FP7 - Grant Agreement no. 283393 - Radionet3

Project name:	Advanced Radio Astronomy in Europe
Funding scheme:	Combination of CP & CSA
Start date:	01 January 2012
Duration:	48 months

Deliverable 8.12

Revised Firmware Design Document
Uniboard^2 Digital Receiver

Due date of deliverable:	30 June 2015
Actual date of deliverable:	30 June 2015
Deliverable Leading Partner:	INAF

An European project supported within the 7th framework programme

1 Document information

Document name:	Revised Firmware Design Document: UniBoard^2 Digital Receiver
Type	Document
Revision	1.0
WP	8
Authors	Giovanni Comoretto
INAF report	02/2015

1.1 Dissemination level

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services	
RE	Restricted to a group specified by the Consortium (including the Commission Services	
CO	Confidential, only for members of the Consortium (including the Commission Services	

1.2 Document history

Revision	**Date**	**Author**	**Modification/Change**
0.1	2015-05-21	Comoretto	Initial draft
0.8	2015-06-25	Comoretto	Revised complete version
1.0	2015-06-28	Comoretto	Revised document with comments from A. Szomoru and B. Quertier

1.3 Distribution list

ASTRON Andre Gunst, Eric Kooistra, Sjouke Zwie, Danil van der Schuur, Harm Jan Pepping

JIVE Arpad Szomoru, Jonathan Hargreaves, Salvatore Pirruccio, Sergei Pogrebenko, Paul Boven, Harro Verkouter

UMAN Ben Stappers, Prabu Thiagaraj

INAF Gianni Comoretto

BORD Benjamin Quertier, Alain Baudry, Stephane Gauffre

UORL Cedric Dumez-Viou, Rodolphe Weber, Nicolas Grespier

MPG Guenter Knittel, Gundolf Wieching

1.4 Terminology

ADC: Analog to Digital Converter

ARP: Address Resolution Protocol

BF: BeamFormer

bps: Bits per second

BW: BandWidth

channel: Frequency band, unit output of the filterbank. For a two stage architecture, *Coarse channel*: and *Fine channel* refer to the unit output of the first and second stage filterbanks.

COTS: Commercial Off The Shelf

DDR: Double Data Rate. DDR3 and DDR4 refer to the 3^{rd} and 4^{th} version of the standard, respectively

DSP: Digital Signal Processing

EMI: Electro-Magnetic Interference

Firmware: Embedded or real-time code that runs on a microprocessor (e.g. written in C), or describes a programmable logic (e.g. written in HDL)

FFT: Fast Fourier Transform

FPGA: Field Programmable Gate Array

Hardware: Boards, sub-racks and COTS equipment

HDL: Hardware Description Language

HEM: HMC Extension Module for the Uniboard[2]

HMC: Hybrid Memory Cube: high density memory standard, with high speed serial interface

IO: Input-Output

IP: Intellectual Property

IP: Internet Protocol

JESD204: A JEDEC (Joint Electron Device Engineering Council) standard for serial ADC interface

JTAG: Joint Test Action Group standard for boundary scan testing and programming of programmable devices

LAN: Local Area Network

PHY: physical interface (layer 1 of OSI model)

QSFP+: SFP for 40Gb Ethernet

PFB: Polyphase Filterbank

RF: Radio Frequency

RFI: Radio Frequency Interference

SFP: Small Form-factor Pluggable transceiver. An optical interface module, available with different performance and range characteristics, pluggable in a common socketed cage.

SFP+: SFP for 10Gb Ethernet

SPEAD: Streaming Protocol for Exchanging Astronomical Data

SOPC: System on Programmable Chip. A processor and associated peripherals implemented using programmable logic. Typically used to control the application on the FPGA.

Subband: Frequency band, unit output of the filterbank

TCP: Transmission Control Protocol, standard safe Internet protocol

UDP: User Datagram Protocol, standard stateless, low overhead Internet protocol

WAN: Wide Area Network

1.5 References

References

[1] Arpad Szomoru: "UniBoard2 Work Package description", RadioNet3 283393 (2011)

[2] Gijs Schoonderbeek: "UniBoard^2 Architecture - Hardware design document", Astron report INFRA-2011-1.1.21 (2014)

[3] G. Comoretto, A. Russo, G. Tuccari, A. Baudry, P. Camino, B. Quertier: "Uniboard Digital Receiver Design document", Arcetri Technical Report 5-2011
`http://www.arcetri.astro.it/images/data/Reports/11/5_2011.pdf`

[4] W.H. Press, S.A. Teukolsky, W.T Vetterling, B.P. Flannery: *Numerical Recipes, 3rd Edition*, chapter 7, "Random number generation".

[5] W.H. Press, S.A. Teukolsky, W.T Vetterling, B.P. Flannery: *Numerical Recipes, 3rd Edition*, chapter 12, "FFT of a single real function"

[6] G. Comoretto, G. Knittel, A. Russo: "Uniboard Pulsar Receiver Design document" (2012)

[7] Fredric J. Harris: "Digital Receivers and Transmitters Using Polyphase Filter Banks for Wireless Communications", IEEE Trans. on Microwave Theory And Techniques, **51**, 4 (2003)

[8] Analog Devices: "JESD204B Survival Guide"
`http://www.analog.com/static/imported-files/tech_articles/JESD204B-Survival-Guide.pdf`

[9] G. Comoretto, A. Russo, G. Tuccari: "A 16 channel FFT multiplexer", Arcetri Technical Report 1-2009
`http://www.arcetri.astro.it/ricerca/rapporti-tecnici/205-reports/564-09-1`

[10] G. Comoretto: "A design method for very large FIR filters", Arcetri memo series 3/2012

[11] J. Manely, M.Welz, A.Parson, S. Ratcliffe, R. van Rooyen: "SPEAD: Streaming Protocol for Exchanging of Astronomical Data", SKA-SA internal memo, 2010/10/07

[12] Comoretto, G., Melis, A., Tuccari, G.: "A wideband multirate FFT spectrometer with highly uniform response" Experimental Astronomy **31**, 59-68 (2011).

[13] VDIF task force: "VLBI Data Interchange Format (VDIF) Specification, release 1.0 (2009),
`http://www.vlbi.org/vdif/docs/VDIF%20specification%20Release%201.0%20ratified.pdf`

Contents

2 Introduction

A radiotelescope receives a wideband RF signal, with typical bandwidths that in current systems are of the order of several GHz. Only a portion of this signal is analyzed, because of hardware limitations, presence of human made interfering signals, or absence of useful astronomic informations. Moreover, for technical limitations, it is often advantageous to process the signal in relatively small bands, up to some tens of MHz, and thus divide the input bandwidth into many small narrow bands (*sub-bands* or *channels* in this document). This task is performed in an astronomic radio receiver. An example of such a system is the MARK-x family of VLBI terminals, in which up to 16 couples of narrow bands (from less than 1 MHz to 32 MHz) can be arbitrarily positioned inside an input band of the order of 1 GHz.

A digital receiver performs the same function using digital signal processing. The radio signal, either just amplified up to a viable level or also converted in frequency, is digitized by a wideband ADC and then decomposed into narrow bands. These bands can be processed locally, e.g. for single dish spectroscopy or for connected element interferometry, stored on disk, for conventional VLBI, or sent to a high speed Internet link for real time e-VLBI.

2.1 Digital receiver concept

The digital receiver concept is shown in figure 1. The signal received from the antenna and divided into n frequency bands (channels) is routed through a off-the-shelf network switch to a series of digital backends. A spectrometer and a pulsar timing analyzer are shown in the figure, but other kinds of processing are possible.

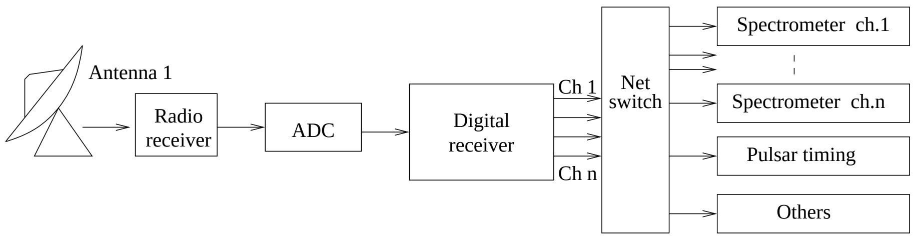

Figure 1: Digital receiver concept: the radio signal is amplified and converted to a band that can be digitized using an ADC. The digital signal is then processed, producing n individually tunable narrow bands, that are routed, through a switch, to a set of digital backends for further analysis

These backends can be implemented either using programmable logics (e.g. other Uniboard based instruments), conventional general purpose computers, or GPU-based computing units. If the routing is static, a direct point-to-point link can be used, but a switch adds a great deal of flexibility, allowing the same digital receiver to be used for simultaneous or alternative observations without the need of reconfiguring the telescope instrument interconnections.

The digital receiver concept is particularly useful for a distributed correlator architecture, both in connected interferometers and in VLBI. In this case (see figure 2) the n channels produced by a digital receiver at each antenna are routed to n separate correlators, each one processing one individual band for all antennas. The switch (or the WAN) performs the task of *corner turning* the signal, i.e. transposing the [antennas-channels] data matrix.

The digital receiver is manly a concept, not a specific application. It is composed of a set of building blocks, for direct data processing (FFT channelizers, tunable filters), packetizing and formatting (SPEAD and VDIF formatting, UDP packetizing), and for test/monitor functions (test signal generator, total power and integrity check measurements, total power spectrometer).

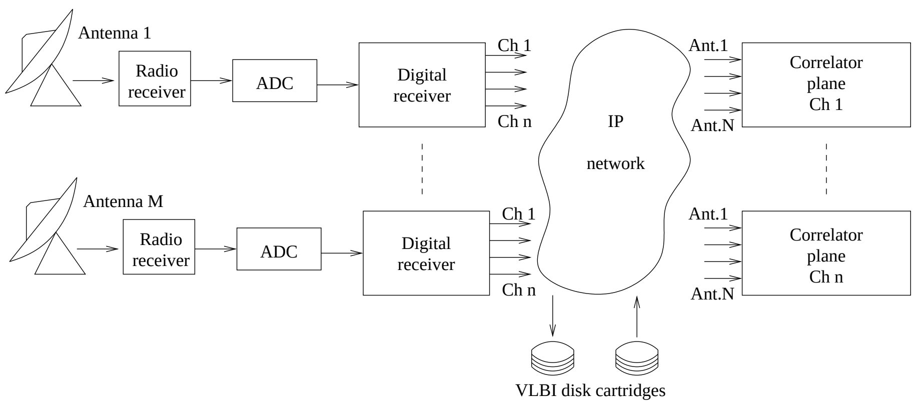

Figure 2: Network correlator: the n individual channels produced by a Digital receiver on each antenna are processed by n correlator planes

These building blocks can be composed, together with the general Uniboard^2 management and control structure, to produce a digital receiver application tailored to the specific needs.

The application is then organized as a general purpose library, with individual fully parameterized modules for high level functions, like serial or parallel FFT, digital downconversion, VDIF formatting, test signal generation. These modules are written within the general framework of the *RadioHDL* signal processing library, and exploits the rich portfolio of generic digital signal processing modules and control structures written as part of the Uniboard^2 project. This ensures portability of the design both to different hardware platforms and FPGA families, and to different telescopes.

2.2 Prior work in the field

As part of the FP7 Radionet project, a digital receiver application has been designed for the Uniboard platform[3]. The design interfaces with an high speed analog-to-digital converter, providing an instantaneous bandwidth of up to 4 GHz. A dual stage filterbank channelizes the digitized signal, providing a set of up to 64 VLBI channels. Each channel represents a portion of the input band with an arbitrary start frequency and width, that is filtered, down-sampled and formatted according to the VDIF format.

Another digital receiver has been developed for the FP7 project “Beacon in the Dark”, for a pulsar timing machine [6]. It receives two polarizations with a bandwidth of 3.2 GHz each, and produces a total of 144 partially overlapping channels of 24 MHz each. These channels are positioned in order to avoid heavily RFI contaminated spectral regions (fig. 3).

2.3 Activities for the Uniboard^2 project

The Digital Receiver application developed for the Uniboard platform has been redesigned for the Uniboard^2. The increased capabilities of the new board can be exploited in various ways:

- The larger number of resources in each FPGA, and the higher clock speed allows to implement a complete digital receiver using a single node. This allows a single Uniboard^2 to implement 4 independent digital receivers, e.g. for a dual polarization, dual frequency receiver for geodynamic VLBI observations.
- The Arria family may use multipliers with a larger word size. This allows for better noise performance and RFI immunity in critical design modules.

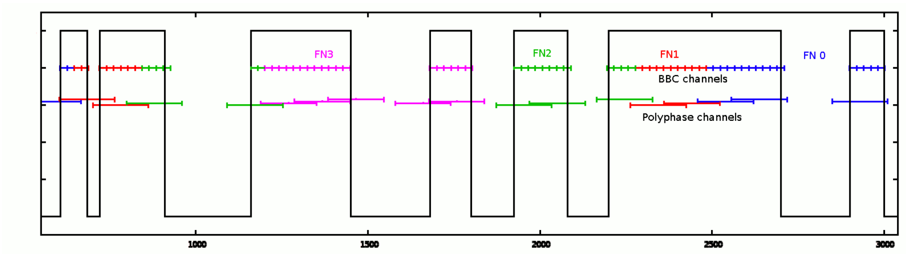

Figure 3: Coarse and fine channelizer bands for the Uniboard Beacon digital receiver. Black line denotes portions of the input band that are observed or filtered out in the RF receiver. Larger bars denote broad channels, color coded for the Uniboard nodes where they are implemented. Smaller bars denote fine channels

- The extra resources available in a single FPGA allows for different options both in the first and in the final channelization stages.

In addition to these optimizations, the general architecture of the original application has been improved in several ways

- The rigid structure of the Uniboard Digital Receiver has been generalized, allowing for a very flexible parametrization of the individual modules. Signals can be represented with different number of bits, filter shapes can be redefined, etc.
- A library of Matlab routines for filter calculation have been written.
- The control structure has been adapted to the AXI4 bus standard, in addition to the original, vendor specific, Avalon bus. This allows porting the modules and applications to a different hardware platform.
- Different architectures for the various modules have been added, allowing for a wide variety of applications.
- The application is designed as a modular library, in which independent elements can be tailored and assembled according to the application needs. In this way it can be adapted to widely different situations.

2.4 The Uniboard^2

The Uniboard^2 platform [2] is a high performance DSP platform that represents an evolution of the previous Uniboard, developed as part of the Radionet FP7 program. It is composed of 4 identical nodes, with one FPGA per node. The initial version will use Arria10 FPGAs, replaced by Stratix10 in the final version.

The Uniboard^2 architecture is shown in figure 4. Each node is interfaced to 6 front panel QSFP+ cages, that can be connected to a 40 Gbps optical link, and to a passive backplane or an extension module using a set of high speed backplane connectors. The backplane connector provides 48 high speed interconnection lines, that are used for a customizable interconnection mesh, to interface Hybrid Memory Cube (HMC) modules, or to provide additional optical links. An additional ring connection, composed of 24 high speed links, interconnects vertically the 4 nodes. The backplane connectors provide also power, JTAG connection, and clock-timing references. The nodes are controlled using an internal 1Gb Ethernet mesh, via 6 front panel RJ45 connectors.

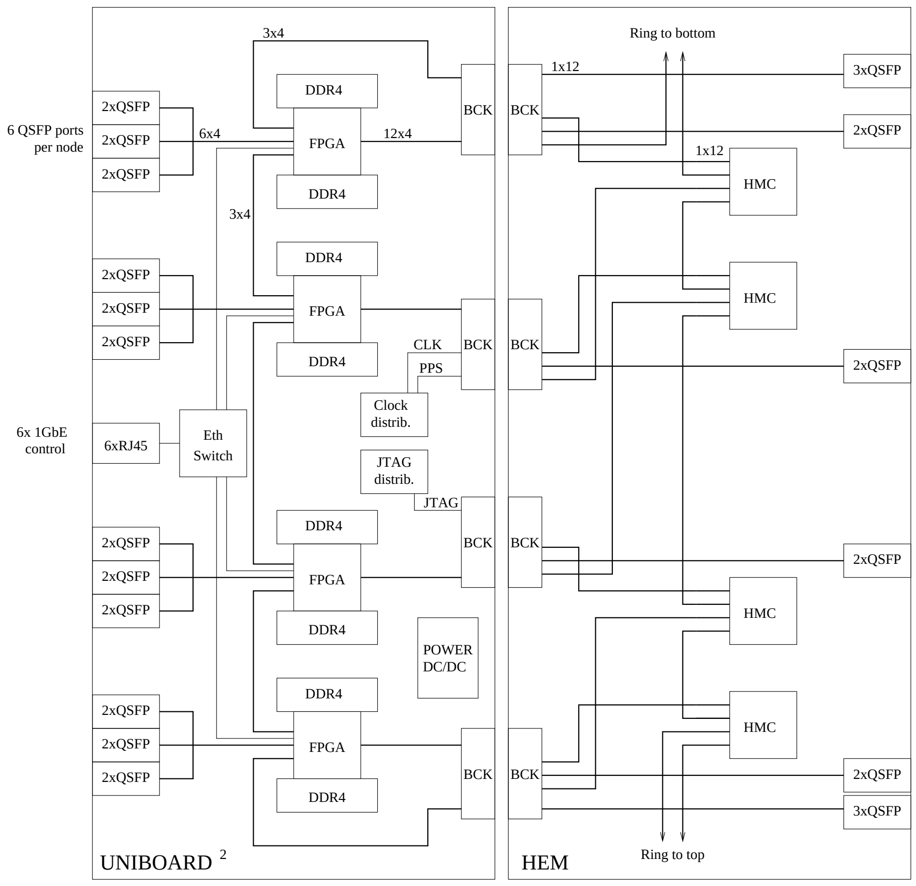

Figure 4: Uniboard 2 and HMC Extension Module block diagrams

The board can be tailored to specific applications using an extension module. The first version for this module (HEM, for HMC Extension Module) is also shown in figure 4, and is composed of 4 HMC modules, one per node. The 4 modules are connected in a ring structure, and to two of the 4 FPGAs each. In this way it is possible from each node to address directly two memory modules, and indirectly all 4, allowing for a full memory based interconnect mesh. The HEM provides also two extra QSFP+ connections per node.

Local memory is provided by two independent 64 bit DDR4 modules, and can be extended using external Hybrid Memory Cube (HMC) modules sited in the backplane or in the interposer module.

Node-to-node interconnection is performed using dedicated backplanes, passive optical interconnects, or standard interconnects using COTS QSFP+ network switches.

The FPGA initially used will be an Altera Arria 10GX1150, with 660 general purpose usable pins, 96 high speed transceivers and a total of 3036 18×19 bit multipliers.

The final board will use Stratix10 FPGAs. This device will have the same pinout, and then the same number of transceivers and pin count, but a much higher number of internal resources, and an internal clock rate up to 1 GHz. The external link bit rates will increase to 26 GHz, with 56 GHz on selected lines. Memory interface will support the maximum transfer rate of 3200 MT/s of the DDR4 standard, and the Hybrid Memory Cube standard interface modes will be fully supported.

This design is based on the expected performances of the Arria10 family, but can be easily scaled to take advantage of the Stratix10 increased performances. The larger number of multipliers can be used both to increase the number of available output channels, and to increase the internal sample size for higher spurious signal rejections in high RFI environments. The higher internal clock speed and link bit rate can be used to scale up the total RF receiver bandwidth to 8-10 GHz, matching the available bandwidth of current centimeter and millimeter radioastronomic receivers.

3 Digital Receiver common control structure

The Uniboard project has defined a standard protocol for communication with the applications. The protocol allows read, write and modify of 32 bit registers across the system, and is agnostic about the register content. Any intelligence resides on an external computer, programmed using an high level language (C, Python, Erlang ...). High level commands or procedures are decomposed in single register access commands, and sent to the FPGA through the control Ethernet link.

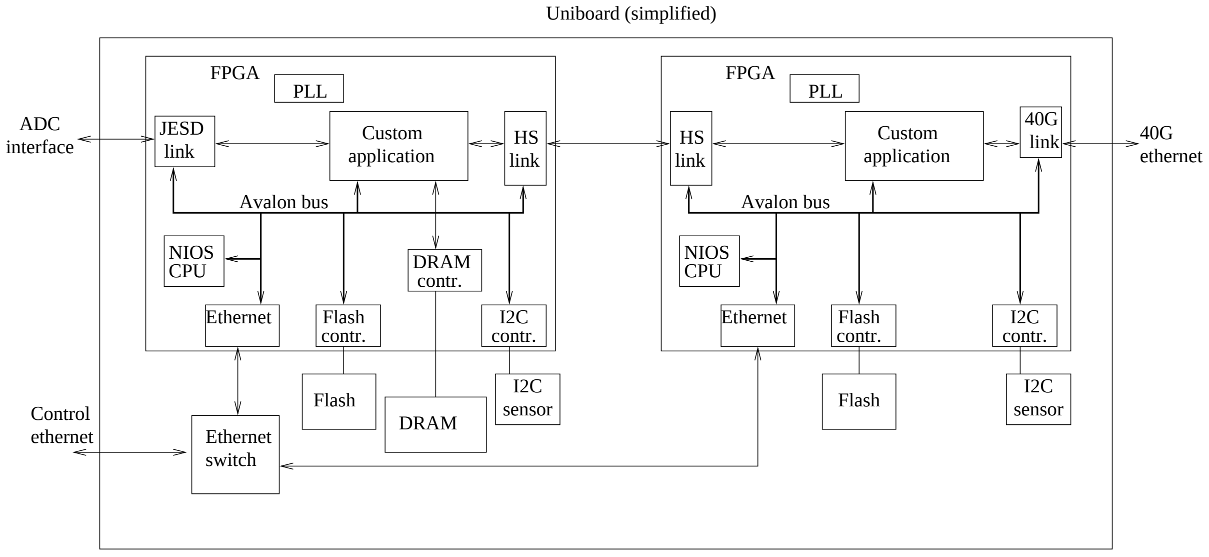

Figure 5: Control structure for an application using two FPGAs

Inside each FPGA a SOPC module runs a standard program, independent from the application, that decodes UDP command packets. The simpler UDP protocol allows to implement the control program without the relatively large overhead of a TCP-IP stack. The control structure is based on the Altera proprietary NIOS SOPC processor, and each module is interfaced to the processor using Altera Avalon processor bus. A set of standard peripherals are used for debugging, program timing, download of personalities in the on-board flash memory, and generic control and monitor functions.

The control structure for a generic, completely hypothetical, Uniboard application is sketched in figure 5. This example application receives samples from an ADC using a JESD104 interface, processes them in a first FPGA using also the on-board DDR4 RAM, and passes the results to a second FPGA that completes the processing and sends the results using a 40 Gb Ethernet port. The standard peripherals for the 1Gb Ethernet interface, the FPGA personality Flash memory, the internal high speed link, a JESD104 serial interface for ADC, a 40 Gb Ethernet MAC and a generic temperature/voltage I2C sensor are present in each node. Other standard interfaces for different serial protocols are available in the Uniboard control library.

The custom application interfaces to the controller CPU using the Avalon bus. Each module has an address space composed by a number of consecutive 32 bit words. In the digital receiver modules we adopted a convention for the first four register positions, as in table 1.

Word	Offset	Write register	Read Register
0	0x00	Test point	Readback test
1	0x04	—	Date code
2	0x08	Control	=
3	0x0c	—	Status
4+	0x10+	App. dependent	App. dependent

Table 1: Common registers in all module interfaces

The first (offset zero) register controls a test point utility. The test point is associated to one (or more) external line, that can be connected, for debug purposes, to an external pin of the FPGA. If the register is set to zero, the external line is at a fixed LOW level, to minimize electric activity and allowing multiple modules to share the same external line (OR-ing together the module lines). A number of internal signals can be connected to these external pins, and probed with an oscilloscope, using the value specified in the register for signal selection. The register can be read back, thus allowing a simple check of the module interface without affecting the module functionality.

The identification register reports a fixed value, hardcoded in the module firmware. It can be used to check for the presence of specific modules, and to report the module version. Writing the register has no effect.

The control register has individual bits, or bit groups, mapped to specific functionalities of the module, at a global level. 32 bits are usually sufficient to set most of simple parameters, like bandwidth, start/stop, synchronization, reset. These bits, when read back, report the current setting of the associated options, allowing for a read-modify-write setting of individual options leaving the rest of the configuration unchanged.

The status register reports the module general status. The register is usually polled and cannot be written. Usually individual bits report a boolean status of some parameter (e.g. an overflow condition, or completion of a total power integration). These bits are reset by pulsing high (writing a 1-0 sequence) a flag reset bit in the control register.

Registers after the first four are module dependent, and are described in the individual detailed module description.

When a total power function is available, it is controlled using a total power register. Often a single total power register can control an array of total power meters, that reasonably use the same configuration. The format of the total power register is the same for all instantiations of the total power function, and is reported in table 2.

bit	control
0x00-01	TP function:
	0 = Total power, 1 = DC offset
	2 = State counter, 3 = RD check
0x02	General enable
0x04-07	Integration prescaler
0x08-0f	Reference status or RD check line
0x10-1f	Integration length (ms)

Table 2: Total power control register

The total power function allows to collect different statistics on the data stream in addition to the total power. The state counter counts the number of samples matching a given pattern, and the Random Data checker checks data integrity by comparing the received bit sequence on a specific sample bit with a pseudo random sequence given by a predefined polynomial. Integration time can be specified as an integer number of milliseconds. To maintain the measurement significance with very different integration times, a programmable number of bits in the accumulator result is discarded before transferring the result in the 32 bit total power register.

Each module register structure is described in a XML file. This file can be used to automatically generate the specific module interface, the relevant software driver structure (e.g. a Python package with appropriate constants for register addressing) and a skeleton documentation file, using an automated document generation tool like Doxygen.

The Avalon interface structure is specific for Altera FPGAs. Other interface standards are widely used, and the overall control structure has also being ported to the AXI4-lite interface standard. This bus is the standard in Xilinx based designs, and is also being increasingly used in Altera designs. The same XML description file can be used to generate the hardware interface for this standard, or a AXI4-Avalon bridge can be interposed between the Avalon interface and the AXI4 bus. Both options are currently implemented.

4 Digital receiver architecture

The Digital Receiver is based on a 2 stage architecture. The general structure is similar to the Uniboard and Beacon digital receivers, and is shown in fig. 6.

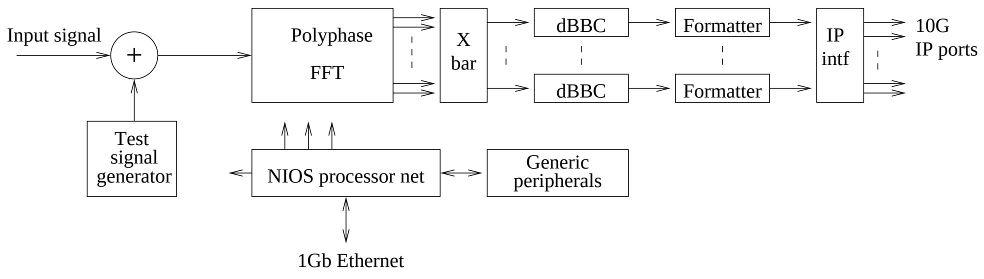

Figure 6: Structure of a dual stage digital receiver

It is composed of an ADC interface (section 4.3), a test signal generator (section 8), a first stage channelizer that provides uniformly spaced *broad channels* (section 4.4), and an array of second stage channelizers, providing the final channelization. A packet formatter then sends the channelized data in an application specific (e.g. VDIF or SPEAD) UDP Ethernet packets (section 9).

The first stage converts a broadband signal, with sample rate largely in excess of the processing rate, to a fixed number of parallel channels, each with a sample rate comparable to the processing rate. This greatly simplifies any further processing, that occurs at the natural FPGA processing frequency. A sample rate lower than the processing rate may also be advantageous, if the second stage processing can exploit resource recirculation. For example the same filter hardware can be used to process in turn separate channels. The second stage is used to provide the final channelization, in a flexible way.

This architecture allows for a relatively simple broadband processing, using an efficient, but rigid, channelization hardware in the first stage, while the final stage allows for a sophisticate, flexible and high performing channelization.

4.1 Architecture variations

Different options are available both for the first and for the second stage channelizers. The basic Uniboard digital receiver provides individually tunable channels, using a *digital baseband converter* architecture (chapter 5). It is possible to individually adjust each channel bandwidth (and sample rate) and position, allowing for maximum flexibility, e.g. when both wideband continuum and narrow band spectral observations must be performed simultaneously. The total number of channels is however limited. This is not usually a problem for VLBI, where the total bandwidth is limited by the data transport, but may constitute a limitation for single dish or connected elements interferometry.

For the situations where most of the input band is to be processed, a dual stage polyphase filterbank is more appropriate (chapter 6.1). Some flexibility can still be retained by choosing which channels to process or transmit for further processing, and allowing some forms of tunability for each block of channels. A standard, non overlapping filterbank can be used when it is not essential to have complete frequency coverage. In this case channel separation is slightly larger than the usable channel bandwidth, with small gaps between channels. Where continuous coverage is required, e.g. for spectroscopic applications, a overlapping polyphase filterbank can be used.

4.2 The Digital Receiver library structure

A generic library of commonly used functions and modules has been written for the application. The library includes commonly used data types like complex number representation and operations, serial and parallel FFT in radix 2, 4 and 5, conventional and oversampling polyphase filtering, and test signal generation.

The library structure is composed of VHDL *libraries*. Each library is described in a specific user manual. Most libraries instantiate mainly a few top level modules, with a large number of parameters, and internally choose the most efficient implementations according to these parameters. For example the polyphase top level function can instantiate a channelizer with overlapping or non overlapping channels, real or complex input samples, and different bit sizes.

These top level modules interface using only standard VHDL types, for maximum portability, and rely on the low level Uniboard^2 `common_lib` for the vendor dependent functions. Other Uniboard libraries are used for standard functions and data types, like records describing streaming data (`dp_lib`) and the Avalon bus, or to implement commonly used peripherals.

In figure 7 the dependences of the libraries used in the package are shown. The top row lists the two mostly used libraries from the UniBoard^2 RadioHDL repository, `common_lib` (that includes also the description for the Avalon bus) and `dp_lib`. These libraries are used extensively, so dependences are drawn only for the first level beyond that.

The remaining libraries are part of this deliverable. The first layer contains a set of commonly used modules.

- The basic functionalities, common to all modules, are enclosed in the `common_OA_lib` library. This library is necessary to compile all other modules, but not to use the top level modules themselves.
- The `axi4lite_lib` is used only for the version of the modules that interface with the AXI4 bus. It is not needed for the Avalon version.
- The `fft_lib` implements various instantiations of the FFT algorithm, described in section 7. These are mainly used in conjunction with polyphase filtering to implement a polyphase filterbank.

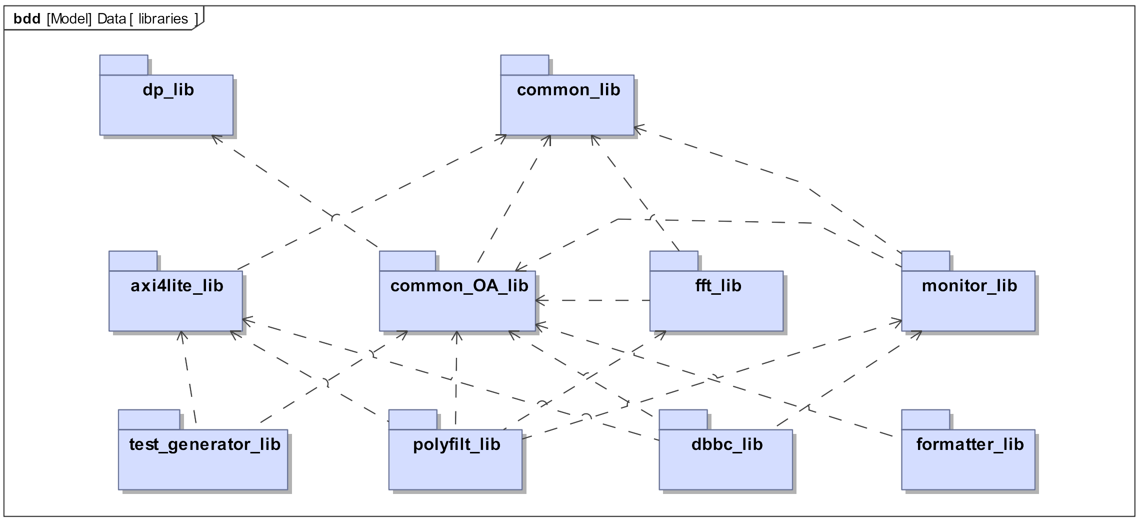

Figure 7: Libraries in the Digital Receiver application

- The `monitor_lib` implements several useful modules for data analysis, like data capture and total power, both for wideband and channelized data.

The bottom layer contains specific high level modules, built by using elements from the previous layer. These elements include:

- `polyfilt_lib` (section 6) Various implementations for parallel and serial polyphase filterbanks, with overlapping and critically sampled channels. A *Matlab* library of filter design functions has also been included, to provide the necessary filter tap coefficient values (section 6.4).

- `dbbc_lib` (section 5) A digital baseband converter module. The module provides a tunable down-converter, that extracts a portion of the input band with programmable bandwidth and center frequency

- `formatter_lib` (section 9) A set of modules that receive framed data and format it in the VDIF or SPEAD formats. They output a set of UDP frames, with programmable IP destination address and port, that must be further packed in an Ethernet frame. The Ethernet and MAC/PHY layers are implemented using generic Uniboard^2 modules and vendor specific IPs.

- `test_generator_lib` (section 8) A test signal generator is very useful for diagnostic purpose, and to characterize the instrument when no signal is available. The signal generator module provided with the digital receiver package includes a white noise, to simulate a realistic astronomic signal, a couple of monochromatic tones and a periodic pulse generator.

4.3 Interconnection with the ADC

The input stage is not developed as part of the design, as it depends heavily on the choice of the ADC. Here we only give some general design considerations, and a few examples of possible ADC interfaces.

The Uniboard^2 has no parallel interface, like the Uniboard, and thus cannot be directly connected with a parallel interfaced ADC. The ADC must be interfaced using high speed serial links, either directly or using an interposed FPGA. For example the industrial standard

JESD204B [8] allows an ADC to interface using multiple links with a serial speed of up to 12.5 Gbps. A single link can transport up to 1.5 Gs/s with 8 bit samples, i.e. a sampled bandwidth of about 750 MHz (figure 8-a). 8 links would be required for a 4 GHz bandwidth, at 8 bit data rate (figure 8-b). These can be physically transported using two QSFP+ optical links.

Commercial ADCs with JESD104 interface and bandwidths up to 1-2 GHz are currently available, but it is difficult to access ADCs that reach the required sample rate (8 GSps). Custom ADCs and a digitization module based on commercial components are now being developed as part of the Uniboard project by the University of Bordeaux. The ADC will be interfaced with a small FPGA, using serial links for the output. Standard Ethernet frames, or frames using the simplified *Uthernet* protocol (a point-to-point addressless version of the Ethernet), developed as part of the Uniboard project, can be used for the transport layer (figure 8-c).

For smaller bandwidth, the system can interfaces with a small CASPER board, using one of the commercial ADCs available in the CASPER project (figure 8-d).

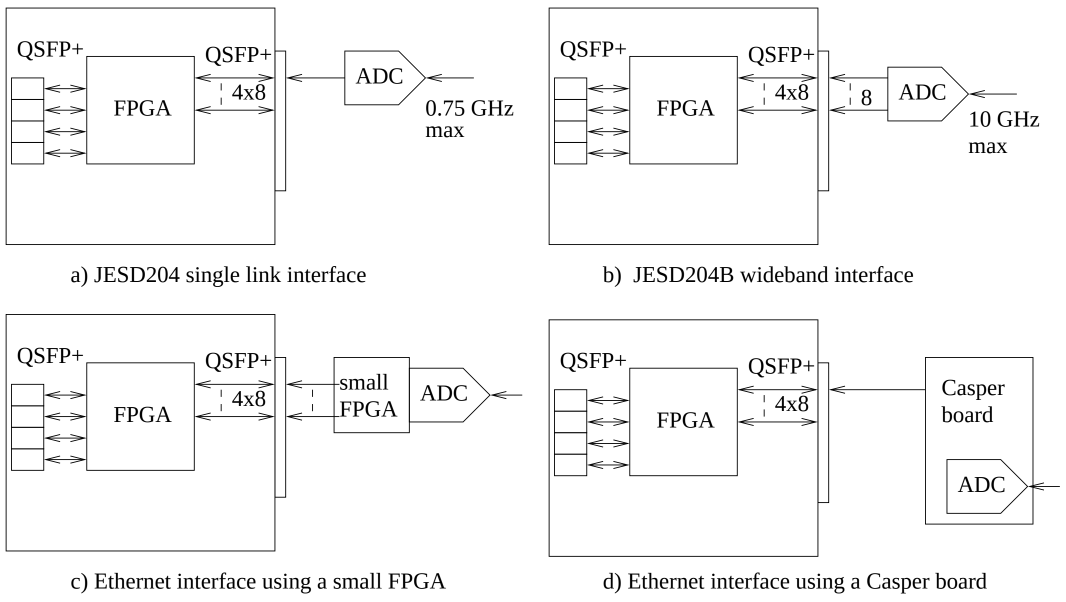
QSFP+
QSFP+
FPGA
4x8
ADC
0.75 GHz max
a) JESD204 single link interface
QSFP+
QSFP+
FPGA
4x8
8
ADC
10 GHz max
b) JESD204B wideband interface
QSFP+
QSFP+
FPGA
4x8
small FPGA
ADC
c) Ethernet interface using a small FPGA
QSFP+
QSFP+
FPGA
4x8
Casper board
ADC
d) Ethernet interface using a Casper board

Figure 8: ADC interfacing to the Uniboard[2]

4.4 First stage polyphase filterbank

In most digital receiver applications the ADC sampling frequency is much higher than the internal frequency of the FPGA logic. ADC data samples are thus represented in time multiplexed form. A first stage of frequency multiplexing, in which the input band is split into adjacent, overlapping channels provide several advantages:

- data is represented in continuous time series, requiring much less routing
- data representation becomes complex valued, allowing for easier signal processing
- any RFI contained in a single channel is less likely to produce interferences outside that channel
- portions of the input band not containing useful astronomic data can be discarded at an earlier stage of the processing

The most efficient channelization scheme for equispaced frequency channels is the polyphase filterbank (PFB)[7]. It is composed of a series of short FIR filters, that convolve the input sample sequence with an appropriate window function, and a conventional N-point FFT, that transposes the FIR response of the windowing function to the N frequency channels of the FFT. Output channels may overlap by an arbitrary amount. In our case we choose to use a 50% overlap, i.e. the channel nominal width is twice the channel spacing (chapter 6.1). This allows for very relaxed requirements on the shaping filter specifications, requiring a minimum of resources for the band-shape filter.

The large overlap region also allows for a great freedom in successive processing. In the digital receiver application the channelized signal is further filtered to a band of typically 1/4 that of the coarse filterbank and, if the effective overlap among adjacent coarse channels is 25%, the narrow channel can be arbitrarily positioned in any point of the input band (figure 9).

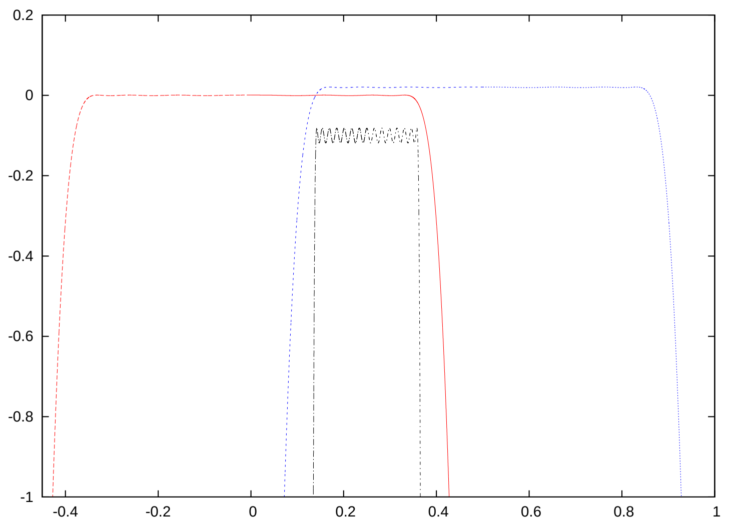

Figure 9: Bands for two consecutive coarse channels, and for a 1/4 band fine channel. Horizontal scale in channel width, vertical scale in dB. Overlap allows the fine channel to be arbitrarily placed in the input band

4.5 Second stage channelizer

The first stage channelizer produces coarse channels with a band equal to the FPGA processing clock frequency. In the Arria10 implementation of the Uniboard[2] a conservative clock frequency, and coarse channel bandwidth, of 250 MHz has been assumed. Usually VLBI correlators work on much smaller channel bands, up to 64 MHz and possibly 128 MHz. Although correlators with complex signal representation are becoming increasingly common, most existing VLBI receivers use real valued samples. For this reason it is necessary to divide the coarse channels into one or more *narrow channels*, with real valued samples.

Three options are possible:

- If the number of required channels is small, one can implement a conventional digital down-conversion structure (section 5), composed of a complex local oscillator and mixer, a complex (real valued) decimating FIR filter, and a complex to real up-conversion stage. This allows the maximum flexibility for channel width and spacing: channels may overlap, and *zooming modes*, in which the same spectral region is observed with different spectral resolution, are available (fig. 10a).

- If the channels are equispaced, with a bandwidth equal to the spacing, and a small *dead region* can be allowed between the channels, a conventional polyphase filter (section 6.2) can be used (fig. 10b). Channel bandwidth/spacing is fixed, requiring a personality reload to change FFT length, but the origin of the fine channel group in any given coarse channel can be shifted, and individual channels may be skipped.
- If the channels are equispaced with a bandwidth different from the channel separation, an oversampling structure can be used (chapter 6.3). This allows a small overlap among adjacent channels, avoiding the *holes* in the spectrum associated with finite filter edges. Again, the bandwidth and overlap is fixed in the design, but each fine channel group can be tuned in block, and individual channels can be skipped. (fig. 10c).
- If the channels are equispaced, and a fixed channel bandwidth is used, a single stage channelizer using a wideband parallel/serial FFT (section 7.3) can be used, and no second stage channelizer is necessary.

These options are described in detail in the sections indicated above.

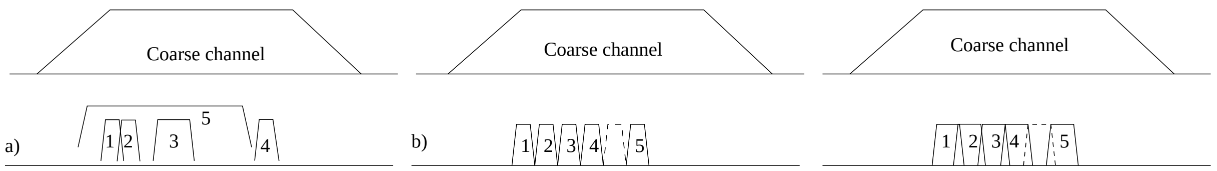

Figure 10: Options for the second stage filter: (a) independent DBBCs; (b) conventional polyphase; (c) Oversampling polyphase

5 Digital baseband converter

This option allows for maximum flexibility, but only a limited number of independent channels can be implemented. Considering the resources in a Arria10 FPGA, and the required multipliers used by the coarse channelizer, about 48 channels can be fitted in a single node. Structure of the filter is shown in fig. 11. An input selector allows the channel to be fed from one of four coarse channels, i.e. to be tuned across about 500 MHz of the input bandwidth. The channel center frequency is selected using a local oscillator and a complex mixer. By synchronizing the local oscillator to an external 0.1 ms timing signal it is possible to tune the local oscillator to an exact multiple of 10 kHz, according to the VLBI standard frequency scheme.

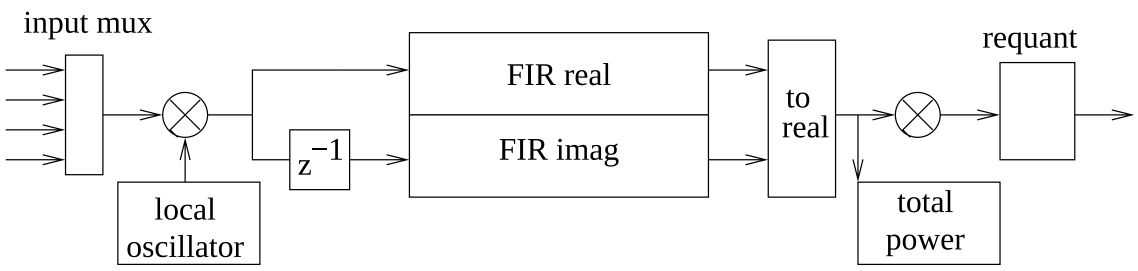

Figure 11: Digital baseband converter structure

The decimating filter uses tap recirculation and symmetry to reduce the number of required multipliers. For a band decimation by a factor D the filter uses $32D$ taps, symmetric, and a total of $32D_m$ tap values are stored to implement all filters up to a maximum decimation of D_m. 8×20 kb memory blocks are used to store tap values for decimations from 2 (128 MHz band) to 256 (0.5 MHz).

With this architecture a total of 32 real multipliers are used to obtain a self similar bandshape, with a usable band of 88% the Nyquist band. Filter coefficients are computed using the Remez algorithm, for an equiripple design. For higher decimations, with more than 500 taps per filter, the technique described in [10] and section 6.4 has been used. Filter coefficient computed using a $sin(t)/t$ and a Dolph-Chebyshew tapering result in slightly worse performance, but filter computation is much simpler, and can be performed automatically in the VHDL code. Parameters for both filters are listed in tab. 3 and plotted in fig. 12.

Filter type	Bandwidth	Pass-band	Stop-band
	% Nyquist	dB (p-p)	dB
Remez equiripple	88%	0.035	80
Dolph-Chebyshew	86%	0.015	72

Table 3: Digital baseband converter filter parameters

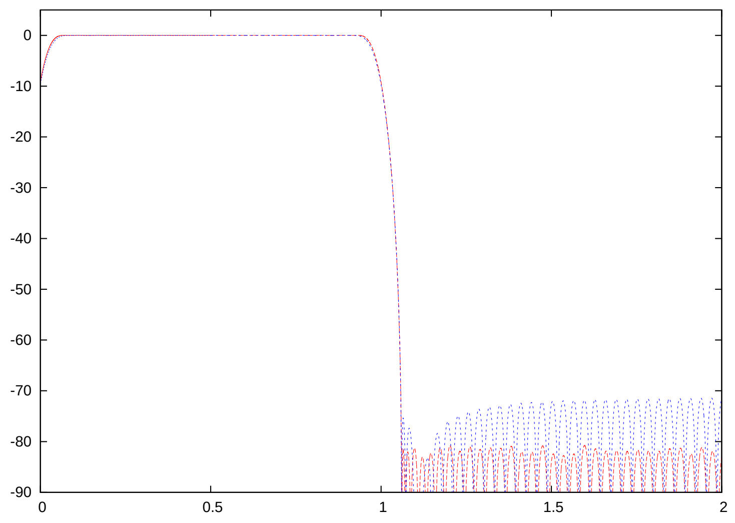

Figure 12: Digital baseband converter example bandwidth, using an equiripple Remez (red) and a Dolph-Chebyshew tapered $\sin(x)/x$ (blue) design

The FIR filters produce decimated and filtered complex samples, at a sample rate equal to the output bandwidth. A real valued data stream at twice the sample rate is obtained by shifting the imaginary portion by half decimated sample, and by up-converting the result by half band.

6 Polyphase filterbank

The polyphase filterbank concept is widely used in radio astronomy. It allows to channelize data with a controlled channel frequency profile, and with high channel-to-channel insulation. The theory behind the polyphase channel is described in [7]. It consists in multiplying the input data stream with an appropriate sliding window function, of a total length equal to a multiple of the

FFT size, before performing the FFT. The resulting response of each FFT channel is equal to the response of the FIR filter corresponding to the window function, translated to the channel center frequency.

The filter allows also to change the sample rate at the input of the FFT, allowing for an *oversampling* of the samples at the FFT output, as described in section 4.5. Common cases are:

- no oversampling (or critical sampling): data samples are just channelized without any data rate change. Channel separation is equal to sample frequency at the FFT output.
- twice oversampling: two samples are created for each input sample. Channel separation is equal to half the sample frequency at the FFT output. This allows a very simple FFT structure, maintaining the power-of-2 relation between original and unchannelized data, and does not require different clock domains. Output data rate is however twice than of the previous case.
- Fractional oversampling: channel separation is n/m times the channelized data rate, with n, m integers. Usually m is a power of 2. This design requires different clock domains for the input and output samples, and produce very odd output data rates, but allows for uniform spectral coverage with very limited increase in data rate.

The implementation of the polyphase filter depends on the parallel or serial nature of the FFT, and on the amount of oversampling. In this section only the polyphase filter is described. The FFT portion of the filterbank uses one of the FFT implementations in the `fft_lib` library, described in section 7.

6.1 Parallel polyphase filter

The structure for a N input double rate polyphase filter structure is shown in fig. 13(left). This structure is particularly suited for channelizing time multiplexed data, as in the 1st stage channelizer.

Each filter computes the convolution of the input data stream for samples i and $i+N$ (fig. 13 right). FFT uses a parallel, decimation in time architecture, which requires the input samples to be presented in bit reversed order. The bit reversal can be performed naturally using a hierarchic time demultiplexing of high speed samples. As the FFT output has a rate double with respect to the natural decimated input sample rate, the phase of the odd output channels rotates by π at every output sample. To correct this, FFT inputs i and $i + N$ are exchanged at each clock cycle.

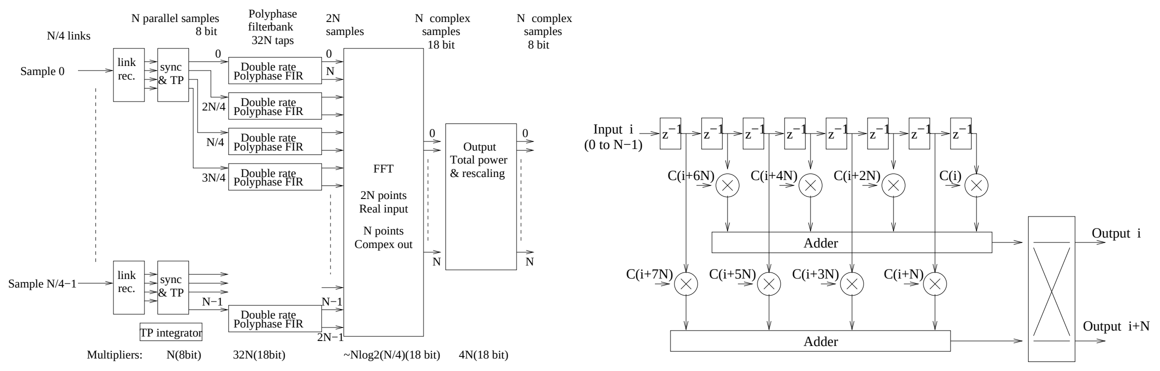

Figure 13: Generic polyphase filterbank structure (left) and filter element of a double rate (50% overlap) PFB (right)

The parallel polyphase filter output is then processed by a parallel FFT, described in section 7.1.

6.2 Polyphase serial channelizer

This channelizer is used on data sampled at the FPGA internal clock frequency (not time multiplexed), and is therefore used for the 2^{nd} stage channelizer, when the application requires uniformly spaced channels with constant channel width. In this case a polyphase filterbank is a more efficient solution with respect to individual channels. The general structure of a PFB is shown in fig. 14, and is similar to that of the first stage channelizer, with some differences:

- both the input filter and the FFT operate on serial data, instead of parallel data;
- the input data samples are complex, while the output samples are real;
- the origin of the FFT channelization can be shifted by $\pm 1/2$ fine channel, using a local oscillator and mixer;
- the FFT uses a radix-4 algorithm, thus allowing 4 channels to be processed in parallel;
- as only the central portion of the coarse channel is used, only half of the fine channels are processed. Thus each base block has 4 inputs and 2 outputs, with the fine channels grouped in serial frames.

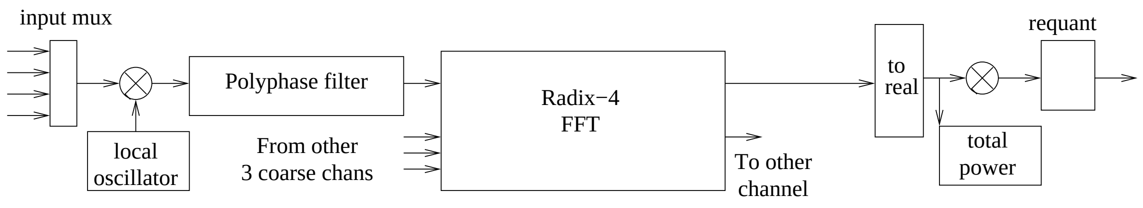

Figure 14: Polyphase filterbank block structure

The channel width and spacing are fixed for any particular design, but can be chosen with some degree of flexibility to suit different applications. Most VLBI continuum observations do not require a continuous frequency coverage and a non overlapping channelization scheme can be adopted. In this case the channel shaping filter is chosen to reject the adjacent channels to a high level, at the expense of a "hole" between channels.

6.3 Oversampling serial polyphase channelizer

For applications where a continuous frequency coverage is required, the filter portion is modified in order to slightly increase the output data rate, while maintaining the same channel spacing. This allows for a small overlap between adjacent channels that cover the "hole" at the filter edges that is needed to prevent aliasing. Feasible oversampling values are listed in table 4, together with the required filter resources, i.e. multipliers in the FIR section, and increase in the clock frequency.

To implement the oversampling a dual clock structure is used (fig. 15). The input samples are placed in a FIFO memory using the original sample rate, and extracted from the FIFO in longer blocks at the oversampled sample rate. Successive stages of the filter delay rewind the sample stream at the beginning of each FFT frame, to provide the required overlap in the time

o_f	Cutoff freq.	Stop freq.	N. of taps	N. of mult.	Data rate incr. (%)
32/26	$0.5/N$	$0.7308/N$	$12N$	24	23.1
32/27	$0.5/N$	$0.6852/N$	$15N$	30	18.5
32/28	$0.5/N$	$0.6428/N$	$18N$	36	14.3

Table 4: Oversampling factor and related polyphase filter characteristics for an N channel PFB

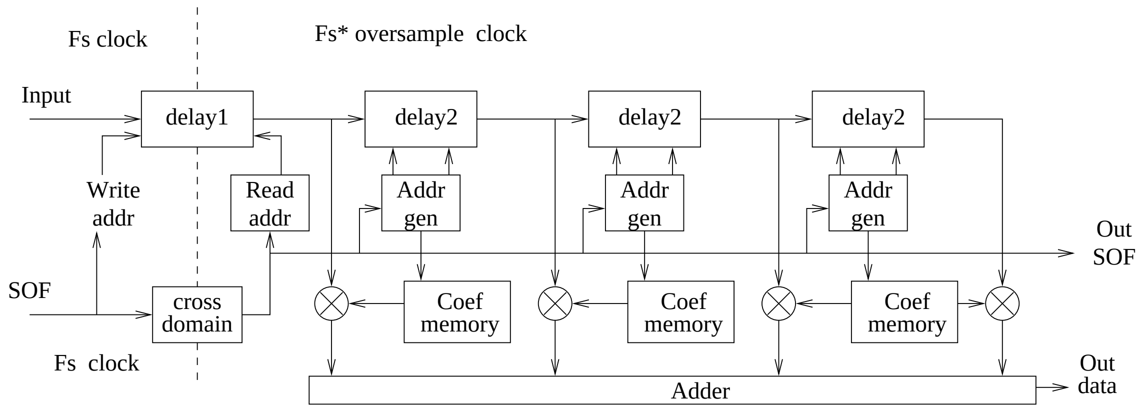

Figure 15: Serial oversampling polyphase filter

sequence. A phase rotation is needed at the FFT output to compensate for the fractional overlap between frames [7].

FFT engine operates in parallel on 4 independent coarse channels, for a total processed bandwidth of $\approx$ 500 MHz. The structure of the channel ordering is such that at any given moment only two outputs present data relative to the inner (good) portion of each overlapping coarse channel (see [12] for details). Therefore the output data rate from the FFT engine is half the input rate, or each FFT block will produce two output streams for a total bandwidth of 512 MHz.

FFT size depends on the required output channel width. For VLBI applications typical FFT sizes range from 4 to 64 channels, i.e. a 512 MHz segment of data produce from 8 to 128 fine channels.

Each block of 4 coarse channels requires about 170 real multipliers, mainly for the filter section. The total number of multipliers for 32 channels is thus 1500, less if not all coarse channels are processed.

Data is then stored in a corner turning memory and data buffer. If the number of fine channels is not excessive, local (internal) memory can be used. Arria10 FPGAs memory blocks have 20 kbit size, i.e. 8 blocks are required for a 8 KB (jumbo frame) dual buffering storage. With about 500 blocks available for buffering, 32 VDIF channels can be implemented. The number of output channels can be much larger, because the VDIF format allows several (up to the packet length) logical channels to be transmitted in one frame.

6.4 Matlab functions for filter coefficient calculation

Polyphase filters, especially serial and parallel/serial wideband filters, with hundreds or thousands of channels and a number of polyphase sections n_f of the order of 10-30, require very large filter orders, up to tens of thousands of taps. Filter tap calculation for very large orders is not simple. The built-in Matlab filter toolbox has been tested, but provided unsatisfactory results for the filter parameters needed in this application. A filter order of $\approx 10^4$ is usually computed using a windowing technique, i.e. is the product of a (infinite) $sin(x)/x$ function with an appropriate (finite) window function. This approach produces a stop-band with a rapidly increasing attenuation and, for a reasonable stop-band attenuation, a very low passband ripple. Both these characteristics result in a over-specification of the filter, at the expense of the resulting filter length. On the other hand equiripple algorithms, like the Parks-McClellan-Remez one, fail for orders exceeding a few hundreds.

For these reasons an ad-hoc design method has been developed. A filter suited for a reduced number of channels, n_s/M is designed using the Remez algorithm, where n_s is the original number of required channels. A C implementation of the algorithm developed by Jake Janovetz (janovetz@uiuc.edu) has been imported as a Matlab function for this purpose. The factor M, usually a power of 2, is chosen in order to obtain a converging solution for the algorithm. Experimentally the chosen C implementation always converges for filter orders $n_f n_s/M$ up to approximately 256.

Once the *reduced* filter is calculated, it is expanded by a factor M to the final size by Fourier transform, zero padding and back-transform. The expansion takes into account the discontinuity present at the FIR response edges, removing them before the expansion and reintroducing them, properly scaled, in the final filter.

The expansion produces a filter with a cutoff frequency scaled by a factor $1/M$, almost exactly the same fractional passband, and a stop-band starting at roughly the same attenuation but slowly decreasing at higher frequencies. The filter length is definitely smaller than for an equivalent filter obtained with a windowing technique. For example a filter for a 512 channel PFB ($n_s = 1024$) and an oversampling factor $O_f = 1.185$ requires $n_f = 14$, while windowed filters range from $n_f = 18$ to $n_f = 22$, i.e. a $\approx 30\%$ longer filter.

A set of Matlab functions implementing this algorithm are available, together with plotting functions for filter performance analysis. An example of a filter with the specifications given in the previous paragraph, for a single step wideband channelizer, is shown in figure 16. The filter tap values are written in a VHDL *package* file, that is directly used in the polyphase (or DBBC) modules to synthesize the filter coefficient memory.

7 FFT library

The FFT is probably the most used signal processing algorithm. The Uniboard[2] library contains some modules for serial and parallel FFT. As similar modules have already been developed for the Uniboard Digital Receiver, and present some advantages in terms of multiplier efficiency, these latter have been expanded in a general purpose library and used for this project.

Different implementations are available:

- Parallel FFT: module 2 Decimation-in-time (DIT) FFT with parallel complex input samples. The module performs a complete FFT at every clock cycle, and is used to channelize wideband time multiplexed signals.
- Real parallel FFT: module 2 Decimation-in-time FFT optimized for parallel real input samples. The unnecessary multipliers and output channels are automatically removed.
- Serial FFT: module 2 and module 4 Decimation-in-frequency (DIF) FFT with complex input samples. The module optimizes the multiplier usage by processing 2 or 4 input signals at the same time.
- Wideband serial FFT: processes wideband time multiplexed signals exploiting the capability of the serial FFT to process multiple signals (in this case multiple consecutive samples) in parallel.

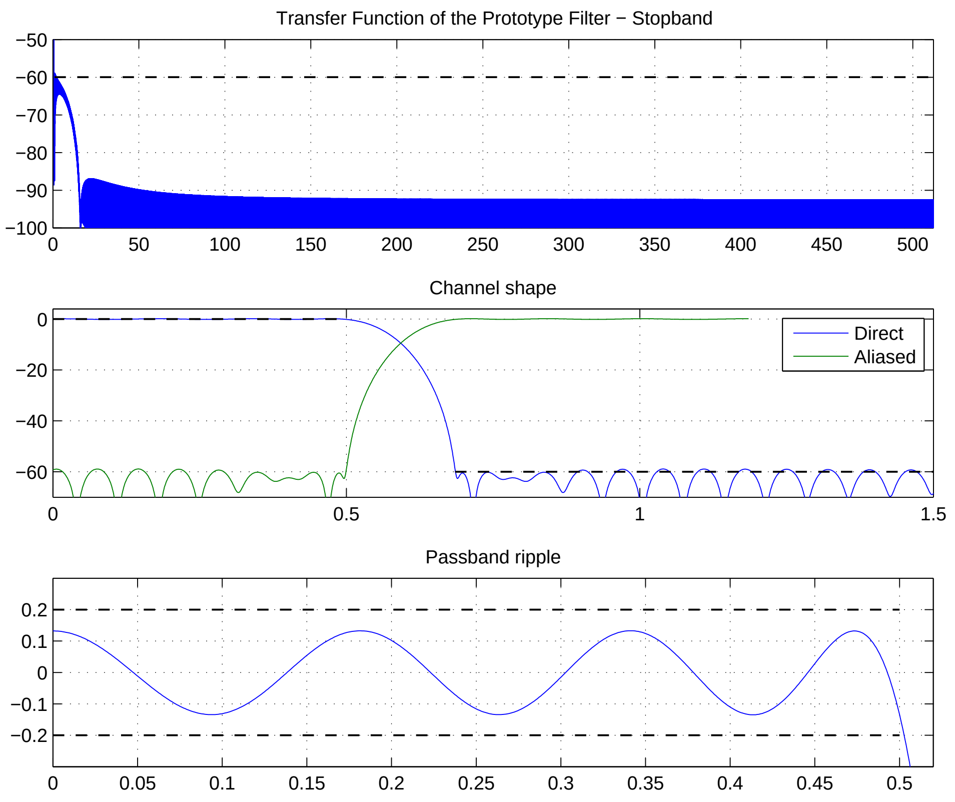

Figure 16: Response for an overlapping wideband serial/parallel PFB, with 1024 complex channels, $n_f = 14$, 32/27 oversampling

- Wideband serial FFT with real inputs: the real valued time multiplexed signal is converted to a complex valued signal, and processed in a half size wideband FFT. The frequencies k and $n/2 - k$ are then separated using the algorithm in [5].

Serial DIF with radix different from 2 and 4 are being added to the library. A radix-5 serial stage based on the Winograd algorithm, with 100% multiplier efficiency, has been developed. This allows for example to channelize a 400 MHz bandwidth signal (800 MS/s) into channels of exactly 1 MHz.

All modules process data internally with 18 bit precision. This value can be changed by adjusting a constant in a library package file. Data sample size is specified as a parameter, and some options for internal rescaling are available. In general as few re-scalings as possible are performed, as they introduce potential offsets with rounding.

Overflow is flagged by mapping the overflown samples to the maximum negative number. Overflown samples are propagated to all affected channels.

7.1 Parallel FFT block

This FFT is used in the double critically sampled polyphase filter described in section 6.1. It is a parallel input implementation for a decimation-in-time (DIT) radix-2 FFT.

The DIT implementation has been chosen because the input bit reversed sequence can be easily implemented in a time multiplexed sample stream. An example for a 8 GS/s ADC interfaced using 8 × 10 Gbps links is shown in figure 17. Groups of 8 consecutive ADC samples are sent to the Uniboard[2] using the 8 links. Data rate on each link is 1 GS/s, so further 1:4 demultiplexing is performed in the link interface, producing blocks of 32 consecutive samples in bit reversed order at a sample rate of 250 MS/s. Each sample is then processed in the double rate polyphase filter, producing two samples separated by 32 ADC clock periods. The polyphase filter is then organized as 64 samples, in the required bit reversed order.

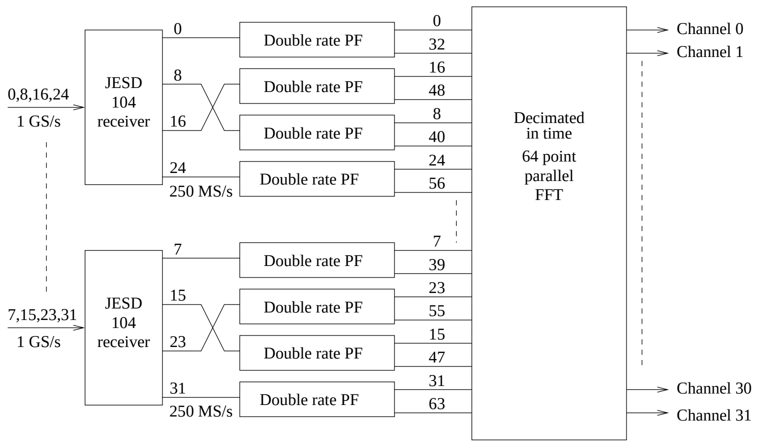

Figure 17: Example of natural bit reversed sorting of input samples for a 1:32 time multiplexed signal

The module is instantiated using a parametrized VHDL code, that replicates an elementary butterfly unit and a twiddle factor multiplicator. The twiddle factor is optimized to automatically recognize the special cases of multiplication by 1, i, and $\exp(i\pi/4)$. In the first two cases no multipliers are instantiated, the latter case is implemented with two real multipliers using the formula $\exp(i\pi/4)x = \sqrt{1/2}\,((1+i)x)$. Four multipliers are used for the general case.

The filterbank receives real valued samples at the input. The FFT can then be optimized for real input, halving the number of required multipliers by exploiting the conjugate relation $y(N-f) = y^*(f)$, and computing only half the outputs at each FFT stage. An example for $N = 16$ is shown in fig. 18. The required number n_m of (real) multipliers for a $2N$ FFT block (N real outputs) is slightly less than $2N\log_2(N/2)$.

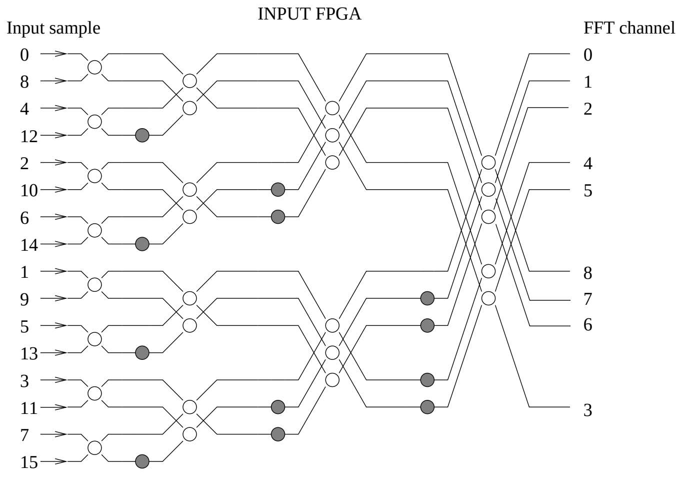

Figure 18: Real valued input 16 channel FFT

The FFT size, $2N$, depends on the input and output sample rates. For the baseline digital filter application the sample rate is 8 GHz and the internal sample frequency is 250 MHz, resulting in a convenient, power-of-2 ratio $N = 32$, and a FFT size of 64. The FFT block requires 204 real multipliers. The filter, for the band-shape of fig. 9, requires 512 taps, and 512 real multipliers.

The output channels must be rescaled, to compensate differences in signal level across the input band, and re-quantized, to reduce the complexity of the following stages. A total power detector is therefore included for each output channel, together with a programmable re-quantization to 8-12 bit resolution.

7.2 Serial FFT

The serial FFT is based on a DIF design with 100% multiplier usage efficiency. Both radix-2 and radix-4 architectures have been implemented. A DIF architecture has been chosen because it requires the input samples to be presented in the natural order. Frequency channels are produced in bit reversed order, but sorting can be implemented in the output channel selection stage.

The radix 2 and radix 4 FFT modules accept 2 or 4 frames (respectively) of n samples each, and transform them into 2 or 4 spectra of n frequency points in n clock cycles. They are composed of a sequence of $log_2(n)$, or $log_2(n)/2$ basic blocks in cascade. The last 2 blocks of the radix 2 FFT, and the last block of the radix 4 FFT do not use multipliers, the others use respectively 4 and 12 real multipliers each. The total number of multipliers n_m per input data stream is thus:

- for radix 2 FFT: $n_m = 2(log_2(n) - 2)$
- for radix 4 FFT: $n_m = 3/2(log_2(n) - 2))$

The radix 4 algorithm uses 25% less multipliers, but allows only transforms with a number of points that is an even power of 2 and require 4 input signals to process.

The basic blocks for radix 2 and radix 4 FFT are shown in figure 19.

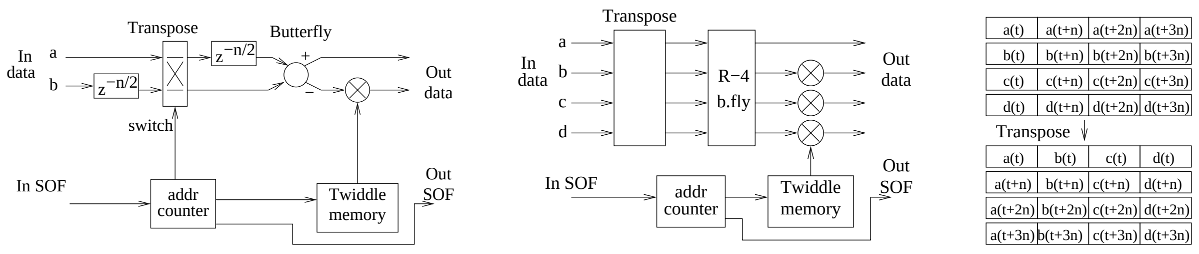

Figure 19: Radix 2 (left) and radix 4 (right) FFT blocks. Input signals are transposed to be processed sequentially, with samples at the proper spacing presented to the butterfly stage

The input signals are transposed in such a way to present at any time samples separated by $n/2$ (n/4 for radix 4) to the butterfly stage. Each input signal is processed in $n/2$ ($n/4$) contiguous clock cycles. By cascading these stages, a complete FFT is implemented. The output of the module is composed of 2 (4) sub-frames, each $n/2$ ($n/4$) samples long, on 2 (4) parallel streams. Each stream contains 1/2 (1/4) of contiguous frequency channels, in bit reversed order within each block.

7.3 Wide serial FFT with real input

In some applications that require a very narrow channelization on the whole bandwidth, a single stage FFT channelizer can be used directly on the wideband, time multiplexed signal. Real samples are combined together as the real and imaginary part of a complex sequence of half size. The algorithm then computes the transform of the sequence of $n/2$ complex values $S_{2j} + iS_{2j+1}$, and then combine channel j and $n/2 - j$ to form the actual real FFT outputs at these two frequencies.

The algorithm uses a radix-4 time multiplexed FFT, with a final parallel stage to combine together the time multiplexed samples. The limitation of the radix-4 FFT forces the possible FFT length to an even power of 2 multiplied by the time multiplexing value. A final specialized stage performs both bit-reversed reordering and separation of the frequencies k and $n/2 - k$, according to the algorithm described in [5].

An example for a 1:16 time multiplexed signal (e.g. 4 GS/s processed at 250 MHz) is shown in figure 20.

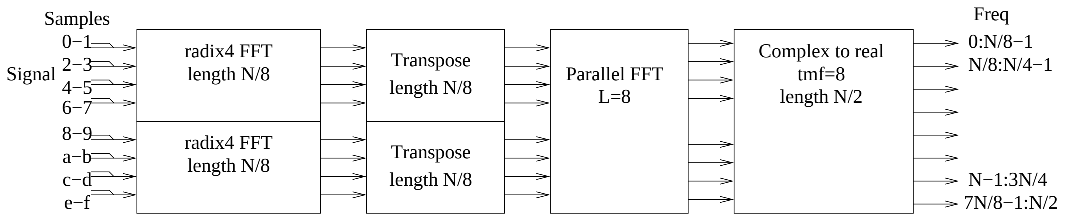

Figure 20: Wideband serial FFT with 1:16 time multiplexed, real valued input data

7.4 Radix 5 FFT

The limitation of a base 2 FFT, often imposes inefficient output sample rates. A radix-5 Fourier stage has been implemented using a Winograd algorithm. The module requires a total of 10 real multipliers, and accepts 5 samples at each clock cycle, producing a complete 5 point transform. Its architecture is shown in figure 21.

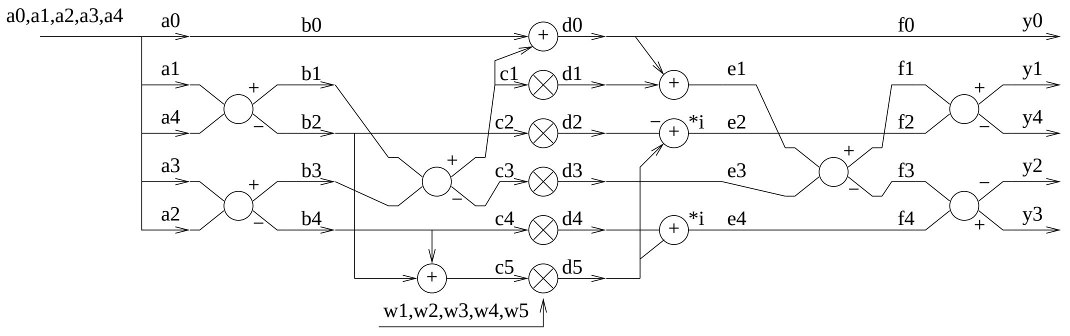

Figure 21: Winograd 5 point butterfly stage

The Winograd algorithm decomposes a Fourier transform of a short vector x into three matrix multiplications, $y = FWBx$, where F and B have only elements valued 0, ± 1 and $\pm i$, and are implemented with adders, and W is diagonal. In the case of $n = 5$ we have:

$$B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 1 & -1 \end{bmatrix} \quad W = \mathrm{diag} \begin{bmatrix} 1 \\ (C_1 + C_2)/2 - 1 \\ S_1 + S_2 \\ (C_1 - C_2)/2 \\ S_1 - S_2 \\ S_2 \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -i & 1 & 0 & i \\ 1 & 1 & 0 & -1 & -i & -i \\ 1 & 1 & 0 & -1 & i & i \\ 1 & 1 & i & 1 & 0 & -i \end{bmatrix} \tag{1}$$

where $C_1 = \cos(2\pi/5)$, $C_2 = \cos(4\pi/5)$, $S_1 = \sin(2\pi/5)$, $S_2 = \sin(4\pi/5)$.

This element can be used to implement FFT with a decimal number of points. As both the clock frequency and the required output sample rate are often expressed as decimal numbers this could overcome the disadvantage of using significant more resources.

An example of a 40 input FFT, appropriate for a clock decimation factor of 20, requiring a total of 170 multipliers, is shown in fig. 22.

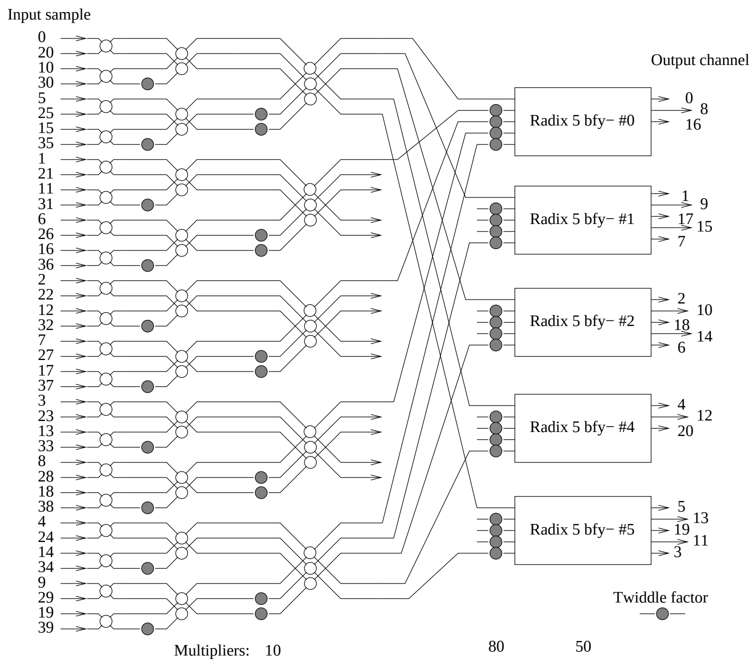

Figure 22: Real valued input 40 channel FFT

A parallel structure with mixed radix is unpractical, as the input data is seldom time multiplexed to a decimal time multiplexing factor. A serial structure with mixed radix also forces the number of parallel signals processed to a common multiple of the two radix, that is typically large (10 or 20 signals). A serial implementation of the 5 point stage is therefore preferable. The structure in figure 21 can be serialized, to accepts 5 consecutive samples and compute the 5-point Fourier transform in 5 clock cycles. Each of the stages shown in the figure is performed serially, using typically one sum/subtraction unit and 5 complex registers per stage. The multiplications are performed sequentially using only two real multipliers, with the 5 (real) coefficients w_i stored in a small table.

The serial input requires a large memory to reorder the data appropriately, both at the input and at the output of each radix-5 stage. An example of a 800 MS/s real valued FFT is shown in figure 23. It accepts 1:4 time multiplexed samples, at 200 MHz, and produces frames of 2×200 channelized samples every microsecond.

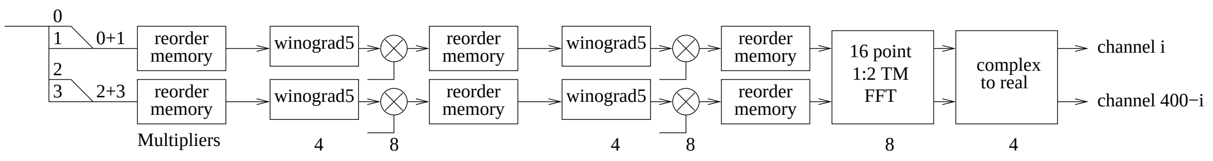

Figure 23: 800 point real valued FFT

8 Test signal generator

A test signal generator is a very useful complement of any digital signal processing system. It allows to test the system without a real signal being available, both during system setup and during actual operations.

For a radioastronomy instrument the test signal should include:

- a truly Gaussian white noise, with a RMS noise spectral density N, and a statistics good enough to perform deep integrations (at least several minutes) with a measurement noise σ_N in accordance to the radiometer equation: $\sigma_N = N/\sqrt{\tau B}$
- a few (2 in the Uniboard^2) pure sinusoidal tones, with power ranging from a small fraction to most (e.g. 90%) of the total power in the samples
- a comb of calibration tones with predictable phase.

A block schematics of the generator is shown in figure 24. The generator operates on a parallel sample representation of the data. Both the parallelization factor and the sample bit width can be as parameters during module instantiation.

All these components have programmable parameters (amplitude, frequency, phase) and are added to the input signal. The input signal is passed through the generator and can be blanked, to provide only the test signal.

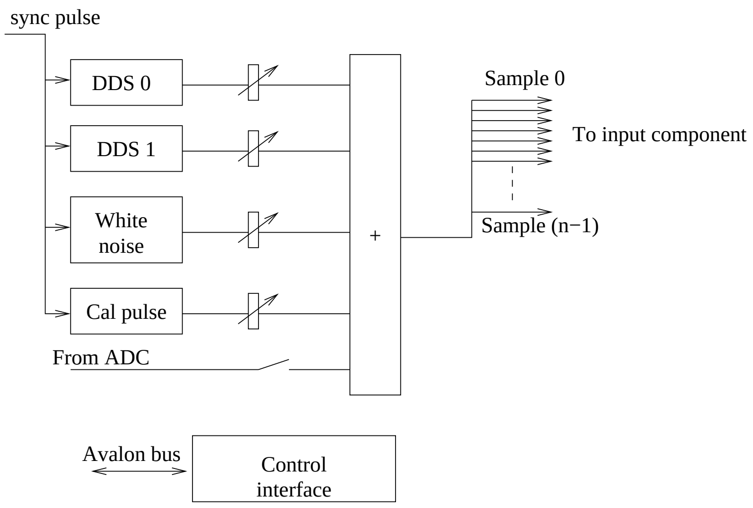

Figure 24: Test signal generator

The component is identical to the one used in the Uniboard and in the Beacon digital receivers, and is described in more detail in the relative Design Documents [3, 6].

The white noise is the sum of a set of 8 uniform random numbers, generated with the algorithm described in [4]. The statistics deviate slightly from a true Gaussian, as can be seen in figure 25. In particular the signal distribution is limited to $\pm 4.89\sigma$, and the kurtosis is $k = 0.2845$, slightly less than the $k = 3$ value of a true Gaussian distribution. The distribution is less peaked, and has lower tails, but approximates quite well the Gaussian for amplitudes up to $\pm 3\sigma$.

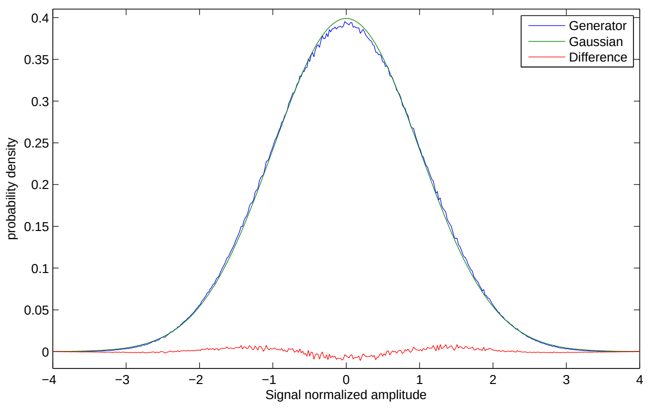

Figure 25: Distribution of samples generated by the white noise generator (blue) compared to a Gaussian distribution (green). Difference in red

A spectrum of a random noise plus two spectral lines is shown in figure 26. The lines have normalized frequencies of 0.071 and 0.25, and amplitudes with respect to the noise of -7 dB and -25 dB respectively.

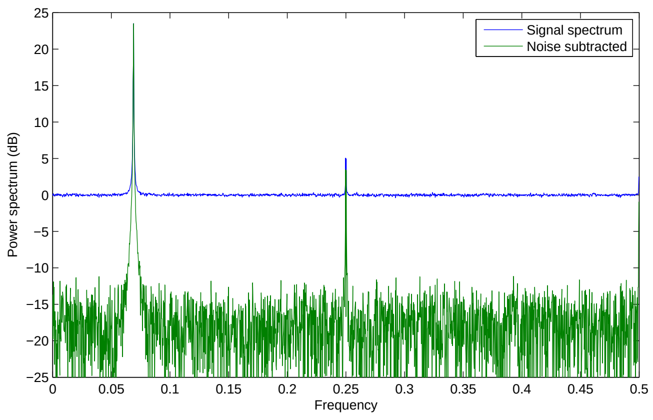

Figure 26: Spectrum of a test signal with noise plus two monochromatic lines. Upper graph: cumulative spectrum. Lower graph: same with noise subtracted

9 Framing and formatting the channelized samples

The channelized samples must be sent to another instrument for actual processing and extraction of the astronomic relevant informations. Using a standard UDP-IP frame format has several advantages:

- Commercial routers are available for this format, with quickly decreasing prices. Routing can then be performed in a completely programmable way, simply changing the address of each packet.
- An instrument can be implemented as an array of identical, relatively narrow-band sections. Different channels can then be routed to separate sections.
- Multicast routers can duplicate the data stream, in order to simultaneously use different instruments on the same data.
- WLAN network connections can be used to route the data over long distances, e.g. for E-VLBI.

The UDP format is better suited for this purpose, as it is quite simple to implement, without the need for a complex state machine and the overhead of a more complex, error correcting scheme like TCP. Individual data packets can be lost, but this does not seriously affect the overall performance of the instrument.

Some metadata must be transferred with the samples. Several formats have been developed in the radioastronomic community for this purpose.

9.1 VDIF format

The VDIF standard[13] has been developed for the storage and interchange of VLBI data. A VDIF packet is composed of a fixed size header and a frame of sequential samples. Samples may refer to a number (constrained to a power of 2) of synchronously sampled independent channels, and are sampled at a constant data rate. Sample size can range from 1 to 32 bits per sample, trading precision for bandwidth.

VDIF has a limitation in the sample frequency, that must exactly divide one second. This may be a problem for heavily channelized data (e.g. for SKA), or when oversampling has been adopted. For example the oversampling factors listed in table 4 produce sample frequencies with periods containing prime factors different from 2 and 5, and thus have non-integer frequencies.

VDIF packets do not need to be contiguous in time (e.g. for pulsar observations). An extra layer of packetization is then usually adopted, by preceding the VDIF header with a packet counter. In this way missing packets can be identified.

9.2 SPEAD format

The SPEAD standard has been developed as a very general protocol for astronomic data transport over a packetized interface (that can also be a generic computer storage file). It is described in [11].

The standard is extremely flexible. The data source and sink agree on the format, meaning and size of the data items being transferred, cumulatively denoted as the *heap*. Each packet may contain an arbitrary subset of these data, for example some or all of the metadata, or some or all of the sample (pixel) vector. Data items can have an arbitrary size and number of dimensions. The header contains, for each transmitted item, the item identifier (a small integer identifying the item) and either a pointer to the data value in the packet payload, or the item value, if it fits in the pointer size (immediate value). A special packet with item descriptors for all the data items is used to fully define the heap.

Due to its flexibility the standard is extremely complex, and not well suited for a FPGA implementation, but by deliberately limiting the implemented features it is possible to have a SPEAD-compliant data source within the capabilities of a FPGA-based formatter.

The implemented formatter uses a small subset of the available standard SPEAD keywords. Most parameters are specified as immediate values, i.e. they are stored in the header as a keyword with a value. Only widely accepted standard keywords are used, in order to avoid the necessity of transmitting a description packet before the actual data transfer. The packet content is always the same, i.e. the whole heap is transmitted with every packet.

If this is necessary, a programmable packet generator can be added to the interface structure. The packet containing the heap description is generated in the control computer, sent to a buffer inside the FPGA, and the buffer content queued to the Ethernet interface.

9.3 Formatter structure

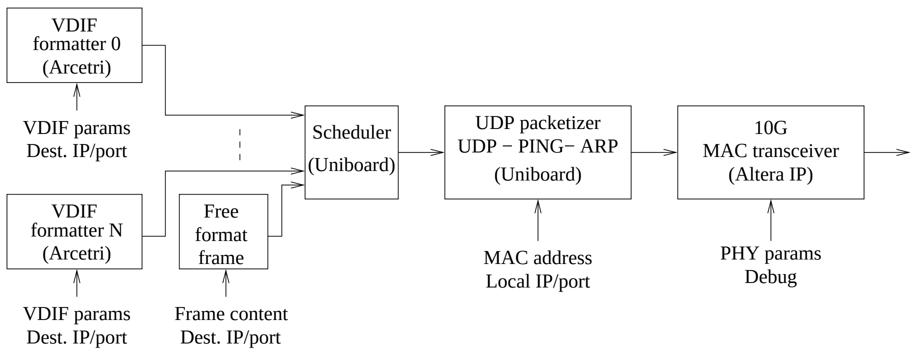
VDIF formatter 0 (Arcetri)
VDIF params Dest. IP/port
VDIF formatter N (Arcetri)
VDIF params Dest. IP/port
Free format frame
Frame content Dest. IP/port
Scheduler (Uniboard)
UDP packetizer UDP – PING– ARP (Uniboard)
MAC address Local IP/port
10G MAC transceiver (Altera IP)
PHY params Debug

Figure 27: Concept of the VDIF-SPEAD/Ethernet interface

The formatting and framing of the channelized data is performed in a similar way for all possible data formats. A block schematics of the formatter and UDP/IP interface is shown in figure 27.

Channelized data, represented as a stream of fixed size packets, is presented to a formatter. Timing information is implicitly contained in the framing, with the formatter knowing only the time for the first frame and the frame frequency. Each channel has its own separate formatter, that is programmed to provide the (fixed) metadata for that particular channel: sample bit size, destination IP address, IP frame length, etc. This allows for different destination addresses and different format for each channel.

The formatter generates a header with the required informations, preceded with a UDP header that includes the IP destination address, and appends the payload with the required number of data samples. For simplicity the UDP checksum is not generated (specified as zero, this is allowed in the UDP-IP protocol). Sample size is constrained to a *jumbo frame* Ethernet packet. The formatted packets are stored in a double buffer memory structure. A stream arbiter, developed as part of the standard Uniboard stream library, selects the first ready packet and queues it to an Ethernet UDP/IP standard core.

A free format buffer, that can be written from the control interface, allows the generation of a packet with a completely arbitrary content. This packet is queues and sent by the interface like the formatter generated ones. This possibility can be used, for example, to generate data item description packets for the SPEAD format.

The UDP/IP core provides the source address and UDP port, and the IP header and checksum, and implements the ARP protocol layer. This core is part of the standard Uniboard library. The formatted Ethernet packet is then transmitted using a commercial, vendor specific, Ethernet MAC IP.

Copyright