Signal Buffering & Vector Plotting

sT-Embed Training

Ric Kolk
Altair Engineering
rkolk@altair.com

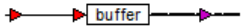
Topics:

- Signal Buffering
- Signal Averaging using Buffer
- Reading & Writing Matrix Elements
- Buffer, PSD, and Plot w/External Trigger
- Using PRBS and PSD to estimate a transfer function

Signal Buffering

Frequently it is necessary to store a sequence of signal data for matrix processing or plotting. The vector "**buffer**" block ("Blocks/Matrix Operation") is used for this purpose.

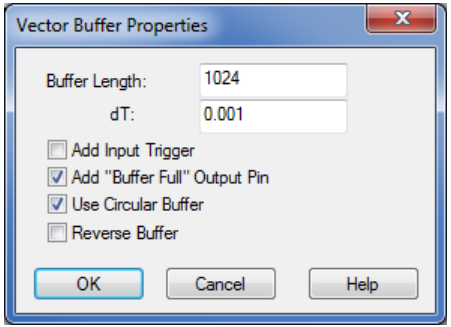
Scalar input signal or sequence



Vector output (1xn); n= "Buffer Length"
Element(1,n) = most recent value

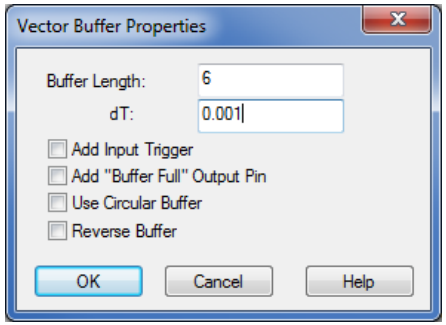
"right click" on the "buffer" block to display its properties:

"Buffer Length": Number of elements, n
"dT": discrete sample time, seconds, to record every signal element, set dT = Time Step
"Add Input Trigger": trigger the buffer with an external signal
"Add Buffer Full Output Pin": Output pin = buffer status = 1/0 = buffer full/ not full
"Use Circular Buffer": When full, buffer begins writing over elements from the beginning
"Reverse Buffer": rotates the output vector 180 degrees, Element(1,1) = most recent value

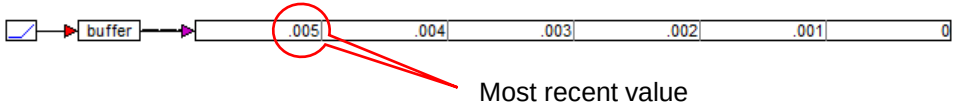


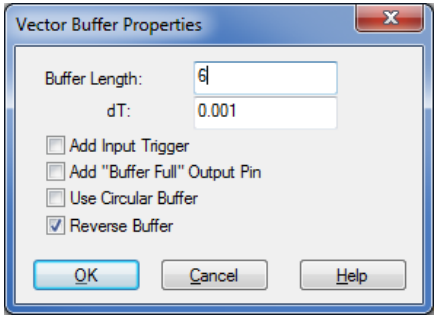
Signal Buffer Examples

Signal Buffering



Example 1: Record 6 elements of a unit “ramp” input signal. [Start (sec), Time Step, End (sec)] = [0,0.001, .005]. As a new value is added, the buffer is shifted right 1 location and the new value is placed in location 1

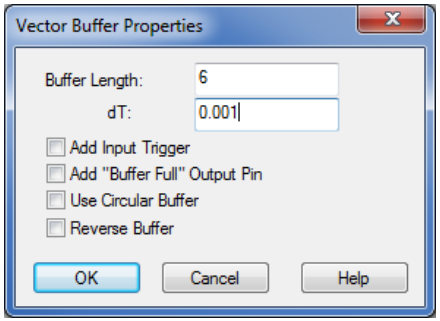


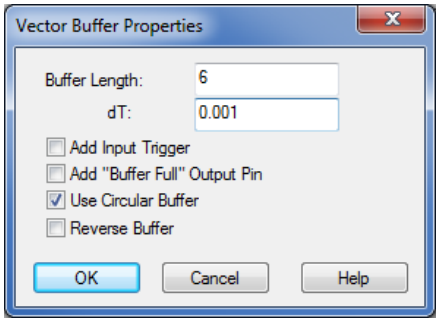


Example 2: “Reverse Buffer” Record 6 elements of a unit “ramp” input signal. [Start (sec), Time Step, End (sec)] = [0,0.001, .005], select “Reverse Buffer”. As a new value is added, the buffer is shifted left 1 location and the new value is placed in location n.

Signal Buffer Examples

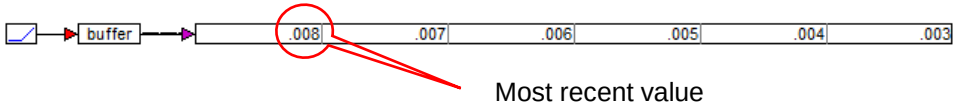
Signal Buffering



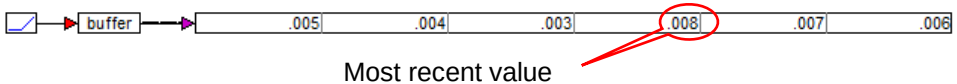


[Signal Buffer Examples]

Example 3: Record 6 elements of a unit "ramp" input signal. [Start (sec), Time Step, End (sec)] = [0,0.001, .008]. As a new value is added, the buffer is shifted right 1 location and the new value is placed in location 1



Example 4: "Use Circular Buffer" Record 6 elements of a unit "ramp" input signal. [Start (sec), Time Step, End (sec)] = [0,0.001, .008]. As a new value is added, the buffer is shifted left 1 location and the new value is placed in location n

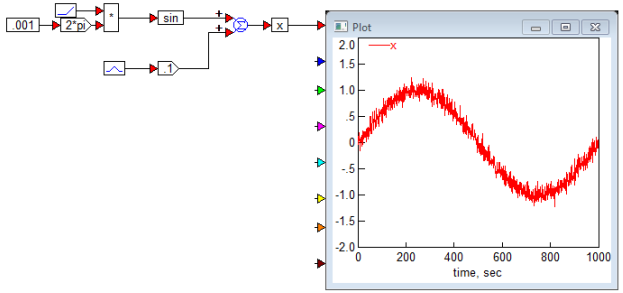


NOTE:
For embedded applications that require a "buffer" block, codegen will execute more efficiently if the "buffer" is configured as "**Circular**". This is because in a "Circular" buffer, only one element is overwritten when a new data point is received. In the Non-Circular configuration, when a new data point is received, every element must be first shifted by one index value. The overhead of shifting every element is time consuming in an embedded application.

Signal Averaging using the Buffer & vSum blocks (1/2)

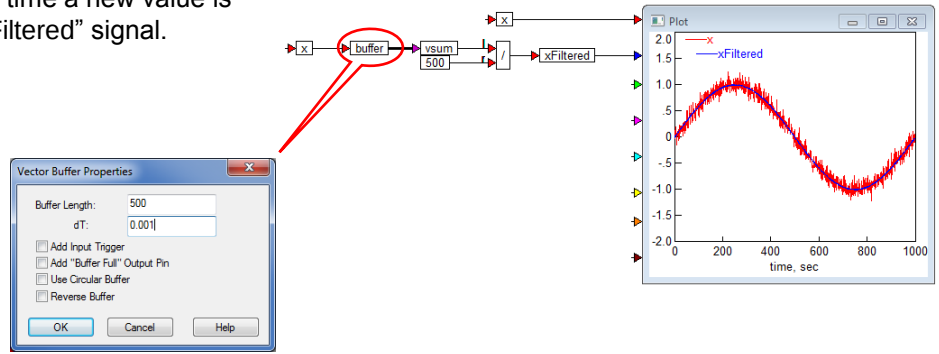
The following block diagram illustrates an application of the “buffer” block to retain a sliding window of the most recent 500 samples of a noisy signal, x. The buffer contents are averaged (using a “vsum” block (“Blocks/Matrix Operation”) every time a new sample is obtained to produce a filtered signal “xFiltered”.

A noisy sin wave signal is created in the block diagram (right). The Simulation is setup with the following values [Start (sec), Time Step, End (sec)] = [0,.001,1000]



[Noisy Sin Wave]

A “buffer” block is configured to buffer the 500 most recent samples of “x”. The buffer contents are then averaged every time a new value is obtained (every 0.001 seconds) to create the “xFiltered” signal.

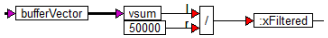


[Noisy Sin Wave with Buffer Filter]

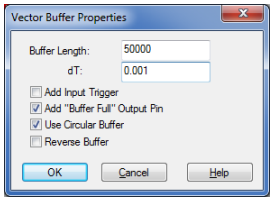
Signal Averaging using the Buffer & vSum blocks (2/2)

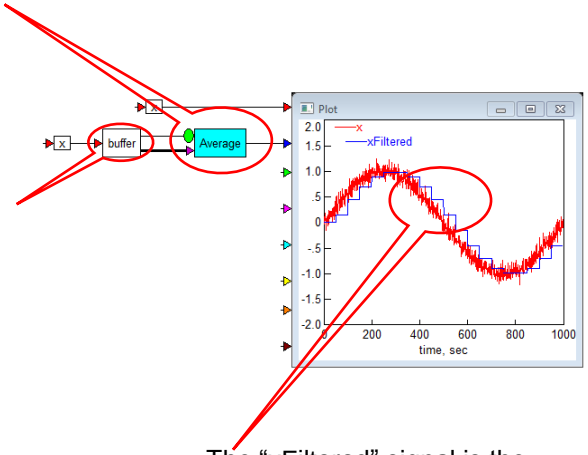
The “Add Buffer Full Output Pin” is normally used with the “Use Circular Buffer” selection to retain a snapshot of the buffer when the buffer becomes full. A full buffer means that every element has been overwritten with new data since the last buffer full event. The signal model, “x” and simulation parameters from the previous page are used.

The averager from the previous page is altered to average 50,000 values and placed in a compound block, “Average” with “Enabled Execution“ checked, will execute once when the “buffer” output pin 1 goes high (1x/50,000 samples)



A “buffer” block is configured to buffer 50000 samples of “x”. When the buffer is full, the “buffer” output pin 1 goes high indicating the buffer is full and triggers the Averager compound block.





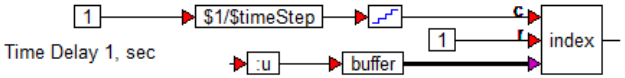
The “xFiltered” signal is the average of the previous 50,000 samples, so, an inherent 50,000 sample delay (50 second delay) is present.

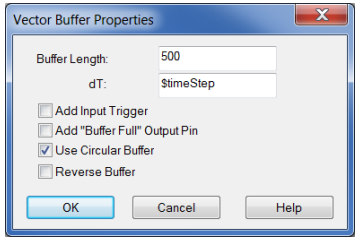
[Noisy Sin Wave with Buffer Filter Triggered Execution]

Delaying a Signal using the Buffer block

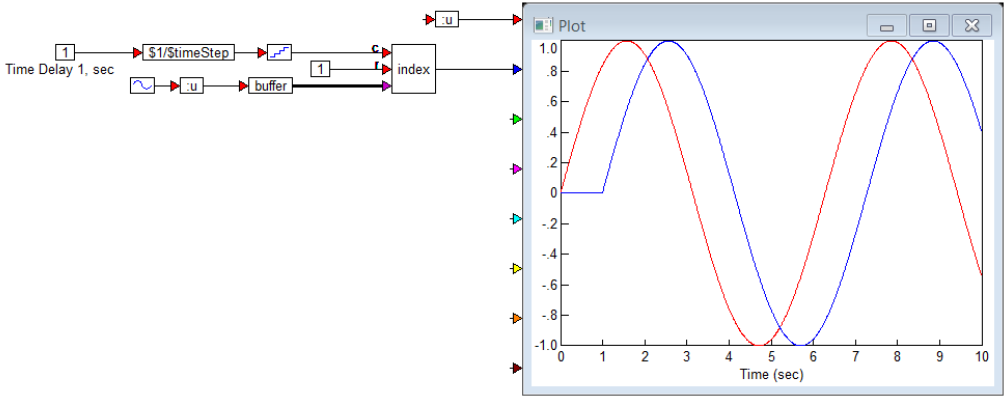
A “buffer” block can be used to delay a signal. The following model produces a 1 second delayed copy of the “u” input signal using a “buffer” block. The “index” block indexes into the buffer row vector and outputs the delayed signal. The delay value is input in seconds and converted to an index by dividing by the simulation step time “$timeStep”. The “quantization” block (configured with “resolution” = 1) is used to ensure the index is an integer value.

The “buffer” is configured as follows:





Setting “u” equal to a 1 rad/sec sinusoid the “buffer” is used to create a 1 second time delayed signal:

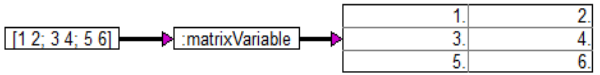


[Buffer Block to Delay a Signal]

Reading & Writing Matrix Elements

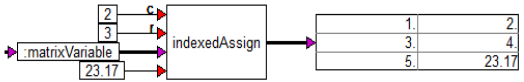
Frequently it is necessary to read or write one element of a vector or matrix signal. The **index** block (“Blocks/Matrix Operation”) is used to read one element and the **indexedAssign** block (“Blocks/Matrix Operation”) is used to write one element.

The “**const**” block (“Blocks/Signal Producers”) is used to define a 3x2 matrix named “:matrixVariable”. The “**display**” block (“Blocks/ Signal Consumers”) is used to view the contents of the matrix.



The “**index**” block (“Blocks/Matrix Operation”) is used to display element [3,2] of the “:matrixVariable” signal.

The “**indexedAssign**” block (“Blocks/Matrix Operation”) is used to modify element [3,2] of the “:matrixVariable”.



Buffer, PSD, & Plot Block with External Trigger (1/2)

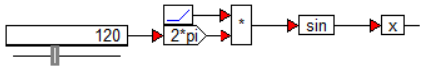
The “**External Trigger**” selection “**plot**” block can be used to display vector data.

When using this option, the “plot” block plots vector data values (y-axis) versus the data index (sample number on the x-axis).

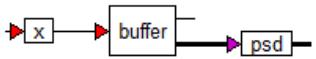
The data index can be scaled to a range defined by the “X Upper Bound” and “X Lower Bound” values in the “Plot Properties” “Axis” tab.

Example: Plot the Power Spectral Density (PSD) of a sin wave with frequency ranging from 0 to 300 Hz.

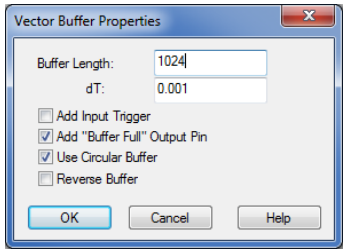
The sin wave signal, “x”, is created using a sT-Embed “slider” block configured for frequencies from 0 to 300 Hz. (right)



The sT-Embed “psd” block (“Blocks/Matrix Operation”) applies an FFT to a signal vector, produced by a “buffer” block, and calculates signal power (amplitude ^2) over a frequency range extending from 0 to 1/(2* “Time Step”) Hz. Since we want the input signal to operate up to 300 Hz, the simulation “Time Step” is selected as 0.001 seconds. This will support a frequency range from 0 to 500 Hz. The Simulation “End (sec)” = 1000 sec.



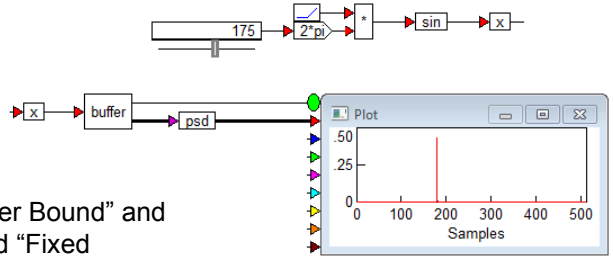
A block diagram is constructed with the signal, “x”, applied to a “buffer” block configured to buffer 1024 values at a sample time, “dT” = “Time Step” = 0.001 seconds. The “buffer” output is then applied to the “psd” block (right).



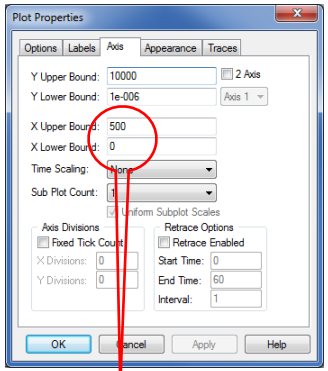
Buffer, PSD, & Plot Block with External Trigger (2/2)

The “buffer” output pin 1 is connected to the “External Trigger” pin of a “plot” block and the “psd” vector is connected to pin 1 of the “plot”.

The “plot” block will now plot the vector data values from the “pst” (y-axis) versus the data index (sample number on the x-axis). Notice when the “External Trigger” configuration is selected, the “plot” “X Label” becomes “Samples”. The plot will be updated every time the 1024 element “buffer” data is refilled. Results using a sin wave frequency = 175 Hz are shown at the right.

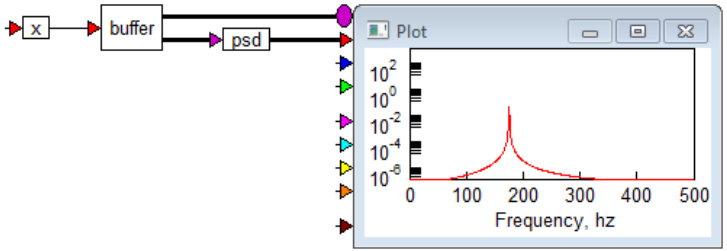


Normally, the y-axis is represented in “log10” units and the x-axis in Hz. This requires the “X Upper Bound” and “X Lower Bound” values to be set as follows in the “Axis” tab and in the “Options” tab, “Log Y” and “Fixed Bounds” are selected.



500 Hz = 1/(2*0.001)

Results using a sin wave frequency = 175 Hz are shown below:



PSD of Sin Wave Example

TFID Example

End of Section

EMBED™
solidThinking