



## RF Engines Limited



### Fixed point Polyphase DFT (radix 2 FFT) model for Matlab

Document Title

Document Ref

H03003

Tel

+44 (0)1983 550330

Fax

+44 (0)1983 550340

E-mail

info@rfel.com

Web

www.rfel.com

Filename

H03003-Fixed Point Polyphase DFT Radix 2 Matlab Model.doc

Approved

Issue	1.00				
Date	25 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	25 May 2003	PW	Initial

## Contents

<b>1</b>	<b>What is it?</b> .....	<b>2</b>
<b>2</b>	<b>Who are we?</b> .....	<b>4</b>
<b>3</b>	<b>Installation</b> .....	<b>5</b>
<b>4</b>	<b>Using the model</b> .....	<b>5</b>
<b>5</b>	<b>Examples</b> .....	<b>9</b>
5.1	Comparison with Standard Floating Point DFT .....	9
5.2	Detection of Low-Level Signals .....	13
5.3	Root Raised Cosine Filter Response .....	14

### 1 What is it?

This document describes the RF Engines Ltd (RFEL) Polyphase DFT (Mixed Radix FFT) 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 Polyphase DFT cores. It has been tested against RFEL's standard Polyphase DFT 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.

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

Parameter	Specification
FFT Stages <sup>1</sup>	Integers from 1 to 25. The upper limit also depends on available memory. <i>Note: If you have installed only an evaluation copy of this software the available number of FFT stages will be restricted. A full installation of the model provides FFTs with fully parameterisable sizes.</i>
Filter Length	Integers from 2 to $5 \times 2^{25}$ taps. The filter must be at least as long as the number of FFT points.
Oversampling Rate <sup>2</sup>	The model is not limited to critical sampling at the output. The output oversampling rate is specified in terms of the block input length through the model (see below).
Block Input Length	Integers from 1 to FFT Size. Setting the block input length equal to the FFT size provides a critically sampled output; setting the block input length to half the FFT size yields a twice oversampled output; and so on.
Bin ordering <sup>3</sup>	Bins may be output in bit-reversed order (i.e. no output bit-reverser), or normally ordered, with the startBin (first bin output) as a parameter. Setting the startBin to -1 provides bit-reversed output, otherwise the following values are given: startBin = 0: output order = [0, 1, 2, ..., N-1] startBin = 1: output order = [1, 2, ..., N-1, 0] startBin = N/2: output order = [N/2, ..., N-1, 0, 1, ..., N/2-1] Where N is the number of FFT bins. The most common settings are -1, 0 or N/2.
Phase Correction	May be turned on or off. If only the amplitude or power spectrum is required, then output phase correction is not required.
Input Precision	Customisable between 1 and 62 bits. See note on Data Precision.
Filter Tap Precision	Customisable between 1 and 62 bits. See note on Data Precision.
FFT Precision	Fully customisable at input and output and at all intermediate stages between 1 and 62 bits. See note on Data Precision.

<sup>1</sup> The maximum number of DFT points supported by the Ventrrix Polyphase DFT range is  $2^{17}$ . This is probably the practical limit to fit on a single FPGA. Larger DFTs may be provided to order.

<sup>2</sup> The Ventrrix Polyphase DFT range currently supports critical sampling and twice oversampled outputs. Other oversampling rates may be provided to order.

<sup>3</sup> The Ventrrix Polyphase DFT range currently provides as options: bit-reversed, normally ordered and N/2 shifted, corresponding respectively to startBin = -1, startBin = 0 and startBin = N/2. Other output orders may be provided to order.

Twiddle Precision	Customisable between 1 and 62 bits. See note on Data Precision.
Processing type	Block Processing.
Block Input Size	Equal to the input size set at initialisation.
Block Output Size	Equal to the FFT Length.
Pipeline Delay	None.
FFT Type	Radix-2 Decimate-in-frequency.
Data Precision <sup>4</sup>	Up to 62 bits of fixed point precision is supported throughout. Intermediate sums and products must not exceed 62 bits precision.
Scaling	Scaling is provided at the output of the filter, the output of the FFT and the output of the phase corrector. This enables the designer to eliminate MSBs which will never be toggled by a real signal and hence minimise the bit growth through the model.
Overflow Detection	Overflow detection is optionally provided to test whether the signal is in range at various points through the model. This is invaluable in verification that the scalings have been correctly specified.
Platforms	Matlab for Windows. Tested on Windows 2000, Matlab version 6.5.

**Table 1: Model Specification**

The Polyphase DFT 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 (FFT size, Filter Taps, etc) 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 Polyphase FFT.
- At destruction, the user calls clear on the model, which then deallocates all associated memory.

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

---

<sup>4</sup> The bit widths used through the core will limit the size of the FFT 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.

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](mailto:info@rfel.com)

### **3 Installation**

To install the Polyphase DFT 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**

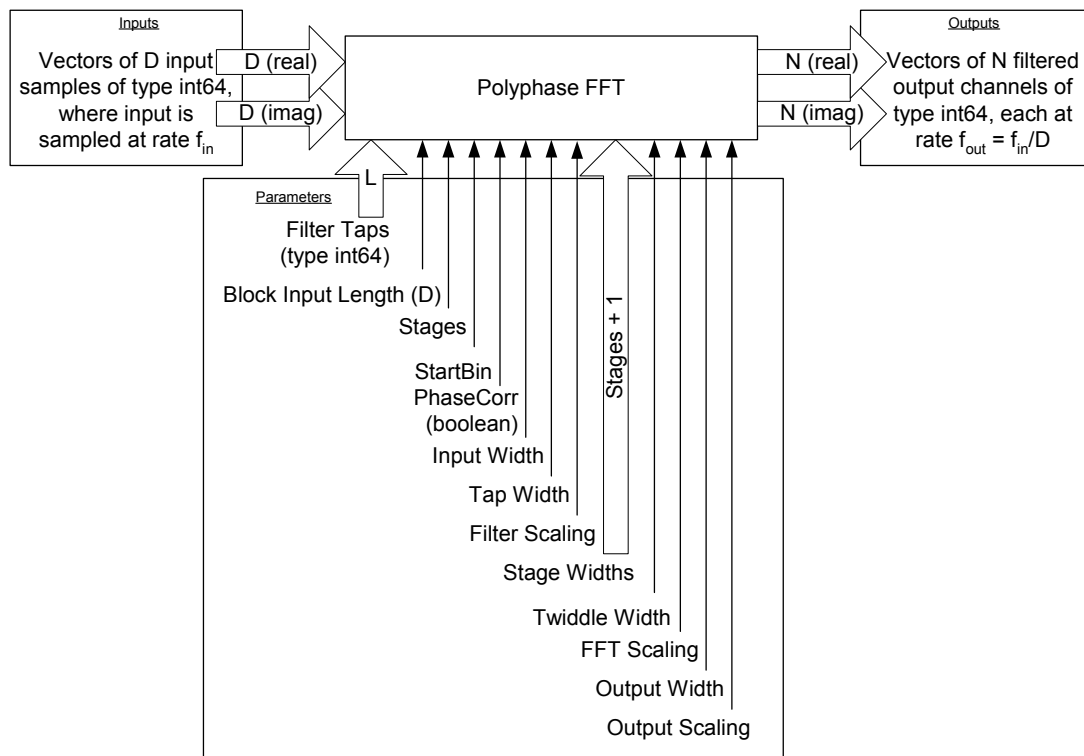
As noted above, use of the model involves the three steps of initialisation, processing and destruction. This enables blocks of data to be continuously “streamed” into the model, rather than passing in a single input array and getting a single output array.

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 parameters specify the dimensions of the input and output vectors and the rate conversion between the input and output. The length of the input vector is the block input length specified at initialisation and the output length is  $N = 2^{\text{fftOrder}}$ . The input length parameter specifies the change in sample rate between the input and each of the N output channels.

For example, if the input to a  $N = 1024$  point Polyphase DFT with an input block length of  $D = 50$  is at  $f_{in} = 100\text{MHz}$ , the input stream must be divided into blocks of 50 samples for input to the model, and the output will be 1024 channels, each at rate  $f_{out} = f_{in} / D = 2\text{MHz}$ .

Note that the model does not require knowledge of the input and output rates; it simply takes the frames as they are passed to it and does the processing. Some examples are described in Section 5.

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: Polyphase DFT 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 carries all the required model parameters.

Note that if phase correction is turned off then the output width and output scaling are, respectively, the width and scaling at the output of the FFT. If phase correction is turned on then the signal passes through the additional phase correction stage and the explicitly specified output scaling and output width are used.

```
[HANDLE] = xpPdff2Mat(FILTER, INLEN, STAGES, STARTBIN, PHASECORR, INWIDTH,
TAPWIDTH, FILTSCALING, STAGEWIDTHS, TWIDDLEWIDTH, FFTSCALING, OUTWIDTH,
OUTSCALING)
```

**FILTER** (type = INT64): This is the impulse response of the filter to be applied to each channel of the polyphase DFT. The filter must have a length at least equal to the number of DFT points.

**INLEN** (type = DOUBLE): This integer specifies the number of samples input to the WOLA for each DFT calculation. It is effectively the decimation through the xpPdff2Mat object for each channel. An input length equal to the number of DFT points yields critically sampled outputs; an input length equal to half the number of

DFT points yields twice-oversampled outputs. Any integer between 1 and the number of DFT points is supported.

**STAGES** (type = DOUBLE): the number of FFT stages

**STARTBIN** (type = DOUBLE): This specifies the start bin for the output of the FFT and is in the range [-1, 0, 1, FFT Points - 1]. If STARTBIN is positive then consecutive output samples appear sequentially from the given start bin. Typical values are 0 and (FFT Points / 2) - which match the output of the standard Matlab FFT and the Matlab FFTSHIFT function, respectively. STARTBIN = -1 specifies a bit-reversed output.

**PHASECORR** (type = LOGICAL): True to do phase correction at the output of the Polyphase DFT. False otherwise. Phase correction is not necessary if you are only interested in the amplitude or power spectrum of the output signal.

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

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

**FILTSCALING** (type = DOUBLE): Number of MSBs to discard at the output of the filter.

**STAGEWIDTHS** (type = DOUBLE): Vector of bit-widths for use through the FFT. Must have (STAGES+1) elements. The first element specifies the bit-width at the input of the FFT; the second element specified the bit-width at the output of the first stage, the (n+1)th element specifies the bit-width at the output of the nth stage.

**TWIDDLEWIDTH** (type = DOUBLE): Scalar specifying the bit-width of the FFT twiddle factors (the same bit-width is used throughout the model).

**FFTSCALING** (type = DOUBLE): Scalar specifying the number of MSBs to discard at the output of the FFT. This appears as a factor of  $2^{\text{FFTSCALING}}$  at the output of the FFT.

**OUTWIDTH** (type = DOUBLE): Bit width at the output.

**OUTSCALING** (type = DOUBLE): Number of MSBs to discard at output.

**HANDLE** (type = UINT32): Handle to a xpPdft2Mat object that is then used in all further processing.

## 4.2 Runtime

The runtime function is used to stream data through the model.

The input data must be either vectors with the number of elements equal to the input length set at initialisation, or matrices with each column a vector of the correct input length to be passed into the Polyphase DFT.

Each input is shifted into the filter at the front end of the DFT, the filter operation is carried out and the DFT is evaluated. The output is then phase corrected (if specified at startup) and provided as an output.

```
[OUTI, OUTQ, OVERFLOW] = xpPdft2Mat(HANDLE, INI, INQ)
```

**HANDLE** (type = UINT32): Handle of valid xpPdft2Mat object obtained from INIT function

**INI** (type = INT64): In-phase input data to the Polyphase DFT. This is either a vector with the number of elements equal to the input length (specified at initialisation) or a matrix with the number of rows equal to the input length.

`INQ` (`type = INT64`): Quad-phase input data to the Polyphase DFT. This is either a vector with the number of elements equal to the input length (specified at initialisation) or a matrix with the number of rows equal to the input length.

`OUTI` (`type = INT64`): In-phase output data. This is either a vector with the number of elements equal to the number of FFT points or a matrix with the number of rows equal to the number of FFT points depending on the format of the input data.

`OUTQ` (`type = INT64`): Quad-phase output data. This is either a vector with the number of elements equal to the number of FFT points or a matrix with the number of rows equal to the number of FFT points depending on the format of the input data.

`OVERFLOW` (`type = STRUCT ARRAY`): A struct element is provided for each frame processed. The struct contains three elements providing overflow information: 'filter', 'fft' and 'out'.

'filter': The LSB of 'filter' is 1 if the input to the filter is outside the specified bit width. The second LSB is 1 if the output of the filter is outside the specified bit width.

'fft': The LSB of 'fft' is 1 if the input of the FFT is outside the specified bitwidth. The second LSB detects overflow at the output of the first stage of the FFT, the (n+1)th LSB detects overflow at the output of the nth stage.

'out': The LSB is 1 if the input to the phase correction is outside the specified bit width. The second LSB is 1 if the output of the phase correction is outside the specified bitwidth.

Note that calculating overflow information is a significant processing overhead - so unless overflow information is required - this output should not be required.

### 4.3 Clear

All `xpPdft2Mat` objects are destroyed when the dll is cleared from memory (i.e. call "clear `xpPdft2Mat`") 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.

```
xpPdft2Mat(HANDLE, 'clear')
```

`HANDLE` (`type = UINT32`): Handle of valid `xpPdft2Mat` object obtained from `INIT` function

### 4.4 Help

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

```
xpPdft2Mat('help')
```

### 4.5 Version

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

```
[DESCRIPTION] = xpPdft2Mat('version')
```

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

## 5 Examples

### 5.1 Comparison with Standard Floating Point DFT

This example compares the performance of a standard floating point DFT with the fixed point polyphase DFT.

*Note: The example is for a radix-2 Polyphase DFT with 10 stages, but is completely parameterisable, so that if your model does not support DFTs of order 10, then simply changing this number will permit the example to run.*

The example is based on a 1024 point Polyphase DFT with the following filter parameters:

Equiripple FIR:  $F_{pass} = 1/2048 * F_s$ ;  $F_{stop} = 3/2048 * F_s$ ;  $A_{stop} = 60\text{dB}$

That is to say that the filter passband is the width of a single DFT bin, there is an overlap of a single DFT bin and the stopband attenuation is 60dB. The filter was designed using FDATool in the Matlab Signal Processing Toolbox.

The input to the system is at a complex sample rate of 102.4MHz (i.e. AtoD rate of 204.8MHz) and the decimation through the polyphase DFT is 1024 samples. This implies an output rate per bin of 1MHz. Thus each output is critically sampled.

The remaining parameters are given by:

Input Width = 10 Bits; Tap Width = 15 bits; Filter Scaling = 2; Twiddle Width = 17 bits; DFT Stage Widths = [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. Therefore, the bit width grows by one bit per FFT stage.

In the example, all scalings applied throughout the system, in quantising the filter taps, etc are accounted for so that the output may be correctly scaled to provide the same output scaling as for the standard floating point DFT.

The various gains that must be accounted for are as follows:

- The scaling applied to the input signal to make it floating point.
- The DC gain of the filter (i.e. the sum of all the taps)
- The scaling applied to the floating point filter taps to convert them to fixed point. This must be divided by the number of DFT points.
- The scaling due to the LSBs dropped at the output of the filter (the input to the DFT). This is given by:  $2^{(-\text{lsbsDropped})}$ .
- The gain through the DFT, which may be found from:  $2^{(\text{STAGEWIDTHS}(\text{end}) - \text{STAGEWIDTHS}(1) - \log_2(\text{FFTPPOINTS}))}$ .

The total gain through the system is the product of all these.

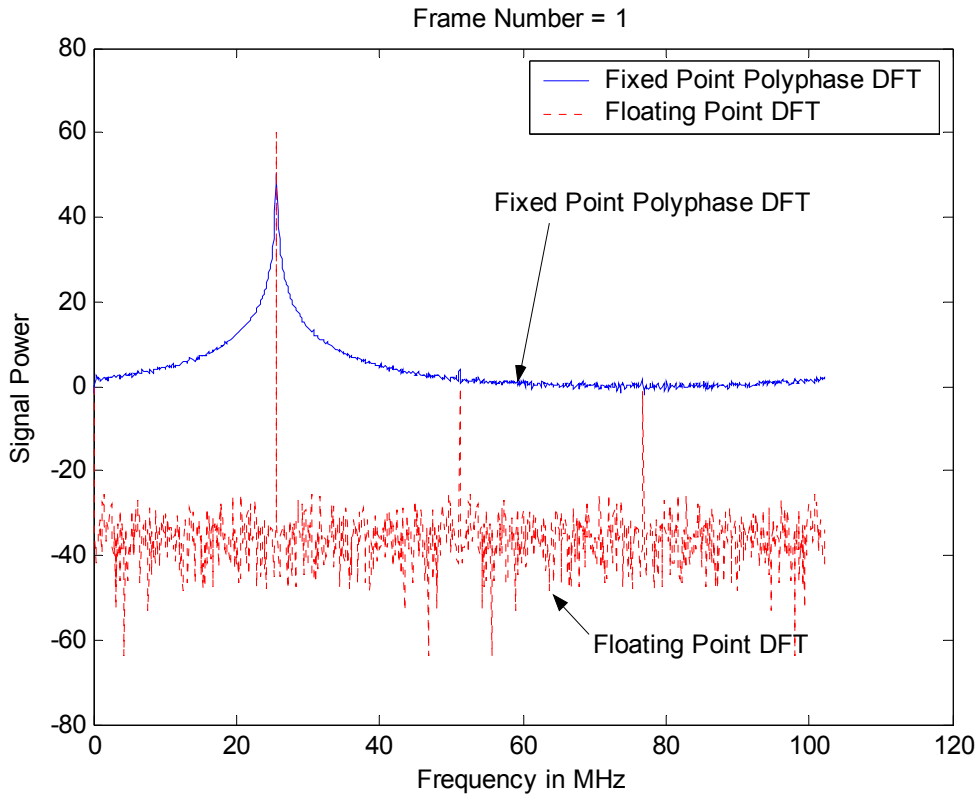
Note also that in order to get the true response from the Polyphase DFT, the filter must be allowed to fill up. This implies that the number of frames must be at least:

$$\text{Number of Taps} / \text{Number Of DFT Points}$$

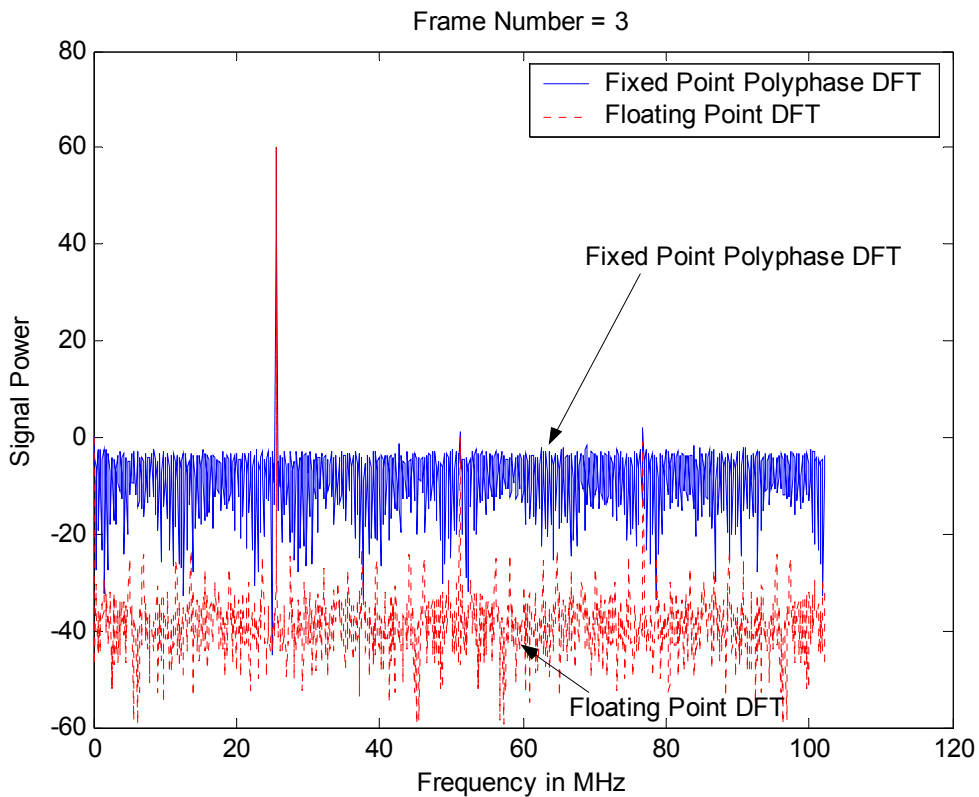
Figure 2 and Figure 3 show the results of the input of a sinusoid at  $F_s/4$  to the system described above. Figure 2 shows the output of the first frame, where the filters are still settling, while Figure 3 shows the output of the third frame, where the filter have been allowed to settle.

The code for this example may be found in the file:

[xpPdf2Mat\\_example1.m](#) (link only works in the Matlab help browser)



**Figure 2: Spectral Plot for Signal at  $F_s/4$  – Showing Filter Fill Time**



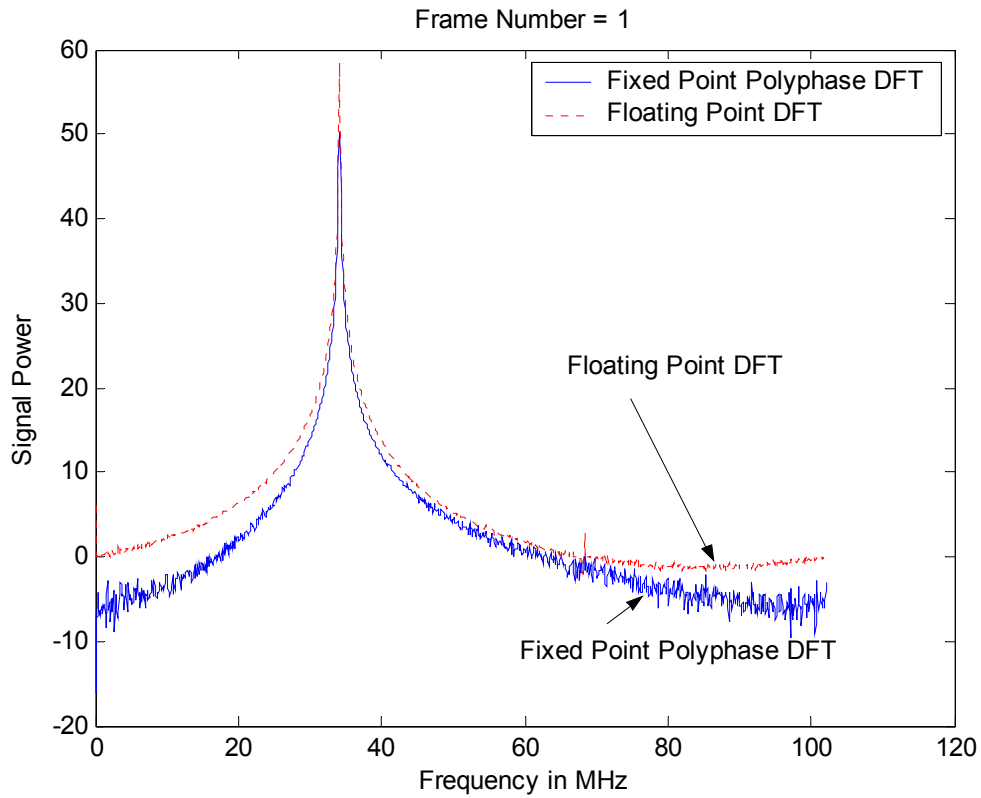
**Figure 3: Spectral Plot for Signal at  $F_s/4$  – Filter Settled**

In fact, this example does not show up the advantages of the polyphase DFT, since the chosen frequency is in the centre of a bin. If we now choose a different

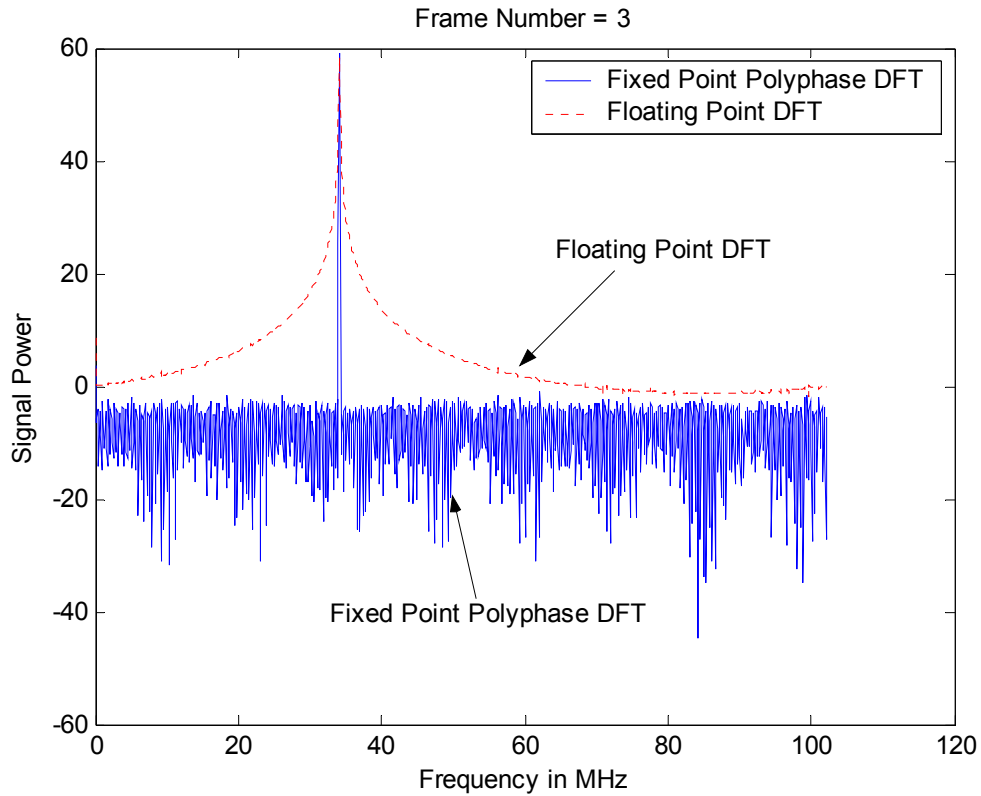
frequency, which is not at the centre of a bin such as  $F_s/3$ , the advantages become obvious. The results are shown in Figure 4 and Figure 5, where it is clear that once the polyphase DFT filter has settled (Figure 5) it has similar frequency selectivity to that shown in Figure 3, while the standard DFT has very poor selectivity.

The code for this example is found in:

[xpDft2Mat\\_example1b.m](#) (link only works in the Matlab help browser)

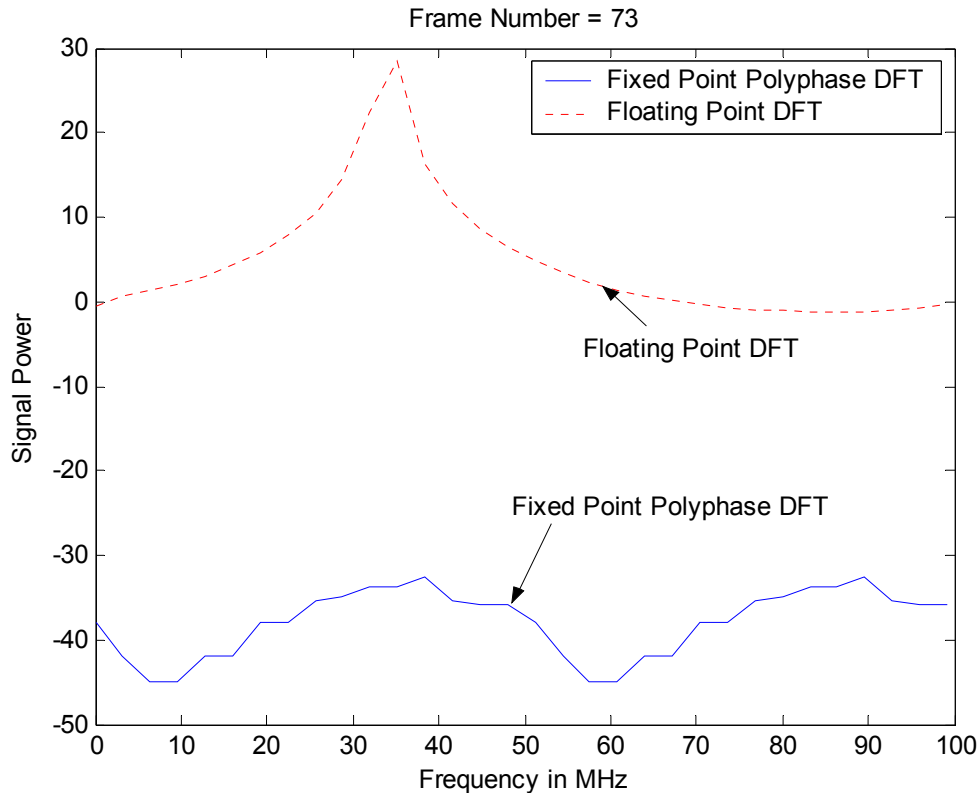


**Figure 4: Spectral Plot for Signal at  $F_s/3$  – Showing Filter Fill Time**



**Figure 5: Spectral Plot for Signal at  $F_s/3$  – Filter Settled**

Note that, if your model is for a lower order Polyphase DFT or you try this example with a lower order, you may find that for the Polyphase DFT response the signal actually disappears. This is shown in Figure 6 for a 32-point DFT. Though this appears to be a fault, it is in fact correct behaviour. For the lower FFT order, the signal actually falls outside the filter mainlobe, so that it is excluded by the higher filter selectivity. This demonstrates the importance of the filter design. Setting the desired frequency closer to the centre of one of the bins would cause it to reappear again.



**Figure 6: Spectral Plot for Signal at  $F_s/3$  – Filter Settled (32-Point DFT)**

## 5.2 Detection of Low-Level Signals

This example presents a similar scenario, but this time the desired signal is at  $-20\text{dB}$  at  $f_s/4 = 25.6\text{MHz}$  and an interfering signal is located 5.5 bins below it at  $26.15\text{MHz}$  with a power of  $0\text{dB}$ .

*Again, note that the example is for a radix-2 FFT of order 10, but is completely parameterisable, so that if your model does not support FFTs of order 10, then simply changing this number will permit the example to run. However, this will affect the spectral response – you may need to play with the parameters to get similar effects.*

In addition, complex white Gaussian noise is added at  $-30\text{dB}$ .

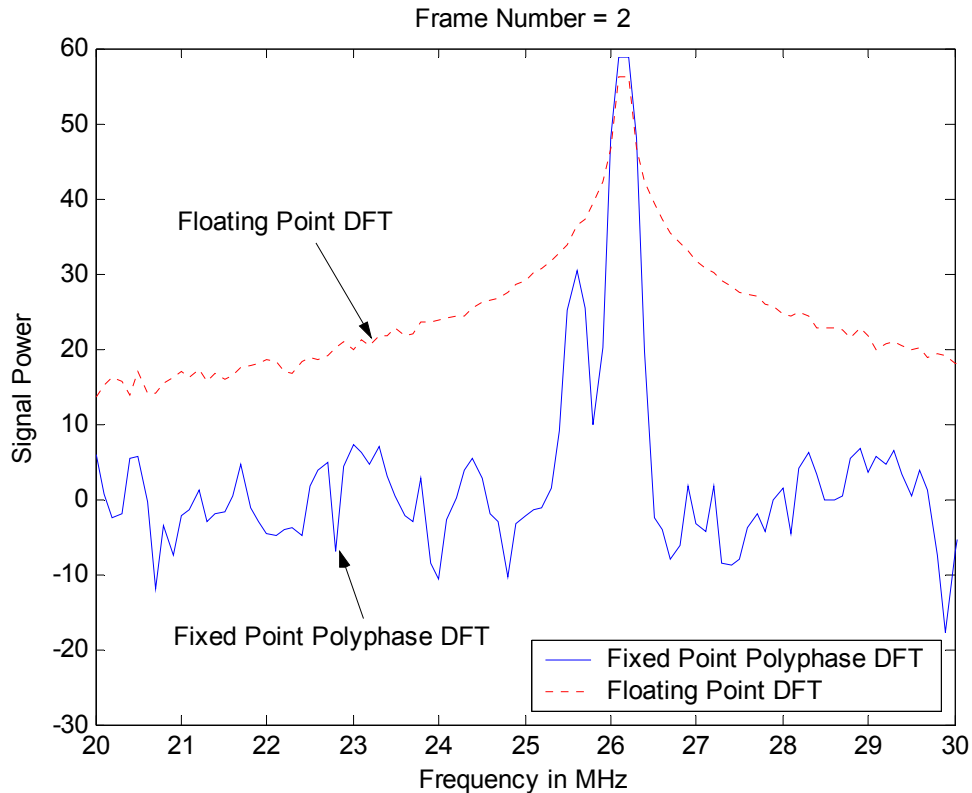
The example is based on a 1024 point DFT with a polyphase filter with the following parameters:

Equiripple FIR:  $F_{\text{pass}} = 1/2048 * F_s$ ;  $F_{\text{stop}} = 3/1024 * F_s$ ;  $A_{\text{stop}} = 80\text{dB}$

Figure 7 shows the output of the Standard DFT and the Polyphase DFT, once the filters have filled up. It is quite clear that it is impossible to detect the presence of the desired source using the standard DFT, but in the Polyphase DFT spectrum it is clearly visible.

The script to run this example is:

[xpPdft2Mat\\_example2.m](#) (link only works in the Matlab help browser)



**Figure 7: Power Spectrum with High-Powered Interfering Signal**

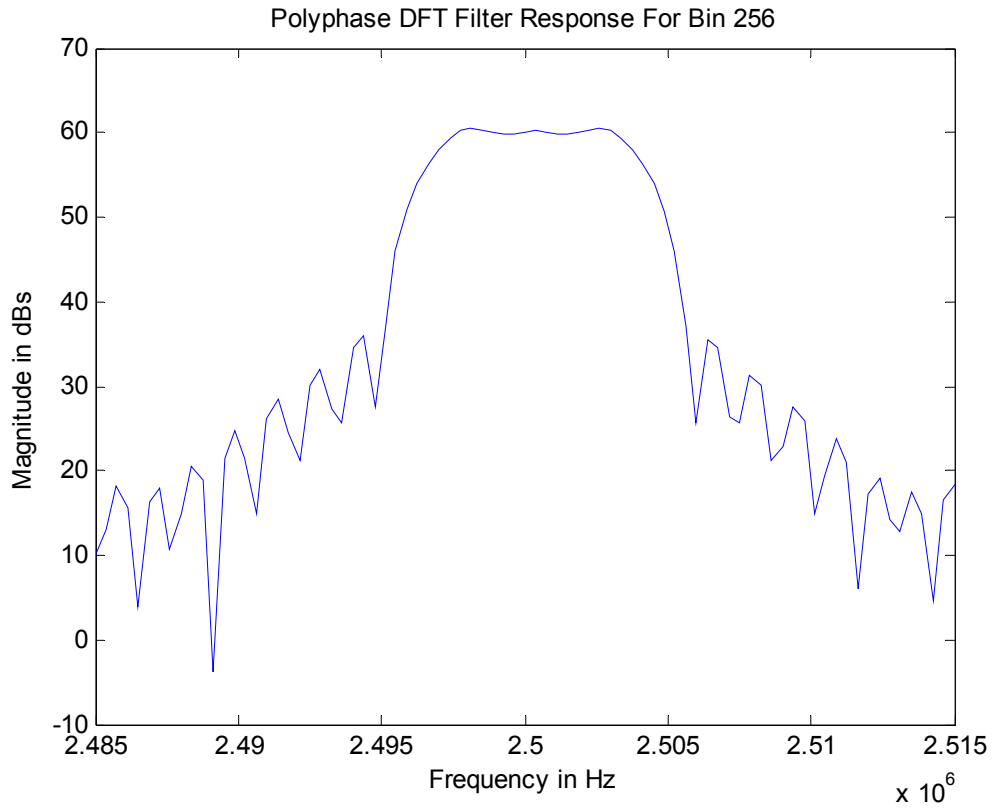
### 5.3 Root Raised Cosine Filter Response

The Polyphase DFT may be used to filter and downconvert a large number of evenly spaced communication channels. The demonstration of this is quite involved, so here we restrict ourselves to demonstrating the filter response for a particular bin of the Polyphase DFT. The filter prototype is a Root Raised Cosine filter with a rolloff of 35%, designed using the FIRRCOS function in the Matlab Signal Processing Toolbox.

The script scans across a frequency range around the desired bin, gives the filter time to settle and the stores the magnitude value of the desired bin output. The resulting frequency response is shown in Figure 8.

The script to run this example is:

[xpPdft2Mat\\_example3.m](#) *(link only works in the Matlab help browser)*



**Figure 8: RRC Filter Response Example**