

Application Note

13

Digital FIR Coefficient Packing

Highlights

Digital-FIR Coefficient Packer

Digital-FIR Optimal Standard

Digital-FIR Root Editor

Digital-FIR Fixed Point Coefficient Export

Variable Fixed Point Implementations

Reducing FPGA logic or LSI circuitry

■ Design Objective

Design several FIR filters and reduce their coefficient complexity.

Examine the changes in the coefficients

View the changes in the filter response

View the changes in the Z-Roots

Packing coefficients is a term frequently used when implementing FIR filters with FPGA logic or directly within LSI circuitry. This can be defined simply as reducing the size of the binary coefficients to a minimum. There are actually two aspects to the size of a coefficient that may be important:

- (1) The fixed point word size in bits
- (2) The number of 1's within each coefficient

Reducing the word size is clearly important. A coefficient of 16 bits requires less logic to implement than a coefficient of 24 bits. The second factor concerns the amount of logic required to implement specific step sizes given the resolution of the word size.

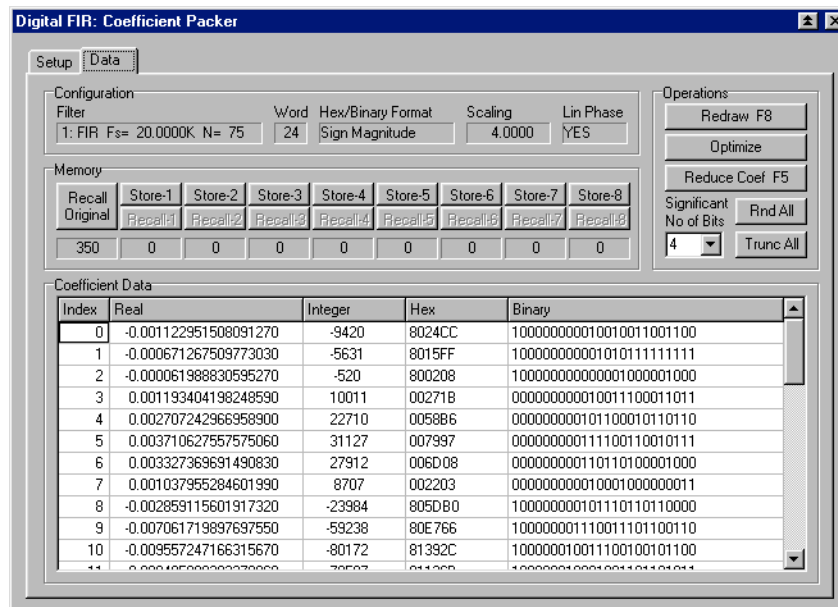
For example, if a coefficient had only a single '1' in the binary word with all other bits zero, then multiplying by the coefficient would merely require a number of bit shifts. This is far easier than a full fixed point multiplication.

A lot depends on what sort of logic algorithm is used to implement the filter operation. However in many cases the number of 1's in a binary coefficient closely correlates with the number of *terms* required in the logic. Therefore it may be important to minimize the number of 1's or terms in the binary coefficients.

■ The Digital-FIR: Coefficient Packer dialog

Packing coefficients is really nothing more than an exercise in sophisticated rounding and/or truncating. However when performed by hand this can be somewhat tedious.

The *Digital-FIR: Coefficient Packer* dialog on the Target menu makes this much easier. Not only does this dialog provide efficient means for manually editing the coefficients, it also provides a *fully automatic* means of optimizing for minimum binary size.



The *Data* page of the packer dialog is shown above. The coefficients are displayed in four different forms: *Real*, *Integer*, *Hex*, and *Binary*. Any of these forms may be edited and the other forms will be automatically recomputed to match. The dialog performs seamless conversion of the coefficients between any of the numeric formats on a continual basis while editing.

At any time during the editing, the FIR filter can be recalculated and viewed on the graphs using the *Redraw* button. A memory system is also implemented to allow previous configurations to be stored and recalled. This allows for easy trial & error comparisons of different coefficient sets.

■ Example #1

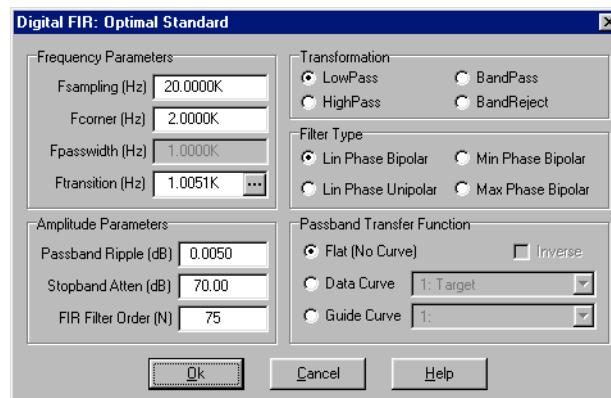
The procedure and results can best be demonstrated by a couple examples. For this first example the following design goal parameters will be used:

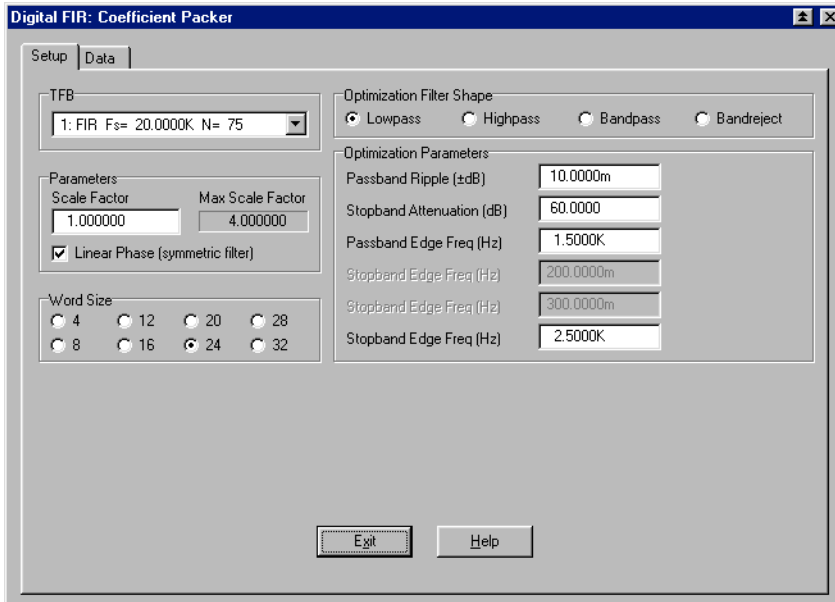
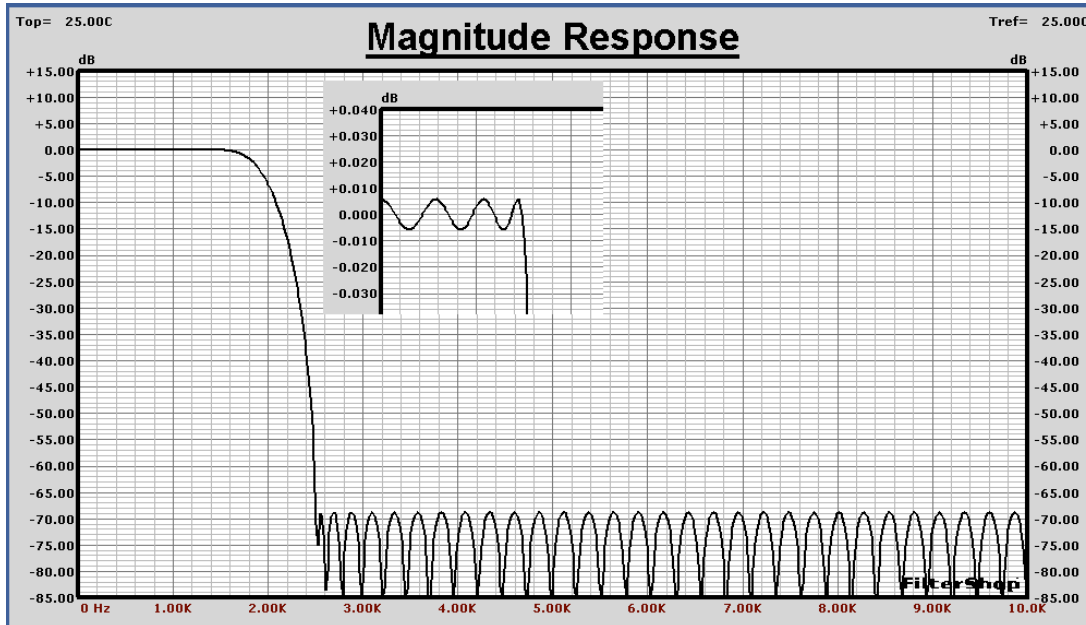
- FIR Lowpass
- $F_s = 20\text{kHz}$
- $F_c = 2\text{kHz}$
- $F_t = 1\text{kHz}$ ($F_{\text{pass}}=1.5\text{kHz}$, $F_{\text{stop}}=2.5\text{kHz}$)
- Ripple $< 0.01\text{dB}$
- Attenuation $> 60\text{dB}$

Before we can reduce the coefficients of the FIR filter, we must first create an FIR filter. For this example a simple optimal equal ripple will be used. The inevitable price paid for reducing or rounding the coefficients, is that the ripple and attenuation may no longer be equal ripple. In fact each may become slightly worse. For this reason it is generally best to *over* design the optimal filter prior to packing. This gives more freedom in rounding the coefficients to achieve better packing.

The *Digital-FIR: Optimal Standard* dialog was used to create the filter as shown here. For this example the ripple was designed at 0.005dB and the attenuation at 70dB. This provides some working room for packing.

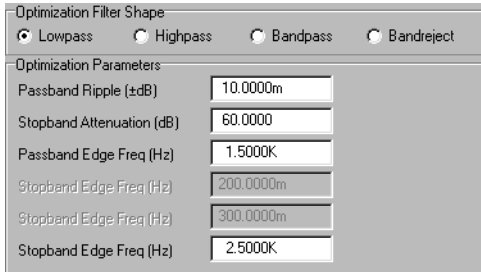
The order of the FIR came out to be about 75 for this design. The response of the filter is shown on the following page.





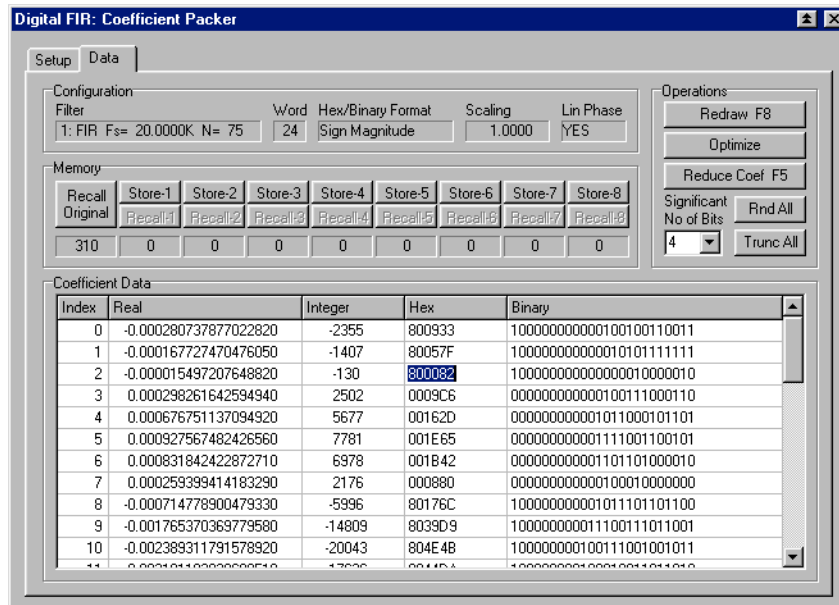
We now open the Packer dialog to setup the parameters. For this design we will simply choose 24 bit fixed coefficients for the initial formatting.

Note that the maximum scaling factor has been computed as 4. This means that we could scale the coefficients up to better use the fixed point word size. However for this example we shall leave the scaling at 1.0.



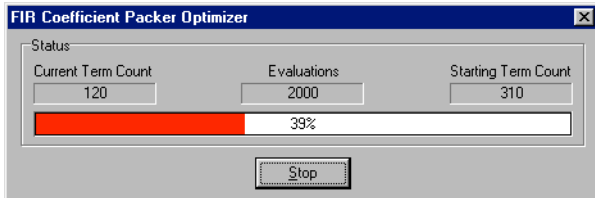
The constraints for the optimization process are setup in the panels as shown here on the left. Here we define the exact goals for the filter. Since the filter is Linear Phase, only half the coefficients are operated on with the other half being mirrored automatically.

When all the setup parameters have been defined, then we selected the *Data* tab page. The 24 bit fixed point coefficients are now shown below.



The Hex/Binary format used for the packer is exclusively Sign-Magnitude. Although Two's Complement is typically used for operation with DSPs, Sign-Magnitude is often the preferred choice for discrete logic implementation. Moreover, both positive and negative coefficients have the same number of 1's (terms) in their magnitude representations. However when exporting the coefficients later either Two's Complement, Sign-Magnitude, or Offset Binary can be deployed.

Note the value shown under the *Recall Original* memory button. This is the number of 1's in the magnitude portion of all coefficients: 310. We seek to reduce this value.

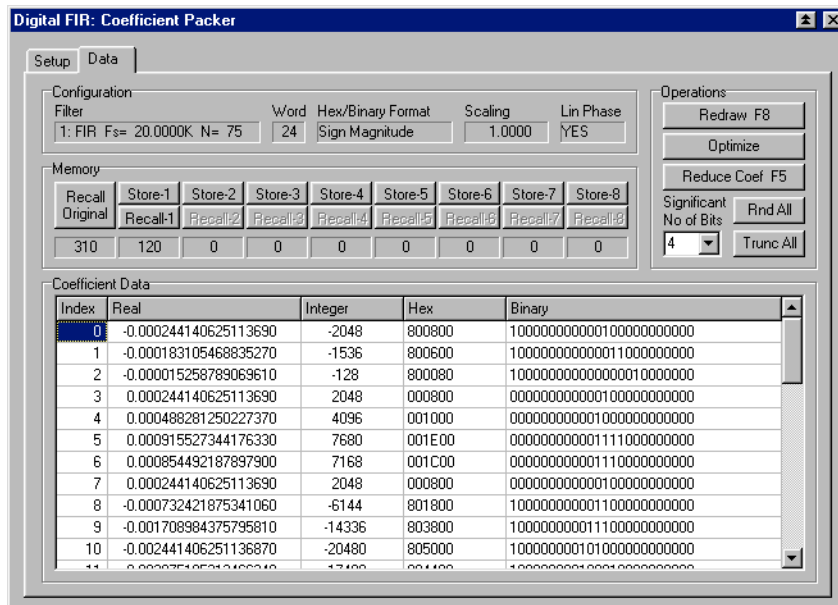


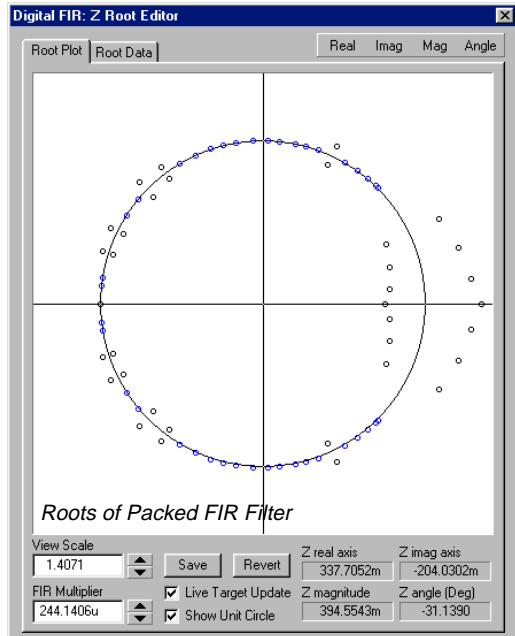
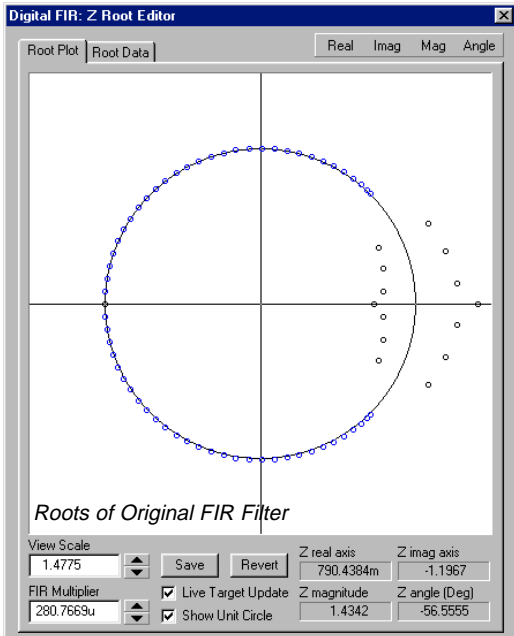
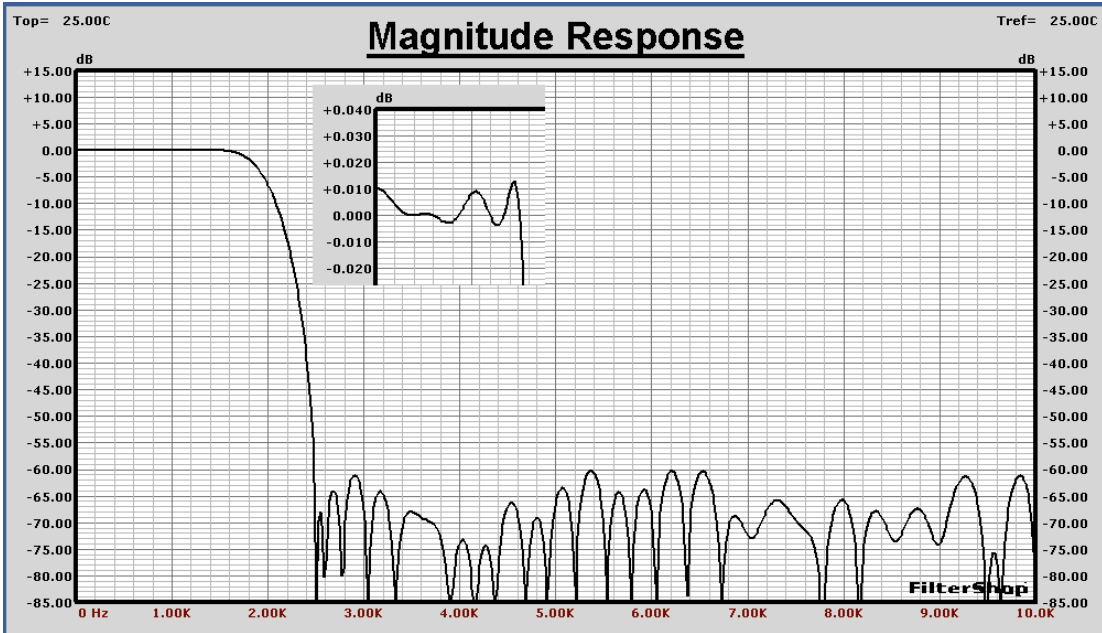
We could certainly attempt to reduce the coefficient sizes through manual editing, but instead we will simply use the *Optimize* button. In about 1 minute the number of terms is reduced from 310 to 120. The filter is now only about 39% of its original size.

The coefficient results are shown below. Note that many more of the Binary coefficients are now zero. Most of the coefficients are only 14 bits long, with a small number of ones in each value.

We could further attempt to reduce the coefficients through manual editing if we desired. Several features are provided on the dialog for this purpose. However in this example we will just go with the optimized result.

The FIR response of the packed coefficients is now shown on the following page, along with a comparison of the root locations. The packed filter has some of the zero roots moved off the unity circle.





The passband ripple is well below the $\pm 0.01\text{dB}$ constraint, and the stopband attenuation is below the 60dB constraint.

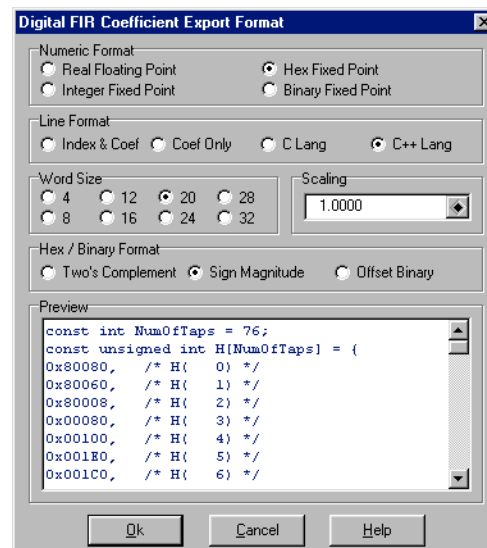
Equal ripple behavior has been lost, but this is generally unimportant for many applications. This was the trade-off for reducing the coefficient complexity.

Exporting the FIR data

There are many options available for exporting the coefficient data from FilterShop. In the case of fixed point coefficients, we must choose a specific word size.

Examining the coefficients we see that the widest coefficient occupies 17 bits. Therefore the 20 bit word size will be used to contain the data in five even Hex digits. The C++ formatting will be used. If we had desired Two's Complement or Offset Binary, this could also be obtained at this time.

The final complete coefficient export file is shown on the following page.



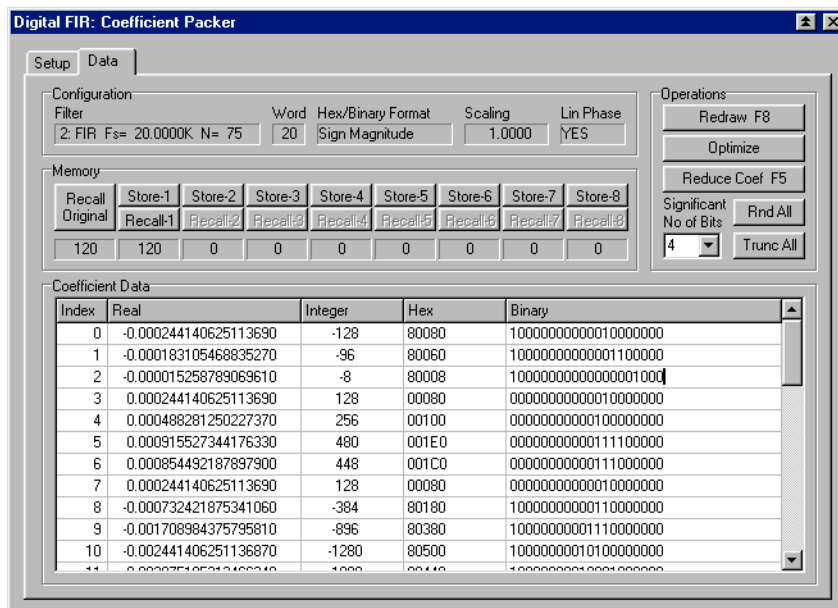
```

/* =====*/
/* FIR Filter Coefficient Data */
/* =====*/
/* (C)opyright 1993-2000 by LinearX Systems Inc*/
/* FilterShop(TM) Version=3.2.0.685*/
/* Date=Sep 4, 2000 Time=Mon 4:13 am*/
/* Design= AppNote13_1.fsd*/
/* =====*/
/* Fs(Hz)= 20.0000K*/
/* Order= 75*/
/* Coef Line Format= C++ Language*/
/* Coef Numeric Format= Hex Fixed Point*/
/* Coef Scaling= 1.0000 */
/* Coef Word Size= 20*/
/* Coef Hex/Bin Format= Sign Magnitude*/
/* =====*/
const int NumOfTaps = 76;
const unsigned int H[NumOfTaps] = {
0x80080, /* H( 0) */
0x80060, /* H( 1) */
0x80008, /* H( 2) */
0x00080, /* H( 3) */
0x00100, /* H( 4) */
0x001E0, /* H( 5) */
0x001C0, /* H( 6) */
0x00080, /* H( 7) */
0x80180, /* H( 8) */
0x80380, /* H( 9) */
0x80500, /* H(10) */
0x80440, /* H(11) */
0x80160, /* H(12) */
0x00340, /* H(13) */
0x00800, /* H(14) */
0x00AA0, /* H(15) */
0x00940, /* H(16) */
0x00340, /* H(17) */
0x80600, /* H(18) */
0x80F80, /* H(19) */
0x814C0, /* H(20) */
0x81200, /* H(21) */
0x806C0, /* H(22) */
0x00B00, /* H(23) */
0x01D00, /* H(24) */
0x02700, /* H(25) */
0x022C0, /* H(26) */
0x00DC0, /* H(27) */
0x81400, /* H(28) */
0x83780, /* H(29) */
0x84E80, /* H(30) */
0x849C0, /* H(31) */
0x82080, /* H(32) */
0x02D80, /* H(33) */
0x094C0, /* H(34) */
0x10200, /* H(35) */
0x15CC0, /* H(36) */
0x19080, /* H(37) */
0x19080, /* H(38) */
0x15CC0, /* H(39) */
0x10200, /* H(40) */
0x094C0, /* H(41) */
0x02D80, /* H(42) */
0x82080, /* H(43) */
0x849C0, /* H(44) */
0x84E80, /* H(45) */
0x83780, /* H(46) */
0x81400, /* H(47) */
0x00DC0, /* H(48) */
0x022C0, /* H(49) */
0x02700, /* H(50) */
0x01D00, /* H(51) */
0x00B00, /* H(52) */
0x806C0, /* H(53) */
0x81200, /* H(54) */
0x814C0, /* H(55) */
0x80F80, /* H(56) */
0x80600, /* H(57) */
0x00340, /* H(58) */
0x00940, /* H(59) */
0x00AA0, /* H(60) */
0x00800, /* H(61) */
0x00340, /* H(62) */
0x80160, /* H(63) */
0x80440, /* H(64) */
0x80500, /* H(65) */
0x80380, /* H(66) */
0x80180, /* H(67) */
0x00080, /* H(68) */
0x001C0, /* H(69) */
0x001E0, /* H(70) */
0x00100, /* H(71) */
0x00080, /* H(72) */
0x80008, /* H(73) */
0x80060, /* H(74) */
0x80080}; /* H(75) */
/* =====*/

```

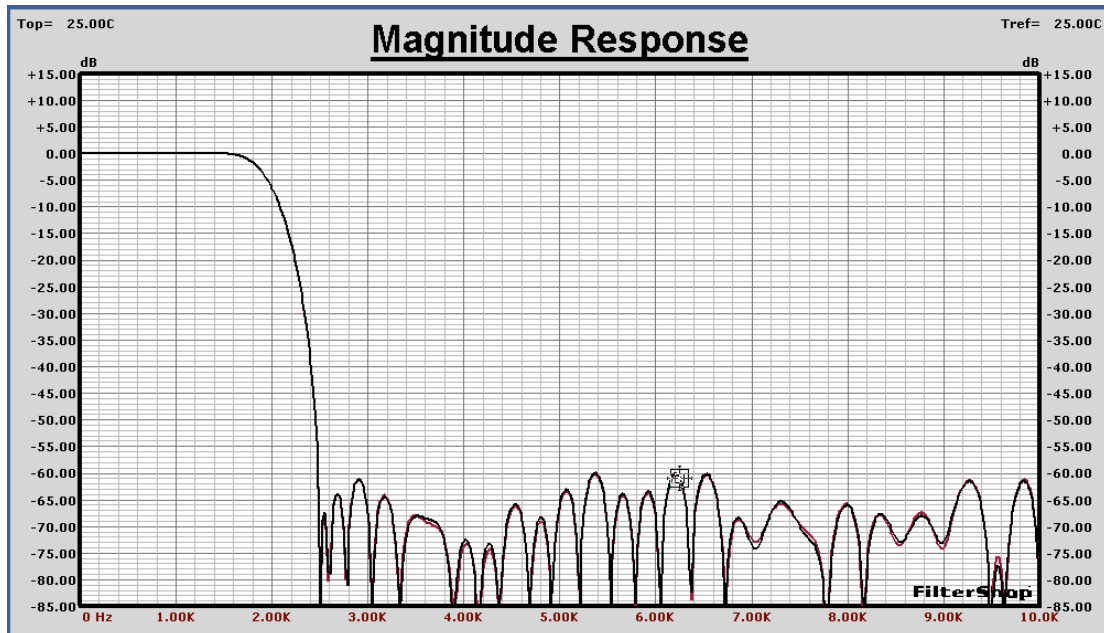
■ **Example #2**

The previous example demonstrated the ease of packing using the Optimize feature. However there is still some benefit to be gained by use of manual coefficient editing. In this example we shall take the previous packed filter and reduce it further.



This time the packer was setup using a 20-bit word size, since the previous optimization reduced the active number of bits to 17. Examining the coefficients we see that #2 is the widest coefficient of 17 bits. In fact it is the only coefficient using the full 17 bit word size.

Since the value of this coefficient is already near zero, we can simply edit the value to be zero and examine the changes in the response. This is shown on the following page. The original response is shown in Red, and the new response in Black. There is not much visible change in the response.



You might wonder why the optimizer did not perform this same adjustment. If we use the cursor to readout the value of the lobe at 5.3kHz we see that it is now -59.96dB. This violates the -60dB constraint so the optimizer could not allow this adjustment. However for this example we shall assume that the -60dB constraint is not that critical, and that we wish to see what manual adjustments can yield.

The widest coefficients are now 15 bits. We can again change the word size to 16 bits to eliminate some of the extra trailing zeros. This is shown on the next page. There are only 4 coefficients using the 15 bit word size: #1, #5, #12, and #15.

Truncating the LSB in each of these coefficients produces the response as shown in the following graph. The worst case lobe is now at 9kHz and at a level of -59.7dB. The passband ripple still remains easily within the ± 0.01 dB constraint.

Therefore with a little manual editing, we have further trimmed the coefficient word size from 17 bits down to 14 bits, and only lost 0.3dB from the stopband. The total term count is now 115. The final coefficients are shown on the following page.

Digital FIR: Coefficient Packer

Setup Data

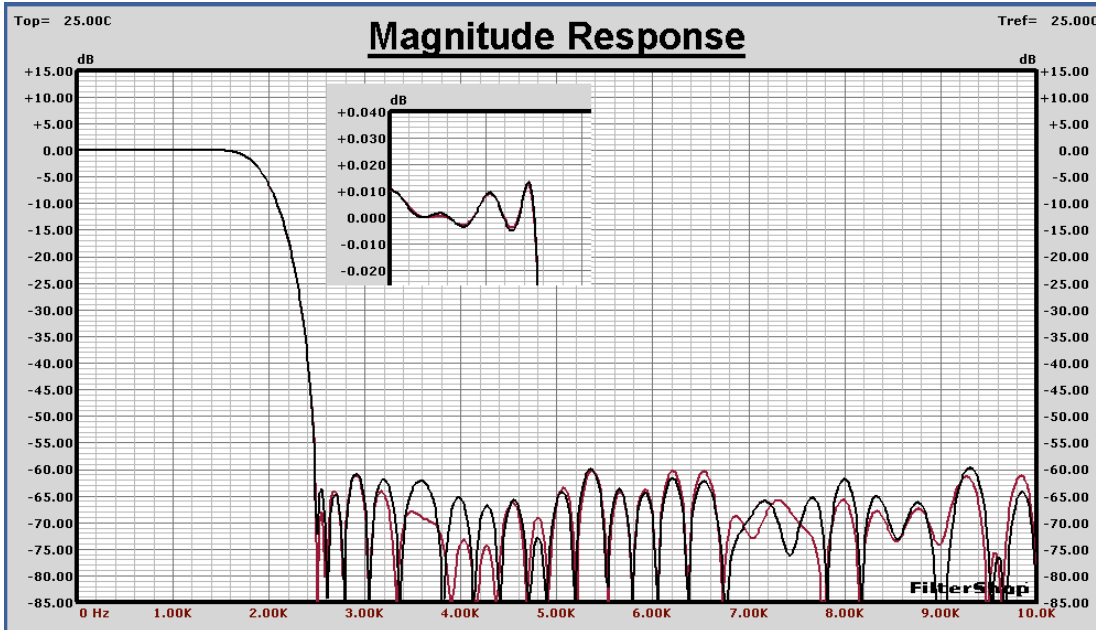
Configuration
 Filter: 2. FIR $F_s = 20.0000\text{K}$ $N = 75$ Word: 16 Hex/Binary Format: Sign Magnitude Scaling: 1.0000 Lin Phase: YES

Memory
 Recall Original Store-1 Store-2 Store-3 Store-4 Store-5 Store-6 Store-7 Store-8
 Recall1 Recall2 Recall3 Recall4 Recall5 Recall6 Recall7 Recall8
 119 0 0 0 0 0 0 0 0

Operations
 Redraw F8
 Optimize
 Reduce Coef F5
 Significant No of Bits: 4 Rnd All Trunc All

Coefficient Data

Index	Real	Integer	Hex	Binary
0	-0.000244140625113690	-8	8008	1000000000001000
1	-0.000183105468835270	-6	8006	1000000000000110
2	0.000000000000000000	0	0000	0000000000000000
3	0.000244140625113690	8	0008	0000000000001000
4	0.000488281250227370	16	0010	0000000000010000
5	0.000915527344176330	30	001E	0000000000011110
6	0.000854492187897900	28	001C	0000000000011100
7	0.000244140625113690	8	0008	0000000000001000
8	-0.000732421875341060	-24	8018	1000000000011000
9	-0.001708984375795810	-56	8038	1000000000111000
10	-0.002441406251136870	-80	8050	1000000001010000



```

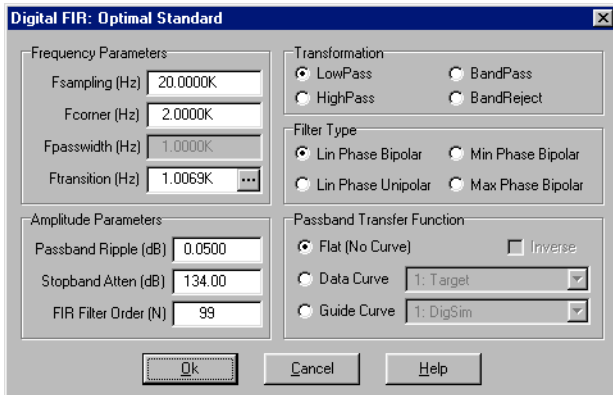
/* ===== */
/* FIR Filter Coefficient Data */
/* ===== */
/* (C)opyright 1993-2000 by LinearX Systems Inc */
/* FilterShop(TM) Version=3.2.0.685 */
/* Date=Sep 5, 2000 Time=Tue 11:42 pm */
/* Design= AppNote13_2.fsd */
/* ===== */
/* Fs(Hz)= 20.0000K */
/* Order= 75 */
/* Coef Line Format= C++ Language */
/* Coef Numeric Format= Hex Fixed Point */
/* Coef Scaling= 1.0000 */
/* Coef Word Size= 16 */
/* Coef Hex/Bin Format= Sign Magnitude */
/* ===== */
const int NumOfTaps = 76;
const unsigned int H[NumOfTaps] = {
0x8008, /* H( 0) */
0x8004, /* H( 1) */
0x0000, /* H( 2) */
0x0008, /* H( 3) */
0x0010, /* H( 4) */
0x001C, /* H( 5) */
0x001C, /* H( 6) */
0x0008, /* H( 7) */
0x8018, /* H( 8) */
0x8038, /* H( 9) */
0x8050, /* H(10) */
0x8044, /* H(11) */
0x8014, /* H(12) */
0x0034, /* H(13) */
0x0080, /* H(14) */
0x00A8, /* H(15) */
0x0094, /* H(16) */
0x0034, /* H(17) */
0x8060, /* H(18) */
0x80F8, /* H(19) */
0x814C, /* H(20) */
0x8120, /* H(21) */
0x806C, /* H(22) */
0x00B0, /* H(23) */
0x01D0, /* H(24) */
0x0270, /* H(25) */
0x022C, /* H(26) */
0x00DC, /* H(27) */
0x8140, /* H(28) */
0x8378, /* H(29) */
0x84E8, /* H(30) */
0x849C, /* H( 31) */
0x8208, /* H( 32) */
0x02D8, /* H( 33) */
0x094C, /* H( 34) */
0x1020, /* H( 35) */
0x15CC, /* H( 36) */
0x1908, /* H( 37) */
0x1908, /* H( 38) */
0x15CC, /* H( 39) */
0x1020, /* H( 40) */
0x094C, /* H( 41) */
0x02D8, /* H( 42) */
0x8208, /* H( 43) */
0x849C, /* H( 44) */
0x84E8, /* H( 45) */
0x8378, /* H( 46) */
0x8140, /* H( 47) */
0x00DC, /* H( 48) */
0x022C, /* H( 49) */
0x0270, /* H( 50) */
0x01D0, /* H( 51) */
0x00B0, /* H( 52) */
0x806C, /* H( 53) */
0x8120, /* H( 54) */
0x814C, /* H( 55) */
0x80F8, /* H( 56) */
0x8060, /* H( 57) */
0x0034, /* H( 58) */
0x0094, /* H( 59) */
0x00A8, /* H( 60) */
0x0080, /* H( 61) */
0x0034, /* H( 62) */
0x8014, /* H( 63) */
0x8044, /* H( 64) */
0x8050, /* H( 65) */
0x8038, /* H( 66) */
0x8018, /* H( 67) */
0x0008, /* H( 68) */
0x001C, /* H( 69) */
0x001C, /* H( 70) */
0x0010, /* H( 71) */
0x0008, /* H( 72) */
0x0000, /* H( 73) */
0x8004, /* H( 74) */
0x8008}; /* H( 75) */
/* ===== */

```

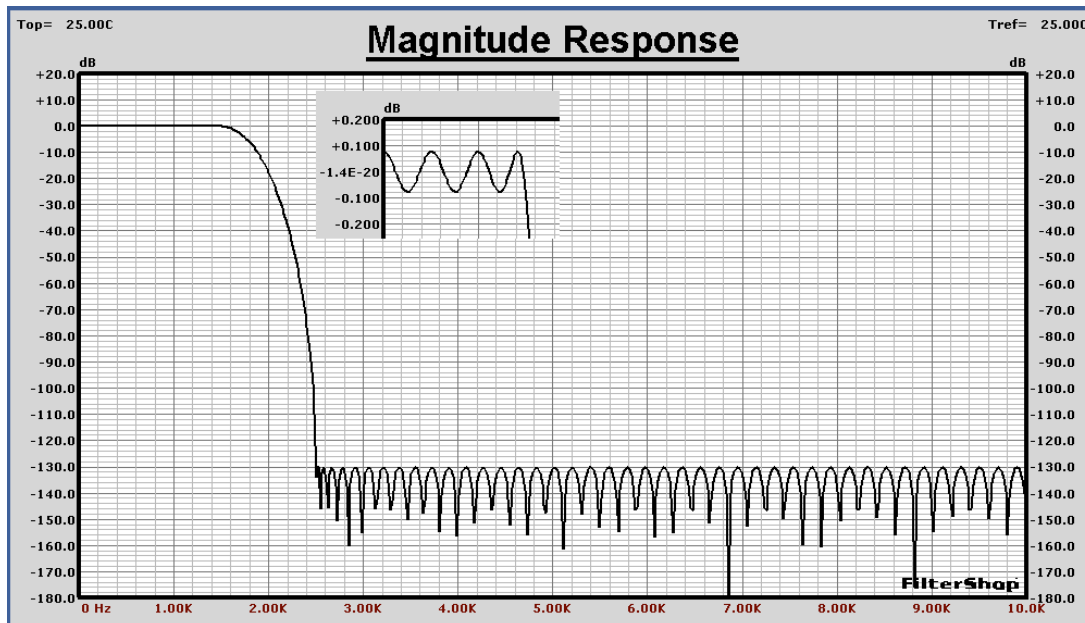
Example #3

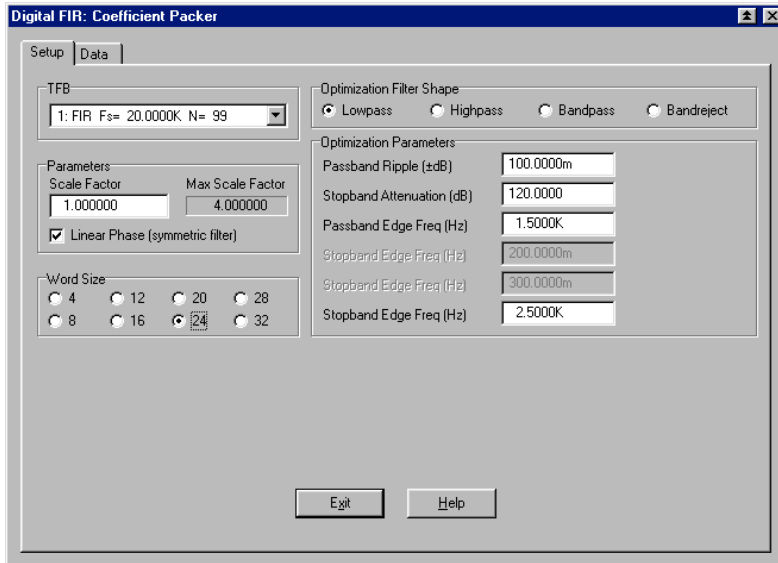
An example of higher stopband attenuation will now be demonstrated. The final design goals for this filter are:

- FIR Lowpass
- $F_s = 20\text{kHz}$
- $F_c = 2\text{kHz}$
- $F_t = 1\text{kHz}$ ($F_{pb}=1.5\text{kHz}$, $F_{sb}=2.5\text{kHz}$)
- Ripple = $\pm 0.1\text{dB}$
- Attenuation = 120dB



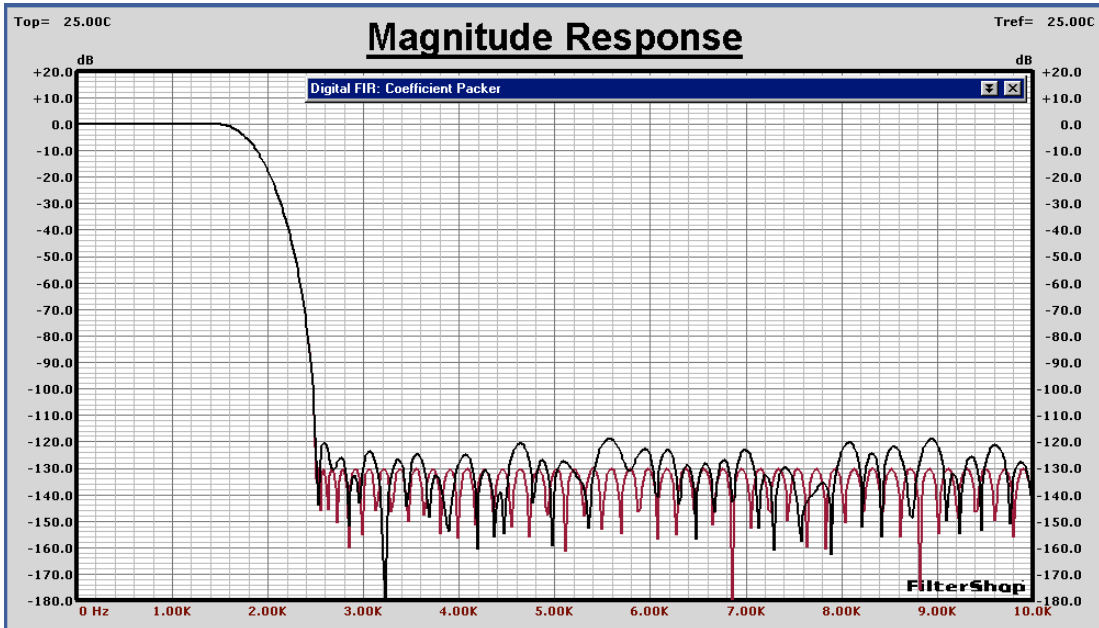
As before an initial FIR design slightly exceeding these specs is produce using the Optimal Standard FIR dialog. The order chosen here was 99. The response of the beginning filter is shown below.

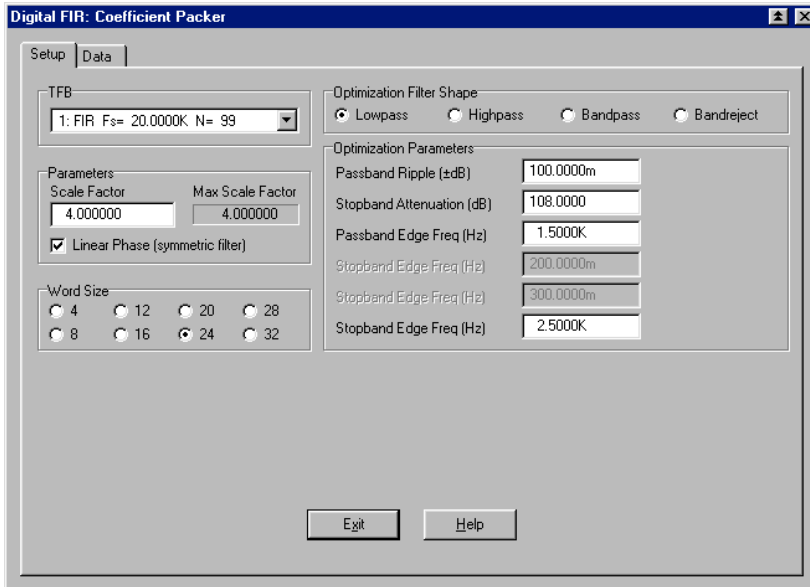




The packing dialog is initially setup with the starting parameters as shown here. A word size of 24 bits is selected. Note that a maximum scaling factor of 4 is possible.

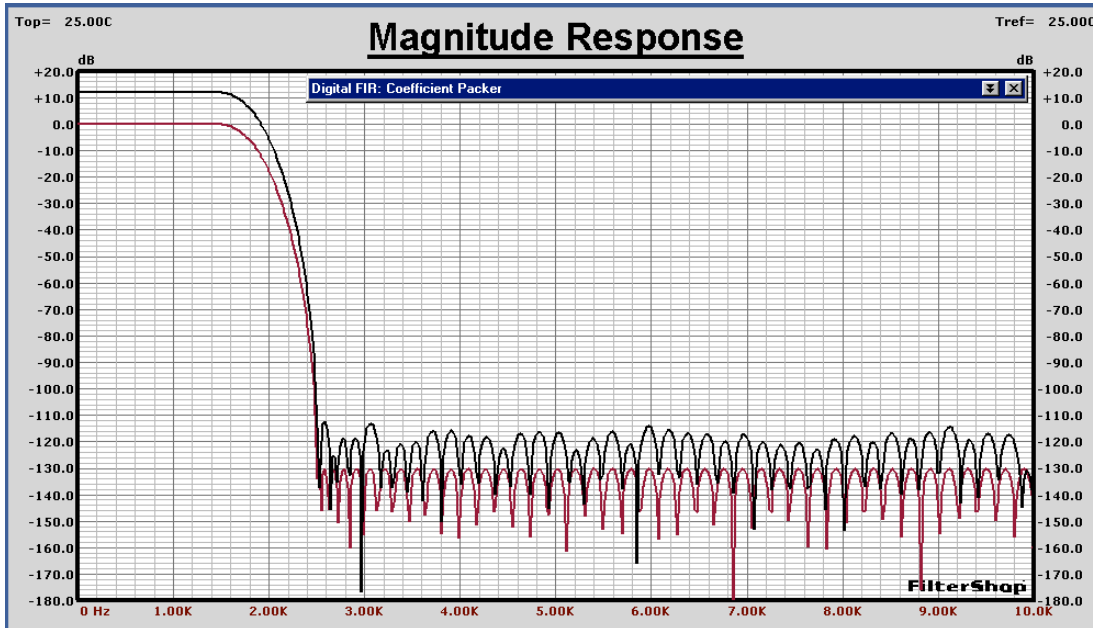
The response below in Black shows the resulting response with 24 bit coefficients. The response already violates the 120dB constraint before any packing even begins. We could choose to increase the word size, or we could scale the coefficients by 4.0 for better utilization.

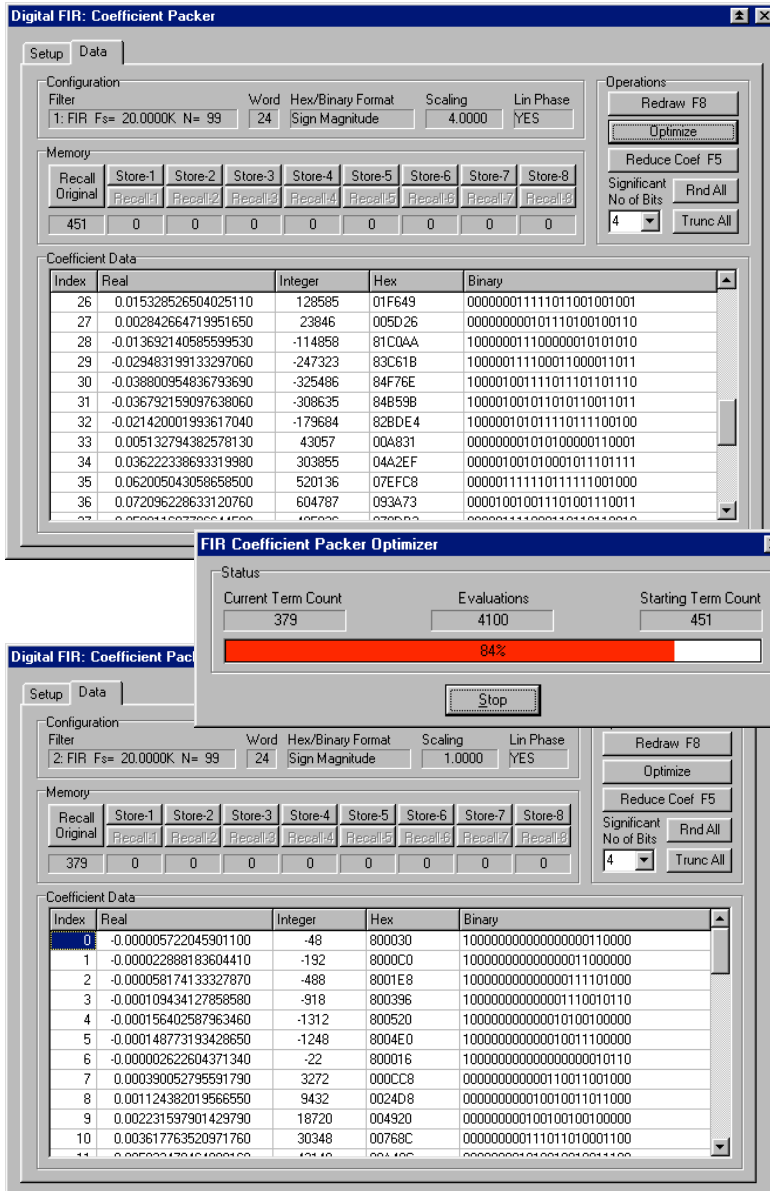




To scale the coefficients a value of 4 is entered, and the stopband attenuation is changed to 108dB. This is necessary if we wish to keep the 120dB stopband *relative* to the passband level. Since we are adding 12dB to the passband (1:4) the same must be added to the stopband. Ripple remains unchanged.

The response is shown below in Black. The atten is now 113dB.





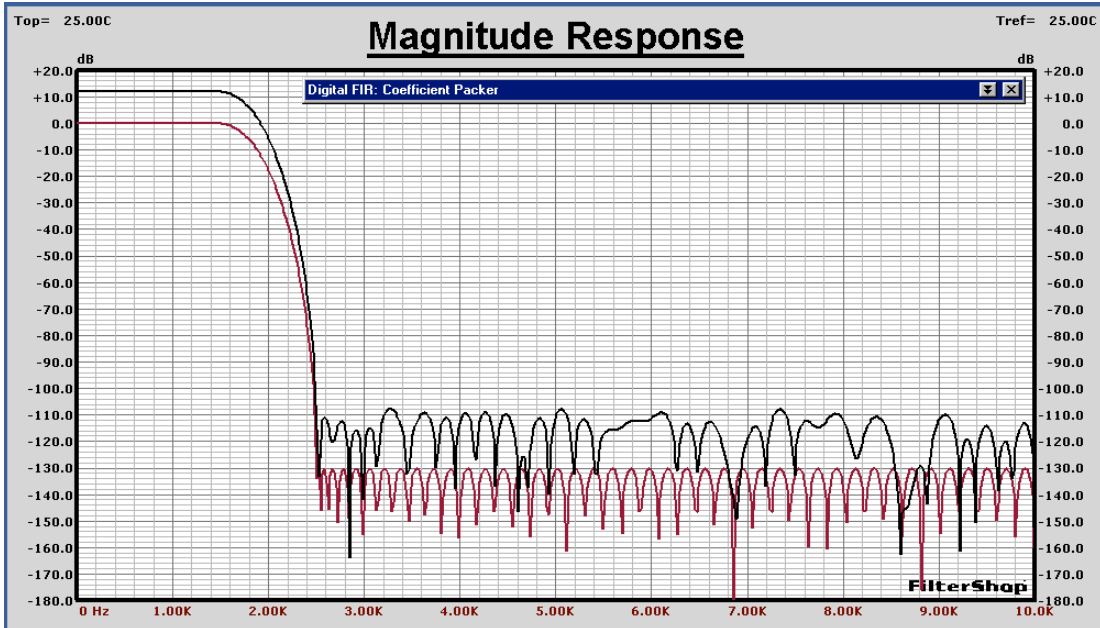
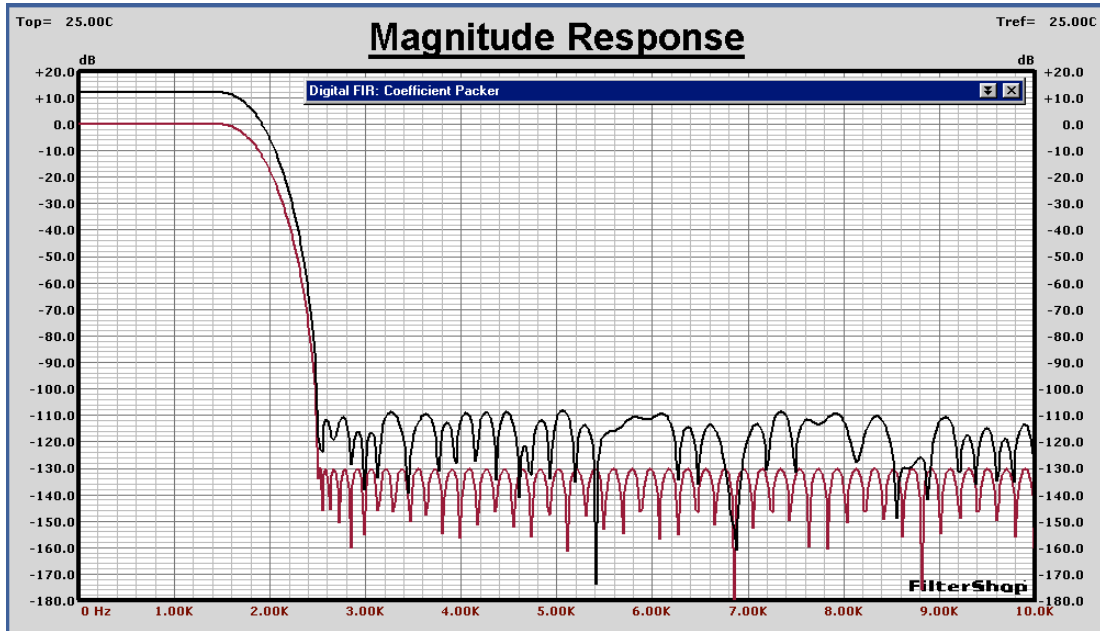
The 24 bit coefficients are shown here. The total number of terms (1's) is 451. Since the response was already slightly difficult to hold in the 24 bit word size, we should expect that less packing will be possible than the previous example.

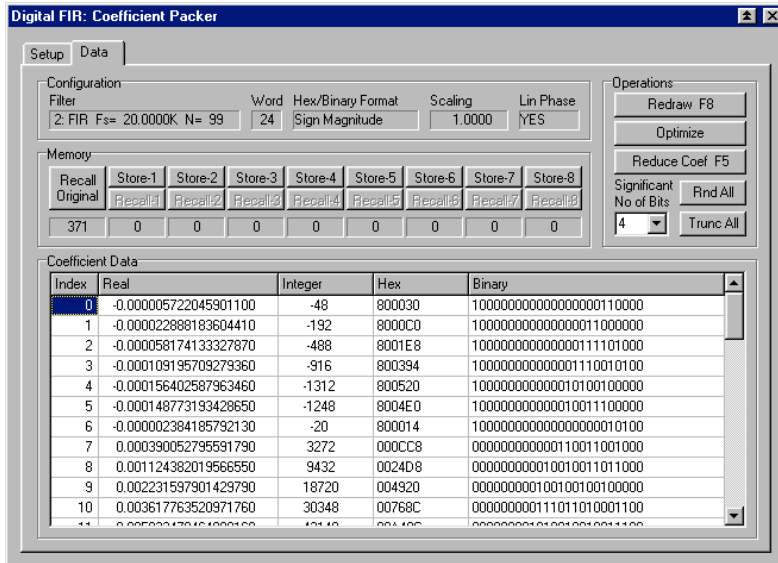
Running Optimize reduces the term count down to about 84% of its original size, or 379. The coefficient results are shown below.

Inspecting the coefficients shows that only three values are using the LSB bit. Manually truncating those bits to zero, the response on the lower graph of the next page is produced. Not much significant change from the upper graph.

The actively used coefficient bits now span 23. Examining the Binary values again shows that only 5 coefficients use the 23rd LSB bit.

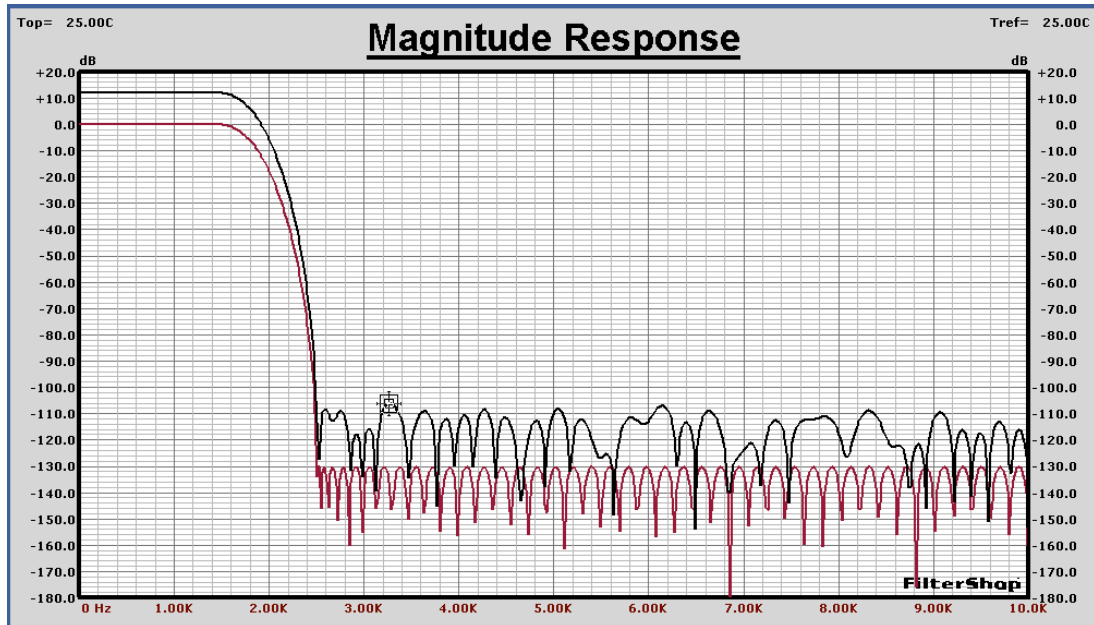
To further reduce the effective coefficient word size, we can truncate these bits to zero, leaving a 22 bit word size.

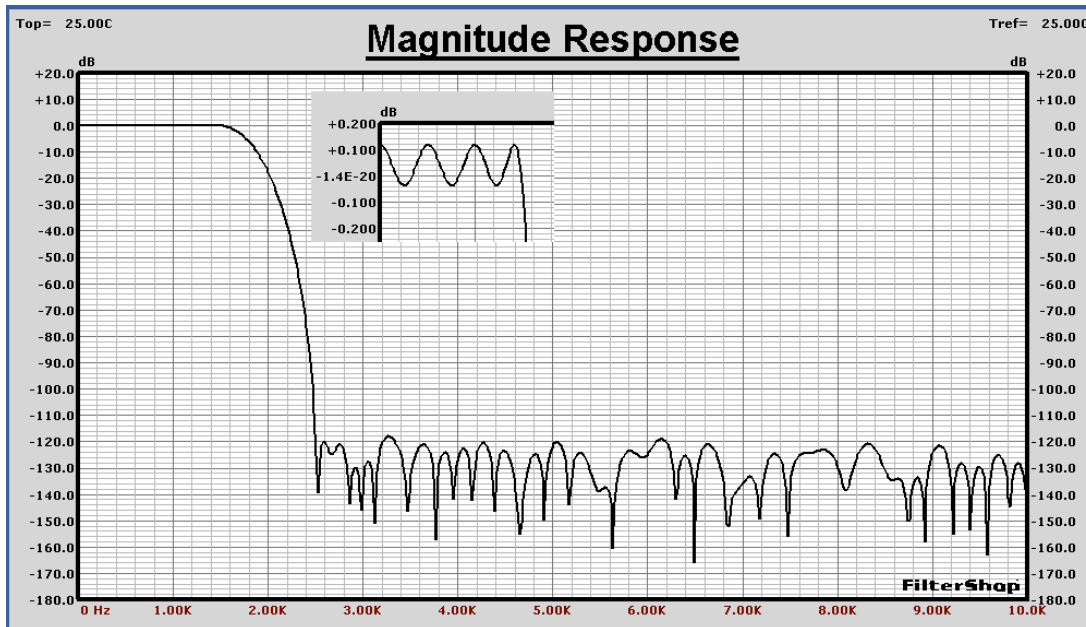




This packer dialog shown here displays the reduced 22 bit coefficients, and the response is shown below in Black.

The worst case attenuation occurs at 3.3kHz with a value of 106dB. This is about 2dB higher than the ideal goal of 108dB. The final term count is 371, down from 451.





We can now scale down the response of the Target TFB containing the FIR using $A_0 = -12\text{dB}$, and the final response is shown above. The passband ripple was unchanged. However note that the average passband level increased about 0.05dB. This is not uncommon when packing coefficients. The nearly 120dB stopband attenuation has been achieved with a 22 bit word size.

The final exported coefficient list is shown on the following page.

Summary

The *Digital-FIR: Coefficient Packing* dialog provides an effective means to reduce the coefficient storage and processing of FIR filters. This can be very important for logic based FIR implementations. Both automatic optimization and manual editing features are provided. Simultaneous multi format display and editing of Real, Integer, Hex, and Binary formats is supported. Changes in the response can be immediately observed during the editing process.

This completes the digital FIR coefficient packing examples.

```

/* ===== */
/* FIR Filter Coefficient Data * */
/* ===== */
/* (C)opyright 1993-2000 by LinearX Systems Inc */
/* FilterShop(TM) Version=3.2.0.685 */
/* Date=Sep 6, 2000 Time=Wed 1:32 am */
/* Design= AppNote13_3.fsd */
/* ===== */
/* Fs(Hz)= 20.0000K */
/* Order= 99 */
/* Coef Line Format= C++ Language */
/* Coef Numeric Format= Hex Fixed Point */
/* Coef Scaling= 4.0000 */
/* Coef Word Size= 24 */
/* Coef Hex/Bin Format= Sign Magnitude */
/* ===== */
const int NumOfTaps = 100;
const unsigned int H[NumOfTaps] = {
0x800030, /* H( 0) */
0x8000C0, /* H( 1) */
0x8001E8, /* H( 2) */
0x800394, /* H( 3) */
0x800520, /* H( 4) */
0x8004E0, /* H( 5) */
0x800014, /* H( 6) */
0x000CC8, /* H( 7) */
0x0024D8, /* H( 8) */
0x004920, /* H( 9) */
0x00768C, /* H(10) */
0x00A49C, /* H(11) */
0x00C590, /* H(12) */
0x00C8F4, /* H(13) */
0x00A0AC, /* H(14) */
0x004758, /* H(15) */
0x8039C8, /* H(16) */
0x80C830, /* H(17) */
0x813B18, /* H(18) */
0x816540, /* H(19) */
0x8123C0, /* H(20) */
0x806E38, /* H(21) */
0x009D9C, /* H(22) */
0x01BB20, /* H(23) */
0x028C00, /* H(24) */
0x02B310, /* H(25) */
0x01F648, /* H(26) */
0x005D28, /* H(27) */
0x81C0A8, /* H(28) */
0x83C61C, /* H(29) */
0x84F770, /* H(30) */
0x29E680, /* H( 53) */
0x1388F0, /* H( 54) */
0x00939C, /* H( 55) */
0x8C50C0, /* H( 56) */
0x9213D8, /* H( 57) */
0x914CAC, /* H( 58) */
0x8BDF40, /* H( 59) */
0x845AF0, /* H( 60) */
0x02BD4C, /* H( 61) */
0x078DB0, /* H( 62) */
0x093A70, /* H( 63) */
0x07EFC8, /* H( 64) */
0x04A2F0, /* H( 65) */
0x00A830, /* H( 66) */
0x82BDE4, /* H( 67) */
0x84B59C, /* H( 68) */
0x84F770, /* H( 69) */
0x83C61C, /* H( 70) */
0x81C0A8, /* H( 71) */
0x005D28, /* H( 72) */
0x01F648, /* H( 73) */
0x02B310, /* H( 74) */
0x028C00, /* H( 75) */
0x01BB20, /* H( 76) */
0x009D9C, /* H( 77) */
0x806E38, /* H( 78) */
0x8123C0, /* H( 79) */
0x816540, /* H( 80) */
0x813B18, /* H( 81) */
0x80C830, /* H( 82) */
0x8039C8, /* H( 83) */
0x004758, /* H( 84) */
0x00A0AC, /* H( 85) */
0x00C8F4, /* H( 86) */
0x00C590, /* H( 87) */
0x00A49C, /* H( 88) */
0x00768C, /* H( 89) */
0x004920, /* H( 90) */
0x0024D8, /* H( 91) */
0x000CC8, /* H( 92) */
0x800014, /* H( 93) */
0x8004E0, /* H( 94) */
0x800520, /* H( 95) */
0x800394, /* H( 96) */
0x8001E8, /* H( 97) */
0x8000C0, /* H( 98) */
0x800030}; /* H( 99) */
/* ===== */

```

