

## **System Maximizer** **User Guide**

*Version 1.1.4*  
*December 10, 2006*

Copyright © 2006 Optunis, Inc. All rights reserved. Optunis, Inc. reserves the right to make changes to any products and services at any time without notice. Optunis, Inc. assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Optunis, Inc.

# Table of Contents

<b>INSTALLING SYSTEM MAXIMIZER .....</b>	<b>6</b>
System Requirements .....	6
<b>GETTING STARTED.....</b>	<b>6</b>
<b>Adding Components To a Design .....</b>	<b>6</b>
Parts of a component .....	6
<b>Parameters .....</b>	<b>7</b>
Setting Parameters .....	7
Showing the Parameters in the Schematic View.....	7
<b>Wires and Connections.....</b>	<b>8</b>
Drawing Wires.....	8
Selecting Wires .....	9
Stretching Wires.....	9
Deleting Wires.....	9
Copying Components .....	9
Selecting Multiple Components and Wires for Moving and/or Deletion.....	10
<b>Making a Hierarchical Design .....</b>	<b>10</b>
<b>Changing Sample Rates .....</b>	<b>11</b>
<b>Simulating .....</b>	<b>11</b>
Start .....	11
Stop.....	12
Viewing the Simulation Results .....	12
<b>Saving / Restoring the Design.....</b>	<b>13</b>
<b>Tips and Techniques .....</b>	<b>13</b>
<b>COMPONENT LIBRARY REFERENCE.....</b>	<b>13</b>
<b>Sources.....</b>	<b>13</b>
Constant .....	13
Pulse .....	13
Sinusoid.....	13
Cosinusoid.....	13
Ramp .....	13
Random .....	13
Gaussian .....	14
Periodic Series .....	14
From File .....	14
1 .....	14
0 .....	14

<b>Sinks</b> .....	<b>14</b>
Time Plot .....	14
Two Input Time Plot.....	14
Frequency Plot.....	14
X-Y Plot .....	14
X-Y Sel Plot.....	14
Polar Plot.....	15
Polar Sel Plot .....	15
To File.....	15
Signal Stats .....	15
<b>Arithmetic</b> .....	<b>15</b>
Gain .....	15
Multiplier .....	15
Adder .....	15
Difference .....	16
Accumulator .....	16
Multiply – Accumulate .....	16
<b>Logical</b> .....	<b>16</b>
Mux .....	16
Comparator .....	16
Barrel Shifter .....	16
<b>Functions</b> .....	<b>16</b>
Sign.....	16
Rectangular to Polar .....	16
Polar to Rectangular .....	16
Look Up Table.....	17
Transcendental (f (x)).....	17
Saturate .....	17
Dead Zone Zero.....	17
Dead Zone Non-Zero.....	17
Hysteresis.....	17
integer.....	18
round.....	18
f(x,y).....	18
<b>Storage / Memory</b> .....	<b>18</b>
Unit Delay .....	18
Serial To Parallel.....	18
Parallel to Serial.....	18
Register .....	18
Shift Register.....	18
Counter.....	19
SRAM .....	19
<b>Specialty Components</b> .....	<b>19</b>
FIR .....	19
IIR .....	19
IIR_Reg .....	20
NCO .....	21
FFT .....	22

Adaptive LMS FIR Filter .....	23
Adaptive LMS FIR Filter with Variable $\mu$ .....	24
Custom Adaptive FIR Filter .....	<b>Error! Bookmark not defined.</b>
<b>Hierarchy Elements .....</b>	<b>24</b>
Hierarchy .....	24
Input Port .....	25
Output Port .....	25
<b>Misc .....</b>	<b>25</b>
Comment Text .....	25
<b>CUSTOM COMPONENTS .....</b>	<b>25</b>
<b>Using Custom Components .....</b>	<b>25</b>
<b>Creating Custom Components .....</b>	<b>26</b>
Methods Required To Be Implemented by the Custom Component .....	26
Sending messages to the Simulator .....	27

## Installing System Maximizer

Installation is simple. Just drag and drop the System Maximizer icon to the Applications library (or your preferred location). Double click System Maximizer to run it.

To install a valid license of System Maximizer, open the System Maximizer -> License Setup ... menu. Enter your username provided in your license email in the username field. Enter the license number in the license number field.

### ***System Requirements***

Operating System: Mac OS X 10.4.0 and above

Processor Requirements: 1 GHz G4 Processor or better recommended

RAM Requirements: 512 MB RAM

## Getting Started

### ***Adding Components To a Design***

To add a component to your design, perform the following steps:

1 – Open the library drawer

You can do this by either:

clicking on the library icon in the toolbar

or by selecting View -> Toggle Library

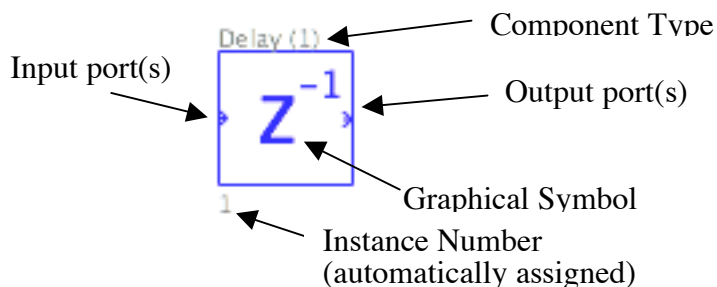
2 – Scroll through the library to find the component you are interested in

3 – Drag and drop this component to your document

### **Parts of a component**

#### **Ports, Type, Instance number, Graphic Symbol**

In a schematic view, a component shows the following:



You may also change parameter values in this panel window

## Parameters

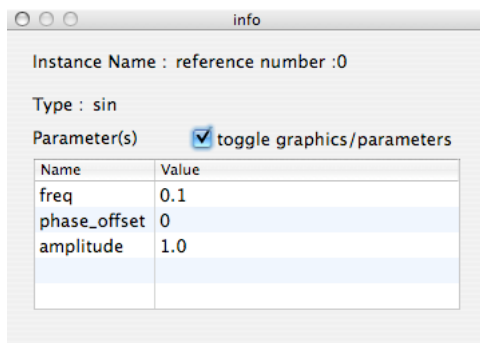
Several components have parameters associated with them. For instance the sin source component has an output that is described by the following mathematical equation

$$\text{amplitude} \cdot \sin((2 \cdot \pi \cdot \text{freq} \cdot k) + \text{phase\_offset}) : \text{ where } k \text{ is the sample number}$$

The user may adjust the amplitude, freq, and phase\_offset parameters.

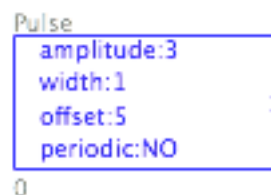
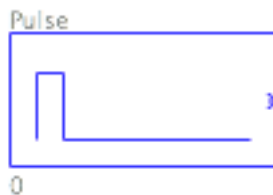
## Setting Parameters

To see the parameters available for a component, select the component and either select info from the toolbar, or choose View -> Show Info... from the menu.



## Showing the Parameters in the Schematic View

Some components also have a parameter view, in which key parameters are displayed.

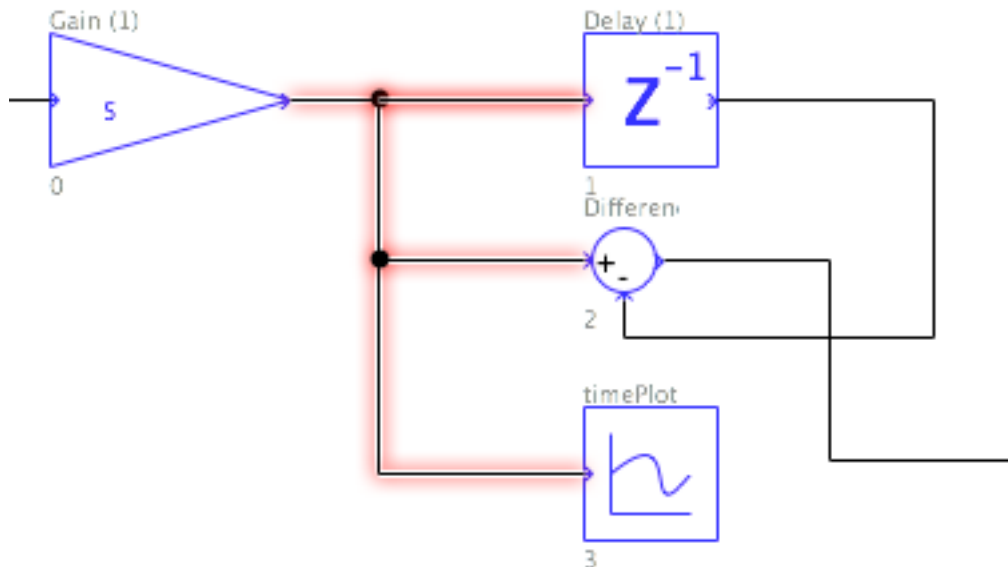


You can choose which view to display by selecting the toggle graphics/parameters button in the info window.

## Wires and Connections

Wires are used to connect the output ports of components to the input ports of other components. There may only a single output port connected to a wire. There may be several input ports connected to this wire. A wire may be composed of wire segments and each segment may have several wire elements.

The following drawing shows several components and wires.



This picture shows 4 wires. There is a wire connected to the input port of the Gain (1) element. There is a wire connecting the output port of the Delay (1) element to the input port of the Difference element. There is a dangling wire connected to the output port of the Difference element. Finally, there is the highlighted wire that connects the output port of the Gain (1) element to the input ports of all the other elements.

A single wire is selected. This selected wire is composed of 5 elements. The segment connected to the timePlot consists of 2 wire elements. All the other segments consist of a single element.

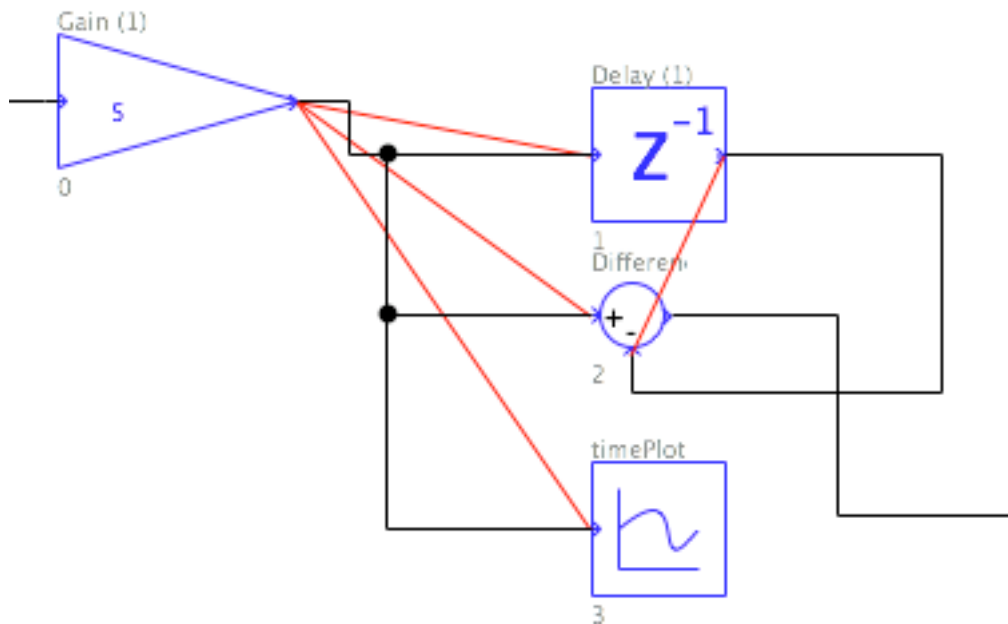
## Drawing Wires

To draw a wire, move your mouse over an input port or an output port (of a component or the end of a wire which is already drawn). The cursor will change to a cross hair, indicating that if you were to drag at this point, a wire would be drawn.

## Selecting Wires

To select a wire, move your mouse over a wire. Click 3 times. The first click will select the wire element. The next click will select the segment. Finally the third click will select the entire wire.

To view the connections between elements, select view -> Show/Hide Port Connections. Red wires will indicate which ports are connected together.



## Stretching Wires

You may stretch a wire segment by selecting a wire element, and dragging, or by selecting a component with a wire already connected to it and moving the element. The wire will stretch to the component. This feature is called rubber banding.

## Deleting Wires

To delete a wire, select the wire, and press the delete key. To delete a wire segment, select the wire segment, and press the delete key.

## Copying Components

To copy a component, press the option key, and drag the component. A copy of the component (with all associated parameters) is then created. Drop this copy someplace in the block diagram.

## Selecting Multiple Components and Wires for Moving and/or Deletion

Multiple components and wires may be selected for moving or deletion. Select multiple items, move your mouse to an empty location in the block diagram. Click and drag. A bounding box will appear. Keep dragging until the bounding box covers all of the desired components. Let go of the mouse button. All the components and wires are now selected. These components may be moved to another place on the block diagram or deleted.

## Making a Hierarchical Design

Hierarchy support includes the following capabilities:

- Hierarchical symbol editor to create, edit, and save hierarchy symbols
- Block diagram editor can push into and pop out of a hierarchical level
- Block diagram editor indicates the hierarchy level in the window title
- Input and output port elements added to the design library
- Hierarchy flattening simulator

Creating a hierarchical design consists of two different steps:

- 1 – Create the base design components for the hierarchy
- 2 – Add the hierarchy elements to your block diagram (and point to the correct base components), and connect up the ports.

To create the base design components for the hierarchy:

- 1 - Open the library drawer
- 2 - Scroll through the library to find input ports and output ports. The input port looks like:



The output port looks like:



The input port sources data to the hierarchy. The output port generates data that leave your hierarchy component.

- 3 – Drag and drop the input and output ports to your base design.
- 4 – create unique port names for your input and output ports. These names will be used later in the hierarchy symbol editor.
- 5 – save the base design.

To add hierarchy elements to your design:

- 1 - Open the library drawer
- 2 - Scroll through the library to find the hierarchy component. It looks like:



- 3 - Drag and drop this component to the design.
- 4 - Right click on the hierarchy component and select edit symbol to open the symbol editor.
- 5 - Set the input and output ports to the exact same name as the input and output port names you used when generating the base components.
- 6 - Set the filename of the base component.

## ***Changing Sample Rates***

All non-constant (dynamically changing) sources, and registered elements have a parameter called `skip_rate`. Changing `skip_rate` will change the sample rate. `skip_rate` determines how quickly this output will change. For single-rate systems `skip_rate` should be set to 0. A `skip_rate` of 2 will cause the source to skip a simulation cycle. In essence, it will run twice as slow, and data will be held constant on the following cycle. A `skip_rate` of 3 will cause the source to run at one-third speed, and hold data for an additional two cycles. (Note: A `skip_rate` of 1 has no effect).

To decimate data by an integer rate, simply set the skip rate. To interpolate data to a higher integer rate, set the skip rate of the slower components appropriately. The result will be the same as increasing the sample rate and applying a zero-order hold. If you desire to insert zeroes instead of holding the data constant, insert a mux element to cycle between zero and the data.

For fractional rate interpolation and decimation, use the least common multiple, and set the skip rates appropriately. For instance, in the case of  $3/4$  rate decimation, start from a base of 12, and set `skip_rates` of 3 and 4 as appropriate.

## ***Simulating***

### **Start**

There are two ways to run a simulation.

Method 1 – setting parameters and running a simulation

- 1) Open the View -> Simulation Parameters ... dialog box, or click the setup sim icon in the toolbar.

- 2) Enter the simulation sample period.
- 3) Enter the number of simulation samples to simulate.
- 4) Click the Run button to run the simulation

Method 2 – re-running a simulation with the same parameters as previously set

- 1) Click the simulate button in the System Maximizer toolbar.

Note: in order to correctly run a simulation, all the input ports of all components must be connected to a valid output.

## **Stop**

If you would like to stop a simulation (currently in progress),

- 1) Open the View -> Simulation Parameters ... dialog box, or click the setup sim icon in the toolbar.
- 2) Click the stop button

## **Viewing the Simulation Results**

### **Scope –**

The time Plot is used to graph the results in the time domain. Click twice on the timePlot component in your design to open the graph. The first time the graph is opened; appropriate settings are chosen for Y-Gain and X-Gain. To change various graph settings click the show controls button on the graph.

### **Cursors, Paper Tape**

There are vertical (sample) and horizontal (value) cursors available for taking measurements. To view the actual data values, click the show paper tape button.

### **Other Controls**

(Note: for longer simulations, this may take some time to generate the paper tape). It is possible to change the title, y-label, and x-label, as well as the grid gains (both time and value), and signal offset. There is a marker on the left edge of the graph, which indicates where 0 would be located in the Y-axis. Available plot types are: line, stairs (default), and stem

## **Dual Time Plot – locked plots**

To view two signals at the same time (locked to each other in time), use the dual time plot.

## **FreqPlot**

This component is used to view a magnitude plot in the frequency domain.

## ***Saving / Restoring the Design***

## ***Tips and Techniques***

## **Component Library Reference**

### ***Sources***

#### **Constant**

The output is always equal to the constant parameter value

#### **Pulse**

The output is a pulse, based on the parameter settings, such as amplitude, offset, repeat rate, and repeating.

#### **Sinusoid**

The output is a sine wave, based on the parameter settings, such as amplitude, frequency, and offset.

#### **Cosinusoid**

The output is a cosine wave, based on the parameter settings, such as amplitude, frequency, and offset.

#### **Ramp**

The output is a ramp function, linearly increasing until it resets.

#### **Random**

This source provides random (uniformly distributed) data as output.

The max\_value parameter sets the maximum allowed value. The min\_value parameter sets the minimum allowed value. Allowed values for the type parameter are real or integer.

## **Gaussian**

This source provides random (Gaussian distributed) data as output. The mean parameter allows you to set the average value. The standard deviation may be set via the `std_dev` parameter.

## **Periodic Series**

### **From File**

The output is read from a text file. The text file contains a list of values. When the values are completed the output may be recycled or stop (depending on the parameter setting).

### **1**

The output is always 1.

### **0**

The output is always 0.

## ***Sinks***

### **Time Plot**

A time plot is displayed.

### **Two Input Time Plot**

A time plot is displayed.

### **Frequency Plot**

A magnitude plot in the frequency domain is displayed.

### **X-Y Plot**

X,Y input data is presented, and the plot draws a straight line from X,Y location to X,Y location. This plot can be used to view trajectories and paths.

### **X-Y Sel Plot**

X,Y input data is presented, and the plot draws a straight line from X,Y location to X,Y location. If the `sel` input is set to 1, then the selected X-Y point is highlighted in red. There is also a trace intensity slider, which may be used to determine how often a trajectory (or path) is taken. Naturally, this plot can be used to view trajectories and paths.

## Polar Plot

Magnitude and phase input data is presented, and the plot draws a straight line from the polar coordinates (mag, phase) location to polar location (mag, phase). This plot can be used to view trajectories and paths.

## Polar Sel Plot

Magnitude and phase input data is presented, and the plot draws a straight line from the polar coordinates (mag, phase) location to polar location (mag, phase). If the sel input is set to 1, then the selected, then the selected polar coordinate point (mag, phase) is highlighted in red. Naturally, this plot can be used to view trajectories and paths.

## To File

Data is dumped to a text file specified in the parameter: filename  
A complete pathname is required.

## Signal Stats

Signal Stats keeps track of and reports statistical information about a signal. The signal is captured on the data\_in port when the enable port is 1 or more. Information reported include : # elements, minimum value, maximum value, mean, variance, and standard deviation.

$$\begin{aligned} \text{mean} &= \frac{1}{N} \sum_{n=1}^N \text{xin}_n \\ \text{variance} &= \frac{1}{N} \sum_{n=1}^N (\text{xin}_n - \text{mean})^2 \\ \text{standard deviation} &= \sqrt{\frac{1}{N} \sum_{n=1}^N (\text{xin}_n - \text{mean})^2} \end{aligned}$$

To view the signal stats, double click the component. Signal stats may be written out to a file (both the statistics, as well as the raw data used to compute the statistics.)

## *Arithmetic*

### Gain

The input is multiplied by the gain factor and sent to the output.

### Multiplier

The two inputs are multiplied together and sent to the output.

### Adder

The two inputs are added together and sent to the output.

## **Difference**

One input is subtracted from the other, and sent to the output.

## **Accumulator**

## **Multiply – Accumulate**

## ***Logical***

### **Mux**

The output is selected from a particular input port. The input ports to select from are 0, 1, etc. The sel port chooses which input port is selected.

### **Comparator**

Inputs A, and B. The A>B output goes high if  $A > B$ , the A < B output goes high if  $A < B$ .

### **Barrel Shifter**

The output is a shifted version of the input (which considered to be a bitwise integer). A direction (applied to the dir input port) of 1 will shift left, and a direction of 0 will shift right. The magnitude (applied to the mag input port) will determine the number of bits to be shifted. Input data is supplied in the dat\_i port, and the shifted data is presented on the dat\_o port.

## ***Functions***

### **Sign**

The output may be one of 3 possible values  $-1, 0, +1$ . If the input is less than 0, then the output is  $-1$ . If the input is greater than 0, then the output is 1. If the input is exactly equal to zero, then the output is 0.

### **Rectangular to Polar**

And X, Y input is converted to Magnitude and Phase. The phase is expressed in radians.

### **Polar to Rectangular**

A Magnitude and Phase (expressed in radians) is converted to X, Y values.

## Look Up Table

The input (which must be a whole number) is the address of the location of the look up table. The output is the table value for that particular input. The table values are determined by a filename, which contains the address and data in the following format:

Addr: data

Unused addresses are set to zero.

## Transcendental (f (x))

The following functions are supported in the transcendental component:

abs, sqrt, sin, cos, tan, sinc, asin, acos, atan, ln, log2, log10, exp, pow2, pow10

Note: inputs to sin, cos, and tan are in radians, and outputs from asin, acos, and atan are also in radians.

## Saturate

The output is the same as the input as long as it within the maximum and minimum values (specified as parameters). If the input exceeds the maximum, it saturates to the maximum value. If the input is less than the minimum, the output is the minimum value.

## Dead Zone Zero

The output is the same as the input as long as it within outside of the dead zone parameter (min value). If the input less than min value and greater than  $-(\text{min value})$ , then the output is zero.

## Dead Zone Non-Zero

The output is the same as the input as long as it within outside of the dead zone parameter (min value). If the input less than min value and greater 0, then the output is min value. If the input is less than zero, and greater than  $-(\text{min value})$ , then the output is  $-(\text{min value})$ .

## Hysteresis

The output is the same as the input as long as it is greater than (min value) or less than  $-(\text{min value})$ . If the input within  $\pm(\text{min value})$ , and the last input was greater than min value, then the output is  $\pm(\text{min value})$ . If the input within  $\pm(\text{min value})$ , and the last input was less than min value, then the output is  $-(\text{min value})$ .

## **integer**

converts the input to an integer value, stripping all fractional components. For instance, an input of 1.5 would convert to 1, and an input of -0.5 would convert to 0.

## **round**

converts the input to an integer value. If the input is positive, it rounds upwards. If the input is negative, it rounds downwards. For instance, an input of 0.5 would round to 1, and an input of -0.5 would round to -1.

## **f(x,y)**

computes two input functions. The value of the parameter function\_name determines the function computed. Allowed values for function\_name are :

pow(x,y), x/y, mag(x,y), phase(x,y)

## ***Storage / Memory***

### **Unit Delay**

The input is delayed by a single sample time.

### **Serial To Parallel**

Serial input is accepted on the ser input. When the convert input goes high, this data is converted to parallel data. Parameter settings are bit width, and endian. Bit Width must be between 2-32. Endian can be either “little” or “big”. If endian is “little” the LSB is entered ser\_I first. If endian is “big” the MSB enters ser\_I first.

### **Parallel to Serial**

Parallel input is accepted on the par\_i input. When the convert input goes high, this data is converted to serial data. Parameter settings are bit width, and endian. Bit Width must be between 2-32. Endian can be either “little” or “big”. If Endean is “little” the LSB is output ser\_I first. If Endean is “big” the MSB is output first.

## **Register**

### **Shift Register**

A programmable shift register function. Values may be loaded (from the dat\_i port) by applying a 1 to the load port. A value of 1 on the dir port will cause the loaded values to

shift left every simulation cycle. Similarly, a value of 0 will cause a shift right every simulation cycle. Updates (loads, and shifts) only occur when the value at the en port is 1. A value of 1 on the rst port, will clear the register to 0.

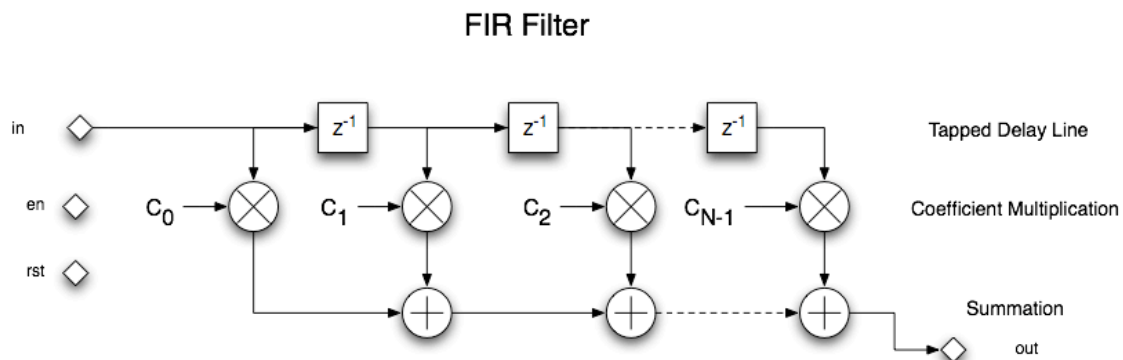
## Counter

This is a loadable counter function, with programmable direction. The rst input will reset the counter to a value of 0. The counter will update when the en port is 1. The direction (up or down) is determined by the value at the up\_dwn port. A value of 1 on the up\_dwn port will cause the counter to count up. A value of 0 will cause it to count down. Finally, it is possible to load a value to the counter, by applying a 1 to the ld\_val port (which will load the value on the value port).

## SRAM

### *Specialty Components*

## FIR



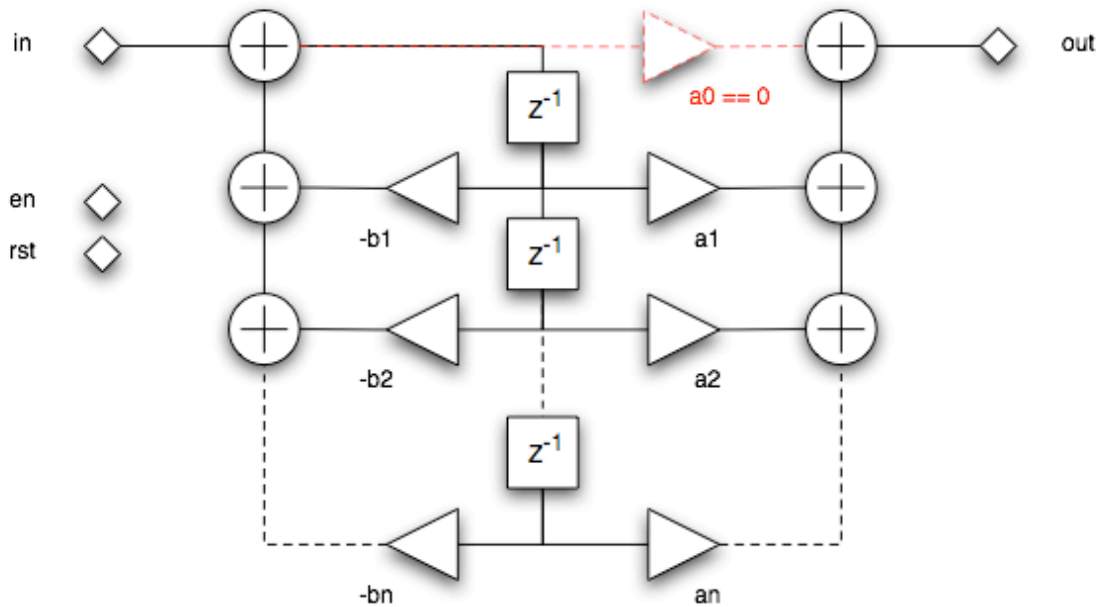
The FIR filter implements a straightforward FIR filter. Filter coefficients are specified via the input filename. Outputs are calculated when the en is equal to 1. The rst signal will clear the tapped delay line.

## IIR

The IIR filter implements a direct form IIR filter.



## IIR Filter (registered) Structure



Coefficient values for  $b$  and  $a$  are specified with an input file. A sample input file is shown below (Note, that the first feed forward coefficient is specified as 0):

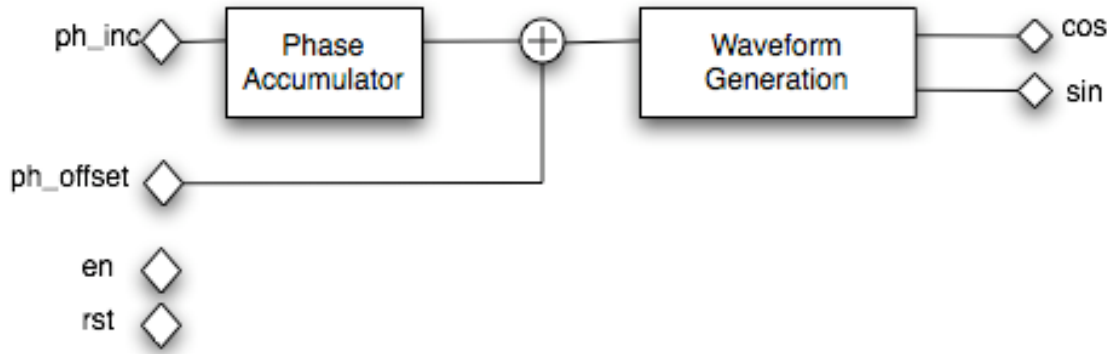
```

feed_foward
0
3
-3
feed_back
-1.3
.6
    
```

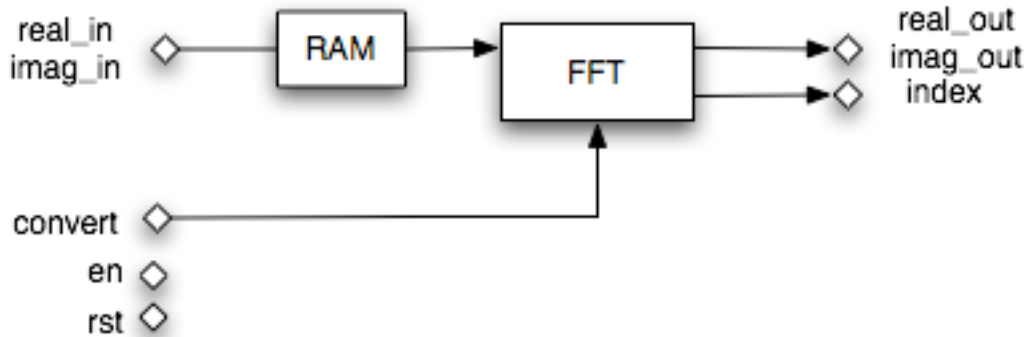
The `feed_foward` command indicates that the following coefficients are for  $a_0$ ,  $a_1$ ,  $a_2$ , ... etc.  $b_1$  is  $-1.3$ . Since the feedback gain is actually  $-b_1$ , the feedback gain is  $1.3$  for the first coefficient, (and  $-0.6$  for the second coefficient). Note that there are no combinatorial paths from input to output, and this component is therefore considered a registered component (the standard IIR filter is considered a combinatorial component).

### NCO

This component is a numerically controlled oscillator

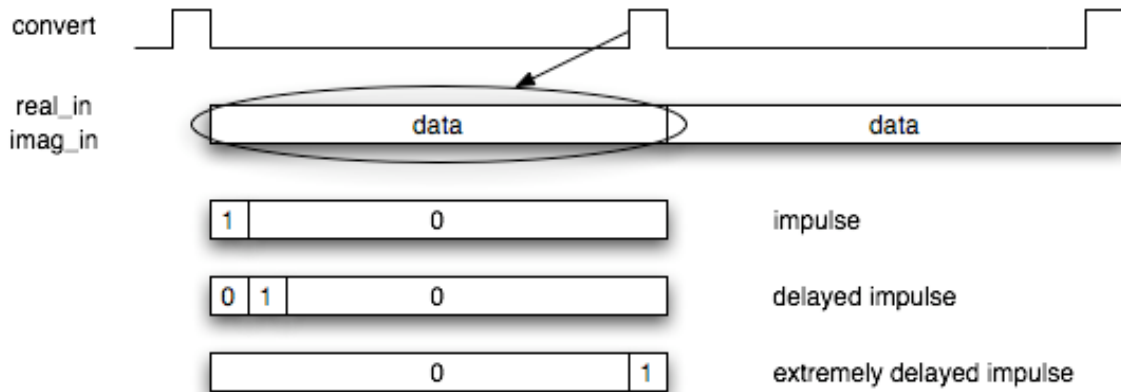


### FFT

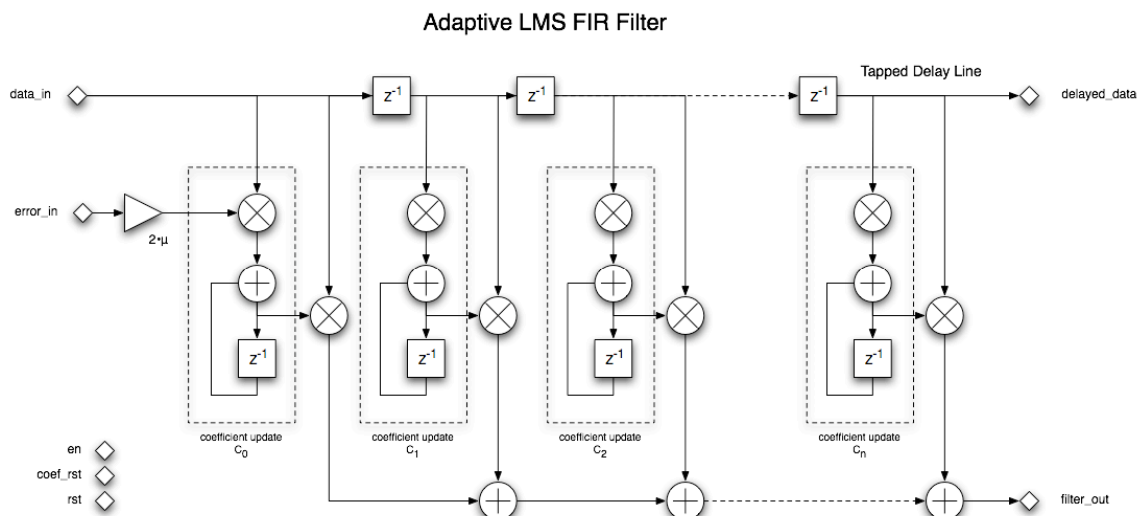


The FFT function is a composite function. Input data is sent to a storage element. When the convert input is provided with an input value of “1”, then the FFT will take data and convert it. Note, input data is sent first data first, and last data last. The input data is valid during when convert is high. The following timing diagram shows impulses and delayed impulse-timing diagrams.

## FFT Usage Timing Diagram



## Adaptive LMS FIR Filter



The function implements an adaptive least mean square finite impulse response filter. Input data is presented on data\_in port. There are two resets. A value of 1 applied to the rst port will reset the entire system, clearing the tapped delay line, and coefficient values. A value of 1 applied to the coef\_rst port will clear only the coefficient values (to zero). An error\_in port supplies the error signal used to calculating coefficient updates for the adaptive LMS algorithm. Coefficients are updated when the coef\_upd input port is 1 (set the input to 0 to stop coefficients from updating). Finally resulting filtered output is available on the filter\_out port. There is an additional output from the tapped delay line for cascading filters. All calculations only occur when the en port is set to 1.

The number of coefficients may be set by changing the num\_coef parameter. The coefficient convergence constant  $\mu$  may be set by changing the mu parameter. Coefficients may be written to an external (tab delimited) file (chosen by the coefficient\_output\_filename parameter) by setting save\_coefficients\_to\_file parameter to YES.

Coefficients adapt according to

$$W_{k+1} = W_k + 2 \cdot \mu \cdot e \cdot X_k$$

where

$$W_k = \begin{bmatrix} c_0(k) \\ c_1(k) \\ \vdots \\ c_n(k) \end{bmatrix} \text{ is the coefficient matrix,}$$

$$X_k = \begin{bmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-n) \end{bmatrix} \text{ is the delayed input data matrix,}$$

$\mu$  is the convergence constant (set as a parameter), and e is the error input.

## Leaky Adaptive LMS FIR Filter with Variable $\mu$

This filter works just like the Adaptive LMS FIR Filter, but the convergence constant  $\mu$  is available as an input port. This allows for dynamically changing  $\mu$ , based upon other factors. For instance, you could perform a power calculation, and base  $\mu$  on the reciprocal of the power (as in the Normalized LMS algorithm), and feed this value to the Adaptive LMS FIR Filter with Variable  $\mu$ .

In addition, the coefficient update algorithm includes a leakage factor (between 0 and 1, where 0.0 = no leakage, and 1.0 = leak everything) :

$$W_{k+1} = [(1 - leakage) \cdot W_k] + (2 \cdot \mu \cdot e \cdot X_k)$$

## Hierarchy Elements

### Hierarchy

This component is used to add a hierarchical block to the current block diagram. By default there is no symbol. After dragging and dropping the component to a block diagram, right-click (or Command-Click) to edit the symbol. The symbol editor allows

you to set the input ports, output ports, and specify the filename of the underlying block diagram.

### **Input Port**

This component is used inside a hierarchical block to obtain data. The input port name must be exactly the same as the input port name on the symbol. This component should not be utilized at the top level of a hierarchy.

### **Output Port**

This component is used inside a hierarchical block to generate data from the hierarchy. The output port name must be exactly the same as the output port name on the symbol. This component should not be utilized at the top level of a hierarchy.

### ***Misc***

#### **Comment Text**

This component is used to drop a text element to the block diagram. The text element has no effect on the simulation.

## **Custom Components**

The premium edition of System Maximizer allows for the use of custom components. The Custom component SDK is included with a download of the latest version of System Maximizer.

The SDK provides source code for a sample custom function and a test-debug utility(ies). It includes a detailed walkthrough of the compilation process using XCode, to get you up and running quickly.

At Optunis, we use the tools we develop with. A premium license will make some of these tools available to you. These tools may speed up your development and provide a competitive edge, however, they are not general purpose, packaged, and released products. If these tools include source code, it expected that you may need to make modifications to the code to be useful.

### ***Using Custom Components***

The premium edition of System Maximizer allows for the use of custom components. To use a custom component, it is necessary to set either a project or user directory. (To set a project directory, select View -> Project Directories, and add your directory path. To set a user directory, select System Maximizer -> User Directories ... , and add your directory path.)

Then simply, copy your custom component to this directory, and System Maximizer – Premium will automatically include it in the library of available components.

## ***Creating Custom Components***

Custom components are .bundle files that contain all the required information about the component. This consists of information describing component behavior in an XML property list, combined executable code. The bundle name must be the same as the principle class of the bundle.

### **Methods Required To Be Implemented by the Custom Component**

The following methods must be implemented by custom component:

`-(NSDictionary *) getCustomElementInfo;`

This method call returns an NSDictionary object that describes the custom component. A sample dictionary is provided in the SDK. It is called prior to simulation.

`-(void) simStart:(id) sim;`

This method is called when the simulator starts. The component may perform any pre-simulation activity at this time (such as allocating memory, or setting initial conditions). A reference is provided to the simulator object for message exchange.

`-(void) simSetReference:(int) refNum;`

This method is called prior to simulation. It sets the ID for the reference object. The custom component must remember this ID for use in any communication to the simulator.

`-(int) refNum;`

This method is call provides the ID originally provided by the simulator.

`-(void) simInitWithParameters:(NSDictionary *) parameters;`

This method is called prior to simulation. An NSDictionary of parameters (consisting of key-value pairs) is sent to the custom object.

`-(void) simOutputGen;`

This method is called during to simulation. When the simulation starts, all objects are sent this message. The custom component **MUST** send a message back to the simulator to set its output values.

Computation may be performed on this cycle or during `-(void) simInputCapture` method call, but a message must be sent back to the simulator before this method exits.

The component may send a message to the simulator to ask for input values (used for combinatorial computations) by calling `-(void) simOutputGen`;

(Note that initial conditions of internal registers should be set in `simStart`).

`-(void) simInputCapture;`

This method is called during to simulation. It provides information to the input ports at time  $t=0$  (for registered inputs). Computation may be performed on this cycle or during `-(void) simOutputGen` method call. Obtain inputs used for combinatorial output

calculation when the simulator calls your `-(void) simOutputGen;` method

`-(void) simEnd;`

This method is called at the end of simulation. It provides a chance to free up any allocated memory.

## **Sending messages to the Simulator**

Messages may (and some messages must) be sent back to the simulator. All messages include a `refNum` (identification number) to indicate which object is sending the message. The simulator will respond to the following messages:

```
-(int) getIntValueFor:(int) refNum
    forInputPort:(int) portNum;
-(float) getFloatValueFor:(int) refNum
    forInputPort:(int) portNum;
-(double) getDoubleValueFor:(int) refNum
    forInputPort:(int) portNum;
```

These messages ask for values on the input port. They may be called during a `-(void) simOutputGen;` method access or during `-(void) simInputCapture;`

```
-(void)setOutput:(int) refNum forOutputPort:(int) portNum
    toIntValue:(float) outval;
-(void)setOutput:(int) refNum forOutputPort:(int) portNum
    toFloatValue:(float) outval;
-(void)setOutput:(int) refNum forOutputPort:(int) portNum
    toDoubleValue:(float) outval;
```

These messages set the output values on an output port. They must be called during a `-(void) simOutputGen;` method access or during `-(void) simInputCapture;`

(Note: it is only necessary to call one of these methods to set a value. The simulator will perform any protocol conversions as required.)

```
-(NSDictionary *) getSimParameters;
```

This message queries the simulator for an `NSDictionary *` object of simulation parameters. The dictionary contains key-value pairs for each simulation parameter. Currently supported parameters include `samplePeriod`, `numCycles`. See the SDK for a detailed list of parameters.

```
-(void) setSimError:(int) refNum
    withMessage:(NSString *) errMessage
    andErrNum:(int) errNum;
```

This message tells the simulator that this component may not function under a given set

of circumstances. The simulator will present this error message to the user. It is typically used when a parameter value does not make any sense (such as a filename pointing to a non-existent file).