AND8384/D

Using Block Floating-Point in the WOLA Filterbank Coprocessor

ON Semiconductor®

http://onsemi.com

APPLICATION NOTE

This application note is applicable to Toccata Plus™, BelaSigna® 200 and Orela® 4500 Series

INTRODUCTION AND SCOPE

The WOLA filterbank coprocessor is an integral part of Toccata Plus, BelaSigna 200 and Orela 4500 series products. This application note describes the block floating−point scheme used in the WOLA filterbank coprocessor to preserve numeric precision in time−frequency transform. It is critical to have a good understanding of the WOLA block floating−point mechanism to optimize and plan tradeoffs for the best numeric precision at all processing stages as well as to avoid saturation effects.

A brief summary of the use of the WOLA filterbank coprocessor is described in this application note. It is strongly recommended that the *Introduction To Audio Processing −− Using the WOLA Filterbank Coprocessor* application note be read in conjunction with this document. This aforementioned application note provides an overview of the features and typical usage of the IOP and the WOLA filterbank coprocessor. It also describes the basic considerations when selecting WOLA filterbank parameters and compromises for specific applications. Keep in mind that the use of the WOLA filterbank coprocessor looks very simple, but care should be taken when considering numeric precision.

WOLA PROGRAMMING BASICS

Before using the WOLA filterbank coprocessor, all the filterbank parameters (R, N, La, DF, even/odd stacking, mono/simple stereo/full stereo/digital mixed mode) have to be defined in the initialization code sections of a signal processing algorithm. Then, the WOLA configuration macro (“WOLA_CONFIGURE”) is run to load all appropriate parameters in the WOLA−related memory, including default windows. If necessary, custom windows can be defined and written in the window locations using appropriate macros (“WIN_CONFIGURE”, “UCODE_CONFIGURE”).

In the main portion of the signal processing algorithm, the WOLA filterbank coprocessor can be called for the desired function (analysis, gain application or synthesis) using the `WOLA_Start` macro, specifying the desired function as a parameter (0 for analysis, 1 for gain application and 2 for synthesis). Consequently, only a “start” macro execution is required for filterbank operations in the signal processing code. When the filterbank operation is completed, an interrupt is generated in the RCore processor and the results are available in the dedicated memory locations.

In normal operation, the WOLA analysis process always gets the input data from the input FIFO. Similarly, the WOLA synthesis puts resulting samples in the output FIFO. In the frequency domain, the values resulting from the analysis are always available in a dedicated memory location. A gain application operation will multiply those values by the gains vector in place. The gain vector data has to be placed in a dedicated memory location. More practical detail and code examples can be found in the documentation from ON Semiconductor’s evaluation and development kit (EDK) and other relevant white papers and application notes.

USE OF THE BLOCK FLOATING−POINT

The WOLA filterbank coprocessor is implemented in the hardware using 18−bit block floating−point numerical precision. This scheme allows for good precision in FFT operations, while still avoiding the considerable computational complexity of floating−point processing. As with any system, it is important to be aware of the way the numerical precision implementation works. In a WOLA filterbank coprocessor−based system, this understanding will help to interpret scale and skillfully manipulate digital data between a WOLA analysis and WOLA synthesis operation. Furthermore, when applying digital gain greater than 1, it is also important to understand how the block floating point works, in order to prevent saturation. The following section explores these two important topics in technical detail.

Relative Scale of Subband Data

It is important for an algorithm designer to understand how the WOLA filterbank coprocessor scales the subband data (frequency domain data between analysis and synthesis). As a first example, the algorithm may need to refer computations back to subband data from the prior processing blocks (the need for history). The WOLA filterbank coprocessor actually scales data optimally in each processing block and often changes the relative scaling between blocks to preserve numeric precision. Consequently, when an algorithm stores and uses subband data from the past, correct calculations will depend on compensating for these scaling differences. As a second example, some algorithms depend on absolute levels in subband data. For instance, some applications may relate the computations on subband data to sound pressure levels. These calculations will also require compensation for the scaling that the WOLA filterbank coprocessor applies to each processed block.

In the RCore, data is represented as 16−bit fractional values (between –1 and $1-2^{-15}$). Hence, the input data passed to the WOLA filterbank coprocessor has this format as well. However, during a standard WOLA filterbank coprocessor operation, the internal representation uses 18 bits: 2 bits are added to the 16−bit fractional representation to prevent saturation in case of data growth. This enables a fractional representation with an integer part between –4 and +3.

In fact, the analysis calculation is split into several steps or passes (windowing and time−folding, successive FFT butterflies, etc). During each calculation pass, the internal precision is 18 bits in the computation unit and the resulting data is stored in 18−bit WOLA temporary memory. Nevertheless, at the beginning of every new pass, the 18−bit results from the previous pass are re−scaled to 16 bits. Only a fractional number is represented in the computation unit at the beginning of each pass. As a consequence, 1 or 2 right shifts may be necessary to map each 18−bit representation to a 16−bit representation. At every pass, a register called `N_FFT` holds the number of right shifts required (0, 1 or 2). The aggregate number of right shifts is accumulated in the block exponent register, located at address D_BLOCK_EXP_DATA in X memory. This register is updated at the end of each pass. After analysis, it contains the total of all the shifts performed. The number of shifts performed during the last pass is still available in the N_FFT register.

The frequency−domain data are completely represented by a two 18−bit signed mantissa per frequency band (one for the real and one for the imaginary components) plus a global exponent across all bands (D_BLOCK_EXP_DATA). The allowed exponent values can go from 0 to 7 (maximum), while its numeric format is 4 bit signed. Two methods can be used to get the analysis results in the 16−bit representation in a desired reference scale:

1. Subband data is available to the RCore from 16−bit memory−mapped locations starting at D_WOLA_RESULT_BASE (X:0x1300). A read will return the most relevant contiguous 16−bit range from the 18−bit memory space. The data can be normalized by a left shift by D_BLOCK_EXP_DATA on each datum (using the “SHIFT A, INV” instruction). Note that it is important to ensure that the algorithm compensates in the case where there would normally be an overflow (advanced algorithm design can often carry the scaling exponent as a variable for future calculation for better overall precision and efficiency). If the other two bits of precision (which are very important for intermediate calculations in the WOLA passes) are not strictly required for algorithmic precision, this method is strongly recommended for computational simplicity.
2. Alternately, the full 18−bit mantissa values can be accessed using three mirrored 16−bit registers: the first one containing bits 0 to 15 (at address X:0x1000 indexed from the LSB side), the second one containing bits 1 to 16 (at address X:0x1100) and the third one containing bits 2 to 17 (at address X:0x1200). This is shown in . The 16−bit data can be scaled to a known reference by reading the exponent in the D_BLOCK_EXP_DATA register and shifting appropriately. The content of the D_BLOCK_EXP_DATA register will then represent the number of left shifts to apply to the analysis results at memory location X:0x1000 (when N_FFT = 0), at location X:0x1100 (when N_FFT = 1) or at location X:0x1200 (when N_FFT = 2). This method is only recommended when extreme numeric precision is required.

When restoring the desired scale of the analysis results in the 24−bit accumulator AE|AH, one must be careful if the resulting data range exceeds 16 bits. Storage in 16−bit memory can pose problems without normalizing to AH first. One suggested strategy is to use a constant normalization (number of right shift) before memory storage or any load operation on the data. In that case, the global number of left shifts applied at each frame to the data in D_WOLA_RESULT_BASE is BLOCK_EXP − NORM (assuming the first method above). Here, BLOCK_EXP is the content of the register D_BLOCK_EXP_DATA (changing at every frame), while NORM is the number of right shifts used for normalization, keeping data as much as possible in the 16−bit fractional range on a dynamic signal (it is the same across all blocks). `NORM` should be selected according to the global system settings (notably amplification level at input stage). It is often determined using experiments to trade off between saturation and precision.

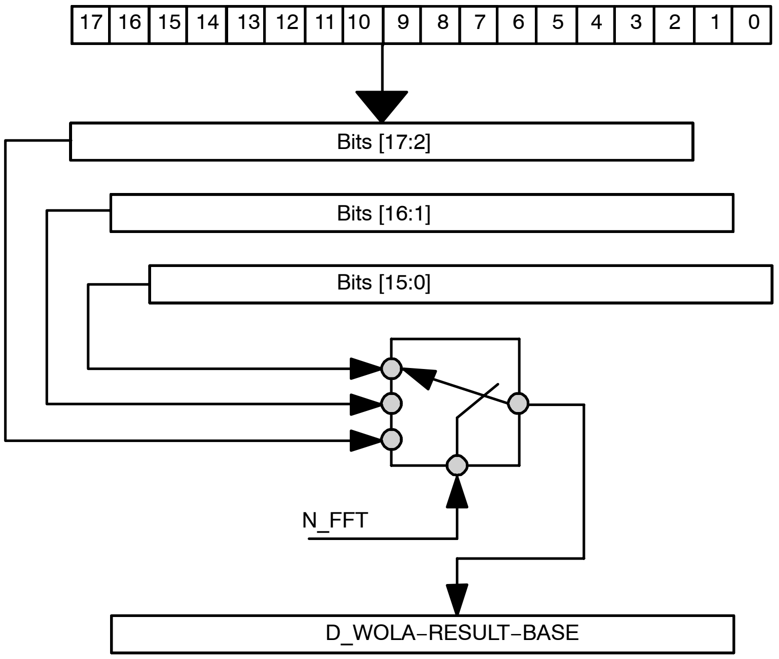


Figure 1. 18−bit Frequency Results

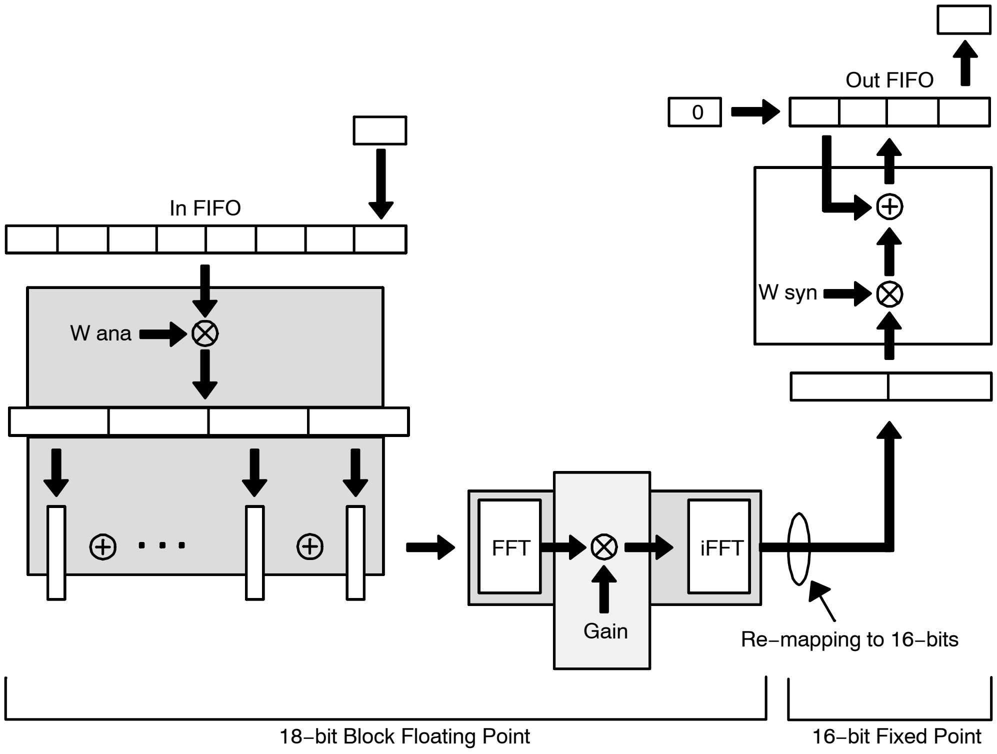


Figure 2. WOLA Operations (Analysis, Gain Application and Synthesis)

Preventing Saturation During Gain Application and Synthesis

The block floating−point scheme is used during gain application (the gain application is one WOLA pass) and synthesis (many passes). The D_BLOCK_EXP_DATA register continues to accumulate the number of right shifts in each pass. The block floating−point mechanism stops just after the IFFT in the synthesis step when the data is forced back to 16 bits. The subsequent operations (synthesis windowing and overlap−add) all use 16−bit fixed−point data. This is shown in , which illustrates the full WOLA process from analysis to synthesis. After synthesis, the time−domain samples available in the output FIFO always have a constant scale, so there is no longer a concern about the relative representation of data.

To remap the 18−bit representation to 16 bits, the mantissa is left−shifted by the final content of D_BLOCK_EXP_DATA to restore the appropriate scale. The FFT/IFFT normalization operation (division by the number of actual FFT points) is taken into account as right shifts, involving FFT_NORM bits. Because the FFTs used in the WOLA implementation are complex, `FFT_NORM` is equal to log2(N/2) bits in mono mode and log2(N) bits in stereo modes. A mathematical representation of the shift factor to restore 16−bit scale after WOLA filterbank coprocessor processing is represented by:

$$RSH = FFT\_NORM - BLOCK\_EXP \text{ [bits]}$$

where: BLOCK_EXP is the content of D_BLOCK_EXP_DATA after the IFFT.
RSH is the number of right shifts applied to restore correct linear fractional scale.

Understanding this mechanism is important in order to manage saturation in the WOLA filterbank coprocessor, especially when applying gains higher than unity in the frequency−domain. The following sections provide some important guidelines, explanations and illustrations.

Saturation in the Frequency−Domain

Saturation of the 18−bit Register

The width of the block floating−point registers in the WOLA filterbank coprocessor is 18 bits. At the end of one pass, if a value has grown higher than this range, it is saturated (where N_FFT is set to 2). This normally never occurs, unless the user selects unreasonably high gains during the WOLA gain application.

Using the Gain Exponent

The gains to be applied to each band (stored at D_WOLA_GAIN_BASE) have 16−bit fractional format and consequently represent values between –1 and 0.99997. The gain exponent register (located at address D_GAIN_EXP_DATA in X memory) is designed for the case where the algorithm requires higher than unity gains and operates by globally left−shifting the frequency−domain data. The number of left shifts to apply is specified in this register, whose format is 4 bits unsigned (the maximum amplification factor is then 2^15−1 = 32767 = 90 dB). This is shown in Figure 3.

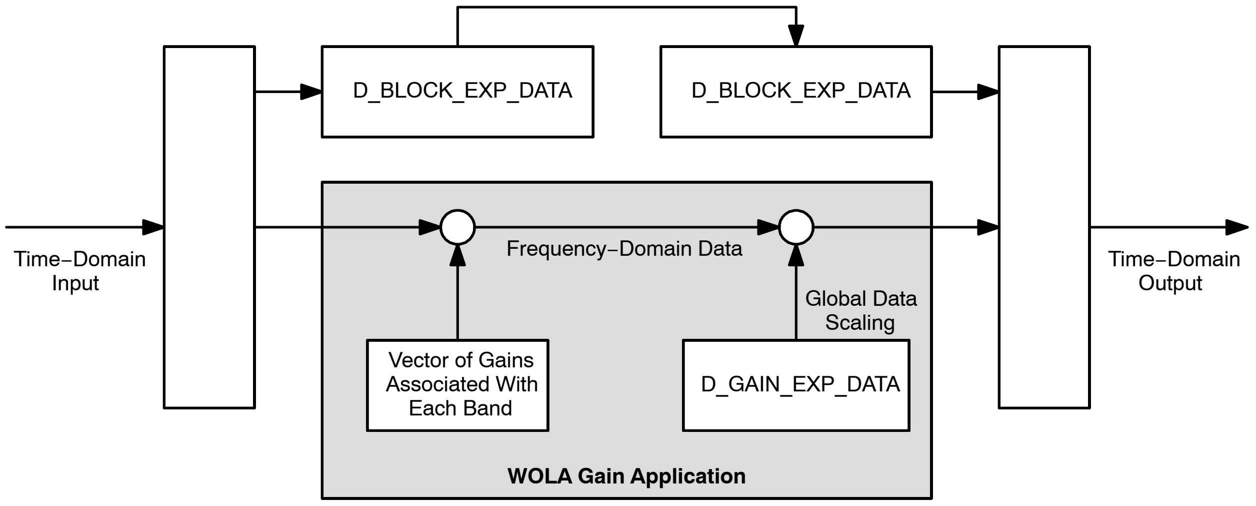


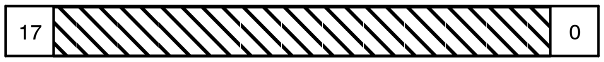
Figure 3. WOLA Filterbank Coprocessor Gain Application

In the last step of the WOLA gain application function, the shift by D_GAIN_EXP_DATA is applied just after multiplication of the frequency−domain data by the gain vector. The shifted data is stored back into 18−bit memory space. Care should be taken in gain application to avoid saturation of this 18−bit memory space. Saturation can occur with D_GAIN_EXP_DATA values greater than 2 and proper care should be taken with input signals and higher gain values. For example, in the case where the analysis results reach 16−bit full−scale range in one particular band, the value in D_GAIN_EXP_DATA should not be higher than 2. Otherwise, assuming a unity gain vector, the datum in this band would be saturated after gain application.

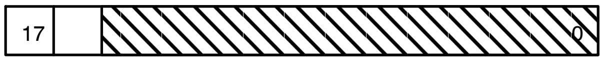
Multiple Gain Applications

To apply very high gains with the D_GAIN_EXP_DATA several gain application stages can run in succession to avoid saturation in the frequency−domain. The block floating−point mechanism re−scales the data to the lower 16−bits of the 18−bit register before every new gain application stage. Below is an example that illustrates how the multiple gain application works.

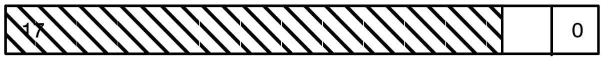
To start, perform a first gain application, with D_GAIN_EXP_DATA = 2, and then a second gain application pass, with D_GAIN_EXP_DATA = 1. As an initial state, it is assumed that data following analysis occupy bits 0 to 16 in the 18 bit register (at least in one band) and N_FFT = 1.



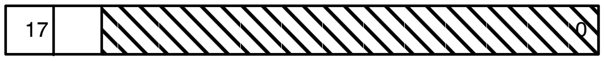
1. **First Gain Application**: As shown above, the analysis results are read from bits 1 to 16 (because N_FFT = 1) and multiplied by the WOLA band gains (which are always lower than 1). As there is no increase in range, N_FFT is set to 0 and the result is put back in the 18−bit register in bits 0 to 15. The gain application is considered as a processing pass. At the end of the pass, the D_BLOCK_EXP_DATA register is updated accordingly (in this case no increase is necessary). N_FFT is always reset at the beginning of the pass, just after data read.



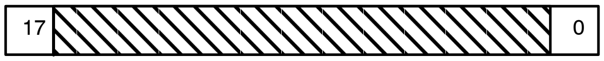
2. **Application of D_GAIN_EXP_DATA = 2 at the End of the First Gain Application**: The data, read from bits 0 to 15 (because N_FFT = 0) is shifted by two bits on the left. N_FFT is set to 2 and D_BLOCK_EXP_DATA is incremented by 2.



3. **Second Gain Application**: The analysis results are read from bits 2 to 17, as N_FFT is 2. They are multiplied by the WOLA band gains (in this case, the programmer may wish to set the gain vector to unity, or the vector will be applied a second time), and the result is put back in the lower part of the 18−bit register from bit 0 to 15, as there is no increase in range. D_BLOCK_EXP_DATA is not increased, and N_FFT is 0.



4. **Application of D_GAIN_EXP_DATA = 1 at the End of the Second Gain Application**: The result is shifted by one bit on the left. N_FFT is set to 1, and D_BLOCK_EXP_DATA is further incremented by 1.



Note: In the example, the application of the D_GAIN_EXP_DATA is shown as a separate WOLA filterbank coprocessor pass for educational purposes only. In the processor, it is included in the same pass as the multiplication with the gain vector, as the final step. However, data is treated in the same way as illustrated.

Applying Gains by Manipulating the Block Exponent

Considering Equation 1 on page 6, it is clear that a global system gain can be realized by manually increasing the block exponent value (BLOCK_EXP). Manually increasing the block exponent using the RCore would result in fewer right shifts (RSH) when the WOLA filterbank coprocessor re−maps the 18−bit range to 16 bits (this modification to BLOCK_EXP can be done just before synthesis). Consequently, a gain would be realized through a purposeful miscalibration of the block floating−point process. Note that this method represents a riskier alternative to using the gain exponent register D_GAIN_EXP_DATA for applying a global gain. As such, great care should be taken if doing this. Remember that the value in the D_BLOCK_EXP_DATA register must not become greater than 7 because the register is 4−bit signed 2's−complement. If this register increments past 7, overflow occurs (not saturation) and a huge attenuation would be applied to the signal. This method should not be used when the number of bands (N) is high because BLOCK_EXP naturally gets very close to 7 in these cases.

Gain Redistribution

In some cases the best way to apply a high global gain without saturation is to split it between the gain exponent D_GAIN_EXP_DATA and the block exponent D_BLOCK_EXP_DATA registers. When a high gain exponent value is required (for example, in a dynamic range compression operation), the value in D_BLOCK_EXP_DATA can be considered as it appears after analysis. A high gain exponent value could saturate the data before synthesis, while a small value for BLOCK_EXP would result in the previously saturated data being scaled down to a low overall level. This is obviously a sub−optimal gain distribution. As a consequence, gain redistribution is sometimes a good practice. In such cases, the algorithm would try and trade off values in D_GAIN_EXP_DATA and D_BLOCK_EXP_DATA. Usually, D_BLOCK_EXP_DATA rarely exceeds 3 after analysis (this depends on the WOLA filterbank coprocessor configuration, it can get up to 7 in extreme cases). As a consequence, after analysis, the BLOCK_EXP could be systematically set to 3, while compensating in the D_GAIN_EXP_DATA register accordingly. For example, if D_BLOCK_EXP_DATA = 1 after analysis in a given frame, and if the specified overall system digital gain is 7, then the algorithm would set BLOCK_EXP = 3 (that is 1+2), and D_GAIN_EXP_DATA = 5 (that is 7−2). As mentioned earlier, avoid increasing the block exponent beyond 7 during synthesis. In addition, recall that the gain exponent should also be monitored, so that it does not exceed 2 when the frequency−domain data reaches full−scale, as this would cause saturation.

The guidelines above show how to avoid or limit saturation in the frequency domain. However, they do not consider saturation in the time domain. In fact, they might cause it. The following section explores considerations of saturation in the time domain.

Saturation in the Time−Domain

Full−Scale Time−Domain Input Signals

Whenever the input time−domain data is 16−bit full−scale (assuming unity gains in every bands, which corresponds to a passthrough application), then the output signal in the output FIFO will have same scale as the input (that is 16−bit full range as well). Consequently, saturation is expected if the gain exponent is used to apply gains greater than unity. For example, setting the value in D_GAIN_EXP_DATA to 2 would amplify data by a factor of 4 in the frequency−domain. There would be no saturation risks at gain application, thanks to the 18−bit memory space. However, the time−domain output data would then need 18 bits to be represented, which is not possible anyway, because the output FIFO is only 16−bits wide. Consequently, the signal would saturate during the re−mapping of 18− to 16−bit representations (just after IFFT). For most applications, this should be avoided.

Use of the Gain Exponent

The use of the gain exponent should be reserved for the amplification of low−level signals. Consider an input time−domain signal represented with 13 bits in the current input block. A value of 3 in D_GAIN_EXP_DATA could be used in the frequency domain without risk of saturating the time−domain signal in the output FIFO (13 + 3 = 16). Of course, care should also be taken to avoid saturation in the frequency domain. Whenever the frequency−domain data is full−range in one band, then the value in `D_GAIN_EXP_DATA` should not be higher than 2.

WDRC Example

Consider the example of a wide dynamic range compressor (WDRC). In such an algorithm, a non−linear compression characteristic is applied in order to amplify the low−level signal components, while leaving the high−level components unchanged (or slightly reduced in the case of limiting). The characteristic is usually stored in a table whose entries correspond to the signal energy level. In the table, the associated amplification factors are listed, from high gains (low signal energy) to low gains (high signal energy). Using the WOLA filterbank coprocessor, such gains would be applied during gain application, being taken into account in the gain vector. Because of the fractional numerical representation in the RCore, those gains would range from 1 to very small values approaching 0. Since low−level signals require amplification beyond the range of the gain vector (0 to 1), a global gain (higher than 1) can be designed together with the compression characteristics to provide adequate gain in aggregate. In the WOLA filterbank coprocessor, this global gain is easily applied using the gain exponent register. The gain exponent can become very high, depending on the compression characteristic.

Such a procedure should be performed with care to prevent saturation when high gains are applied and for keeping best precision when the gain vector values are low. The amplification characteristics and the global gain value should be designed to avoid saturation while maintaining dynamic range and meeting the requirements of the signal processing scheme. Effectively, band gains are maximized in case of low input signals, and the gain exponent would be increased as expected. The WDRC design should make sure that data amplification does not produce saturation of the 18−bit band registers and that the output signal in time (output FIFO) does not require more than 16 bits for a complete representation. Conversely, the gain vector values will be very low in case of high−level time−domain input signal, while the gain exponent will be high, resulting in a total gain close to 1. In this situation precision is compromised (remember the gain vector is applied before the gain exponent). The loss in SNR should not be critical because the level of the signal is high, masking the noise. However, better precision can be maintained by making the process adaptive. Typically, when the input signal is high, the low values in the gain vector could be increased to values between 0.5 and 1. To compensate accordingly, the gain exponent can be reduced in the opposite amount on a frame−by−frame basis. This could significantly reduce the risk of loss in numeric precision.

The Time−Domain Saturation Criterion

Consider again Equation 1 and re−mapping the 18−bit representation into 16−bits, as performed when terminating the block floating−point mechanism (see Figure 2). The number of right shifts applied for restoring the original scale is:

$$RSH = FFT\_NORM - BLOCK\_EXP \qquad \text{(eq. 1)}$$

In the (FFT + IFFT) process, the data range always grows by a factor corresponding to the number of FFT points (that is by `FFT_NORM` bits). Consequently, if a signal before the FFT needs U bits for its representation (U <= 16), then the signal after IFFT would need U + `FFT_NORM` bits. This assumes no data modification between FFT and IFFT (no manipulation of the gain or block exponent registers). The WOLA block floating−point mechanism runs during the FFT/IFFT operations, shifting the data to the right at every pass that ends with the data in the upper 2 bits of the 18−bit space, adding `FFT_NORM` – (16–U) right shifts in aggregate to the `D_BLOCK_EXP_DATA` register. The subtraction by 16−U expresses the fact that no shift is required until the scale of the data exceeds 16 bits. Consequently, the same signal only needs U + FFT_NORM – (FFT_NORM – (16 – U)) = 16 bits. This shows that the block floating−point scheme prevents saturation during the FFT/IFFT process, while continuously keeping the best precision, which is the very purpose for which it was designed.

However, the correct scale must be restored by Equation 1. After the IFFT, the WOLA filterbank coprocessor performs right shifts (RSH) to rescale the data. After this, the data will be represented with U' = 16 – RSH bits. Notice that only 16 bits (fixed−point) will be available from here on for representing the re−scaled data, so U' should not be more than 16 bits. We can derive the criterion to avoid saturation as:

$$RSH \geq 0$$

or similarly: $$\boxed{BLOCK\_EXP \leq FFT\_NORM} \quad \text{(eq. 2)}$$

This criterion can be checked at the end of the WOLA synthesis operation, verifying whether saturation did occur during the process and if so to what extent (how many bits). The WOLA filterbank coprocessor configuration determines FFT_NORM, while BLOCK_EXP can be read in the D_BLOCK_EXP_DATA register after synthesis. This information can be used to determine RSH.

Implications of Criterion (Equation 2)

The criterion is simple, but it has implications for the application of gains greater than unity. Looking at Equation 2, it becomes obvious that:

1. Manually increasing the block exponent can cause saturation.
2. Applying a greater−than−zero value to the D_GAIN_EXP_DATA register generally causes D_BLOCK_EXP_DATA increase accordingly, unless there are very small values in the gain vector. This could make BLOCK_EXP grow to the point of saturation where criterion (Equation 2) is no longer satisfied. A good implementation will fundamentally depend on a careful calibration and understanding of the time−domain input signal range.

Criterion (Equation 2) should be checked for any system with higher−than−unity digital gain. Both gain redistribution (in which the D_BLOCK_EXP_DATA is manually increased), and methods mentioned previously (running two subsequent gain application processes in order to apply gain exponents higher than 2 without saturating the 18−bit registers in the frequency domain) can be used to try and adhere to the criterion as necessary.

Saturation Caused by the Time−Folding Operation

Figure 2 shows that the windowing and time−folding operations (performed as one pass before the FFT) can cause the block floating−point mechanism to increase D_BLOCK_EXP_DATA automatically. The amount of potential increase depends on the time−domain input data range, on the analysis window shape and on the particular WOLA filterbank coprocessor configuration (specifically the L/N ratio). This increase never requires more than 2 left shifts with a usual analysis window. However, be aware that if an increase of the block exponent happens at this time, (for instance, FOLDING_SHIFTS = 1 or 2 bits before starting the FFT processes), then criterion (Equation 2) will not be satisfied unless the gain vector is less than unity, thereby reducing the data scale. If unity gains are used, then saturation will occur at re−mapping time. In this case, the range of data before FFT would be 16 bits (otherwise no increase of the block exponent would be required during time−folding) and the block exponent will increase by FFT_NORM bits during FFT/IFFT. The final value of the block exponent after synthesis will be BLOCK_EXP = FFT_NORM + FOLDING_SHIFTS, RSH will be negative, and this will cause saturation.

Saturation in the Overlap−Add Operation

Figure 2 also shows that saturation can occur in the 16−bit fixed−point overlap−add operation. Data range may need to be more than 16−bits wide to prevent overflow because of the addition operations in the overlap−add. Increasing the ratio OS/DF has direct influences on data scales in the overlap−add process. Typical synthesis window shapes make saturation rare, however high OS/DF ratios can cause saturation in theory.

CONCLUSION

To summarize the main points:

1. The gain exponent was designed to apply gains to low−level signals. It should never be used if the time−domain input signal uses the full 16−bit range (assuming unity gains in the bands). Saturation can occur after the synthesis operation in the time domain if high gains are applied to full−scale signals. The block−floating point mechanism does not solve for this, because the output data range is only 16−bit (as is the input).
2. D_GAIN_EXP_DATA values higher than 2 (gain > 4) should never be used when the frequency−domain data occupies the full 16−bit range. This would cause saturation to the frequency−domain data (just after gain application) because the memory space is only 18 bits wide and is not sufficient to hold larger values.
3. After the WOLA synthesis operation is completed, the user can check criterion (2). If it is not satisfied, then saturation has occurred. Conversely, if this criterion is satisfied, there is no guarantee that saturation has not occurred. The condition is not totally sufficient, as it is possible to have undetected saturation from the overlap−add operation.

Block Floating−Point in Stereo Modes

The WOLA filterbank coprocessor can be configured to process in monaural or in stereo mode (simple stereo or full stereo). In stereo modes both channels are processed together as the real and the imaginary part of the same complex FFT / IFFT. Consequently, all the block floating−point and gain registers are common to both channels. The gain exponent register is also the same for both channels. This has the following implications:

1. The block exponent register will behave according to the values in the channel with the highest level. Scaling factors (right shifts) will be applied similarly to both channels. Consequently, precision in the lower channel will be affected − a reduction in performance may occur in a channel that has a level significantly lower than in the other one.
2. Any use of the gain exponent (D_GAIN_EXP_DATA) on one channel will force the same global shift on the other channel; therefore, gain application in stereo system requires special care. For example, a compression algorithm running on channel 0 would update the gain exponent according to the energy of its input signal similar to the WDRC sample code). If a simple passthrough (or noise reduction) algorithm were running on channel 1, the same gain exponent would also be applied here causing unwanted modulation. The output of channel 1 would be modulated by the energy of the signal in channel 0. In this case, variations of the gain exponent should be compensated on channel 1 by applying shifts to modify the gain vector in the opposite way. This is a typical situation when processing a transmit and a receive channel simultaneously, as in a telecom application. In most cases, the input signal on channel 0 (typically speech from the near end) can be very different from the signal on channel 1 (typically silence, or noise from the far end).

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910
Japan Customer Focus Center
Phone: 81−3−5773−3850

ON Semiconductor Website: **www.onsemi.com**

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative