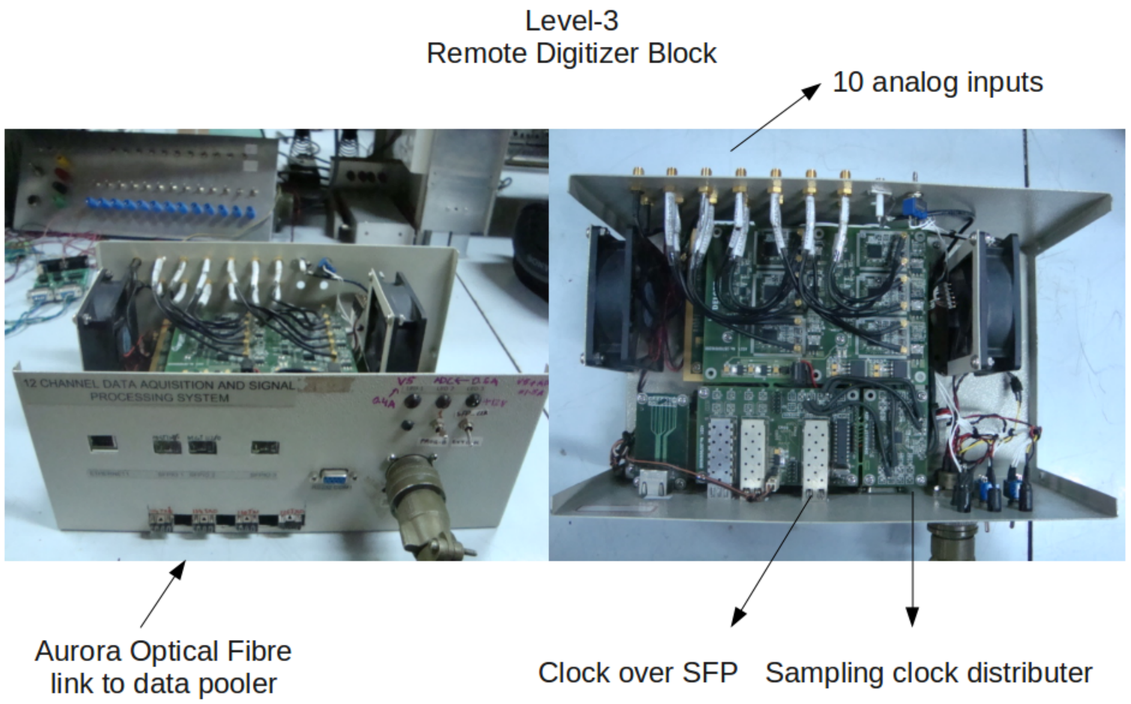

Level-3
Remote Digitizer Block
10 analog inputs
12 CHANNEL DATA AQUISITION AND SIGNAL PROCESSING SYSTEM
Aurora Optical Fibre link to data pooler
Clock over SFP
Sampling clock distributer

Figure B.2: *Level-3 remote digitizer units, with onboard clock distribution.*

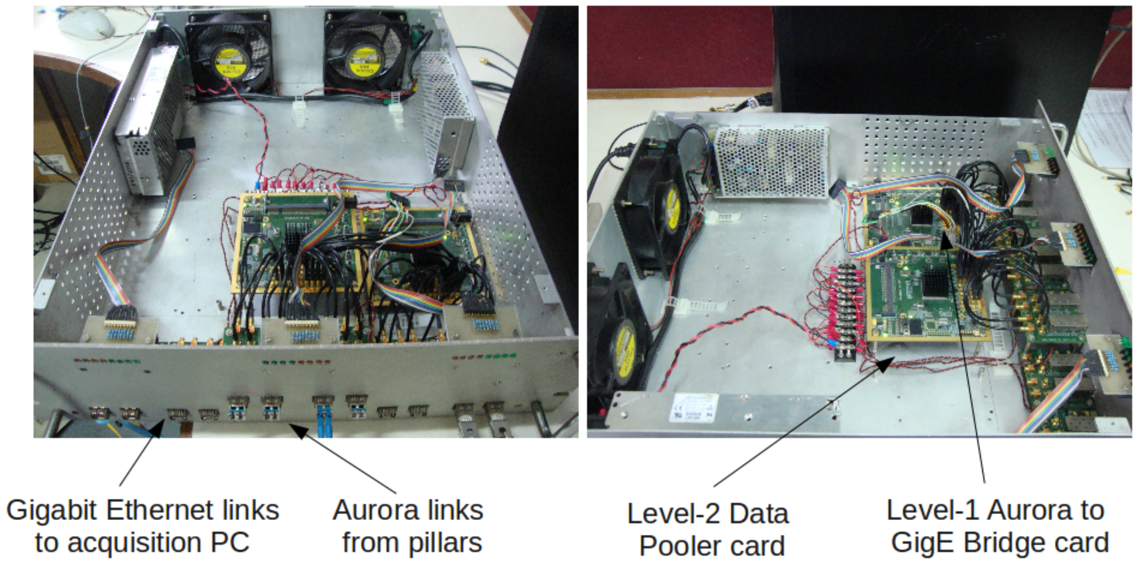

Gigabit Ethernet links to acquisition PC
Aurora links from pillars
Level-2 Data Pooler card
Level-1 Aurora to GigE Bridge card

Figure B.3: *Level-2 and Level-1 data pooler and bridge cards shown in their chassis.*