ASTRON
Netherlands Institute for Radio Astronomy

APERTIF Filter bank Firmware Specification

	Organisatie / Organization	Datum / Date
Auteur(s) / Author(s): Eric Kooistra	ASTRON	
Controle / Checked: Andre Gunst	ASTRON	
Goedkeuring / Approval: Andre Gunst	ASTRON	
Autorisatie / Authorisation: **Handtekening / Signature**	ASTRON	

Distribution list:

Group:	Others:
Rajan Raj Thilak Andre Gunst Gijs Schoonderbeek Daniel van der Schuur	Albert Jan Boonstra

Document history:

Revision	Date	Author	Modification / Change
0.1	2011-02-21	Eric Kooistra	Creation.
1.0	2011-03-04	Eric Kooistra	Updated after review with AG, GS, AJB.

Table of contents:

Terminology:

ADC	Analogue to Digital Convertor
APERTIF	Aperture Tile In Focus
DIF	Decimate In Frequency
DIT	Decimate In Time
DUT	Device Under Test
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HW	Hardware
IO	Input Output
IP	Intellectual Property
LOFAR	Low Frequency Array
MM	Memory-Mapped
PFB	Poly-phase Filter Bank
PFS	Pre Filter Structure
PFT	Pipelined FFT
RTL	Register Transfer Level
SNR	Signal to Noise Ratio
ST	Streaming
SW	Software
Subband	Frequency bin of the PFB
UNB	Path to UniBoard Firmware directory [9]

References:

1. “Detailed Design of the Digital Beamformer System for APERTIF”, G. Schoonderbeek, A. Gunst, E. Kooistra
2. "Understanding Digital Signal Processing", R. Lyons
3. "Handbook of Real-Time Fast Fourier Transforms", W.W. Smith, J. M. Smith
4. "RSP Firmware Design Description", LOFAR-ASTRON-SDD-018, Wessel Lubberhuizen
5. “Polyphase filter bank quantization analysis”, 2004, LOFAR-ASTRON-MEM-109, J. Stemerdink
6. “Quantization Error Analysis of Digital Signal Porcessing Blocks”, 2004, LOFAR-ASTRON-MEM-129, J. Stemerdink
7. "A New Approach to Pipeline FFT Processor", Shousheng He and Mats Torkelson
8. “Exploiting the modularity of pipeline FFTs for reusable design”, Raj Rajan Thilak
9. https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk, the UniBoard FP7 SVN repository ($UNB)
10. $UNB/Firmware/doc/howto
11. $UNB/Firmware/modules
12. $UNB/Firmware/modules/LOFAR
13. $UNB/Firmware/modules/LOFAR/pfs, contains the PFS
14. $UNB/Firmware/modules/LOFAR/pft2, contains the PFT and the PFB
15. $UNB/Firmware/modules/dsp
16. http://casper.berkeley.edu/wiki
17. "A Scalable Correlator Architecture Based on Modular FPGA Hardware and Data Packetization", Aaron Parsons e.a.
18. "Low-Frequency Interferometry: Design, Calibration, and Analysis Towards Detecting the Epoch of Reionization", Thesis 2009, Aaron Parsons
19. "A New Approach to Radio Astronomy Signal Processing: Packet Switched, FPGA-based, Upgradeable, Modular Hardware and Reusable, Platform-Independent Signal Processing Libraries", 200509URSI.pdf, Aaron Parsons e.a.
20. www.rfel.com
21. “UniBoard Firmware Platform”, ASTRON-RP381, E. Kooistra
22. “Specification for module interfaces using VHDL records”, ASTRON-RP380, E. Kooistra
23. “Heater Logic Module Description”, ASTRON-RP416, E. Kooistra
24. www.altera.com, stratix4_handbook.pdf
25. www.altera.com, quartusii_handbook.pdf

1 Introduction

1.1 Purpose

This document specifies the Poly-phase Filter Bank firmware module required for APERTIF [1].

1.2 Overview

Figure 1 shows the Poly-phase Filter Bank (PFB) with M=1024 that was used for LOFAR [4, 12]. The PFB consists of a FIR Pre Filter Structures (PFS) for complex input and a complex FFT. The FFT is a pipelined complex FFT (PFT) hence it can process two real inputs. Hence with M=1024 this then results in M/2 = 512 complex frequency subbands per real input. The PFB is a critically sampled filter bank, because the down sampling factor is the same as the FFT size. The PFB uses a pipelined architecture and is clocked at the same rate as the sample rate. The rate factor of sample clock frequency divide by the digital clock frequency is called P.

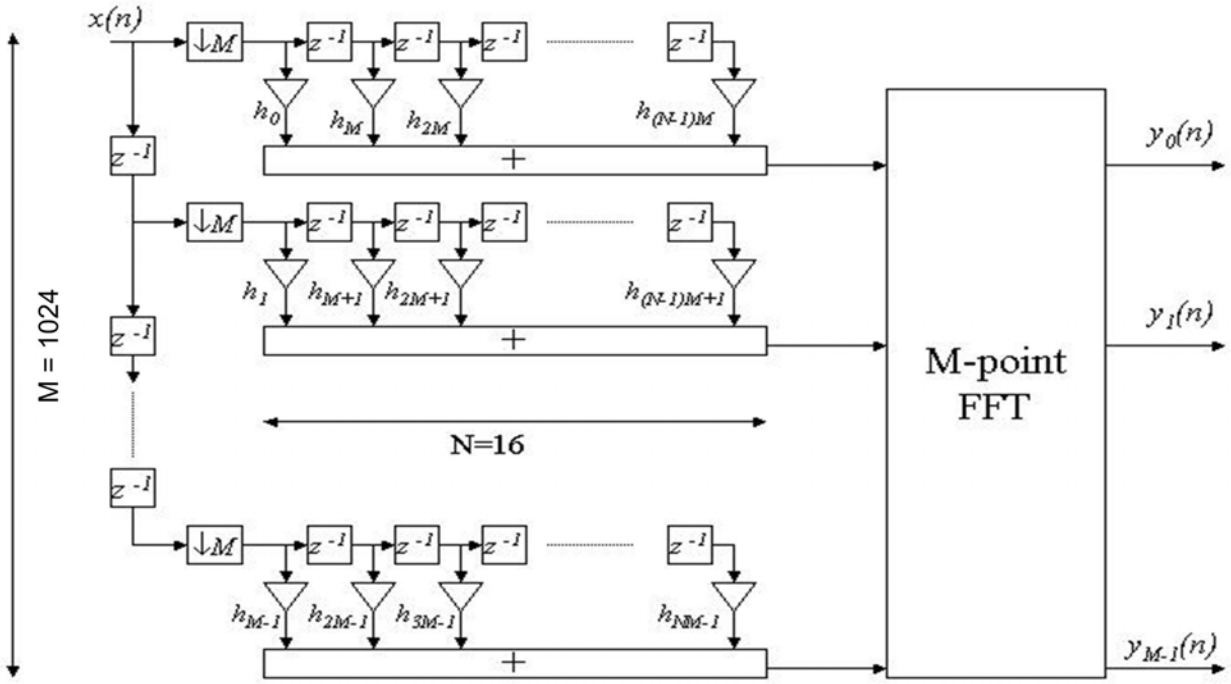

Figure 1: Poly-phase Filterbank

The APERTIF Poly-phase Filter Bank (PFB) must be wideband, so P ≥ 1. Typically P is a power of 2, so for APERTIF P = 4 to be able to process 800 Msps using a clock frequency of 200 MHz [1]. The pre filter structure (PFS) from LOFAR may be reused for the APERTIF PFB. Therefore the initial design focus lies on the Fast Fourier Transform (FFT) part of the PFB.

Determining the appropriate FIR coefficients for the PFS is not part of this filter bank firmware specification. For the PFB of Figure 1 the NM=16*1024 FIR coefficients from the LOFAR PFB can be used, these are available in Coeffs16384Kaiser-quant.dat [13]. Running the MATLAB script top.m in [14] nicely shows how the LOFAR PFB works for a sinus wave input.

1.3 Planning

The DSP IP work for the PFB is divided into phases that have clear goals.

- Phase I defines what is minimally required for a wide band PFB for the APERTIF beam former.
- Phase II defines more efficient usage of FPGA resources.
- Phase III defines additional PFB features.

Phase I starts with the selection of the pipelined FFT architecture, followed by the firmware implementation according to the specifications of section 2 and 3, and it ends with the deliverables listed in section 4.

1.3.1 Phase I

- Complex FFT
- Complex FFT with two real inputs
- PFB with two real inputs using a PFS
- Wideband FFT with two real inputs
- Wideband PFB with two real inputs using a PFS

1.3.2 Phase II

- Complex FFT with multiple ports (share twiddle factors)
- Wideband PFB with multiple ports (share FIR coefficients)

1.3.3 Phase III

Preliminary:

- Scramble rounding crosstalk that occurs between two real inputs using the (un)switch feature [4, 14]
- Oversampled or overlap PFB using a down sample factor that is less than the FFT size
- Support for FFT sizes and wideband factors P that are not a power of 2 [3]

2 Functional specification

2.1 General

The PFB firmware should be implemented in a generic and modular way. However its performance only needs to be evaluated for two settings:

1. APERTIF beam former: A wide band PFB for 512 subbands, 8-bit real input samples, 800 Msps
2. APERTIF correlator: A complex PFB for 64 channels, 4,4-bit complex input samples

2.2 Complex FFT

Figure 2 shows the PFT from LOFAR [4, 14]. The transform is a pipelined FFT.

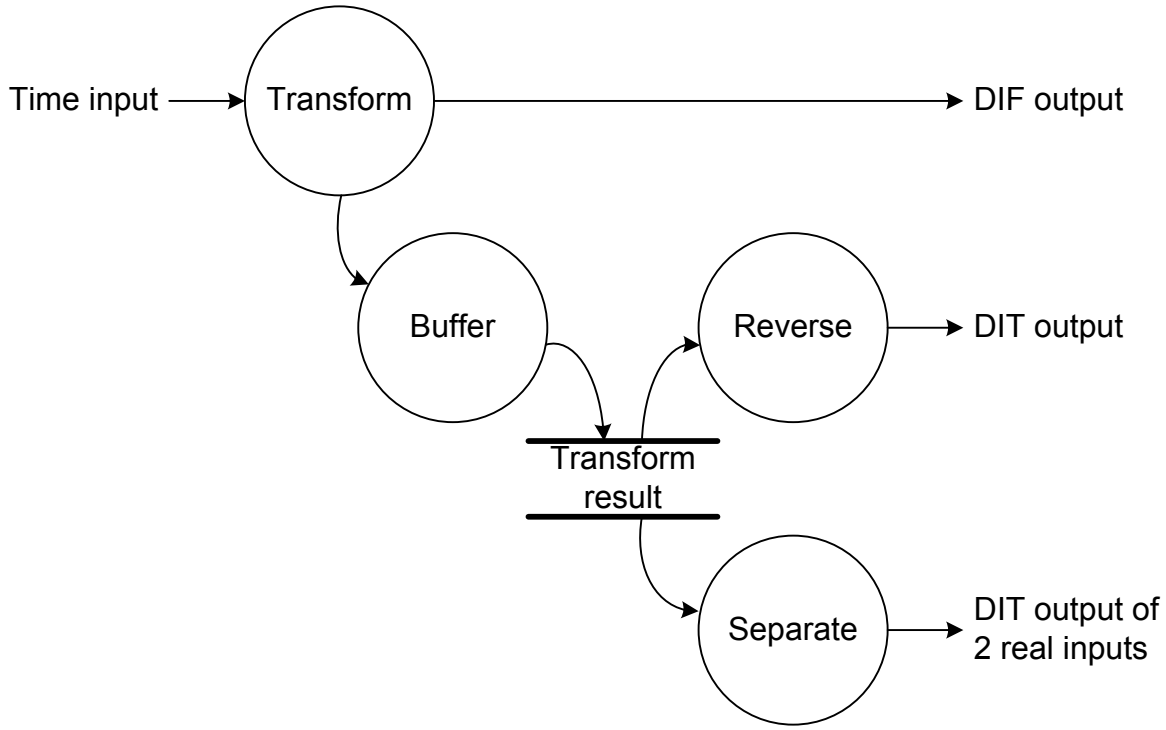

Figure 2: Data flow diagram of the PFT from [4]

Requirements:

1. The FFT size must be a generic parameter. At least powers of 4 have to be supported, so FFT sizes of 4, 16, 64, 256, 1024 points. Preferably also powers of 2 should be supported.
2. The FFT input data width and output data width must be independent generic parameters.
3. The FFT must support a generic parameter to choose the type of internal quantization, e.g. truncation or rounding.
4. The implementation loss due to internal quantization in the FFT must less be than 0.5 dB (?)
5. The FFT must use its multipliers efficiently (≥ 75% in time)
6. The FFT must support a data valid signal. During an FFT transform all input samples are valid. Between transforms there can be idle cycles.
7. The FFT must support a sync pulse that marks the (re)start of a transform.
8. The FFT must propagate the input sync and the input data valid taking account for the FFT latency.

In the LOFAR PFT the transform achieves 75% multiplier efficiency in time by using the radix 2^2 scheme [4, 7]. The transform in the CASPER PFT [16, 17, 18, 19] (called F-engine) achieves 100% multiplier efficiency in time by using a biplex, radix 2 scheme. Biplex implies that the CASPER PFT has two ports and operates on two complex input signals at the same time to achieve 100% multiplier efficiency instead of only 50%.

For more information on (re)quantization see [5, 6]. It is ok to assume that internally only FPGA DSP elements with 18x18 multipliers [24] are used, so only IO widths up to 18 bit need to be supported.

The sync input acts as a synchronous reset that marks the start of a new transform. If the sync pulse occurs in the cycle before the start of a new FFT transform, i.e. at most once every FFT size number of data valid clock cycles, then the sync input has no effect. If the sync is not used then the first valid sample starts the sequence of transforms.

2.3 Complex FFT with two real inputs

Requirements:

1. The FFT must also support using two real inputs instead of one complex input.

The separate function in Figure 2 separates the FFT output for the two real inputs. The theory for using a complex FFT to calculate the FFT for two real inputs is explained in [2].

2.4 PFB with two real inputs

Requirements:

1. The number of taps in the PFS must be a generic parameter.
2. The PFB must support using a complex input.
3. The PFB must also support using two real inputs instead of one complex input.

2.5 Wideband FFT with two real inputs

Requirements:

1. The wideband FFT must at least support factors P that are a power of 2.

The factor P equals the ADC sample clock frequency divided by the digital processing clock frequency. Figure 3 shows the principle drawing from [20] for a wide band FFT. Note that we define P=M, to reflect that the factor P results in P parallel input sections. A wideband PN-point PFT can be constructed from P N-point PFTs in parallel. The outputs of these P N-point PFTs are then passed through P P-point PFTs (?) to get the wideband FFT output.

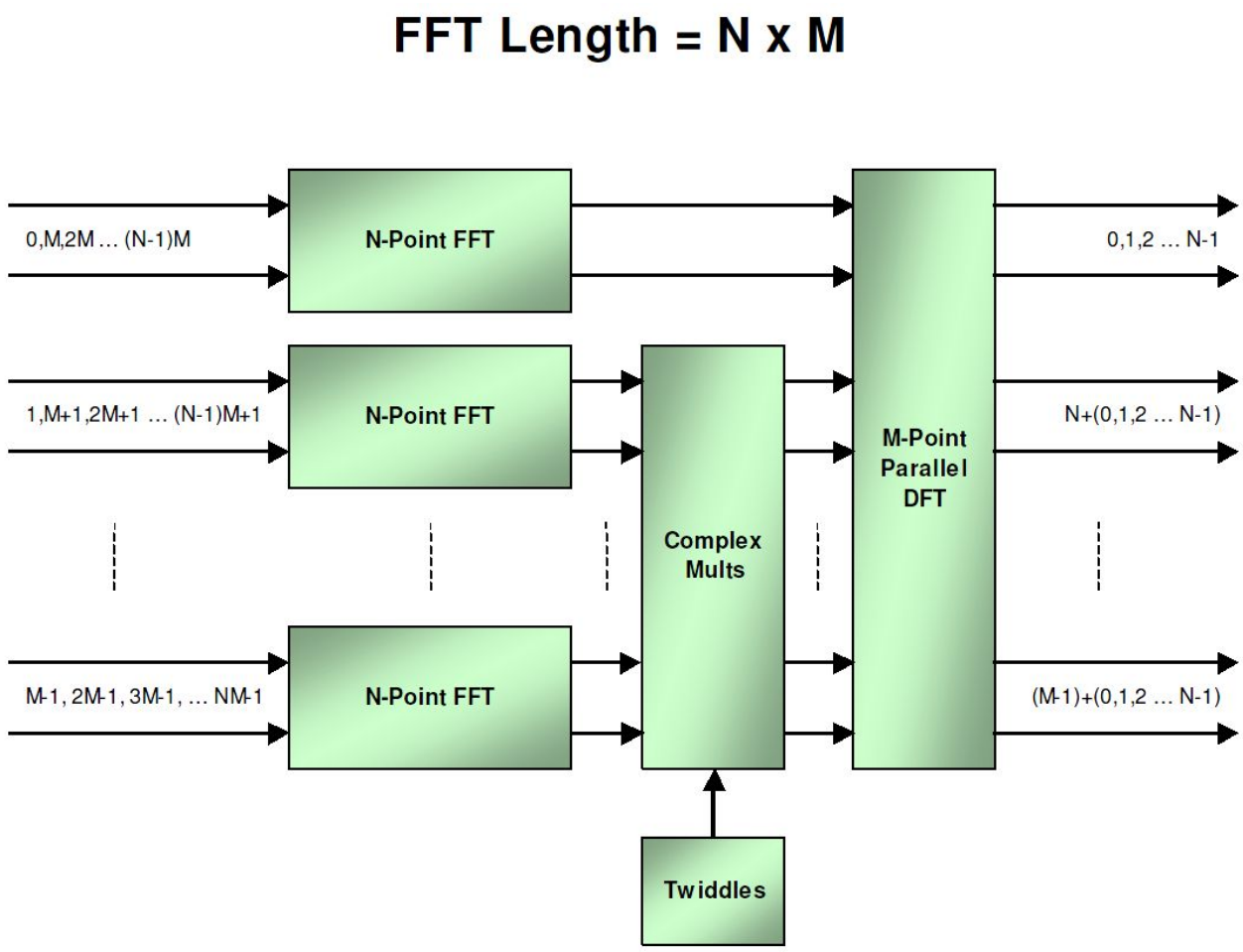

Figure 3: Wideband FFT (principle drawing from [20])

2.6 Wideband PFB with two real inputs

Requirements:

1. The wideband PFB must support real inputs.
2. The wideband PFB must at least support factors P that are a power of 2.

The wideband PFB combines the PFS of Figure 1 with the wide band PFT of Figure 3.

3 Implementation specification

3.1 Firmware

Requirements:

1. All PFB source files must be maintained in the UniBoard SVN repository [15] and follow the directory structure that is used for VHDL modules [11]. See how_to_use_SVN.txt [10].
2. The PFB firmware must be written in VHDL.
3. The PFB firmware must follow the VHDL coding style that is described in how_to_write_vhdl.txt [10].
4. The FPGA specific resources like multipliers and RAM must be instantiated via VHDL wrappers [21, 22].
5. The PFB usage of FPGA resources both in time and in space must be close to what can be theoretically expected, the LOFAR PFB forms the benchmark see Table 1.
6. The PFB should be synthesized for the Stratix IV PFGA (type EP4SGX230KF40C2) using the Quartus II synthesis tool.
7. The PFB must be able to clock at >300 MHz. The PFB should preferably be able to clock at >400 MHz. If >400 MHz is not feasible, then it should be made clear what the bottleneck is.

IP	Nof ALM	Nof DSP 18x18	Nof block RAM M9K	Fmax [MHz]
Complex FFT with DIF	2499	16	26	319
Complex FFT with DIT	2532	16	35	328
Complex FFT with 2 real inputs	2695	16	35	336
PFS for 2 real inputs	798	32	70	509
PFB with 2 real inputs	3427	48	105	328

Table 1: LOFAR PFB resource usage for the Stratix IV FPGA using the Quartus II synthesis tool

Table 1 shows the FPGA resource usage for the LOFAR PFS, PFT and PFB. The numbers were obtained with the pf_top.vhd entity and the pf_top.qpf synthesis project. The utilization numbers come from the resource utilization section by entity of the Quartus fitter report. The Fmax comes from the TimeQuest timing analyzer Fmax report. The Fmax numbers are indicative, because no effort has been made to investigate or improve them.

3.1.1 FPGA IP

It is possible to infer FPGA IP like DSP, RAM and FIFO that map efficiently on Stratix IV resources [24, 25]. However inferring FPGA IP and especially DSP elements is not trivial, because it depends on:

- the properties of these FPGA IP resources
- the capabilities of the synthesis tool

Therefore in practice it is better to use FPGA IP that is generated with the MegaWizard. The common module in [11] contains RTL to infer FPGA IP and generated MegaWizard FPGA IP. These FPGA IP components for DSP, RAM and FIFOs are wrapped by an entity in the common module. This wrapped component then has multiple architectures. E.g. one architecture can be the inferred RTL description and another can be an instantiated Stratix4 MegaWizard component. Also when the DSP or RAM is inferred, then it must be done so via a wrapped VHDL component [21] in the common module.

3.2 Verification requirements

Requirements:

1. The FFT output must be compared to the (ideal) floating point FFT for a full scale, uniform noise input to show that the implementation loss (due to quantization and internal rounding) is conform what can be expected.
2. The FFT and PFB must be verified for some characteristic signals like an impulse and a sinus
3. The PFS must be verified by checking that an impulse input signal yields the FIR filter coefficients.
4. The PFB must also be verified using a sinus signal or a stream of impulses with a period that does not integer divide the FFT size
5. A VHDL test bench must self check that the PFB VHDL works OK, i.e. conform the algorithm.

The LOFAR PFB firmware contains examples of test benches for the PFS [13] and of test benches with test signals for the PFT [14]. The test bench to self check that the PFB firmware works OK can operate similar as the tb_pfb2.vhd test bench that verifies the LOFAR PFB. This test bench merely verifies that the PFB output value after the PFB impulse response time is equal to a pre-determined known correct value. Note that the tb_pfb2.vhd test bench can only verify an input signal for which the signal period integer divides the FFT size.

3.2.1 Implementation loss

Due to limited internal word widths and limited output data width the FFT will decrease the SNR somewhat. This decrease is called the implementation loss. The implementation loss is defined by the difference of the output SNR of the implemented FFT with respect to the output SNR of an algorithm FFT that works with infinite precision. In practice infinite precision is assumed to be achieved by using double floating point numbers for the FFT calculations. Table 2 shows how the implementation loss of the FFT can be determined starting with a floating point time series signal xf. The function Q() denotes (re)quantization. The power of a signal is defined by the function P() as the sum of the squared sample values for one ore more FFT transforms. A VHDL test bench can read the xq signal from a file, apply it to the VHDL implementation of the FFT and then write the Xqqq output signal to another file for further analysis. The generation of xq and the analysis of Xqqq can be done e.g. with MATLAB.

Input	Function	Output	Comment
xf	FFTf	Xfff	= Float input, algorithm FFT, float output
xq=Q(xf)	FFTf	Xqff Xqfq=Q(Xqff) Eqfq=Xfff-Xqfq	= Quantized input, algorithm FFT, float output = Quantized input, algorithm FFT, quantized output = Algorithm FFT output error due to quantized input and output → SNRqfq = P(Xfff)/P(Eqfq)
xq	FFTq	Xqqq Eqqq=Xfff-Xqqq	= Quantized input, implementation FFT, quantized output = Implementation FFT output error for quantized input and output → SNRqqq = P(Xfff)/P(Eqqq)
			→ FFT implementation loss = SNRqfq – SNRqqq [dB]

Table 2: FFT implementation loss definition

4 Deliverables

4.1 Firmware

All source code must be available in the UniBoard SVN repository [9]. The code must compile, simulate and synthesize correctly.

4.2 Documentation

The APERTIF PFB must be documented according to the ASTRON report module document template [21]. Reference [23] serves as an example module document that uses this template:

Title
Distribution list
Document history
Table of contents
Terminology
References
1) Introduction
2) DSP algorithm
 - FFT architecture choice
 - Complex FFT, DIT, DIF, multiplier efficiency, memory efficiency
 - Generation of the twiddle factors
 - Separate function for two real inputs
 - Rounding schemes and impact on implementation loss compared to using floating points
 - Relevant MATLAB programs
3) Hardware interface
 - Parameter (generics)
 - Streaming IO (data, valid, sync)
4) Application
  - Simulation: A Modelsim do file that compiles the IP and the test benches
  - Synthesis:
    . Resource usage (compare with LOFAR PFB)
    . Speed
  - Known issues
5) Design
  - Top level (similar to Figure 2)
6) Implementation
  - Some relevant details, e.g. on how the FPGA DSP, RAM and FIFO resources are used
7) Verification
 - A VHDL test bench must self check that the PFB VHDL works OK, i.e. conform the algorithm
8) Appendix
 - List of files (VHDL, MATLAB, Python, Modelsim project files, Quartus II project files, etc)

The PFB module document should have no forward references. The template is such that chapter 1 and 2 are relevant to a DSP engineer. Chapter 3, 4 are relevant to a DSP / digital engineer that needs to apply the PFB. Chapter 5, 6, 7 and 8 are relevant to a digital engineer that needs to maintain the PFB. Chapter 6 can be small or may even be omitted. The PFB has no memory mapped (MM) interface. Therefore there is no Software interface section. Instead there is a DSP Algorithm section.