



RF Engines Limited



RFEL Fixed Point Halfband Filter Model for Matlab

Document Title

Document Ref

H03002

Tel

+44 (0)1983 550330

Fax

+44 (0)1983 550340

E-mail

info@rfel.com

Web

www.rfel.com

Filename

H03002-Fixed Point Halfband Filter Model Manual.doc

Approved

Issue	1.00				
Date	23 May 03				
Author					
Technical					
Commercial					
QA					

Copyright RF Engines Limited 2003

All Rights reserved

This document is issued by RF Engines Limited (hereinafter called RFEL) in confidence, and is not to be reproduced in whole or in part without the prior written permission of RFEL. The information contained herein is the property of RFEL and is to be used only for the purpose for which it is submitted. None of this document's contents can be disclosed to a third party without the prior written permission of RFEL.

Document History

Revision No	Date Issued	Issued by	Summary of changes
1.00	23 May 2003	PW	Initial

Contents

1	What is it?	2
2	Who are we?	4
3	Installation	4
4	Using the model	4
4.1	Initialisation.....	5
4.2	Runtime	6
4.3	Clear.....	6
4.4	Help.....	7
4.5	Version	7
5	Examples	7
5.1	First Nyquist - Sinusoid at $fs/4$	7
5.2	Second Nyquist - Sinusoid above $3*fs/4$	10

1 What is it?

This document describes the RF Engines Ltd (RFEL) Distributed Halfband Filter (DHBF) Software Model for Matlab. The version of the software considered here is the fixed point version – which uses fixed point arithmetic throughout and has customisable bit-widths. There is also a floating point model which uses floating point arithmetic, but the floating point model is not described here.

The model is designed to accurately represent the functionality of RFEL halfband filter cores. It has been tested against RFEL's standard halfband filter implementation and shown to be bit-true.

The model is designed to be fully parameterisable, to encompass the widest range of possible requirements. Many of the modelled configurations are provided by RFEL as optimised off-the-shelf FPGA cores; cores may also be tailor-made to meet higher specification requirements.

A technical description of the model is given in Table 1.

Parameter	Specification
Filter Length	The filter must have $4K-1$ taps, where K is any integer between 1 and 10000. The upper limit also depends on available memory.

Phase	The initial phase of the filter and the LOs may be set to one of four values: 0, 1, 2 or 3. The meaning of this will be described later.
LO Frequency	The DHBF filter may be used to downconvert signals from the first Nyquist region (in which case the LO is at $fs/4$) or the second Nyquist region (in which case the LO is at $-fs/4=3*fs/4$).
Input Precision	Customisable between 1 and 62 bits. See note on Data Precision.
Tap Precision	Customisable between 1 and 62 bits. See note on Data Precision.
Output Width	Customisable between 1 and 62 bits. See note on Data Precision.
Output Scaling	Customisable between 0 and 62 bits. See note on Data Precision.
Clipping	The model provides the same clipping as the FPGA core in that when the filtered output signal overflows the output width, the signal saturates. An additional array is output to provide clip detection.
Processing type	Block Processing
Block Input Size	Any even integer.
Block Output Size	Half the input length.
Pipeline Delay	None
Data Precision ¹	Up to 62 bits of fixed point precision is supported throughout. Intermediate sums and products must not exceed 62 bits precision.
Platforms	Matlab for Windows. Tested on Windows 2000, Matlab version 6.5.

Table 1: Model Specification

The DHBF model is implemented as a fully optimised MEX add-in for Matlab and can therefore be called from Matlab in the same way as any other function. Use of the model involves three stages, initialisation, processing and destruction.

- At initialisation, the user specifies the model parameters (filter taps, phase, Nyquist region and all appropriate bit widths) and the function returns a handle. This handle is then used in all future processing.
- During runtime, the user supplies input data blocks to the model and retrieves the output of the halfband filter.
- At destruction, the user calls clear on the model, which then deallocates all associated memory.

¹ The bit widths used through the core will limit the size of the DHBF that will fit on a particular FPGA device. Signals with up to up to 17 bits may be multiplied by a single 18-bit multiplier and up to 34 bits may be multiplied using two 18-bit multipliers. Therefore it is in the interest of the designer to keep the bit widths as low as possible to meet a given performance.

Use of the model is described in detail in Section 4.

2 Who are we?

RF Engines Limited specialises in advanced digital signal processing hardware designs for high quality signal filtering and conditioning. Utilising patented architectures and high-speed Field Programmable Gate array (FPGA) devices, the company supplies off-the-shelf or specialist system solutions for advanced electronic products in the defence, instrumentation and communications markets worldwide.

Based on the Isle of Wight in the UK, the company provides:

standard designs sold as Intellectual Property Cores,

complete turn-key designs developed against a particular specification, and

consultancy services to establish suitable architectures for specific tasks.

For more information go to: <http://www.rfel.com/>

For a list of available products, go to: <http://www.rfel.com/products/Products.asp>

Or contact us to discuss your needs at: info@rfel.com

3 Installation

To install the DHBF software model, run the installation file and follow the onscreen instructions. The model should be installed to the required directory and then this directory should be placed on the Matlab path.

4 Using the model

As noted above, use of the model involves the three steps of initialisation, processing and destruction. This enables blocks of data to be continuously “streamed” into the model, rather than passing in a single input vector and getting a single output vector.

The model is shown graphically in Figure 1; all parameters are double precision numbers (i.e. the Matlab standard data type) unless otherwise noted. The length of the input vector is always an even number and each output vector is one half the input length.

During runtime, the model takes inputs and supplies outputs as Matlab’s int64 type. The filter taps during initialisation are also of type int64. This permits fixed point bit widths of up to 62 bits to be supported.

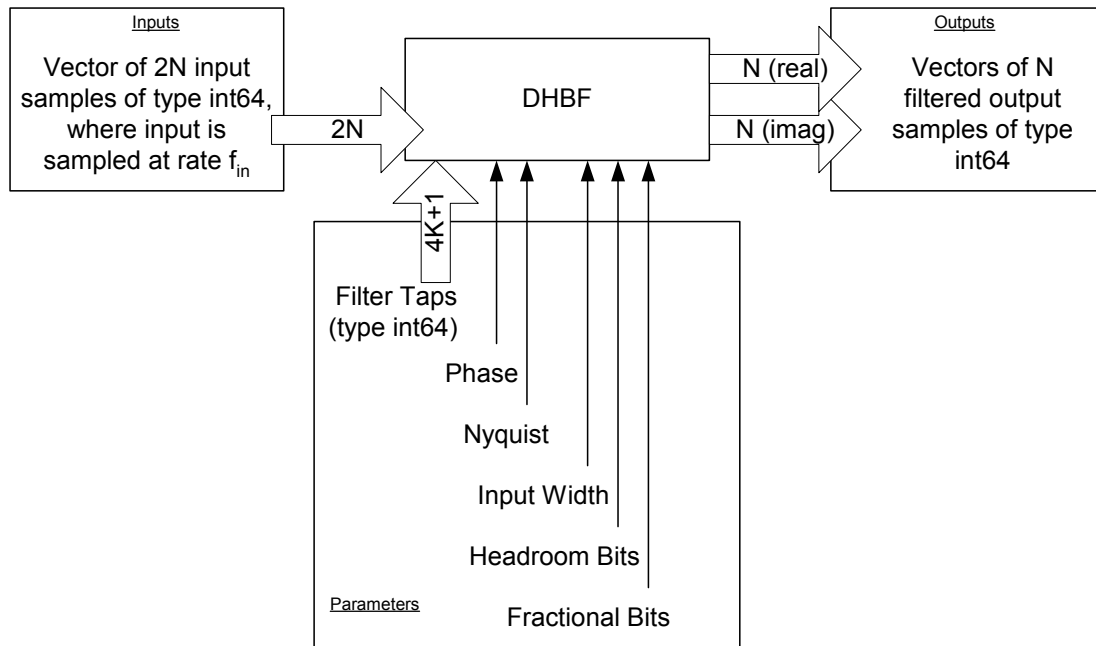


Figure 1: DHBF Inputs, Outputs and Parameters

The model is designed to accurately represent the functionality of RFEL FPGA cores. It has been tested against RFEL's standard implementation and shown to be bit-true provided the modelled bit-widths are correctly set to avoid overflow.

The model is designed to be fully parameterisable, to encompass the widest range of possible requirements. Many of the modelled configurations are provided by RFEL as optimised off-the-shelf FPGA cores. Cores may also be tailor-made to meet higher specification requirements.

4.1 Initialisation

The constructor contains all the parameters relating to the halfband filter.

The output width is given by:

$$\text{OUTPUTWIDTH} = \text{INPUTWIDTH} + \text{HEADROOMBITS} + \text{FRACTIONALBITS}$$

where $(\text{TAPWIDTH} - \text{FRACTIONALBITS} - 1)$ LSBs are dropped off the bottom.

```
[HANDLE] = xpDhbfMat(TAPS, PHASE, FIRSTNYQUIST, INWIDTH, TAPWIDTH,
HEADROOMBITS, FRACTIONALBITS, ISSYM)
```

TAPS (type = INT64): The complete set of halfband filter taps. These must follow the standard format for a halfband filter, where:

- there are an odd number of taps,
- the centre tap is equal to $2^{(\text{TAPWIDTH} - 1)}$,
- all other even numbered taps are zero (except centre tap),
- the taps are reflected around the centre tap

PHASE (type = DOUBLE): This describes the initial phase of the halfband filter. There are four possible phases. These are (for the first Nyquist):

- 0: first sample to I channel (odd filter taps), $\text{LO} = \exp(-jn\pi/4)$
- 1: first sample to Q channel (even filter taps), $\text{LO} = \exp(-j(n+1)\pi/4)$

2: first sample to I channel (odd filter taps), $LO = \exp(-j(n+2)*\pi/4)$

3: first sample to Q channel (even filter taps), $LO = \exp(-j(n+3)*\pi/4)$

For the second Nyquist the LO direction is reversed

FIRSTNYQUIST (type = LOGICAL): This is true to take the spectrum from the first Nyquist region (i.e. with the IF at $fs/4$) and false for the second Nyquist region (i.e. with the IF at $-fs/4 = 3*fs/4$).

INWIDTH (type = DOUBLE): Bit width at input to the filter.

TAPWIDTH (type = DOUBLE): Bit width of filter taps.

HEADROOMBITS (type = DOUBLE): The number of additional bits to allow above unity gain at the output of the DHBF.

FRACTIONALBITS (type = DOUBLE): Number of additional fractional bits output.

ISSYM (type = LOGICAL): True for fixed symmetrical rounding at the output of the filter, false for asymmetrical rounding at the output of the filter.

HANDLE (type = UINT32): Handle to a `xpDhbfMat` object that is then used in all further processing.

4.2 Runtime

The input is passed through the local oscillator, filtered using the filter supplied at initialisation and decimated to provide complex output onto OUTI and OUTQ. Clipping is detected on the in-phase output and is output as an array of INT64s where 0 denotes no clipping and 1 denotes clipping.

```
[OUTI, OUTQ, CLIP] = xpDhbfMat(HANDLE, IN)
```

HANDLE (type = UINT32): Handle of valid `xpDhbfMat` object obtained from INIT function

IN (type = INT64): Real input data - must have an even number of elements.

OUTI (type = INT64): In-phase output of DHBF. Half the length of the input.

OUTQ (type = INT64): Quad-phase output of DHBF. Half the length of the input.

CLIP (type = INT64): This optional output provides information as to whether the DHBF has clipped. The model will run faster without this output so it should only be required when necessary.

4.3 Clear

All `xpDhbfMat` objects are destroyed when the dll is cleared from memory (i.e. call "clear `xpDhbfMat`") or specific instances may be destroyed using this function.

All memory is deallocated and the handle is then useless and will generate an error if used.

```
xpDhbfMat(HANDLE, 'clear')
```

HANDLE (type = UINT32): Handle of valid `xpDhbfMat` object obtained from INIT function

4.4 Help

Detailed online help may be displayed in the help browser by typing:

```
xpDhbfMat('help')
```

4.5 Version

A string giving version information for the function may be retrieved by calling:

```
[DESCRIPTION] = xpDhbfMat('version')
```

DESCRIPTION (type = CHAR): String describing the current function version

5 Examples

The following examples illustrate the use of the halfband filter.

5.1 First Nyquist - Sinusoid at fs/4

In the first example, the halfband filter is set up to downconvert the signal from the first Nyquist range (from 0 to $fs/2$ where fs is the AtoD sample rate). The AtoD rate is set to 100MHz and a cosine signal is generated at 25MHz ($fs/4$).

Figure 2 shows the spectrum of the real input signal. Components can be seen at +/- 25MHz. Figure 3 shows the in-phase and quad-phase outputs of the DHBF. It can be seen that after settling, the output is a DC level in the in-phase component and 0 in the quadrature-phase input. The signal has clearly been shifted down by the appropriate $fs/4$. Figure 4 shows the spectrum of the DHBF output.

The script to run this example may be found at:

[xpDhbfMat_example1.m](#) *(link only works in the Matlab help browser)*

If the input signal had been a sinusoid, the output signal would have been a DC level in the quadrature-phase component with no energy in the in-phase component. This is shown in Figure 5. Note that there is no settling on the quadrature-phase output – only a delay corresponding to half the length of the DHBF at the full input rate. The script to generate this plot is:

[xpDhbfMat_example1b.m](#) *(link only works in the Matlab help browser)*

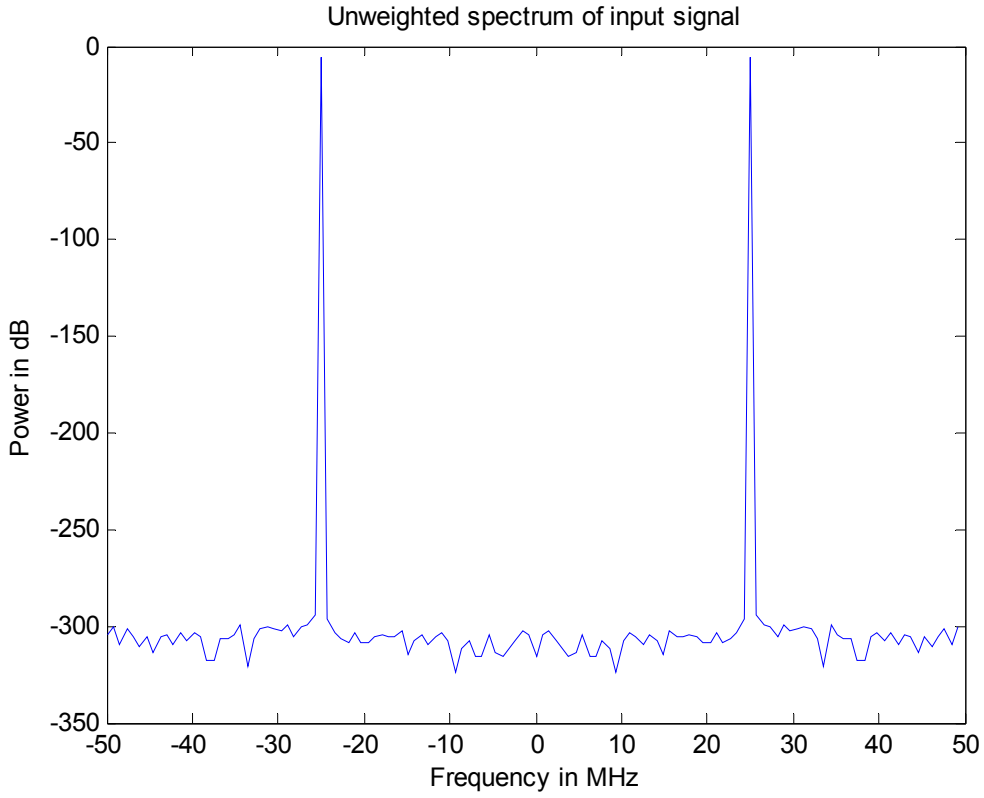


Figure 2: Input Spectrum

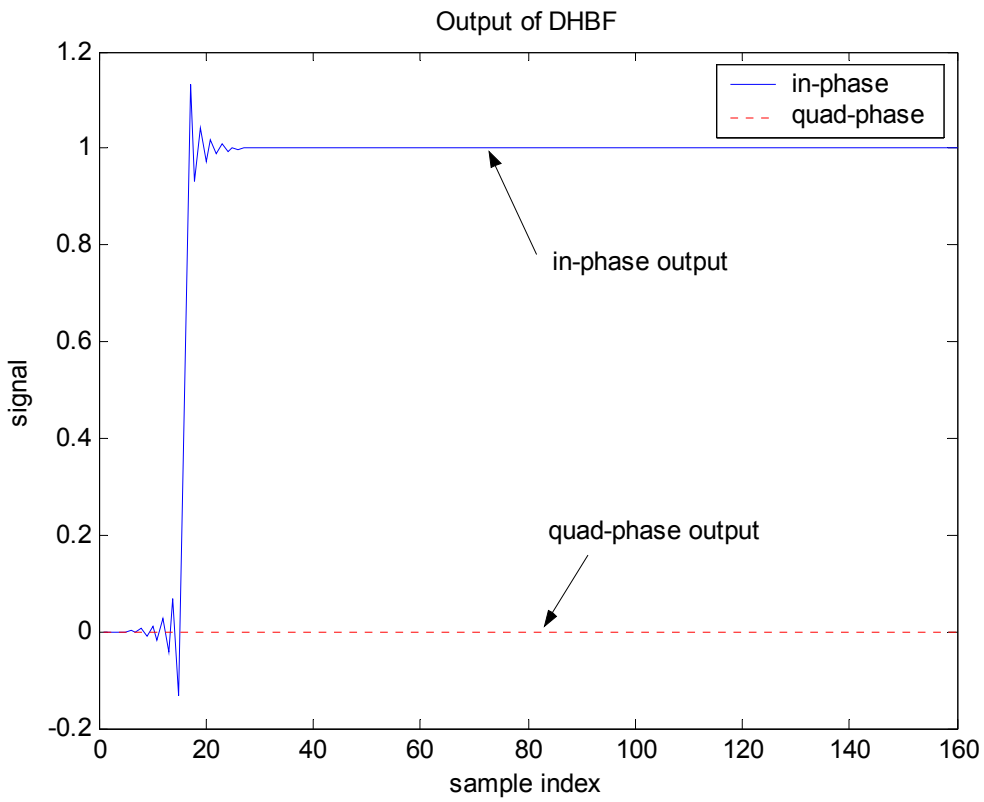


Figure 3: DHBF output signal (using cosine)

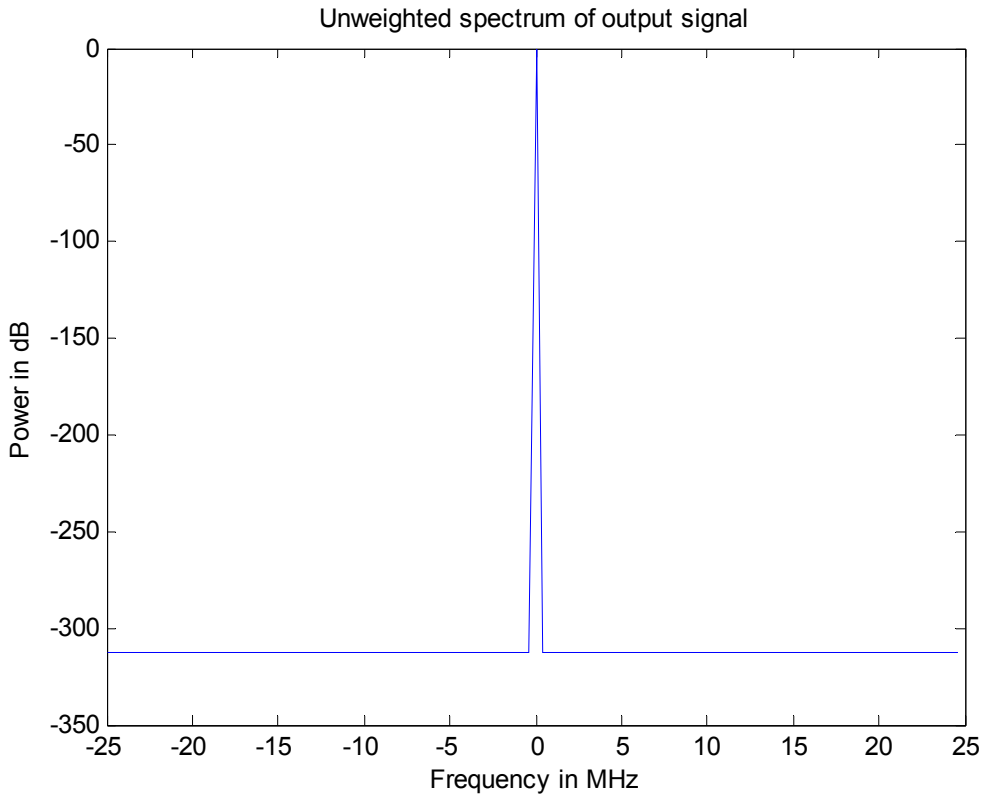


Figure 4: Spectrum of output signal

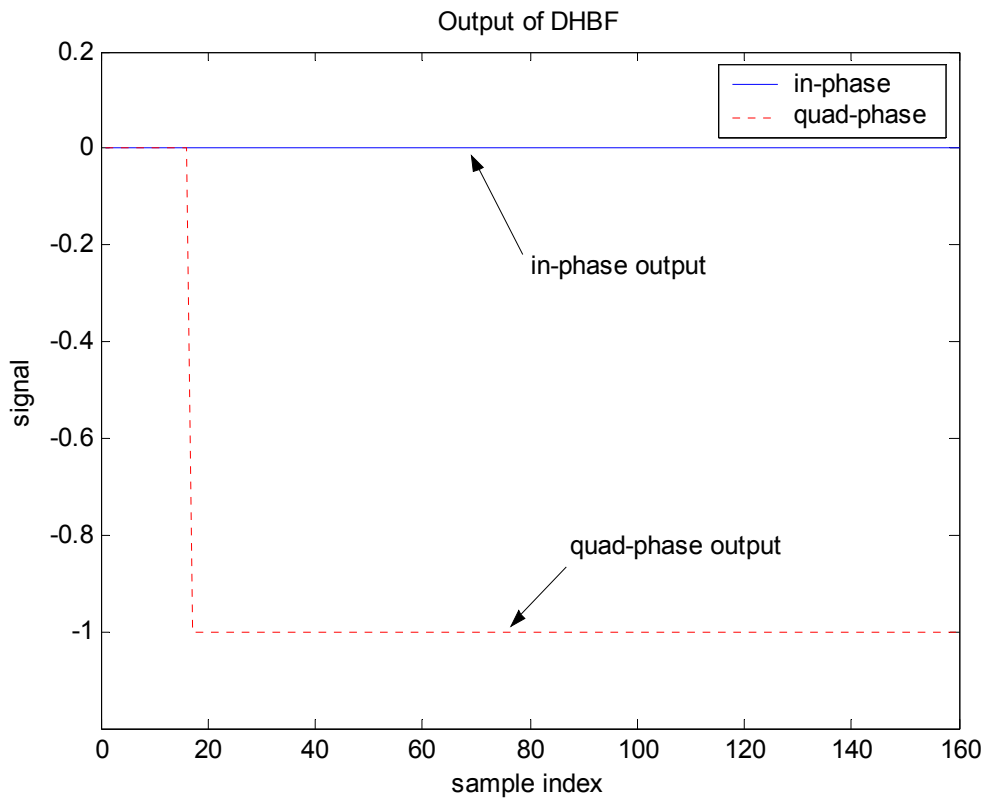


Figure 5: DHBF output signal (using sine)

5.2 Second Nyquist - Sinusoid above $3*fs/4$

This example demonstrates the use of the model to downconvert a signal centred at $3*fs/4$ (the second Nyquist). It also demonstrates the use of the model in its streaming mode. In the previous example a single vector of input data was passed into the model, with a single vector output. In this example, blocks of data are sequentially passed into the model with the model retaining its state between blocks. This could be advantageous where a very large data set was being processed, for example streaming to and from a file.

The input sample rate is again 100MHz. The signal is this time located at 7.54MHz ($3*fs/4 + 400kHz$), where the second Nyquist range is in use. The output signal will be located at 400kHz. The model is initialised with the given taps, data is then generated and divided into frames of 100 samples. These frames are streamed through the model and the output plotted.

Figure 6 shows the spectrum of the real input signal. This is located at 65.4MHz, aliased down to $-24.6MHz$, with an image at $24.6MHz$. The spectrum is not clear as with the previous example, since this frequency is not located at the centre of one of the analysis FFT bins. Clarity could be improved by using a weighted FFT.

Figure 7 shows the in-phase and quadrature-phase outputs of the DHBF. The resultant complex sinusoid is output at 400kHz. The filter build-up can be seen in the in-phase component and the delay can be seen in the quad-phase component as before.

Figure 8 shows the unweighted output spectrum of the DHBF. The frequency of the signal can be clearly seen – though, because the FFT is unweighted, the response of the FFT makes the DHBF stop-band performance look very poor. This may be improved by using a Blackman-Harris weighting as shown in Figure 9. The DHBF was designed to have a stopband attenuation of at least 80dB.

The script for this example is:

[xpDhbfMat_example2.m](#) *(link only works in the Matlab help browser)*

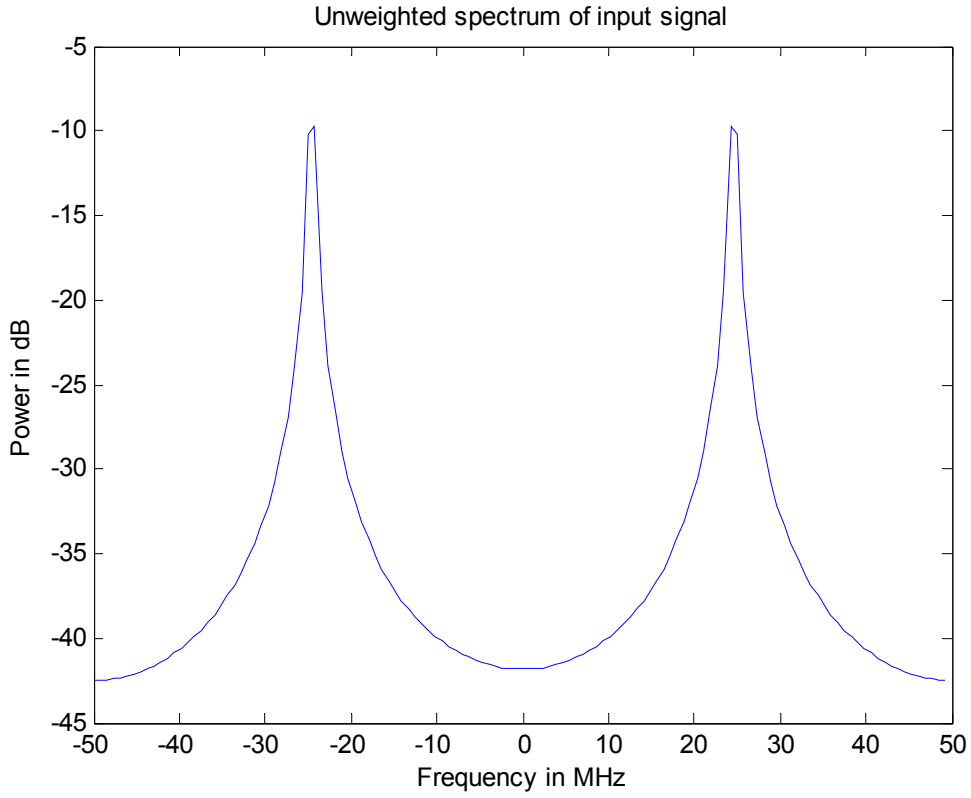


Figure 6: Spectrum of input signal

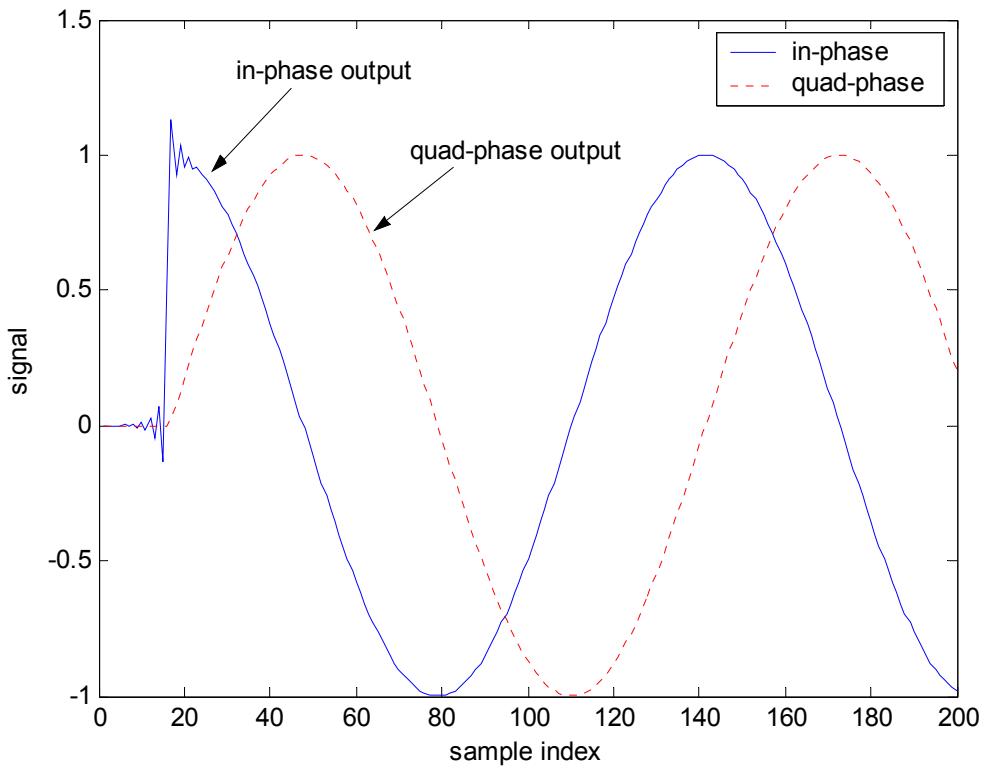


Figure 7: DHBF output signal

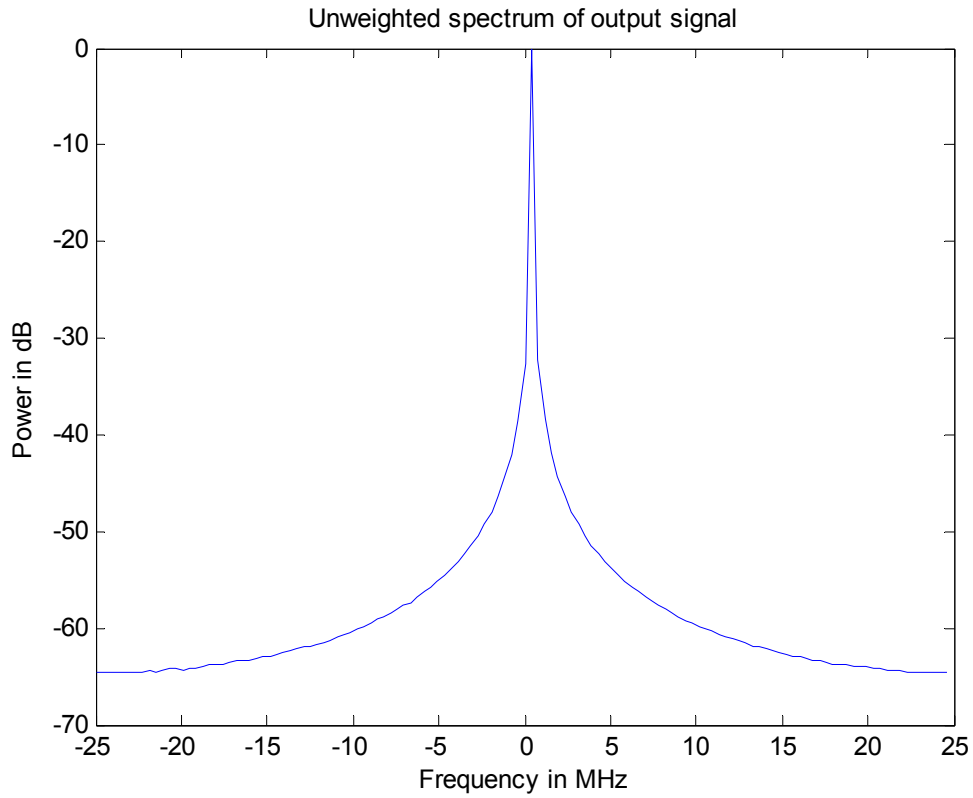


Figure 8: Unweighted spectrum of output signal

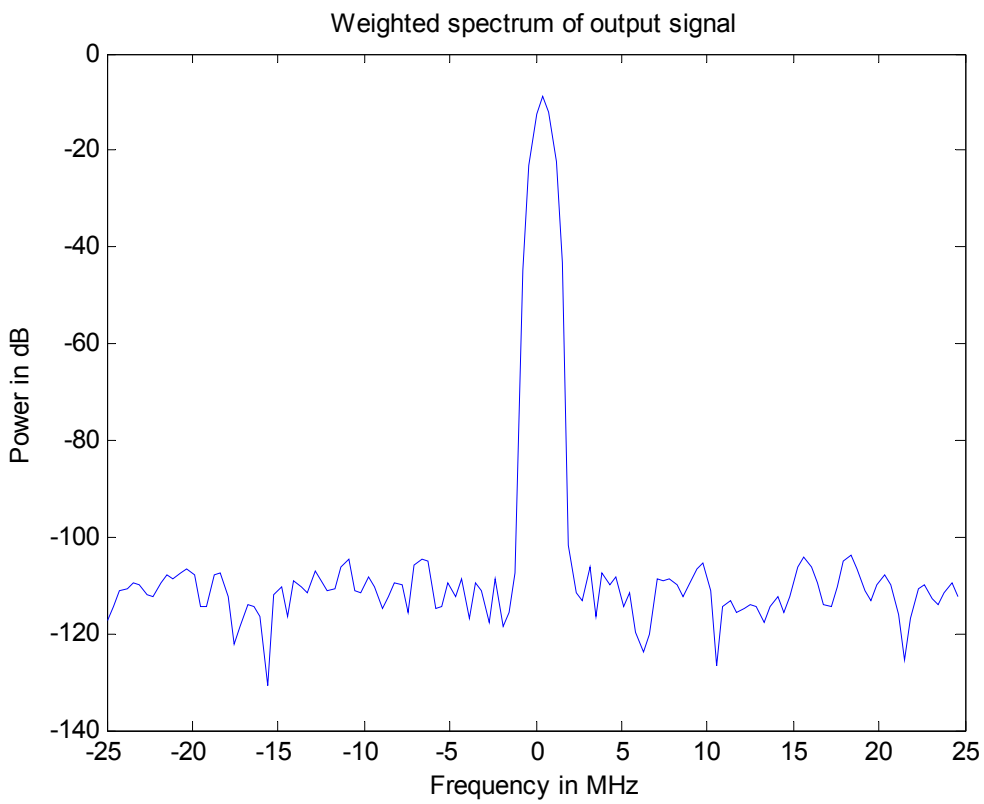


Figure 9: Weighted spectrum of output signal (Blackman-Harris)