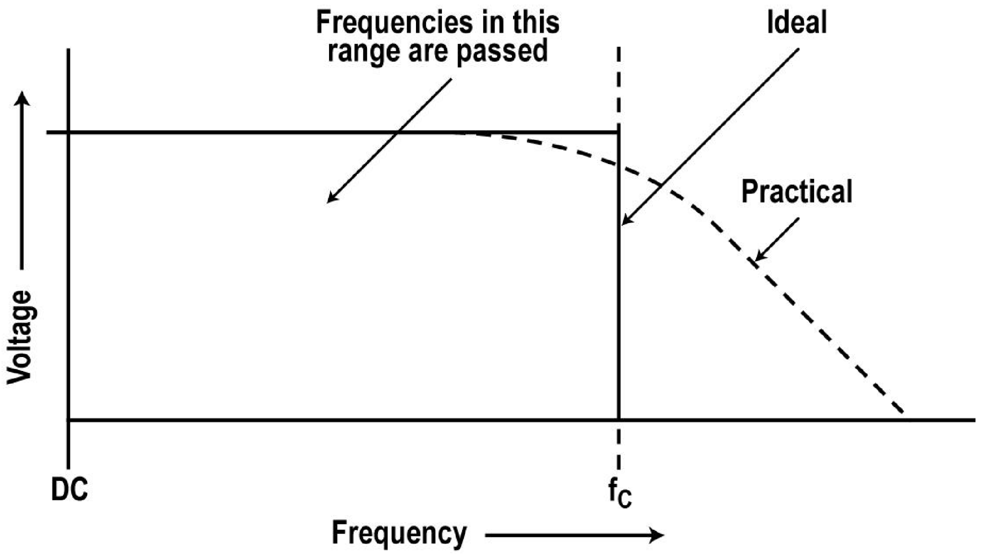
Digital Filters

Digital Filters

The basic function of any filter is to perform some desired frequency discrimination or selection process. Filters are frequency sensitive circuits that pass some signal frequencies with ease but greatly attenuate or eliminate signals of other frequencies. Selectivity refers to how well a filter can separate or distinguish between signals on very close frequencies. Selectivity is determined by how steep the filter response curve is.

A digital filter is a circuit that produces traditional analog filter results by digital methods. Digital filtering is by far the most common application of DSP. Digital filters deliver superior performance over analog filters. The selectivity is better (steeper skirts) and the phase response can be adjusted to deliver a desired result.

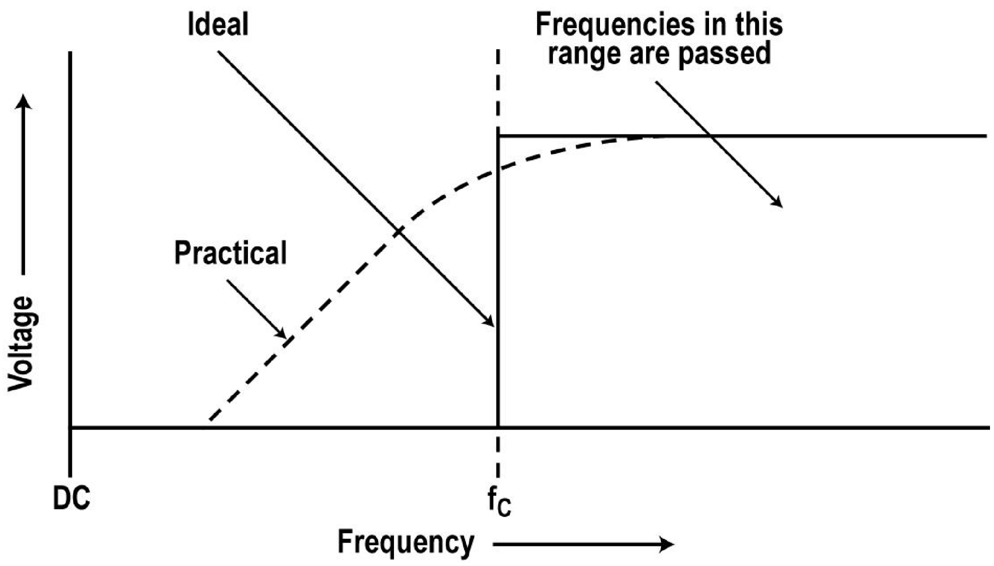
Analog Filters: Low Pass



There are four basic filter types: low pass, high pass, band pass, and band reject. The ideal response curves of the low and high pass filters are shown.

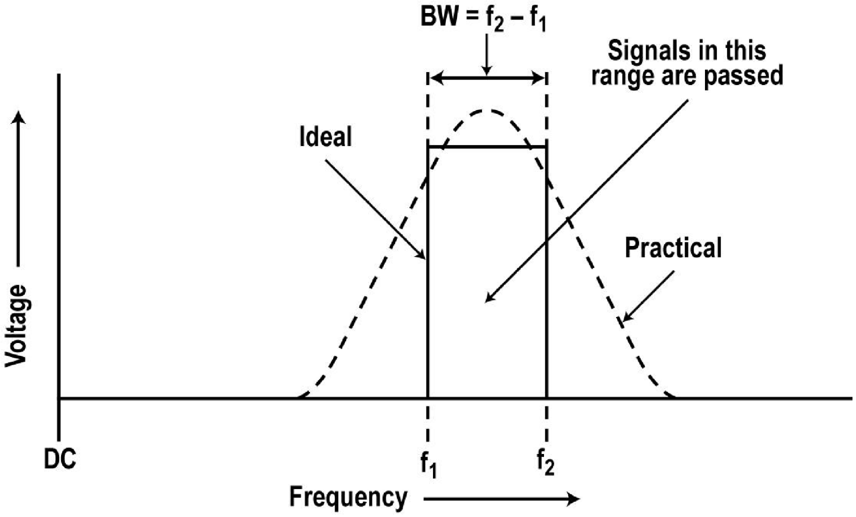
The low pass filter passes signals below a specific cut-off frequency (f_c) but rejects signals above the cut-off.

Analog Filters: High Pass



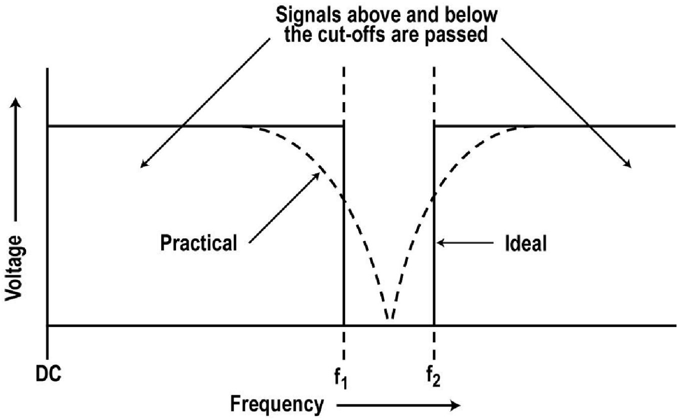
The high pass filter passes signals above the cut-off and rejects those below the cut-off. The dashed curves show typical response curves from practical analog filters.

Band Pass Filter



The ideal and practical response curves for the band pass filter is given in the figure above. The band pass filter passes a narrow band of frequencies between two cut-off frequencies, f_1 the lower cut off and f_2 the upper cut-off. The bandwidth (BW) of the filter is $f_2 - f_1$.

Band Reject Filters

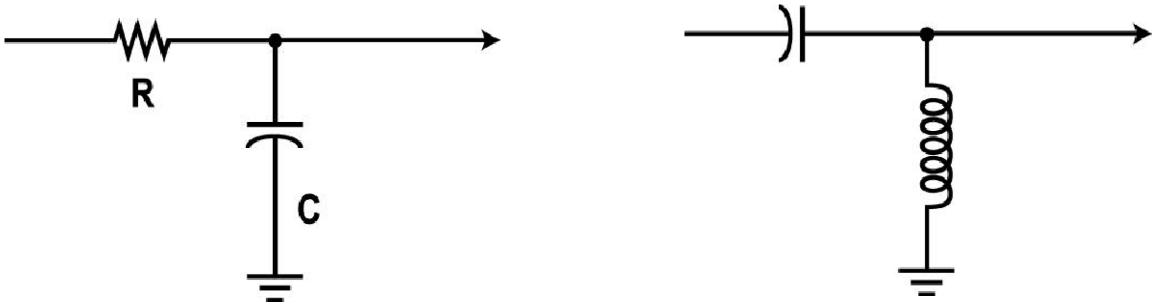


The band reject filter, also called a notch filter, passes all frequencies above and below the two cut-off frequencies but eliminates the signals with frequencies between the two cut-off frequencies. The bandwidth (BW) is $f_2 – f_1$.

Many times the filter is designed to filter out one single frequency in which case it is referred to as a notch filter.

The dashed curves show the response provided by traditional analog circuits.

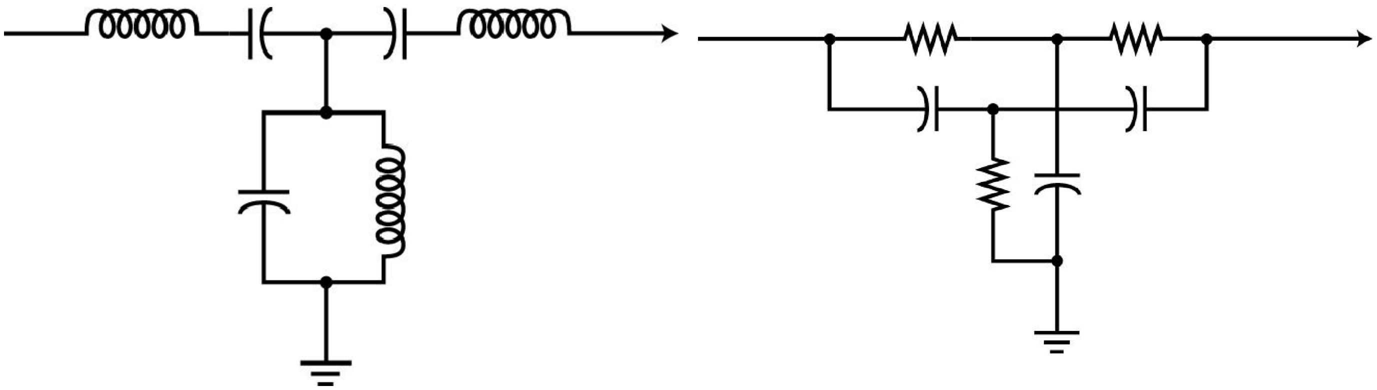
Low and High Analog Filter Implementation



Analog filters are made with resistor-capacitor (RC) or inductor-capacitor (LC) sections. This figure on the left shows a low pass filter made with an RC section.

The figure on the right shows a high pass filter made with an LC section. These sections are cascaded to achieve greater selectivity but with increasing attenuation.

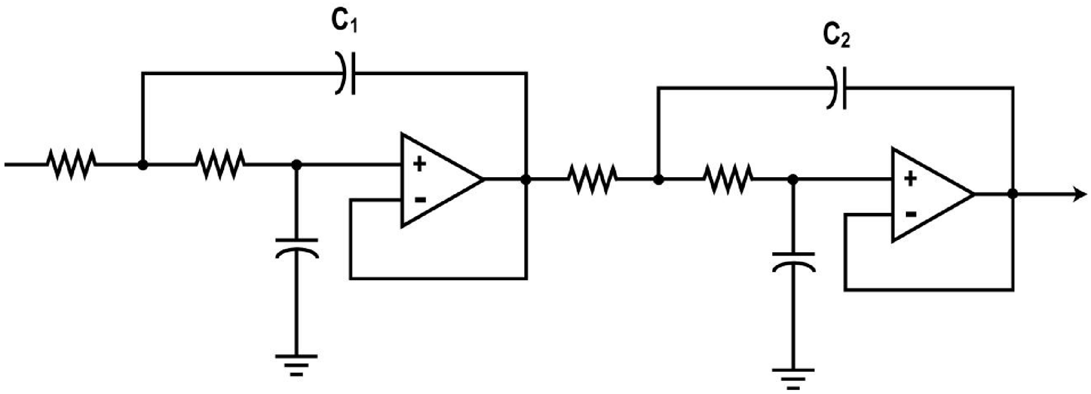
Analog Band Pass and Band Reject Filters



The left figure shows a band pass filter made with LC sections.

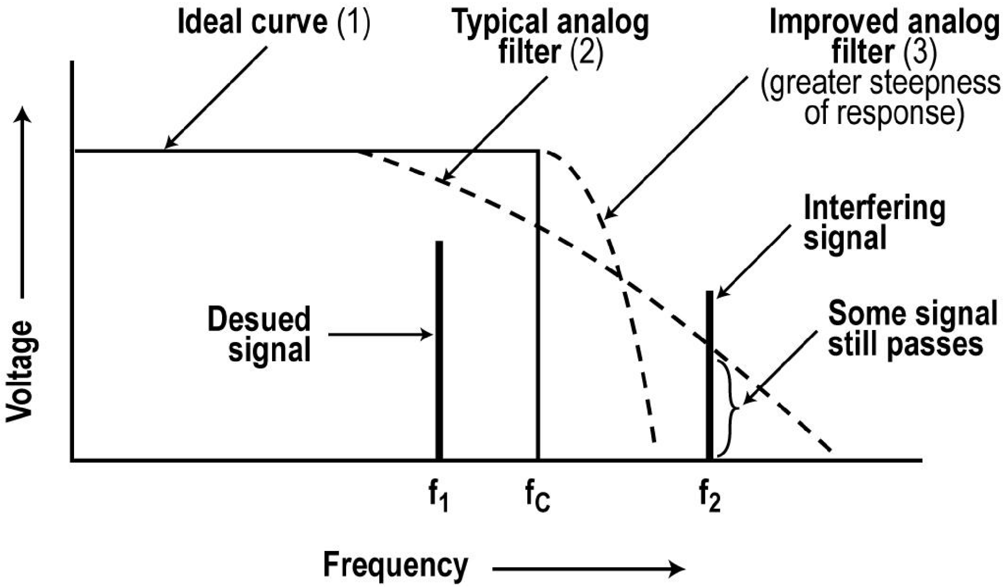
The right figure shows a notch filter made with RC sections.

Active Filters



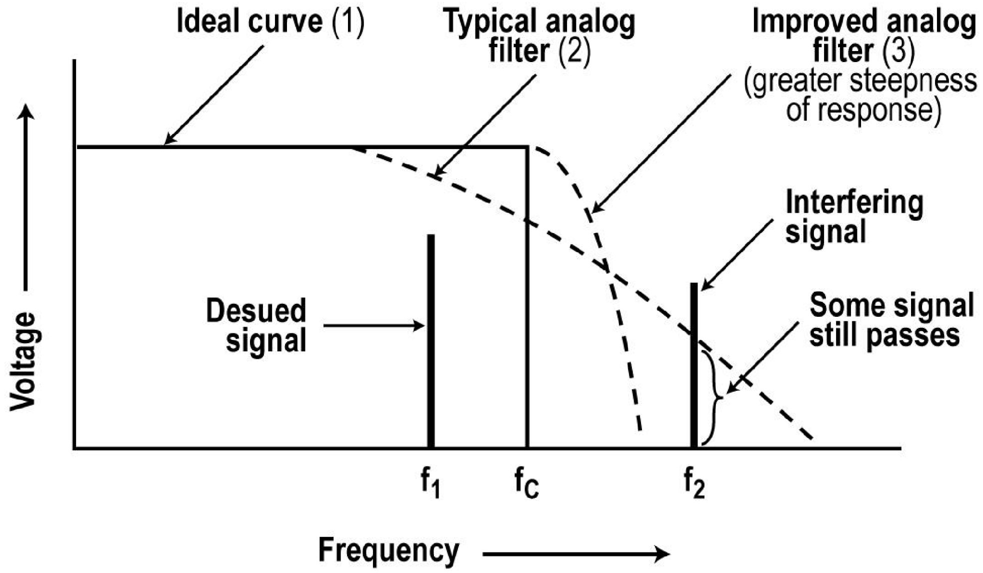
An active filter is one made with operational amplifiers (op amps) and RC sections as shown in the example above. By using multiple RC sections and feedback (via C_1 and C_2), highly selective filters can be made. Selectivity is improved by cascading these filters. A benefit is that the circuit can have gain thanks to the op amps.

Analog Filter Limitations



While analog filters are widely used in many applications, they are not perfect. Performance approaching the ideal curves cannot be achieved even with multiple cascaded filters. This limits the ability of the filter to discriminate against signals very close in frequency to one another.

Limitations of Analog Filter



Analog filters require very precise values of R, L, and C. These are difficult to obtain and very expensive. Furthermore, these values can change with temperature or vary over time due to aging.

DSP filters eliminate these problems completely.

Phase Response

Another limitation of analog filters is the phase response. Capacitors and inductors have a reactance value that varies with frequency. This is the reason that these components are useful in making filters. If the input frequency changes, the phase of the output also varies. While this may not hurt some analog signals, this phase change does cause signal distortion of pulse and digital signals.

Remember according to the Fourier theory, all rectangular pulses are made up of a fundamental sine wave and multiple harmonic sine waves. When all these sine waves are added together, they form the rectangular wave. A filter circuit will produce different phase shifts for the different harmonic frequencies. Shifting the harmonics causes the output signal to be greatly distorted from the original. This phase shift is called group delay.

Digital Filter Benefits

Digital filters overcome these analog filter limitations. A digital filter can be made to provide whatever degree of selectivity is desired. Performance closely duplicating the ideal response curves can be achieved. The exactly vertical response curve is sometimes referred to as a “brick wall” filter. Interfering signals hit the brick wall and are eliminated.

In addition, the digital filters can be made with linear phase response. This means the harmonic content of a signal is preserved since phase shift will be a linear variation with frequency.

Finally, digital filters are fixed by the software and not by R, L, or C values that can change.

Types of Digital Filters

Three common types of digital filters are the averaging filter, the finite impulse response (FIR), and the infinite impulse response (IIR).

The averaging filter is used primarily for minimizing noise on a signal.

The FIR filter is also called a non-recursive filter. Its output is calculated only with current and previous inputs.

The IIR filter is a recursive filter. Recursive means that the processing algorithm repeatedly applies a mathematical process to the data as well as the outcomes of the process. In other words, the IIR filter uses feedback. The IIR filter output is calculated with the current and previous inputs but also previous output states as well.

The averaging and FIR filters are the simplest of the three but the IIR filter can be made to produce superior selectivity with less processing time and effort.

Basic Digital Filter Algorithm

The processing algorithm for most digital filters is given mathematically as:

$$y(n) = \sum (a_i)\, x[n - 1]$$

The term y(n) is the computed output for a given a given input sample value n.

∑ means a summation or addition of all values of the coefficients a multiplied by each input sample x[n – 1].

The term a_i refers to a constant value or coefficient value. The i refers to the number of the coefficient meaning there are different values of coefficient to be used in the various multiplications.

The expression x[n – 1] does not mean x multiplied by n – 1. The [n – 1] designation simply means the next sample in sequence. It would be followed by sample [n – 2] then [n – 3] and so on.

Explaining the Algorithm

Expanding the mathematical expression given earlier:

$y(n) = \sum (a_i)\ x[n - 1]$ where i may vary from 0 to some selected maximum value k.

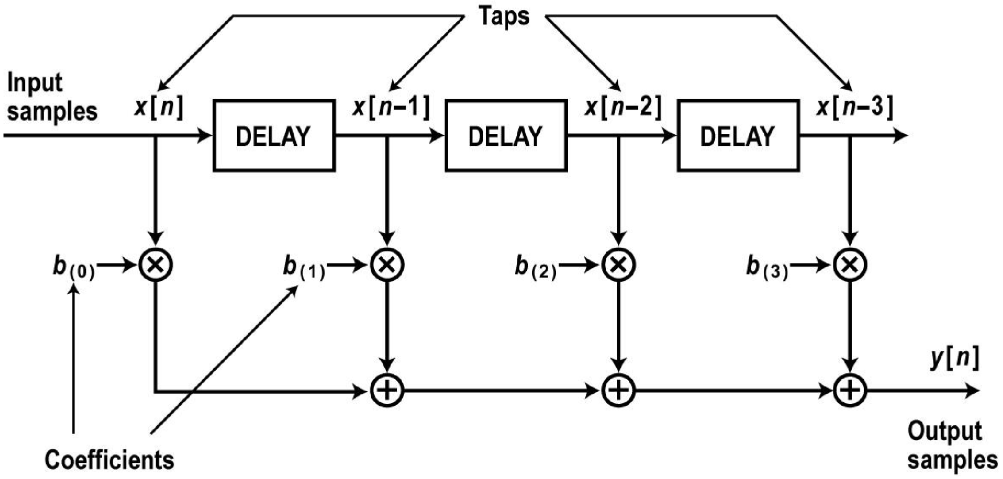
The number of coefficients used in multiplying sequential samples of the input is called the number of taps in the filter.

This gives us:

$y(n) = a_0\ x[n] + a_1\ x[n - 1] + a_2\ x\ [n - 2] + a_3\ x[n - 3] + \ldots\ldots..$

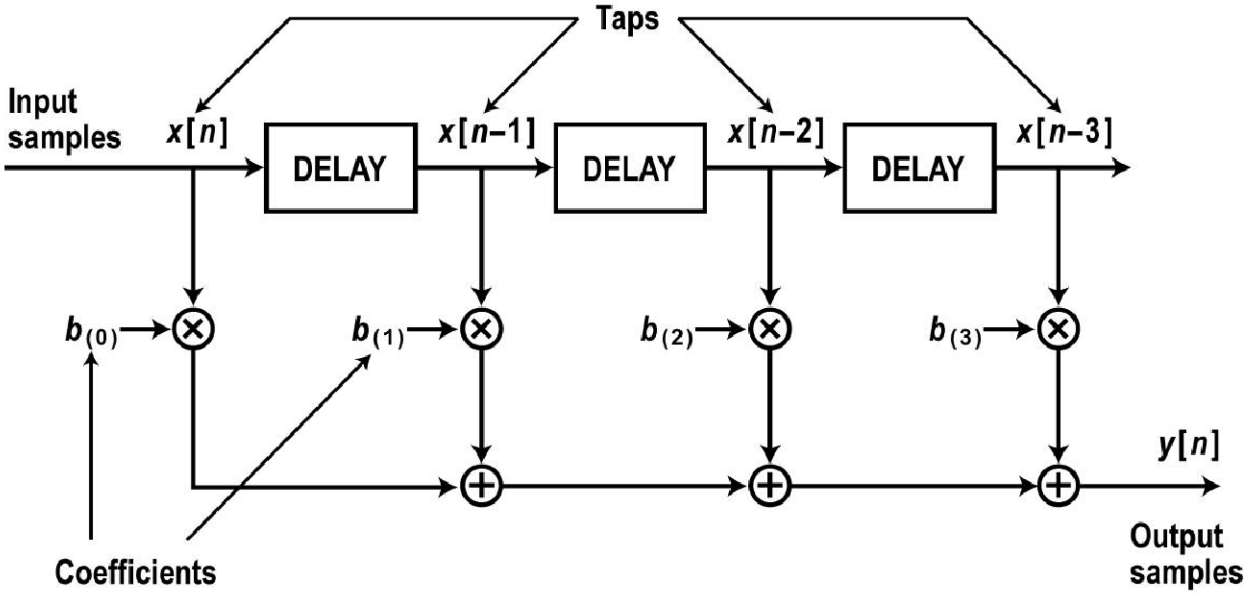
What this expression is telling you is that an output value y(n) is calculated by first multiplying the first binary sample value [n] by a constant coefficient a_0 . This is added to the next sample x[n – 1] multiplied by another coefficient a_1. The process repeats for as many values of i are selected.

Implementing the Algorithm



As you can see, the basic filtering algorithm requires three basic operations: multiplication, addition, and sample sequencing. All of these operations can be performed in any DSP chip. Individual add and multiply instructions are available on any processor. Special multiply and add or multiply and accumulate (MAC) instructions can also combine the multiply-add process to speed up the operations.

Illustrating the Algorithm



One way to represent the algorithm is in a block diagram form. This visual diagram better illustrates the flow and sequence of operations.

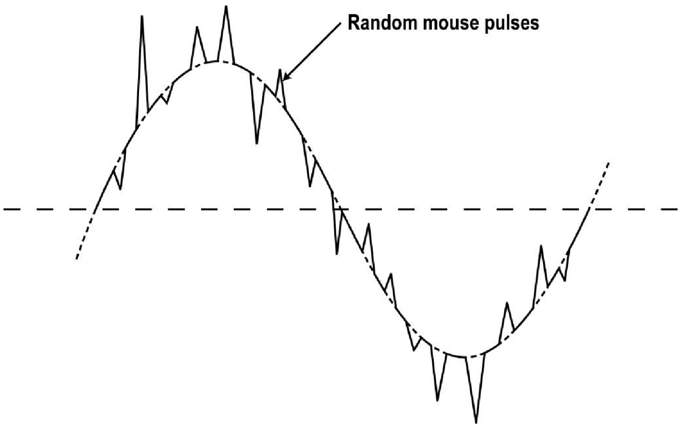
Block Diagram Symbols

There are three symbols used in the block diagram. The circle with the X (⊗) is a multiplier. It is used to multiply the binary value of one sample by its corresponding coefficient in binary form.

The circle with the + (⊕) is an adder that adds two binary numbers.

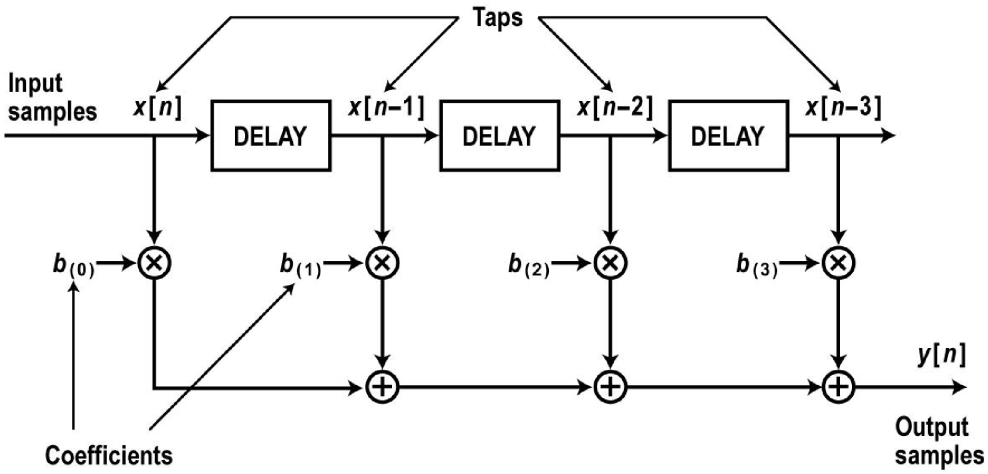
The rectangle containing the word DELAY represents a time shift or delay. Some diagrams use the designation Z^{-1} to indicate the delay. It could be implemented with an actual delay circuit called a delay line whose output occurs some time after its input. However, here this symbol is just a way of representing a time difference between samples. The delay could also be a shift register for binary values. But typically what happens is that the delays are just ways of showing the sequential samples of the original analog signal as they are received from the ADC or read out of a RAM. Each sample is called a tap.

An Averaging Filter



An averaging filter is used to take an average of a waveform usually for the purpose of minimizing high frequency noise on a signal. The figure shows a sine wave signal with high frequency noise superimposed on it. An example is power line or auto ignition noise on a radio signal. Since noise is random, the occurrence and amplitude of the noise spikes will vary from cycle to cycle of the sine wave. If we than take an average of the signal over several samples and cycles, the noise will average itself out.

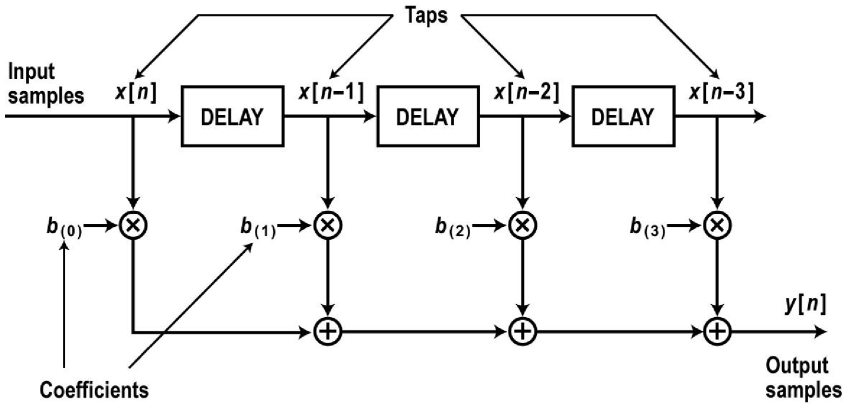
Sample Averaging



The algorithm illustrated in the figure will perform the average with the appropriate coefficients. To obtain an average of any number of values, you add all the values together then divide by the number of values. This average is also called the mean. Assume the following the four numbers 7, 10, 12, and 8:

Average = (7 + 10 + 12 + 8)/4 = 9.25

DSP Sample Averaging



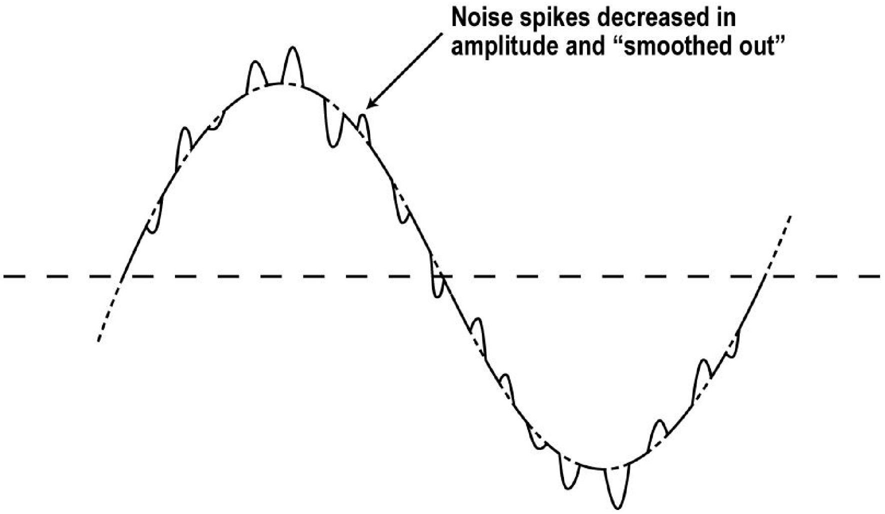
In DSP, the divide operation can be used but it is usually slower than a multiply operation, so we modify the averaging process as follows:

Average = 0.25 (7 + 10 + 12 + 8) = 1.75 + 2.5 + 3 + 2 = 9.25

Remember that multiplying by a fraction is the same as dividing by its reciprocal: 0.25 = ¼. In a computer, multiplication is much faster.

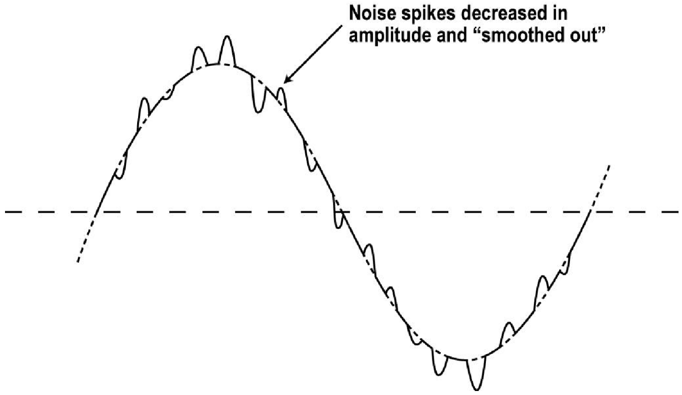
In the figure, each of the “b” coefficients is set to 0.25.

Averaging Filter



By repeatedly sampling the noisy sine wave and passing the samples through the filter, the filter will produce output samples. When these samples are passed through a digital-to-analog converter (DAC), they will produce the same sine wave but the noise spikes will be greatly minimized.

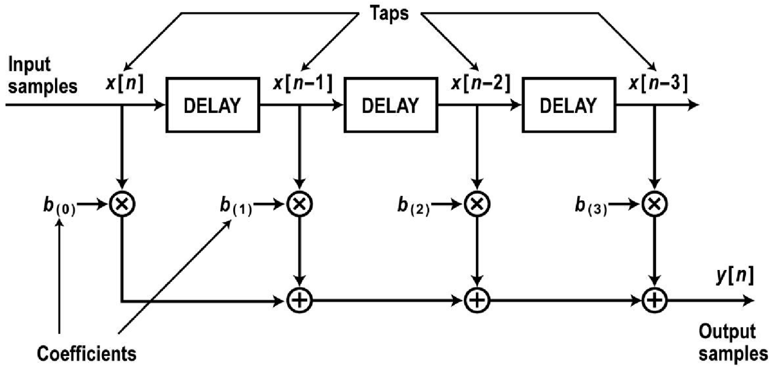
Averaging Filter Operation



Because the noise is random, some pulses will be positive while some are negative, some will be large and some will be small. The averaging process makes them smaller. The result is a cleaner sine wave with less noise that can be more easily recovered and processed again.

Since the noise is mainly high frequency signals, the averaging filter is really a low pass filter. It passes the lower frequency sine wave but attenuates the higher frequencies.

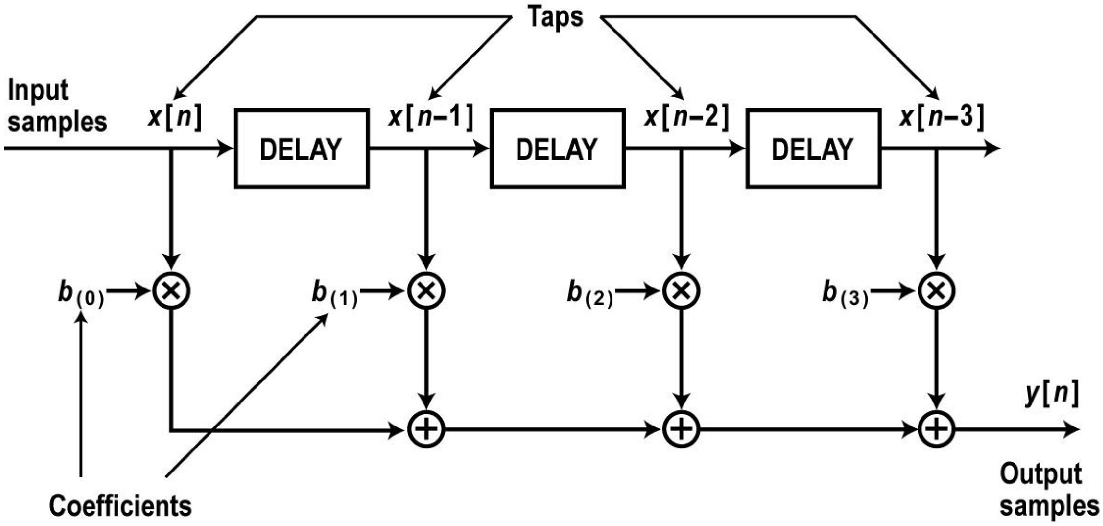
Filter Operation



Keep in mind that this diagram is simply a block diagram illustrating the steps of the mathematical algorithm programmed into the DSP. Here how it works.

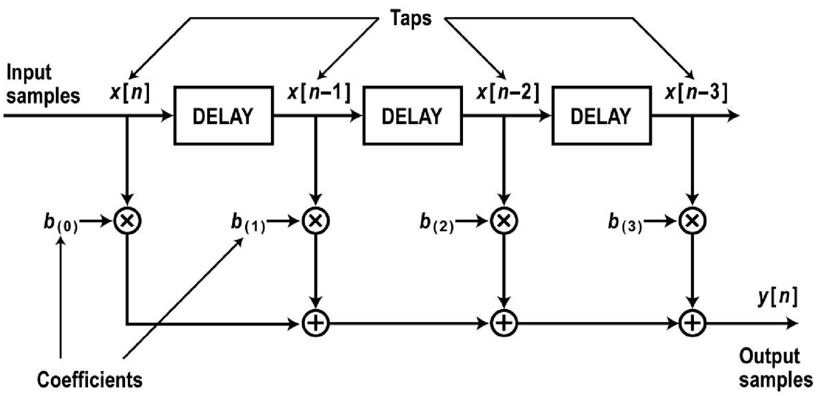
The samples of the analog signal feed into the input. These may be coming directly from an ADC or they may be stored in sequential memory locations in a RAM. Let’s assume the latter condition. The speed of processing is determined by how fast the ADC sampling rate is or how fast the samples are read out of RAM.

Filter Operation Example



Note that the first sample [n] is multiplied by the coefficient b_0. The sum goes to an adder. The adder output is simply the product of a_i and [n]. There is no output from any of the other multipliers or delay segments since the next samples have not been read in yet.

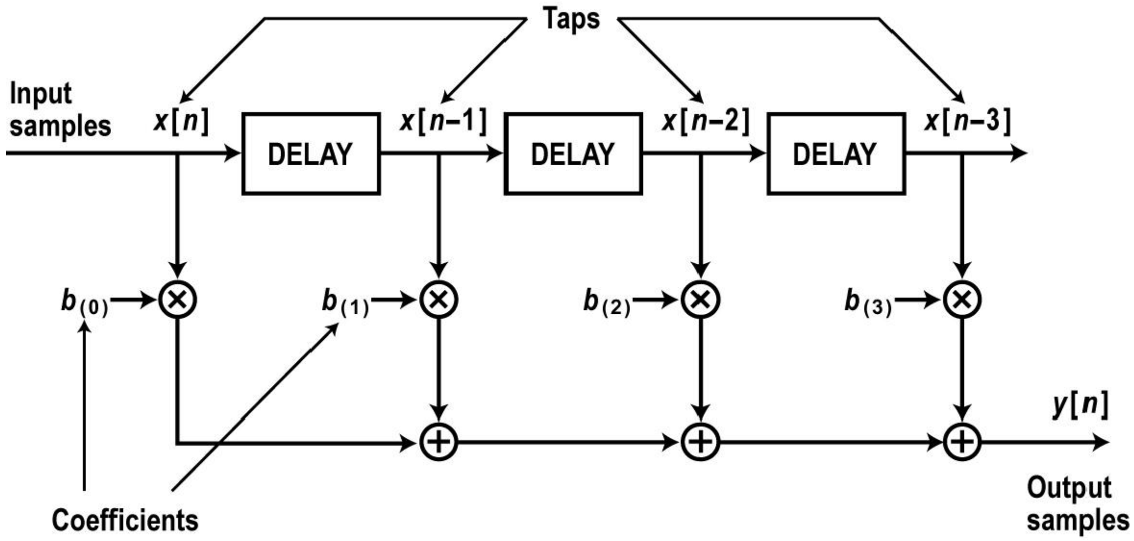
Filter Operation: Sample 2



Next, the second sample [n – 1] is fed into the input. At this time, the first sample [n] has moved over to the output of the first delay unit. Now sample [n] is multiplied by b_1. The current input sample [n – 1] is multiplied by b_0. The two values are now summed to produce the output.

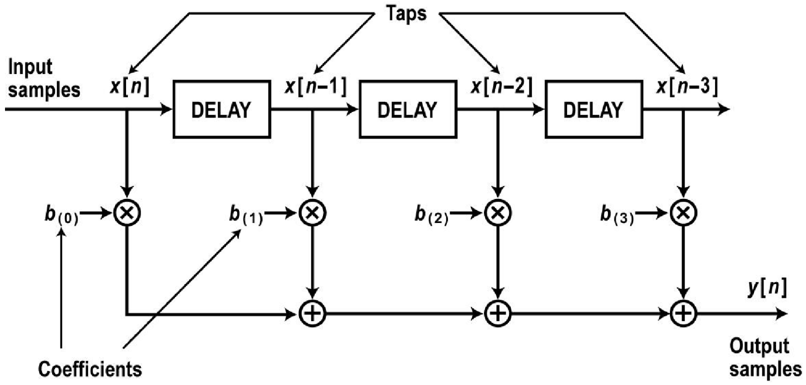
Note that for each new input sample, the multiply and add process occurs on the samples that have been fed in and a single new output value is created. These outputs are then fed to either a DAC for direct output or to a RAM for later output.

Filter Operation: Sample 3



The third input sample is then fed in where it is multiplied by b_0. The second input sample shifts over one position to the right and the first input sample is further shifted to the right where it is multiplied by b_2. Again, the samples are multiplied by the coefficients and a new output sample is produced.

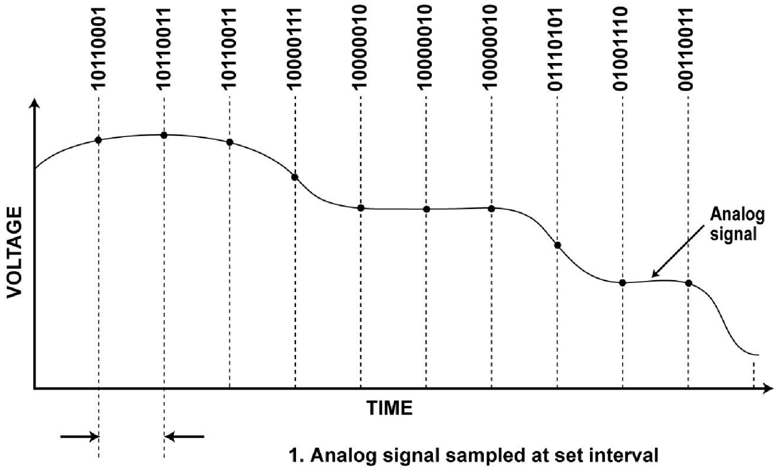
Filter Operation: New Input Samples



As you can see, for each new input sample, the other samples more over to the right and are multiplied by another coefficient. Just keep in mind that what is happening here is that the samples are not really moving over. Instead, they are still stored in sequence in the RAM so the DSP simply accesses them in the correct order defined by the algorithm.

All of the circuits are being used when the fourth sample is shifted in. When the fifth sample is read in, it and the others are effectively moved over. The first sample drops out of the picture at this point. The process continues with remaining samples.

Summarizing Filter Operation



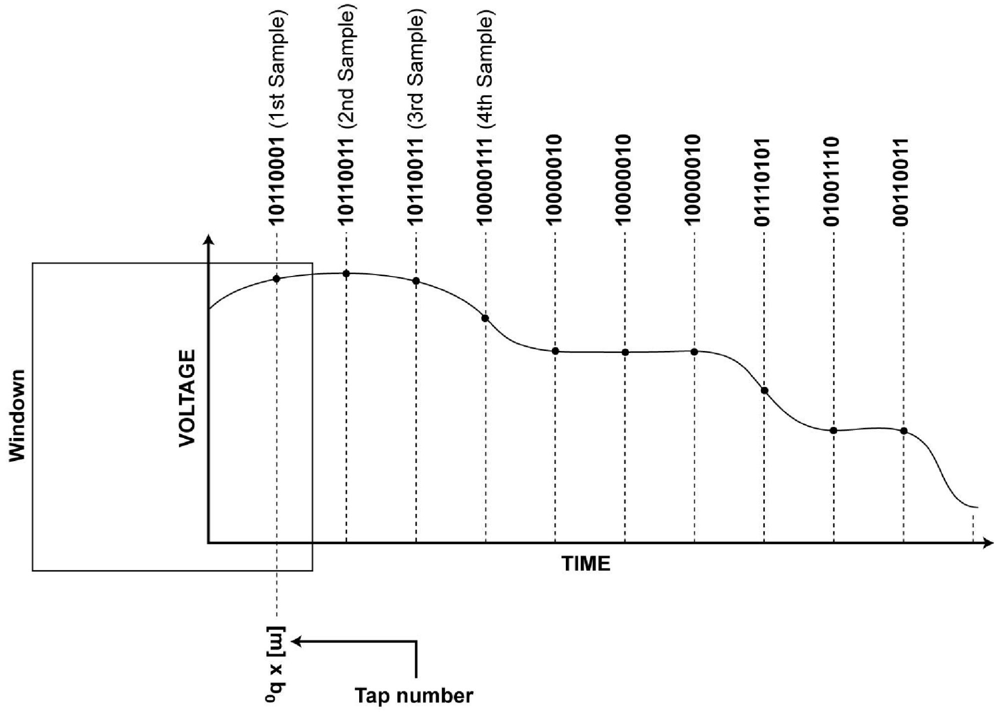
Sampling time or interval (t) also Delay interval

1. Analog signal sampled at set interval points and measured.
2. Measured signal is converted into a proportional binary number.

One way to view what is happening is to think in terms of the DSP algorithm as only processing four samples at any given time. It is as if a sliding window is passing over the samples with only the DSP using a group of four at a time. The complete operation is illustrated by the sequence shown in the following slides.

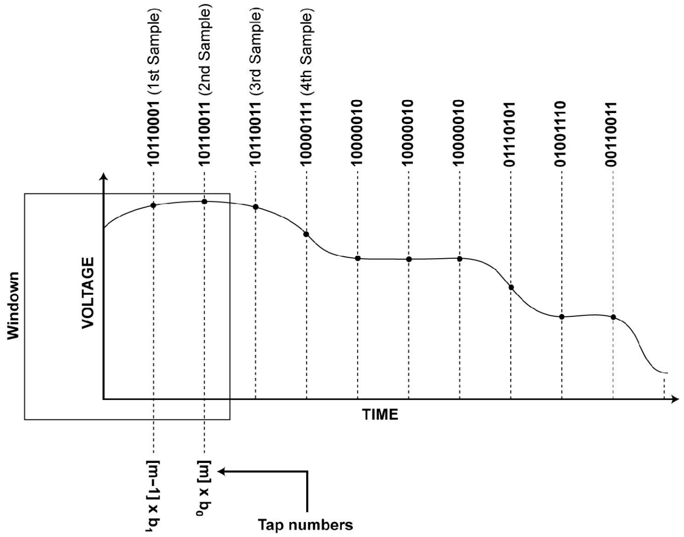
The figure above shows a sampled analog signal to be averaged.

First Sample



This figure shows the first sample is read in and multiplied by coefficient b_0. Note the window.

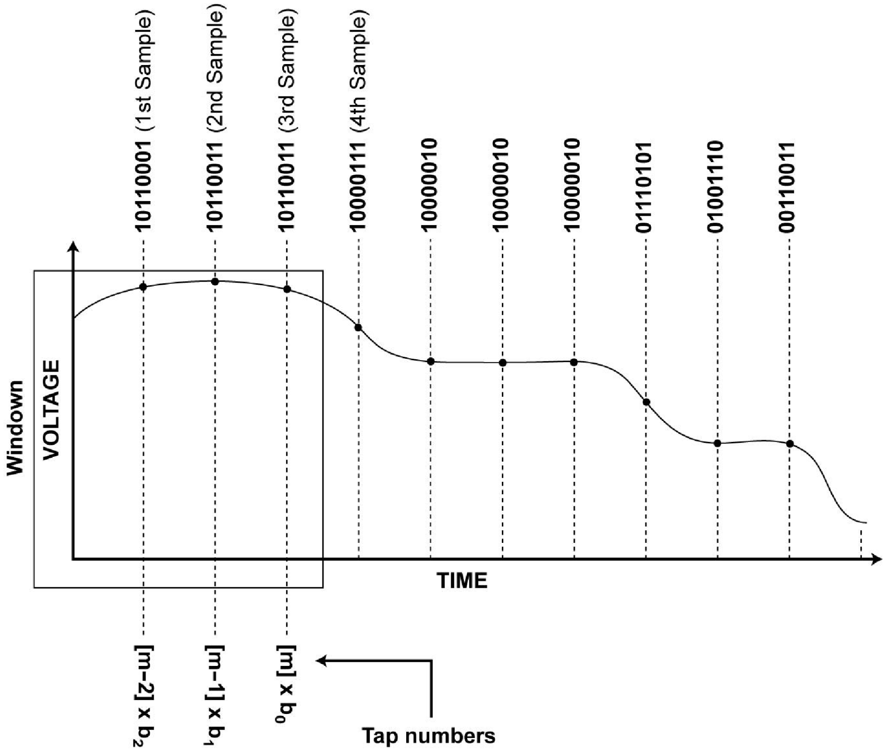
Second Sample



This figure shows the second sample being read in and multiplied by b_0. The first sample is now multiplied by b_1.

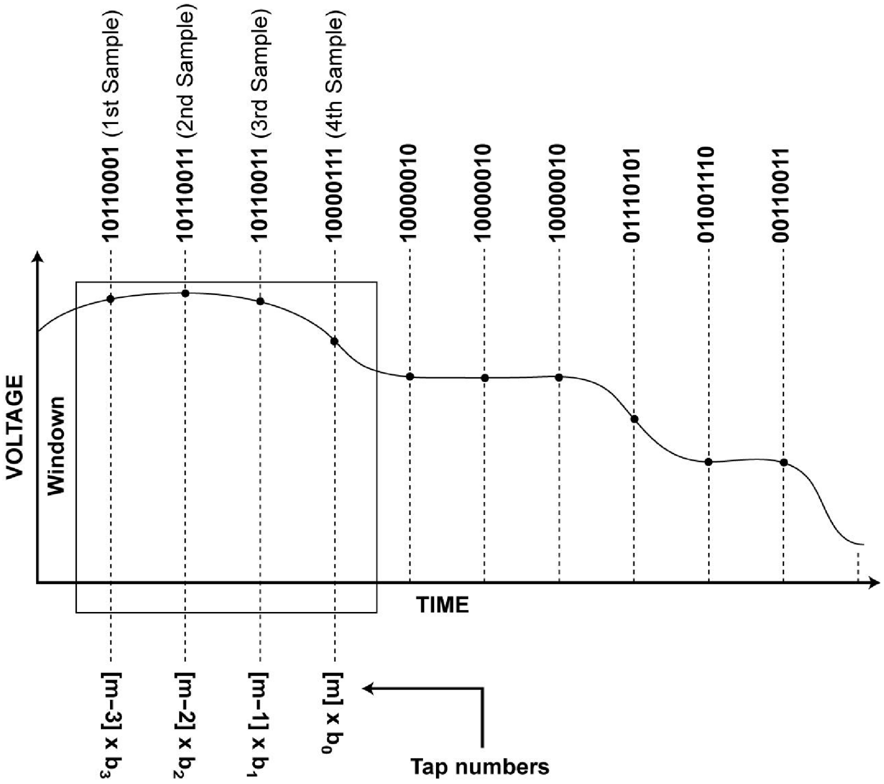
Be sure that you understand that the numbers [n], [n – 1], [n -2] etc. are the designations for the taps in the earlier figure and not the sample numbers. Note that the samples occur from left to right.

Third Input Sample



This figure shows the third input sample which is multiplied by b_0, the second sample is now multiplied by b_1 and the first sample is multiplied by b_0.

Final Step



Here we see the final step of the window.

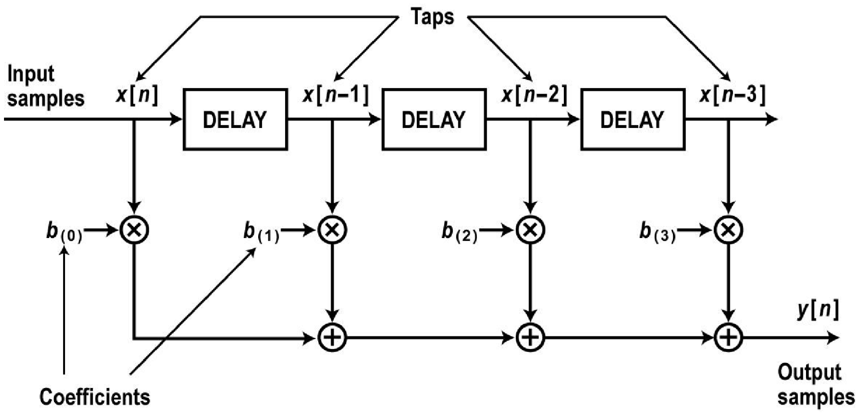
Filter Operation Conclusion

The new values created by the DSP represent the filtered output. As indicated earlier, they may go directly to a DAC for conversion back to analog or they may go into a RAM where they are used in another calculation or read out later.

Most processing is real time meaning that all the calculations in the algorithm are carried out very fast and occur between the sample of delay times.

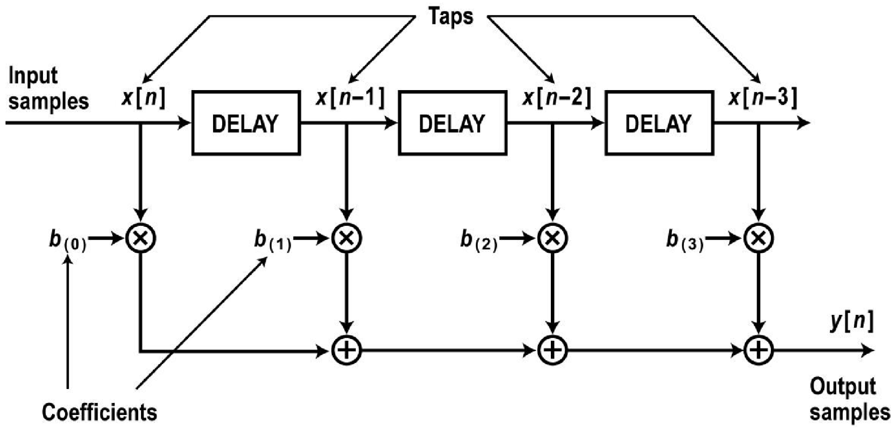
When the filter output samples are converted to analog, the resulting signal is similar to that which would be obtained with an analog filter with similar characteristics.

FIR Filter



An FIR filter has the same algorithm and basic process as shown in the earlier example. What makes the FIR filter different from the averaging filter is the coefficients. By selecting the correct set of coefficients, the filter can be made to perform any of the four basic filter types. The coefficients may be positive, negative, or some combination. The coefficient values and their sequence determine what the filter does to the sample data.

FIR Filter Processing Stages



Furthermore, the number of processing stages in the algorithm determines the selectivity. In this example, only four stages are used. To produce an even steeper selectivity curve, more stages of processing are added. FIR filters with 8 or more taps are often needed to get the desired degree of selectivity. However, the greater the number of taps, the greater the number of mathematical processes and the longer the processing time.

The Mystery of the Coefficients

Where do the coefficients come from? These are derived by several different mathematical processes beyond the scope of this presentation. Engineers who design DSP filters must know this math and how to use it.

In some cases, the DSP chip manufacturer furnishes libraries of algorithm subroutines with predefined coefficients or programs that can be run to determine the coefficients without having to solve the higher math usually required. With this kind of software, the designer without a massive knowledge of the math can create “cook book” digital filters that work perfectly.

The Infinite Impulse Response (IIR) Filter

Selectivity in a FIR filter is improved as you add sections or taps. But just remember that as you add taps, you automatically increase the computing time. Since you want to do real time processing, that is processing the data as it occurs, the processor must be very fast to make all its calculations between samples. The more taps, the more samples being processed and the greater the amount of computing time.

IIR filters allow the designer to produce very good selectivity with fewer taps and less processing speed. It does this by using a kind of feedback. In other words, it calculates the output samples using not only the current samples but also uses previous results from calculations to produce new output samples.

The IIR Algorithm

The mathematical representation of an IIR filter is similar to that for FIR filters but includes a second part. The expression for the output y[n] is:

$$y(n) = \sum (a_i)\, x[n - i] \;-\; \sum (b_j)\, x[n - j]$$

where i and j may vary from 0 to some selected maximum value k.

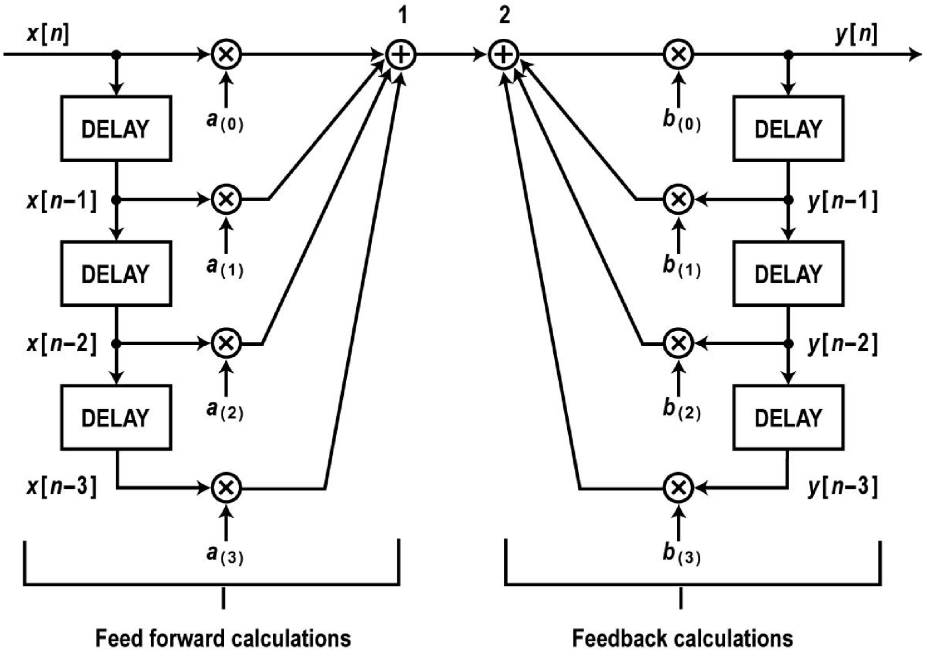
Note that there are two sets of coefficients designated a_i and b_j.

Expanding gives:

$$y(n) = a_0\, x[n] + a_1\, x[n - 1] + a_2\, x[n - 2] + a_3\, x[n - 3] + \ldots\ldots\ldots -$$
$$b_0\, x[n] + b_1\, x[n - 1] + b_2\, x[n - 2] + b_3\, x[n - 3]\ldots\ldots$$

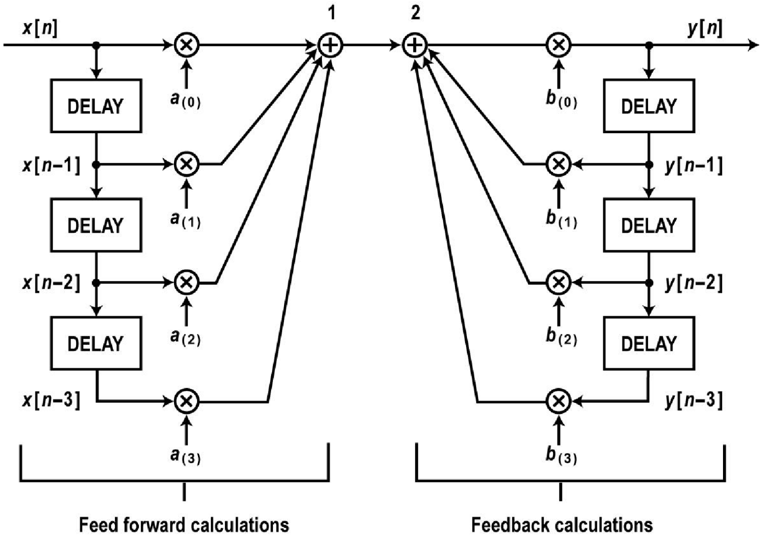
The subtraction is carried out in the adders by making the b coefficients negative. Remember adding a negative number is like subtracting.

IIR Filter Architecture



This figure shows a drawing of one way an IIR filter can be implemented. It does not literally correspond to the equation in the previous frame. However, with algebraic manipulation, the equation can be transformed to the diagram shown.

IIR Filter



The algorithm has two basic sections, the input called the feed forward section on the left and a feedback section on the right. There are four taps per section. The feedback comes by taking the output y[n] and sending it to a second sequence of delays where the output samples are multiplied by the b coefficients then added together. The result is fed to adder 2 where it is combined with the output of the feed forward section.

IIR Filter Design

The IIR design can be applied to any of the four basic filter types, low pass, high pass, band pass, and band reject as described earlier for FIR filters. Selection of the coefficients determines the filter output response. Multiple sections may be cascaded to further shape the output response and improve selectivity.

Again, the determination of the a and b coefficients is beyond the range of this module but various mathematical techniques have been implemented in software to simplify the process.

FIR vs. IIR Filters

FIR filters are the simpler of the two to design and implement. Their response is very predictable and easily duplicated. The phase response is linear which is a major benefit.

On the other hand, FIR filters require many taps to give the kind of selectivity often needed. This in turn requires a very high speed processor which is typically more expensive and power hungry.

The IIR filter can give much better selectivity with fewer taps. These filters require less computing time so they respond very fast. On the down side, they do not have a linear phase response which may or may not be a disadvantage depending upon the application. IIR filters are also much more difficult to design.

Both types are widely used.

Test your knowledge

Digital Signal Processing
Knowledge Probe 2
Digital Filters

Click on Course Materials at the top of the page.

Then choose **Knowledge Probe 2**.