# Wideband Poly Phase Filter Module Description

| | Organisatie / Organization | Datum / Date |
|---|---|---|
| **Auteur(s) / Author(s):**<br>Harm Jan Pepping | ASTRON | 1 October 2012 |
| **Controle / Checked:**<br>Eric Kooistra | ASTRON | |
| **Goedkeuring / Approval:**<br>Andre Gunst | ASTRON | |
| **Autorisatie / Authorisation:**<br>**Handtekening / Signature**<br>Andre Gunst | ASTRON | |



**UniBoard**

**Doc.nr.:** ASTRON-RP-1390
**Rev.:** 0.1
**Date:** 13-05-2012
**Class.:** Public

## Distribution list:

| Group: | Others: |
|---|---|
| Andre Gunst<br>Eric Kooistra<br>Daniel van der Schuur | Gijs Schoonderbeek<br>Sjouke Zwier<br>Harro Verkouter (JIVE)<br>Jonathan Hargreaves (JIVE)<br>Salvatore Pirruccio (JIVE) |

## Document history:

| Revision | Date | Author | Modification / Change |
|---|---|---|---|
| 0.1 | 2013-5-13 | Harm Jan Pepping | Creation |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Table of contents:

## Terminology:

| | |
|---|---|
| Beamlet | Beamformed subband |
| BF | Beamformer |
| DIAG | Diagnostics (VHDL module) |
| DP | Data Path (VHDL module) |
| DSP | Digital Signal Processing |
| DUT | Device Under Test |
| EOP | End Of Packet |
| FFT | Fast Fourier Transformation |
| FIFO | First In First Out |
| FIR | Finite Impulse Response |
| FPGA | Field Programmable Gate Array |
| HDL | Hardware Description Language |
| IO | Input Output |
| MISO | Master In Slave Out |
| MM | Memory-Mapped |
| MOSI | Master Out Slave In |
| Nof | Number of |
| PFS | Poly Phase Filter Structure |
| PPF | Poly Phase Filter |
| RAM | Random Access Memory |
| Signal Path | Time series signal |
| SISO | Source In Sink Out |
| SOP | Start Of Packet |
| SOPC | System On a Programmable Chip (Altera) |
| SOSI | Source Out Sink In |
| SRC | Source |
| ST | Streaming |
| Subband | Frequency signal |
| WPFB | Wideband Poly Phase Filter Bank |

## References:


1. ‘APERTIF Filter Bank Firmware Specification Part 2’, ASTRON-SP-054, Eric Kooistra
2. ‘A Radix-2 Single Delay Feedback (R2SDF) architecture based generic Fast Fourier Transform for firmware implementation’, ASTRON-RP-755, R.T. Rajan
3. ‘UniBoard Wideband-FFT Module Description’, ASTRON-RP-1350, Harm Jan Pepping
4. ‘DIAG Module Description’, ASTRON-RP-1313, Eric Kooistra, Harm Jan Pepping
5. https://casper.berkeley.edu/wiki/The_Polyphase_Filter_Bank_Technique
6. $UNB/Firmware/dsp/filter/
7. $UNB/Firmware/dsp/wpfb/
8. $UNB/Firmware/dsp/wpfb/tb/python/
9. ‘Apertif Beamformer Firmware Description’, ASTRON-RP-1377, Daniel van der Schuur

# 1 Introduction

## 1.1 Purpose

The wpfb unit is a Wideband Poly Phase FilterBank which can process a datastream that is partly applied in serial and partly applied in parallel. It consists of a Poly Phase Filter (PPF) and a Fast Fourier Transform (FFT). The wpfb distinguishes from a single FFT by the PPF that functions as a window function in order to produce a flat response across the channel, but also provides excellent suppression of out-of-band signals.

## 1.2 Module overview

An overview of the wpfb unit is shown in Figure 1. The wpfb is build out of two N-point wideband PPFs (one for the real input and one for the imaginary input) and a single N-point wideband FFT. All P outputs are connected to a subband statistic unit that calculates the power in each subband. The MM interface is used to read and write the filter coefficients and to read out the subband statistics.

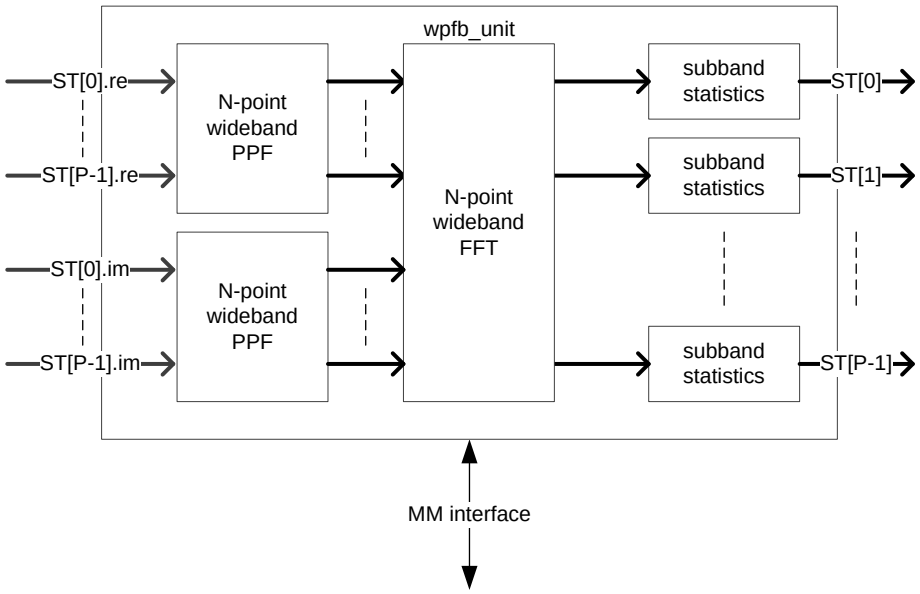


**Figure 1 wpfb unit overview**

The wideband FFT is already developed and described in detail in [3]. This document describes the wideband PPF in detail and the wpfb unit as a whole.

# 2 Firmware interface

This chapter covers all firmware interface related topics of the wpfb unit. It describes the functionality of the in- and output ports.

## 2.1 Clock domains

The wpfb unit receives two clocks. The memory mapped clock, mm_clk that allows interfacing with the host (SOPC system with NIOS processor) and the datapath clock dp_clk. Both clocks are accompanied with a reset, mm_rst and dp_rst.

| Name | Frequency (MHz) | Description |
|---|---|---|
| DP_CLK | 200 MHz | Clock for the datapath |

| | | |
|---|---|---|
| MM_CLK | 125 MHz | Clock for the memory mapped interface. |

**Table 1 wpfb unit clocks**

## 2.2 Parameters

| Generic | Type | Description |
|---|---|---|
| g_wpfb | t_wpfb | This record in defined in wpfb_pkg.vhd. It contains parameters that define the behavioural of the wpfb. Therefore it contains parameters that configure the filter part and parameters that configure the FFT part. An overview of the content of the t_wpfb record is detailed in **Error! Reference source not found.**. |
| g_use_bg | boolean | When set to TRUE the output of the wpfb will be driven by a block generator. The block generator is configurable via a memory mapped interface. |
| g_file_index_arr | t_nat_natural_arr | Array with numbers used to index the files with the coefficients. |
| g_coefs_file_prefix | string | File prefix, including the path for pointing the files that contain the filter coefficients. |

**Table 2 wpfb unit parameters**

The items of the t_wpfb record are listed in. The column named “value” specifies the default value that is used.

| Generic | Type | Value | Description |
|---|---|---|---|
| use_reorder | BOOLEAN | true | When set to ‘true’, the output bins of the FFT are reordered in such a way that the first bin represents the lowest frequency and the highest bin represents the highest frequency. |
| wb_factor=P | NATURAL | 4 | The number that defines the wideband factor. It defines the number of parallel pipelined FFTs. |
| nof_points=N | NATURAL | 1024 | The number of points of the FFT, but also the number of subbands for the PPF filter. |
| nof_taps=T | NATURAL | 16 | The number PPF taps per subband. |
| fil_in_dat_w | NATURAL | 8 | The width of the input data for the PPF filter. |
| fil_out_dat_w | NATURAL | 16 | The output width of the data of the PPF filter. |
| coef_dat_w | NATURAL | 16 | The width in bits of the coefficients that are applied in the PPF filter. |
| use_reorder | BOOLEAN | true | When set to true the output of the FFT will be reordered. |
| use_separate | BOOLEAN | true | When set to ‘true’ a separate algorithm will be enabled in order to retrieve two separate spectra from the output of the complex FFT in case both the real and imaginary input of the complex FFT are fed with two independent real signals. |
| fft_in_dat_w | NATURAL | 8 | Width in bits of the input data for the FFT. This value specifies the width of both the real and the imaginary part. |
| fft_out_dat_w | NATURAL | 14 | The bitwidth of the real and imaginary part of the output of the FFT. The relation with the in_dat_w is as follows: out_dat_w = in_dat_w + (log2(nof_N))/2+1 |
| stage_dat_w | NATURAL | 18 | The bitwidth of the data that is used between the stages (= DSP multiplier-width) |
| guard_w | NATURAL | 2 | Number of bits that function as guard bits. The guard bits are required to avoid overflow in the first two stages of the FFT. |
| guard_enable | BOOLEAN | true | When set to ‘true’ the input is guarded during the input resize function, when set to ‘false’ the input is not guarded, but the scaling is not skipped on the last stages of the FFT |

| | | | |
|---|---|---|---|
| | | | (based on the value of guard_w). |
| stat_data_w | POSITIVE | 56 | Width of the output of the subband statistics unit. This value must be high enough to accommodate the highest possible bitgrowth in the subband statistic unit. Highest integration period is set to one second. |
| stat_data_sz | POSITIVE | 2 | This number specifies how many 32-bit registers are required to read out one accumulated value. |
| pft_pipeline | t_fft_pipeline | | This record contains pipeline settings for the parallel FFTs. The record is defined in rTwoSDFPkg.vhd, which is located in the rTwoSDF library. More info can be found in [3]. |
| fft_pipeline | t_fft_pipeline | | This record contains pipeline settings for the pipelined FFTs. The record is defined in rTwoSDFPkg.vhd, which is located in the rTwoSDF library. More info can be found in [3]. |
| fil_pipeline | t_fil_ppf_pipelilne | | This record contains pipeline settings for the PPF filter. An overview of the t_fil_ppf_pipeline record fields Is given in Table 4. |

**Table 3 t_wpfb record fields**

| Generic | Type | Value | Description |
|---|---|---|---|
| mem_delay | NATURAL | 1 | Specifies the read latency for the memory. |
| mult_input | NATURAL | 1 | Input register of the multiplier. |
| mult_product | NATURAL | 1 | The latency of the multipliers that are used in the FIR filter. |
| mult_output | NATURAL | 1 | Output register of the multiplier. |
| adder_stages | NATURAL | 1 | Latency of each stage in the adder tree. |
| requant_remove_lsb | NATURAL | 1 | Register in the requantizer. |
| requant_remove_msb | NATURAL | 0 | Register in the requantizer. |

**Table 4 t_fil_ppf_pipeline record fields**

## 2.3 Interface signals

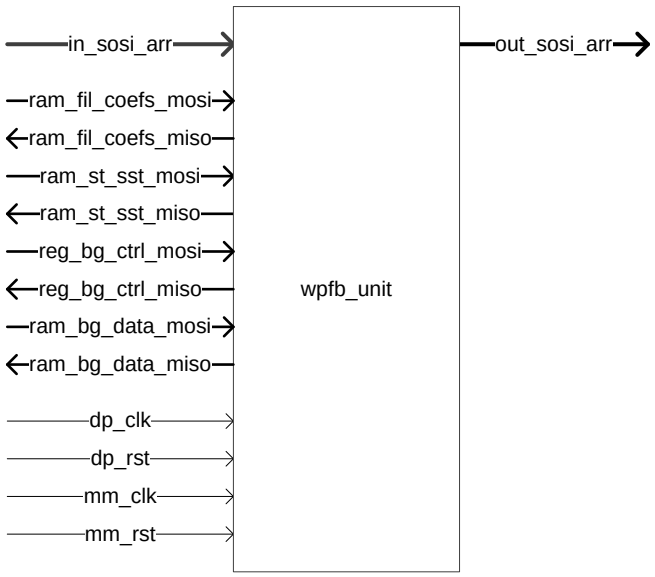


**Figure 2 wpfb unit interface signals**

| Interface | Type | Size or Span | Description |
|---|---|---|---|
| in_sosi_arr | t_dp_sosi_arr | wb_factor | Array of input streams where each stream holds 1/P portion of the time-domain input data. Data can be single complex or dual real. Format of the data is explained in detail in |
| out_sosi_arr | t_dp_sosi_arr | wb_factor | Array of output streams containing the frequency-domain data. Format of the data is explained in detail in 2.3.2. |
| ram_fil_coefs_mosi | t_mem_mosi | 2*nof_taps * nof_points | A mosi interface to read and write the filter coefficients. |
| ram_fil_coefs_miso | t_mem_miso | 2*nof_taps * nof_points | A mosi interface to read and write the filter coefficients. |
| ram_st_sst_mosi | t_mem_mosi | nof_points | A mosi interface to read out the subband statistics data. |
| ram_st_sst_miso | t_mem_miso | nof_points | A miso interface to read out the subband statistics data. |
| reg_bg_ctrl_mosi | t_mem_mosi | nof_points | A mosi interface to read and write the control settings for the block generator. |
| reg_bg_ctrl_miso | t_mem_miso | nof_points | A miso interface to read and write the control settings for the block generator. |
| ram_bg_data_mosi | t_mem_mosi | nof_points | A mosi interface to read and write the waveform data for the block generator. |
| ram_bg_data_miso | t_mem_miso | nof_points | A miso interface to read and write the waveform data for the block generator. |
| dp_clk | std_logic | na | Datapath clock |
| dp_rst | std_logic | na | Datapath reset |
| mm_clk | std_logic | na | Memory mapped interface clock |
| mm_rst | std_logic | na | Memory mapped interface reset |

**Table 5 wpfb unit interface signals**

### 2.3.1 IN_SOSI_ARR interface

The format for the in_sosi_arr is explained in detail in chapter 2.3.1 of [3].

### 2.3.2 OUT_SOSI_ARR interface

The format for the out_sosi_arr is explained in detail in chapter 2.3.2 of [3].

### 2.3.3 RAM_FIL_COEFS_MOSI interface

The filter coefficients can be read and written via the ram_fil_coefs_mosi interface. The address span of this interface is determined by 2*nof_taps*nof_points, where the 2 is due to complex input which allows to provide different coefficients for the real and imaginary channel.

| Signal | Type | Description |
|---|---|---|
| ram_fil_coefs_mosi.address[14:0] | MOSI | Word address range, supporting 2*16*1024 = 32768 registers. |
| ram_ fil_coefs _mosi.wrdata[31:0] | MOSI | Write data word, must be valid when wr is asserted. |
| ram_ fil_coefs _mosi.wr | MOSI | Write strobe. |
| ram_ fil_coefs _mosi.rd | MOSI | Read strobe. |
| ram_ fil_coefs_miso.rddata[31:0] | MISO | Read data word which is valid one clock cycle after assertion of rd. |

**Table 6 ram_fil_coefs_mosi interface**

### 2.3.4 RAM_ST_SST_MOSI interface

The ram_st_sst_mosi is explained in detail in chapter 2.3.3 of [3].

### 2.3.5 REG_BG_CTRL_MOSI and RAM_BG_CTRL_MOSI

The interface for the block generator is described in detail in [4].

### 2.3.6 Clocks and resets

Table 7 shows an overview of the clocks and reset signals that are available on the wpfb unit.

| Signal | Type | Description |
|---|---|---|
| dp_clk | Clock | Clock input for the datapath interface of the wpfb unit. |
| dp_rst | Reset | Reset input for the datapath clock domain registers. |
| mm_clk | Clock | Clock input for the memory mapped interface parts of the wpfb unit. |
| mm_rst | Reset | Reset input for the memory mapped interface parts of the wpfb unit. |

**Table 7 Clocks and resets**

# 3 Software interface

This chapter describes the software interface for the wpfb unit. The wpfb unit contains three register spans: the filter coefficients, the subband statistics and the block generator. Every span is described in the following paragraphs.

## 3.1 Filter Coefficients span

The filter coefficients can be read and written via the ram_fil_coefs_mosi interface. The order of the coefficients in the address span is determined by P, nof_taps, nof_points and the number of inputs (=2, real and imaginary). The coefficients are grouped according to the tap they belong to. Table 8 gives an overview of the registers where the filter coefficients are stored, where P = 4, nof_taps =16, nof_points = 1024 and the number of inputs = 2. The register name is composed of a “coef” number that corresponds to the sample/subband number, a tap number, a path number and a tag that shows which input (real or imaginary). The width of the registers is 16-bit.

| Name | Address (words) | Size (bits) | Read/ Write | Description |
|---|---|---|---|---|
| coef_0_tap_0_path_0_real | 0x0 | 16 | r/w | Coef for tap 0 for sample 0 on path 0, real |
| coef_1_tap_0_path_0_real | 0x1 | 16 | r/w | Coef for tap 0 for sample 1 on path 0, real |
| ---------- | ------ | --- | ----- | ---------- |
| coef_254_tap_0_path_0_real | 0xFE | 16 | r/w | Coef for tap 0 for sample 254 on path 0, real |
| coef_255_tap_0_path_0_real | 0xFF | 16 | r/w | Coef for tap 0 for sample 255 on path 0, real |
| coef_0_tap_1_path_0_real | 0x100 | 16 | r/w | Coef for tap 1 for sample 0 on path 0, real |
| coef_1_tap_1_path_0_real | 0x101 | 16 | r/w | Coef for tap 1 for sample 1 on path 0, real |
| ---------- | ------ | --- | ----- | ---------- |
| coef_254_tap_1_path_0_real | 0x1FE | 16 | r/w | Coef for tap 1 for sample 254 on path 0, real |
| coef_255_tap_1_path_0_real | 0x1FF | 16 | r/w | Coef for tap 1 for sample 255 on path 0, real |
| coef_0_tap_2_path_0_real | 0x200 | 16 | r/w | Coef for tap 2 for sample 0 on path 0, real |
| coef_1_tap_2_path_0_real | 0x201 | 16 | r/w | Coef for tap 2 for sample 1 on path 0, real |
| ---------- | ------ | --- | ----- | ---------- |
| ---------- | ------ | --- | ----- | ---------- |
| coef_254_tap_15_path_0_real | 0xFFE | 16 | r/w | Coef for tap 15 for sample 254 on path 0, real |
| coef_255_tap_15_path_0_real | 0xFFF | 16 | r/w | Coef for tap 15 for sample 255 on path 0, real |
| coef_0_tap_0_path_1_real | 0x1000 | 16 | r/w | Coef for tap 0 for sample 0 on path 1, real |
| coef_1_tap_0_path_1_real | 0x1001 | 16 | r/w | Coef for tap 0 for sample 1 on path 1, real |
| ---------- | ------ | --- | ----- | ---------- |
| ---------- | ------ | --- | ----- | ---------- |
| coef_254_tap_15_path_1_real | 0x1FFE | 16 | r/w | Coef for tap 15 for sample 254 on path 1, real |
| coef_255_tap_15_path_1_real | 0x1FFF | 16 | r/w | Coef for tap 15 for sample 255 on path 1, real |
| coef_0_tap_0_path_2_real | 0x2000 | 16 | r/w | Coef for tap 0 for sample 0 on path 2, real |
| coef_1_tap_0_path_2_real | 0x2001 | 16 | r/w | Coef for tap 0 for sample 1 on path 2, real |
| ---------- | ------ | --- | ----- | ---------- |
| ---------- | ------ | --- | ----- | ---------- |
| coef_254_tap_15_path_2_real | 0x2FFE | 16 | r/w | Coef for tap 15 for sample 254 on path 2, real |
| coef_255_tap_15_path_2_real | 0x2FFF | 16 | r/w | Coef for tap 15 for sample 255 on path 2, real |
| coef_0_tap_0_path_3_real | 0x3000 | 16 | r/w | Coef for tap 0 for sample 0 on path 3, real |
| coef_1_tap_0_path_3_real | 0x3001 | 16 | r/w | Coef for tap 0 for sample 1 on path 3, real |
| ---------- | ------ | --- | ----- | ---------- |

| | | | | |
|---|---|---|---|---|
| ---------- | ------ | --- | ----- | ---------- |
| coef_254_tap_15_path_3_real | 0x3FFE | 16 | r/w | Coef for tap 15 for sample 254 on path 3, real |
| coef_255_tap_15_path_3_real | 0x3FFF | 16 | r/w | Coef for tap 15 for sample 255 on path 3, real |
| coef_0_tap_0_path_0_imag | 0x4000 | 16 | r/w | Coef for tap 0 for sample 0 on path 0, imag |
| coef_1_tap_0_path_0_imag | 0x4001 | 16 | r/w | Coef for tap 0 for sample 1 on path 0, imag |
| ---------- | ------ | --- | ----- | ---------- |
| ---------- | ------ | --- | ----- | ---------- |
| coef_254_tap_15_path_3_imag | 0x7FFE | 16 | r/w | Coef for tap 15 for sample 254 on path 3, imag |
| coef_255_tap_15_path_3_imag | 0x7FFF | 16 | r/w | Coef for tap 15 for sample 255 on path 3, imag |

**Table 8 Filter Coefficients span**

## 3.2 Subband Statistics span

The subband statistics span is well described in chapter 3.1 in [3].

## 3.3 Block Generator span

The block generator span is well described in detail in [4].

# 4 Module Design

## 4.1 Algorithm

A proper explanation of the poly phase filter algorithm can be found in [5].

## 4.2 Architecture

As stated before this document focusses on the poly phase filter structure since the FFT that is used to compose the poly phase filterbank is described in detail in [3]. In order to create a wideband PPF unit a single channel PPF was designed first.

### 4.2.1 Single poly phase filter structure

Figure 3 shows the filter structure for an L=16-taps poly phase filter that supports M=512 subbands. A total of 16*512= 8192 coefficients *(h)* are applied to the incoming datastream *x(nN)*. The M=512 outputs *y(nM)* can be merged into one output that can be considered as *y(nN)*. Note that the vertical aligned delay elements ($z^{-1}$) operate in the *N* time domain, while the horizontal aligned delay elements work in the decimated time domain *M*. This type of a poly phase filter is considered as a single channel poly phase filter.

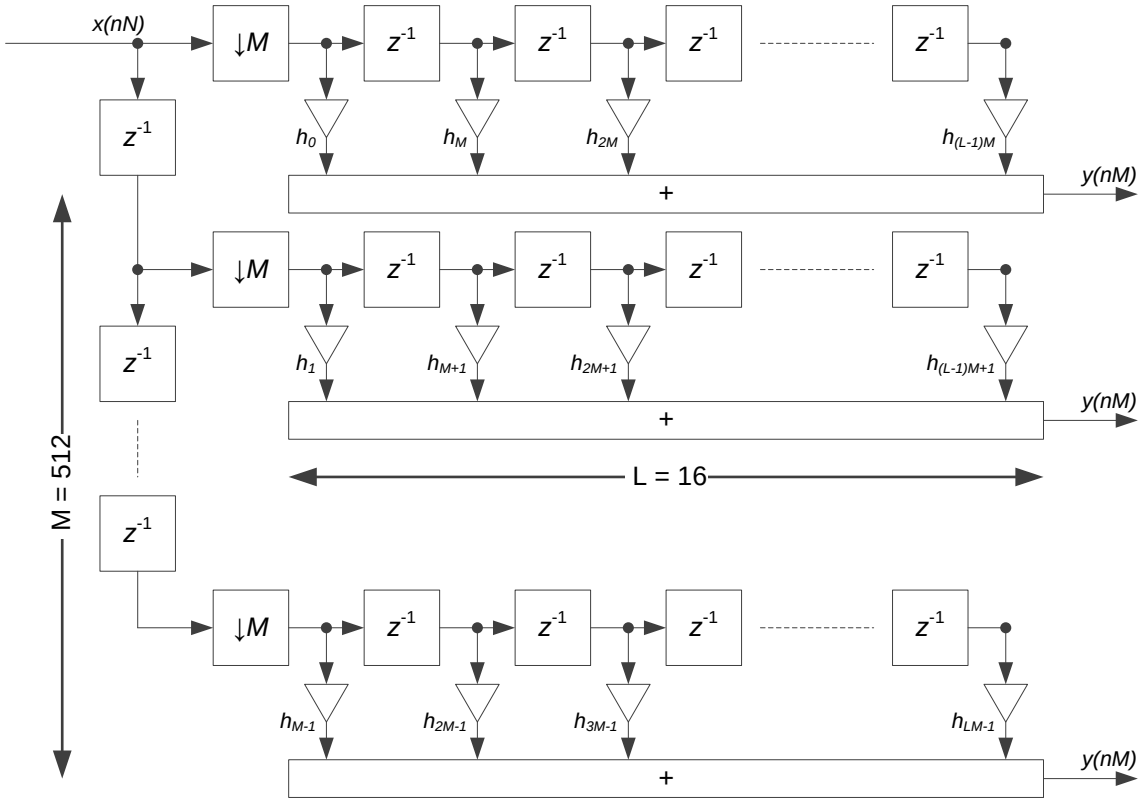


**Figure 3 Filter structure for a single poly phase filter**

An architect for the above filter structure is given in Figure 4, which is partly inspired by the Lofar PFS (poly Phase Filter Structure). The architecture consists of memories for the “historical” tap-data and the filter coefficients that are controlled by a controller. A separate filter unit takes care of the actual filter-operations (multiplications).

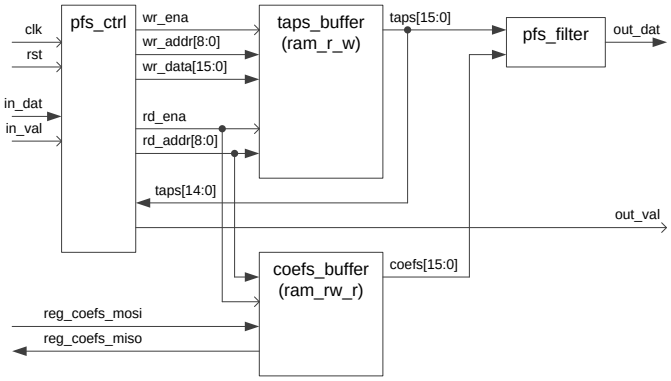


**Figure 4 Architecture single channel poly phase filter**

The coefficients memory can be written and read via a memory mapped interface

### 4.2.2 Wideband poly phase filter structure

The structure for a wideband variant of the poly phase filter is shown in Figure 5. It shows the same L=16-taps poly phase filter that supports M=512 subbands as given in Figure 3, but now with a wideband factor of P=4 being applied. This results in P=4 parallel poly phase filters. Where each single channel filter handles M=512/4 = 128 subbands. The number of coefficients remains 8192 as they are divided over the P=4 single channel poly phase filters. All outputs of each single channel filter can be merged to generate the *y(nNP+p)* outputs.

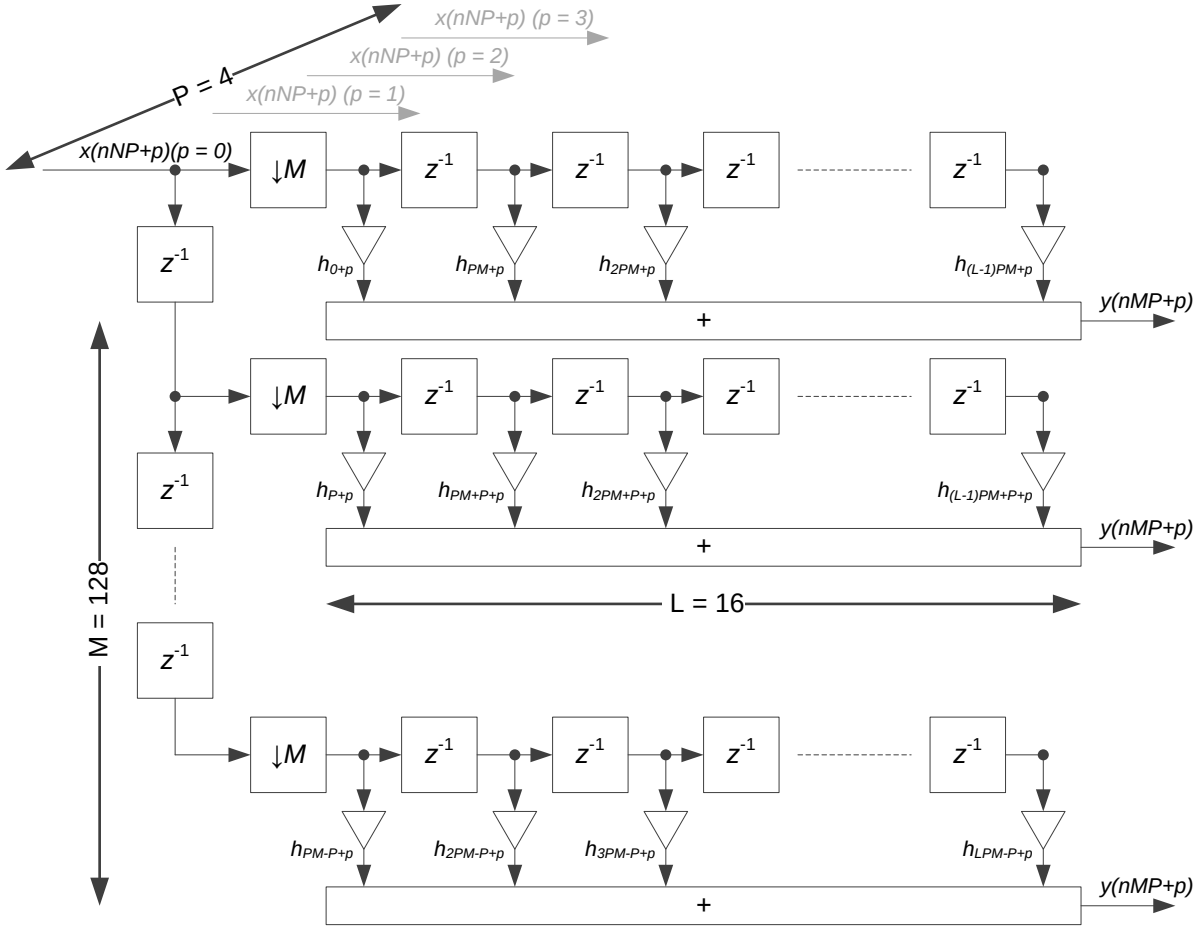


**Figure 5 Filter structure for a wideband poly phase filter**

The architecture for a wideband poly phase filter can be regarded as a number of single channel poly phase filters as depicted in Figure 4 placed in parallel.

# 5 Implementation

This chapter describes the implementation of the wpfb unit and in particular the poly phase filter, since the wideband FFT has already been described in detail in [3]. First the implementation of the single poly phase filter is described, then the wideband poly phase filter and the last paragraph describes the wideband poly phase filter bank (filter + FFT). The filter parts are stored in the filter_lib (see [6]). For the wpfb unit a separate library is created (see [7]).

## 5.1 fil_ppf_single

The fil_ppf_single is based on the architecture as shown in Figure 4 and consists of a memory that holds the coefficients for every tap, a memory that holds the history data for every filter, the filter part that performs the multiplication and addition and a controller part that drives the filter operation. Figure 6 gives a more detailed overview of the implementation of the fil_ppf_single unit.

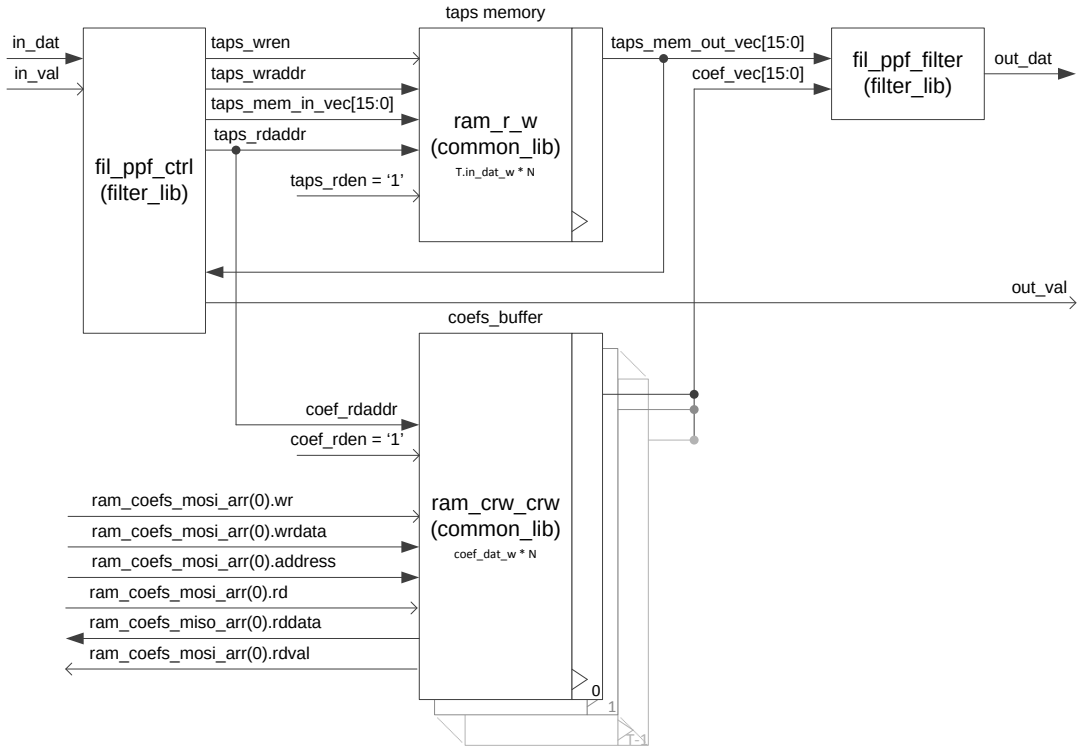


**Figure 6 Detailed block diagram of fil_ppf_single**

### 5.1.1 parameters

For the parameterization of the fil_ppf_single unit two generic records are defined: t_fil_ppf and t_fil_pipeline. The content of t_fil_ppf is shown in Table 9. The record is defined in a package, fil_pkg.

| Generic | Type | Value | Description |
|---|---|---|---|
| wb_factor | NATURAL | 1 | The wideband factor (only applicable to fil_ppf_wide) |
| nof_bands | NATURAL | 1024 | The number of subbands is N (N=nof_points for the FFT) |
| nof_taps | NATURAL | 16 | The number of filter taps in the filter. |
| in_dat_w | NATURAL | 8 | Width of the input data. |
| out_dat_w | NATURAL | 16 | Width of the output data. |
| coef_dat_w | NATURAL | 16 | Width of the filter coefficients. |

**Table 9 t_fil_ppf record fields**

The content of the t_fil_pipeline record is depicted in Table 10.

| Generic | Type | Value | Description |
|---|---|---|---|
| mem_delay | NATURAL | 1 | The read latency of both the taps and coefficients memory |
| mult_input | NATURAL | 1 | Input registers of the multiplier |
| mult_product | NATURAL | 1 | Register for the multiplication |
| mult_output | NATURAL | 1 | Output registers of the multiplier |
| adder_stage | NATURAL | 1 | Number of pipeline registers between every stage in the adder tree |
| requant_remove_lsb | NATURAL | 1 | Register in the requantizer. |
| requant_remove_msb | NATURAL | 0 | Register in the requantizer. |

**Table 10 t_fil_pipeline record fields**

### 5.1.2 taps memory

The taps memory is a single port read write memory that is instantiated from the common_lib. History data from the first T-1 taps results are stored in the taps memory along with current data value. The datawidth of the data is therefor defined by T*in_dat_w. The depth of the taps memory is determined by the number of subbands of the poly phase filter: nof_bands.

### 5.1.3 coefficient buffer

The buffer that holds all the coefficients for the filter is an array of dual ported rams from the common_lib. Every ram in the array holds the coefficients for a single tap. The a-port is used to read the coefficients to feed them to the filter block. The a-port runs on the dp_clk. The b-port is for writing the coefficients and optional reading back via a memory mapped interface. The b-port runs on the mm_clk. The datawidth of each buffer is defined by the width of the coefficients. The depth is determined by the number of subbands of the filter: nof_bands.

### 5.1.4 fil_ppf_filter

The fil_ppf_filter unit is a simple concatenation of building blocks form the common_lib as shown in Figure 7. The first stage consists of T multipliers from the common_lib. The output of the multipliers is connected to the input of a T-input addertree that performs the summarization of the filter function. The single output of the addertree is led through a requantize unit from the common_lib that quantizes the data to meet the desired output width.

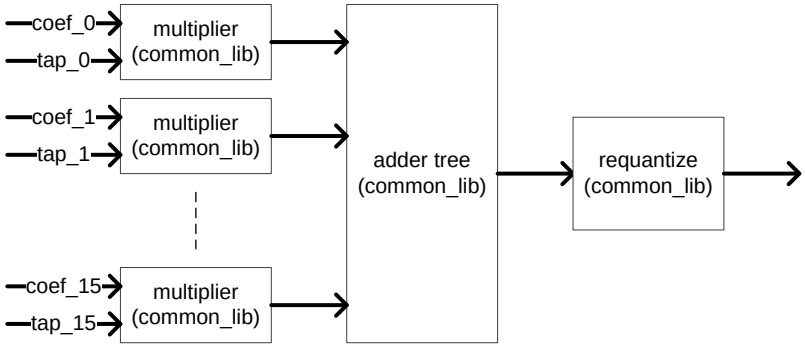


**Figure 7 Block diagram of the ppf_filter unit**

### 5.1.5 fil_ppf_ctrl

The control of the filter is performed by the fil_ppf_ctrl unit. This unit is responsible for writing and reading the taps memory and reading the coefficients memory. It also takes care of the incoming data and the outgoing out_val signal. A block diagram is shown in Figure 8 where two independent counters are used to create the

read address and the write address. All functionality is triggered by the in_val signal or a delayed version of the in_val signal. The depths of the different pipeline stages for the in_val signal are defined as follows:

a = 1
b = g_fil_ppf_pipeline.mem_delay (= 1)
c = 1
d = c_ctrl_latency + c_mult_latency + c_adder_latency + c_filter_zdly + c_requantize_latency – 2

where

c_ctrl_latency = 1
c_mult_latency = total latency of the multiplier in the fil_ppf_filter unit
c_adder_latency = delay of the adder tree in the fil_ppf_filter unit
c_requantize_latency = latency in the requantizer unit in the fil_ppf_filter unit

c_filter_zdly = the delays that are required to settle the filter (= nof_bands)

The read address drives both the taps history memory as the coefficients buffers, so these are perfectly aligned. The in_data is concatenated with the lower tap data from the taps history memory and then written back to the history memory.

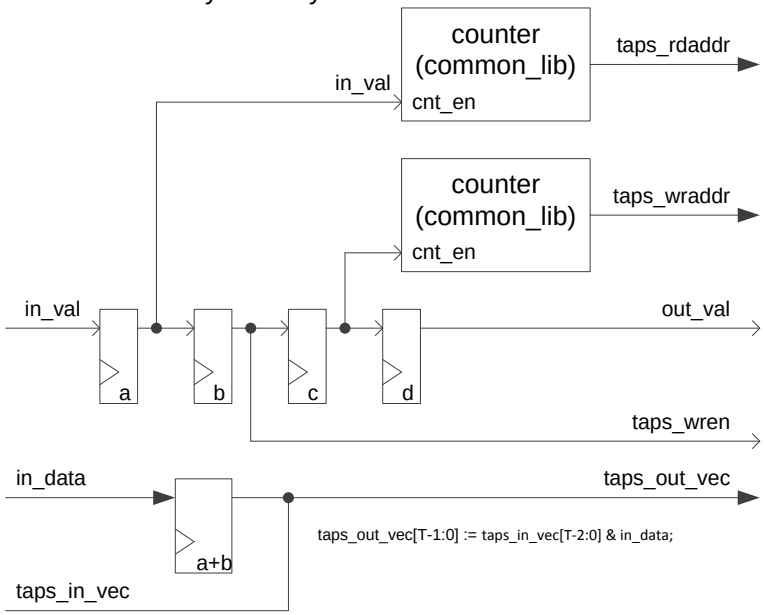


**Figure 8 Detailed block diagram of fil_ppf_ctrl**

## 5.2 fil_ppf_wide

The wideband variant of the poly phase filter is created by putting a number of fil_ppf_single units in parallel. Each single unit will then process nof_bands/P subbands. The memory maps for the coefficients of each single unit are merged together to one memory map using a common_mem_mux component from the common_lib.

## 5.3 wpfb unit

The wpfb unit is the fft_wide_unit (as described in paragraph 5.4 in [3]) extended with a wideband poly phase filter. Figure 9 shows a detailed block diagram of the wpfb unit. Note that a unique wideband filter

instantiation is required for both the real and imaginary input. The memory maps for the coefficients of both filters are combined via a common_mem_mux unit from the common_lib.

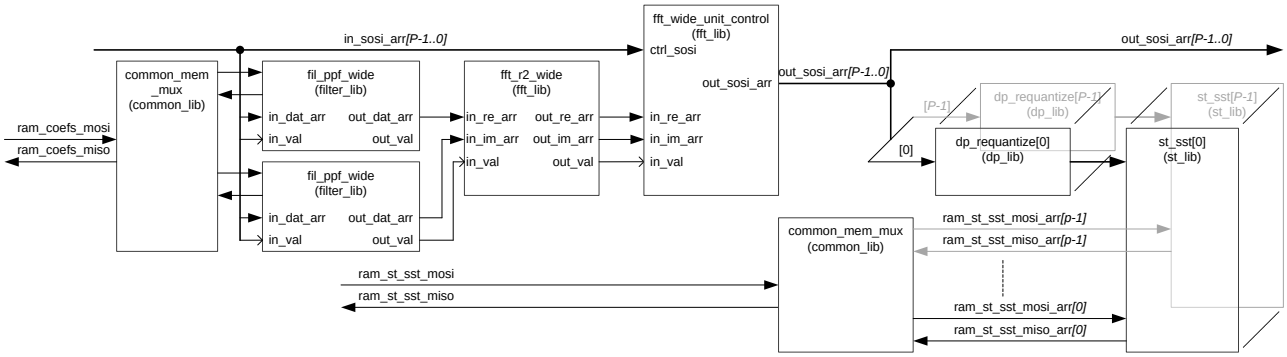


**Figure 9 Detailed block diagram of the wpfb unit**

# 6 Verification

Verification of the wpfb unit is performed using a co-simulation setup with Modelsim and Python. The python script, tc_mmf_wpfb_unit.py can be found in [8] and it has the option to start Modelsim automatically using the c_start_modelsim variable. The tc_mmf_wpfb_unit.py is the same as the tc_mmf_fft_r2.py script (see [3]) but it is extended with functionality that writes the filter coefficients.

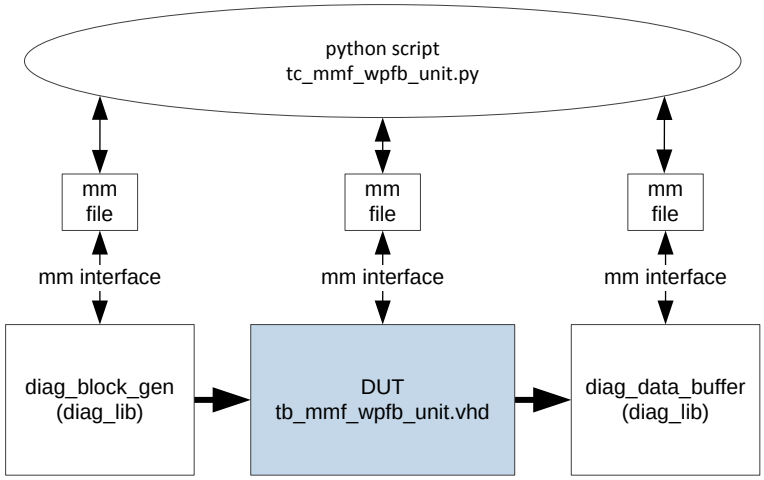


**Figure 10 Testbench based on co-simlulation**

A visual representation of the script based testbench is displayed in Figure 10. The script sends stimuli data to the diag_block_gen unit and possible the filter coefficients to the DUT. Then it will monitor the amount of data that is written in the diag_data_buffer. Once the specified amount of data is written the script will read the data from the diag_data_buffer and process is further for displaying. The script plots both the FFT result from the VHDL DUT as the reference FFT output that is determined within the Python script. The effect of the filterbank is seen at best when a frequency is chosen that is not on a bin of the FFT.

# 7 Validation

Validation of the wpfb_unit is performed with the bn_filterbank design of the Apertif project. This design contains two wpfb units as described in [9].

# 8 Appendix – list of files

## 8.1 Firmware VHDL

All VHDL source files that are used for the wpfb unit can be found in the following four directories, where the last library contains the toplevel file:

$UNB/Firmware/dsp/rTwoSDF/src/vhdl
$UNB/Firmware/dsp/fft/src/vhdl
$UNB/Firmware/dsp/filter/src/vhdl
$UNB/Firmware/dsp/wpfb/src/vhdl

The next table gives an overview of the VHDL source files in the wpfb library:

| VHDL File | Description |
|---|---|
| wpfb_pkg.vhd | Package that contains record en function definitions that are specific for the wpfb unit. |
| wpfb_unit.vhd | Design of a wideband poly phase filter bank that is extended with streaming input and output interfaces and a subband statistics module. |

## 8.2 Testbench

The vhdl testbench files for simulation are in the following directory:

$UNB/Firmware/dsp/wpfb/tb/vhdl

The python scripts that are used for the testbenches can be found in:

$UNB/Firmware/dsp/wpfb/tb/python