

CHAPTER 8

Synplify DSP FPGA Tutorial

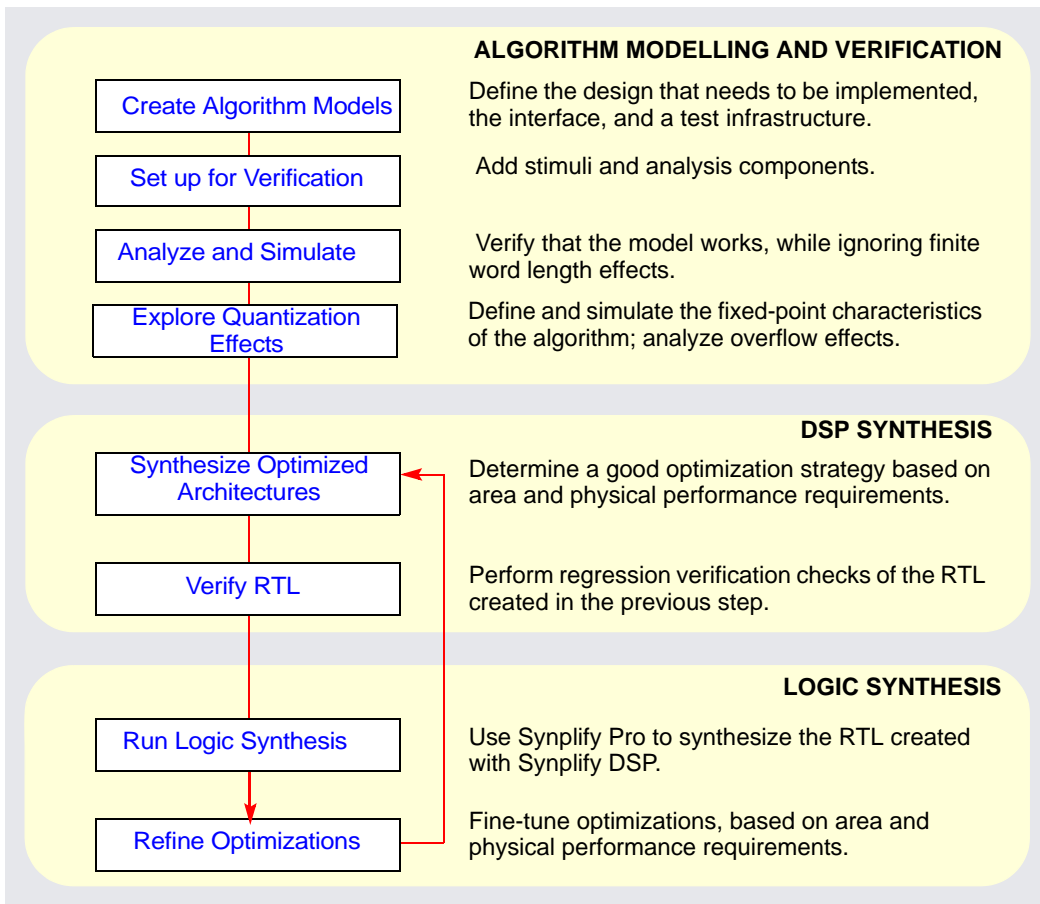
This tutorial gives you a quick introduction to working with the Synplify[®] DSP software for FPGA technologies. It shows you how the Synplify DSP product bridges the technology gap between MathWorks Simulink and the FPGA synthesis product line from Synopsys.

The following topics first describe the flow and then describe the stages in the Synplify DSP FPGA tutorial:

- [Tutorial Design Flow, on page 8-2](#)
- [Create Algorithm Models, on page 8-3](#)
- [Set up for Verification, on page 8-11](#)
- [Analyze and Simulate, on page 8-15](#)
- [Synthesize Optimized Architectures, on page 8-24](#)
- [Verify RTL, on page 8-29](#)
- [Run Logic Synthesis, on page 8-29](#)
- [Refine Optimizations, on page 8-31](#)

Tutorial Design Flow

This tutorial follows the flow for an FPGA DSP design, from algorithm concept to FPGA implementation, using the Synplify DSP software and the Synplify Pro software for synthesis. The tutorial follows an example that has already been set up, describing the steps along the way. It is created for the FPGA Actel, Altera, Lattice, and Xilinx technologies. If you are targeting another FPGA vendor, you can follow the sequence of the flow to familiarize yourself with it.



Create Algorithm Models

The design used in this tutorial is a simple, low-pass FIR filter. In this design capture stage, you use Synplify DSP blocks to capture the functionality that must be implemented in FPGA hardware. To capture a Synplify DSP design, there are two simple rules:

- The design must be bounded by Synplify DSP blocks. It must have a Synplify DSP Port In block for each input and a Synplify DSP Port Out block for each output.
- Use the Synplify DSP blockset to implement your algorithm behavior. Any functionality that is to be implemented in hardware must be instantiated from the Synplify DSP blockset.

This section describes these stages:

- [Start the Demo Tutorial, on page 8-3](#)
- [Add Port In and Port Out Blocks, on page 8-5](#)
- [Add the FIR Block, on page 8-7](#)

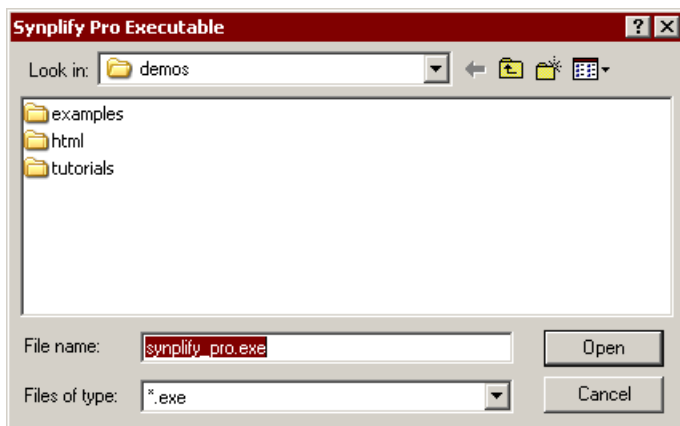
Start the Demo Tutorial

This tutorial uses the FIR example from the demos directory. The example has already been set up, so the tutorial describes the steps that are automatically implemented in the example.

To follow along and open this example, do the following:

1. Start the demo tutorial:
 - From the MATLAB window, select Help->Demos.
 - Go to Synplify DSP->Tutorials
 - Double-click FIR Tutorial.

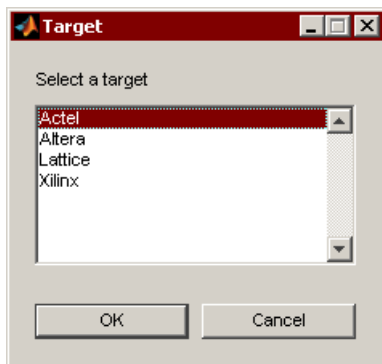
2. If required, specify the path to the Synplify Pro executable in the dialog box that opens.



The tutorial looks for the Synplify Pro executable in the default installation folder. You only need to specify the path to the executable if it does not find it at the default location.

3. Select the FPGA target architecture when you are prompted.

This tutorial follows an Actel target, and the dialog box settings and examples reflect this. If you choose one of the other three vendors, some settings might be different. You can still run through the sequence in the tutorial and get comfortable with the design flow, even if you are not using any of these four vendors.



The demo tutorial opens with two windows.

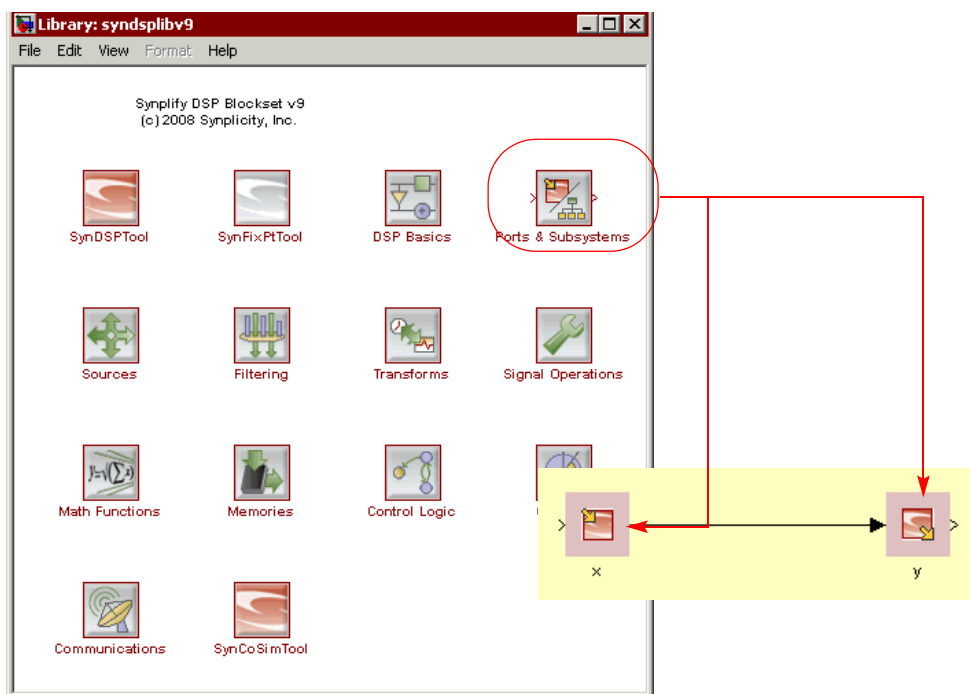
Add Port In and Port Out Blocks

When you start the demo, two windows open:

- The model window with the first screen of the demo tutorial
- A dialog box for port parameters

The following describes the sequence of steps that was run automatically. If you are working on your own design, you would do these steps manually.

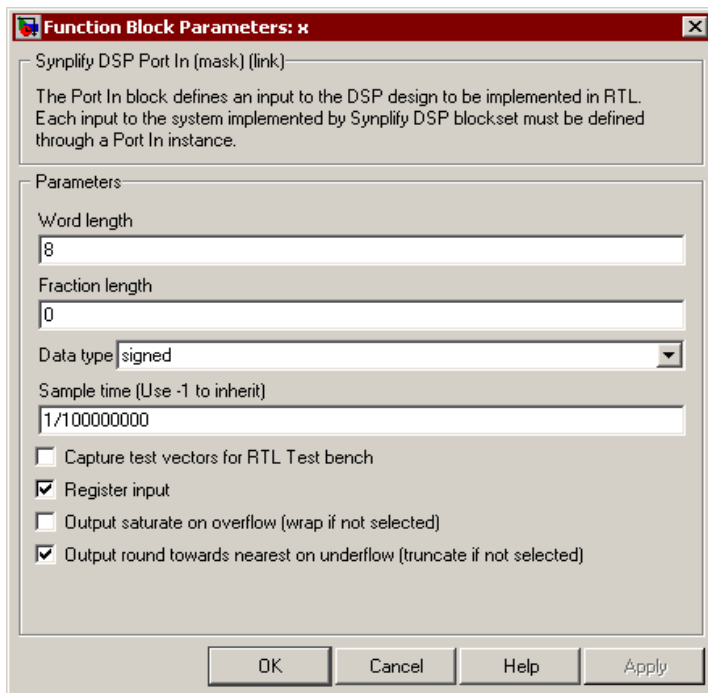
1. The demo first instantiates the Synplify DSP Port In and Port Out blocks from the Synplify DSP Ports & Subsystems library.



The model window shows the Synplify DSP Port In and Port Out blocks instantiated as *x* and *y*, respectively. Putting in these blocks satisfies the first rule for Synplify DSP design (see [Create Algorithm Models, on page 8-3](#)), which is to bound the design with these two blocks.

2. Set parameters for the Port In block.

The parameters that were set automatically are displayed in the open dialog box. Notice the settings for Word length and Sample time as displayed in the dialog box. The value of the settings might vary slightly, depending on the FPGA technology you selected. The following figure shows the Actel parameters.



3. Go to the next screen.

- Close the dialog box.
- Double-click Next in the model window to go to the next stage of the design, which is to add the FIR block.



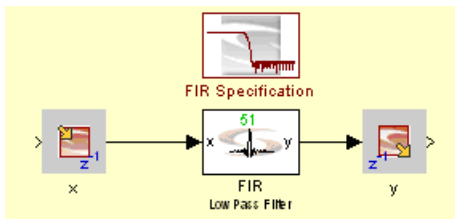
Add the FIR Block

When you double-click Next, three things happen:

- The model window is updated to include new blocks, including the FIR.
- A dialog box opens with parameters for the FIR.
- A FIR specification toolbox window opens.

The following describes these steps that the demo runs automatically.

1. The demo automatically instantiates the following blocks:
 - The FIR block from the Synplify DSP DSP Filtering library, which it names FIR Low Pass Filter.
 - The Synplify DSP FDATool block, which it renames FIR Specification.

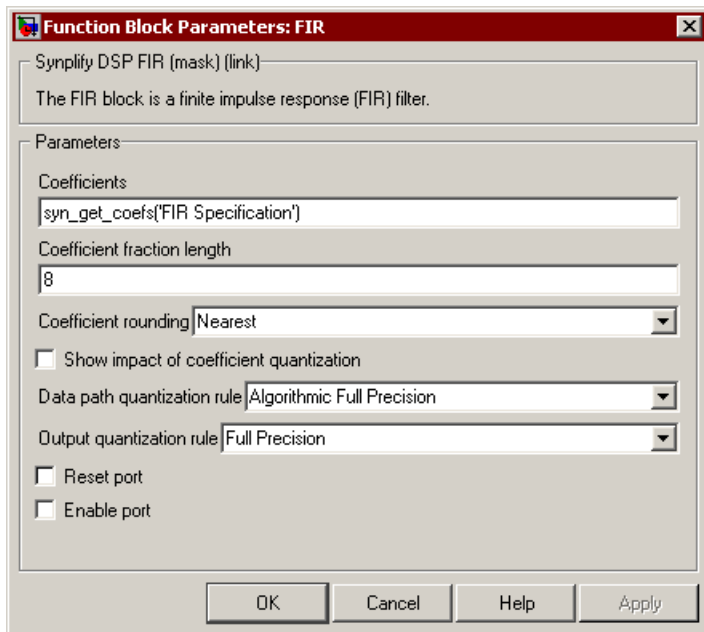


2. The demo sets parameters for the FIR Low Pass Filter block. In particular, note the following settings:

Coefficients	The <code>syn_get_coefficients</code> function in this field specifies that the tool use the coefficients set in the FIR Specification (FDATool) block. Alternatively you can use a MATLAB vector variable.
--------------	---

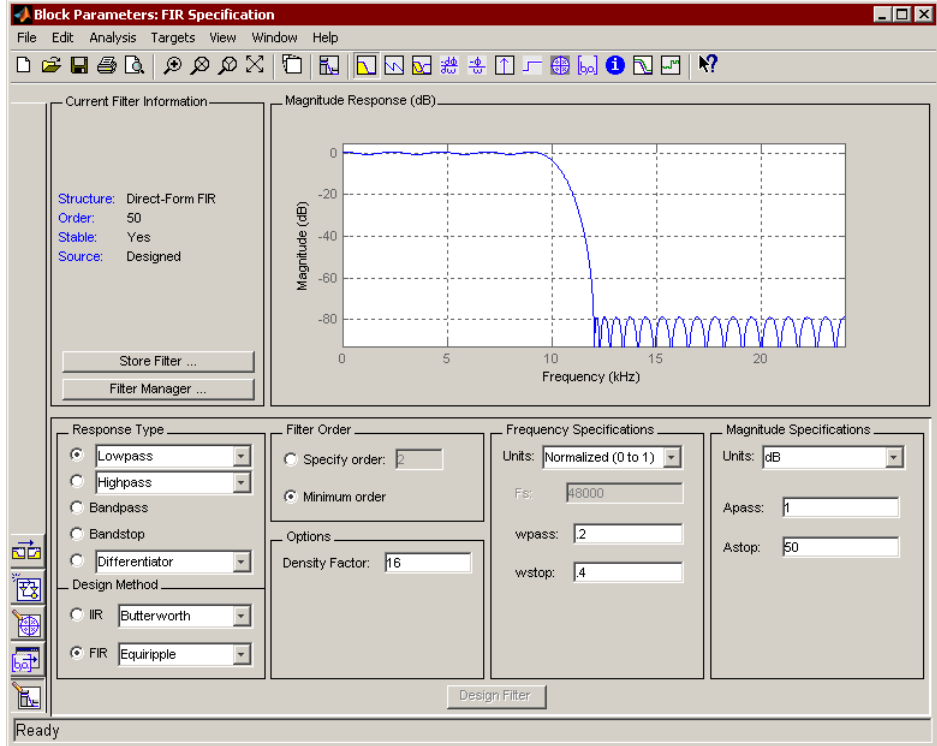
Coefficient word length	This sets the precision of the coefficient quantization.
-------------------------	--

Data path format and Output format	This selects the precision of the internal format.
------------------------------------	--



3. Next, the demo defines the FIR coefficients with the FIR Specification (FDATool) block and the MathWorks Filter Design and Analysis Tool. Note the following settings in the MathWorks tool window:
 - Order: The default is 50.
 - Frequency specifications wpass and wstop
 - Magnitude specification: astop

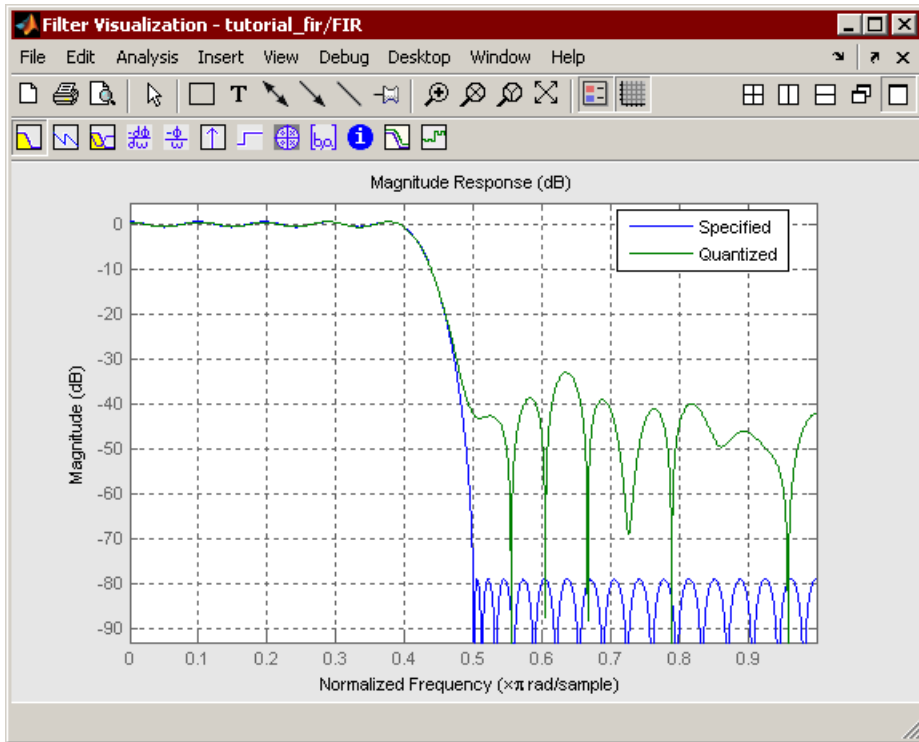
This sets full-precision FIR coefficients. The FIR block quantizes these coefficients. The FIR block icon reflects the settings, showing a 50th order FIR filter with 51 taps, because the number of coefficients (taps) specified was 50.



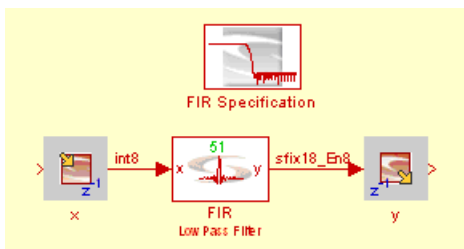
4. To view the results, do the following:

- In the FIR parameters dialog box, click Show Impact of Quantization. Another window opens and shows how the quantized coefficient compares to the full coefficient.

The quantization of a signal is determined by the quantization propagated from input signals. Each block in the Synplify DSP blockset calculates the quantization of the outputs based on block-specific rules and the quantization on the inputs. You can also manage the quantization of a signal directly with a block cast operation inside the block.



- To view the propagation of parameters, select Format->Port/Signal Displays in the model window, and enable the following to configure the display: Sample Time Colors, Port Data Types, and Signal Dimensions.



5. Go to the next screen by closing the dialog boxes and tool windows and double-clicking Next in the model window.

Set up for Verification

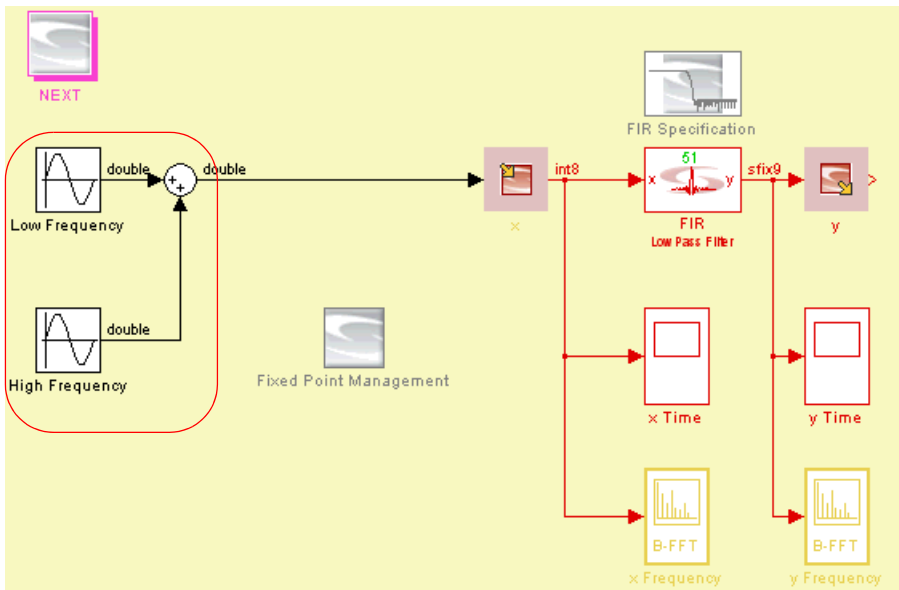
In this stage, the demo adds Simulink stimuli and analysis components to the schematic to verify the model.

- [Add Stimuli Components, on page 8-11](#)
- [Add Analysis Components, on page 8-12](#)

Add Stimuli Components

The demo automatically runs the following steps.

1. It creates low and high frequency signals to the input (x) of the algorithm. You see the following:



- The demo automatically adds two instances of the Simulink->Sources->Sine Wave block to the design schematic to generate sine waves. The demo names them Low Frequency and High Frequency.
- It adds a Simulink->Math Operations->Sum block to the design. Note how the blocks are connected to the x input.

2. The demo then sets sine wave block parameters. Check the dialog boxes of the Low Frequency and High Frequency blocks, and note the settings for the following:

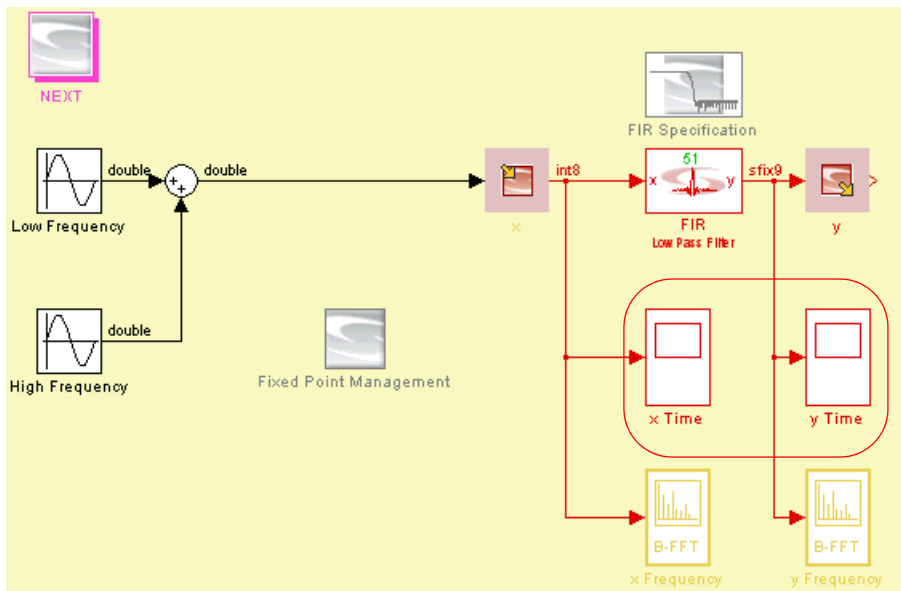
- Amplitude
- Frequency

3. Close the Low Frequency and High Frequency source blocks.

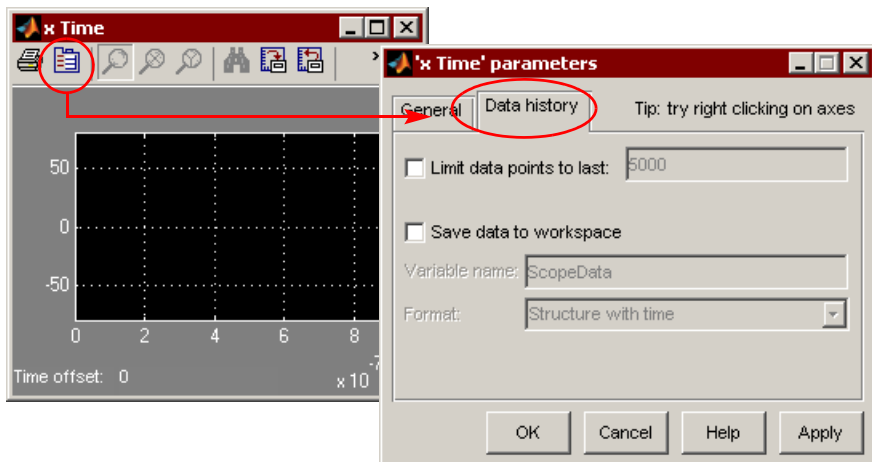
Add Analysis Components

By default, Simulink does not store the data and you must explicitly set up scopes to store the data for subsequent plotting as described below. Set up the design to store data and analyze the simulation results in the time domain. For this tutorial, these steps have been run automatically, and the model window shows the finished results.

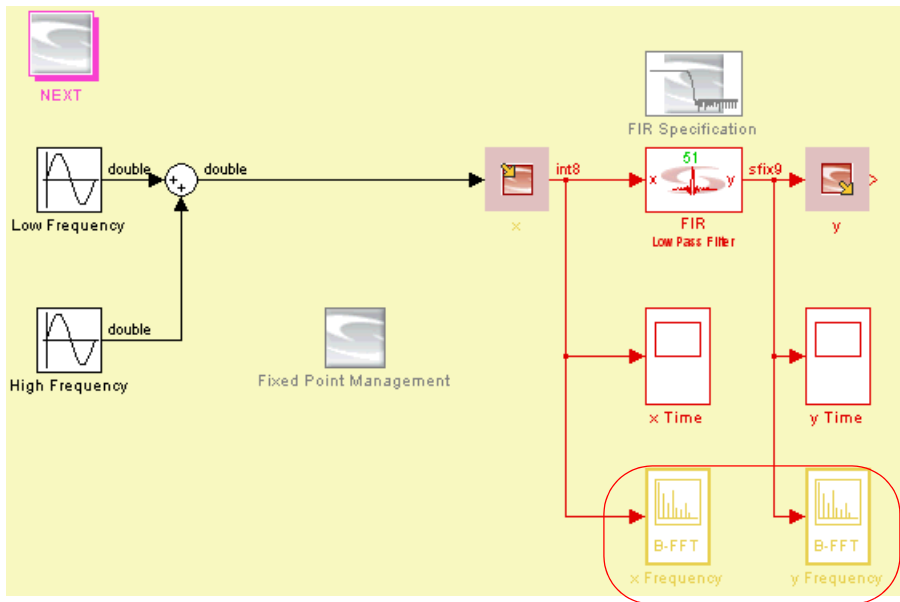
1. The demo automatically adds two instances of the Simulink->Sinks->Scope block for time domain analysis. The scopes are named x Time and y Time. The model window shows the following:



- Note how they are connected to the input and output of the FIR instance.
2. The demo sets scope parameters. You can view the settings by doing the following:
- Double-click the x Time scope to open the scope window. Click the Parameters icon to open the x Time parameters dialog box. Note that Data History->Limit data points to last has been disabled.

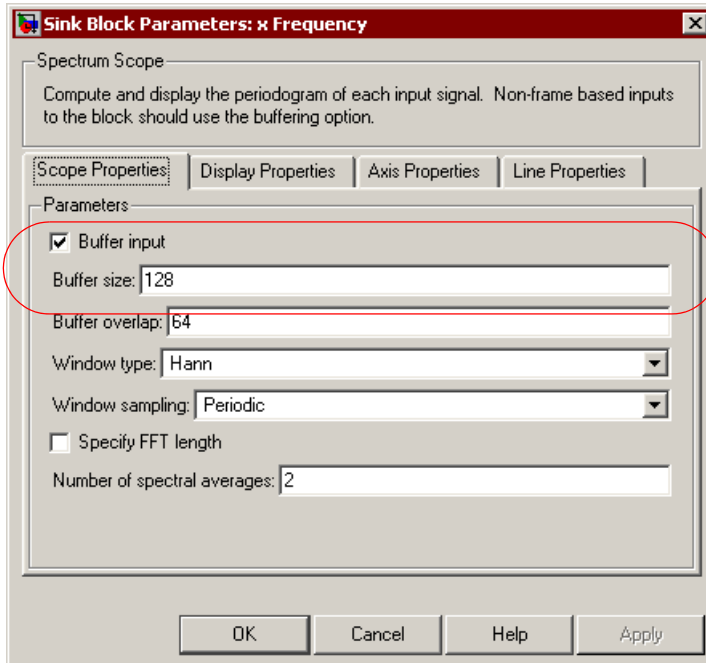


- Repeat the previous step for the y Time scope.
 - Close the scope windows.
3. The demo automatically adds two instances of the Signal Processing Blockset->Signal Processing Sinks->Spectrum Scope block for frequency domain analysis.
- In the demo, the scopes are named x Frequency and y Frequency.
 - Note how they are connected to the input and output of the FIR.



4. View the settings for the spectrum scopes.

- Double-click x Frequency and y Frequency.
- Note that the buffer size is set to accommodate a 128-word signal for the FFT frame (Buffer Input is enabled, and Buffer size is set to 128). The following shows the settings in the x Frequency dialog box.

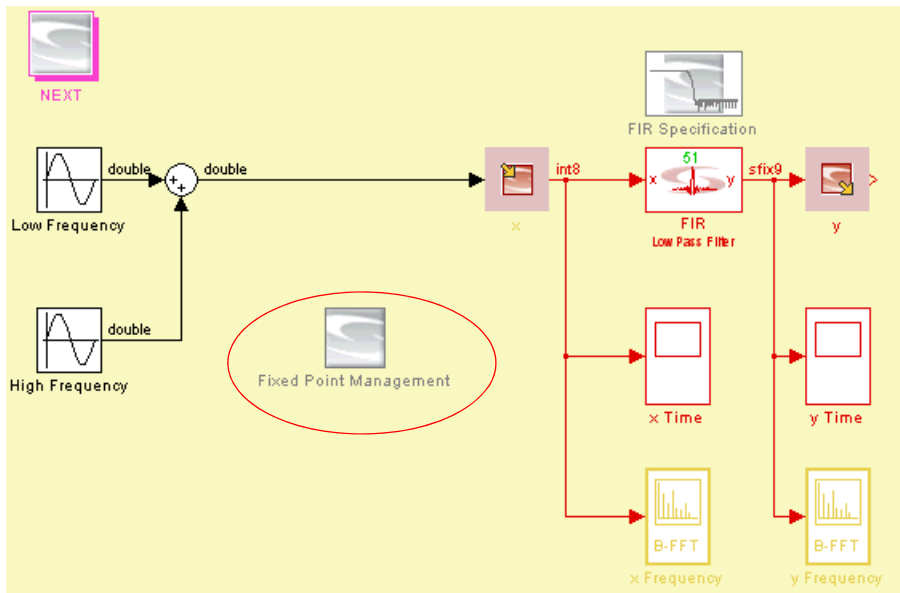


5. Close the dialog boxes for the scopes.

Analyze and Simulate

1. The demo instantiates SynFixPtTool from the Synplify DSP Blockset to manage fixed-point settings.

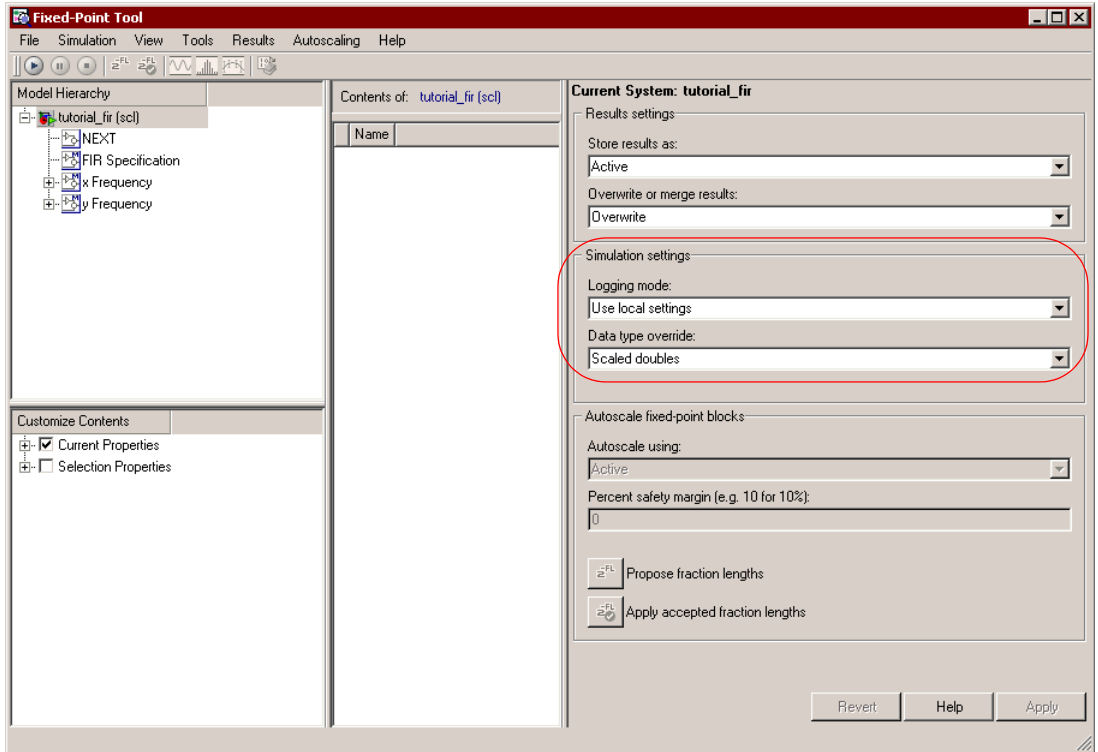
Note that the demo renames the block Fixed Point Management. This block instantiates the library path. It lets you explore quantization by overriding the different blocks in the design with floating-point settings and events.



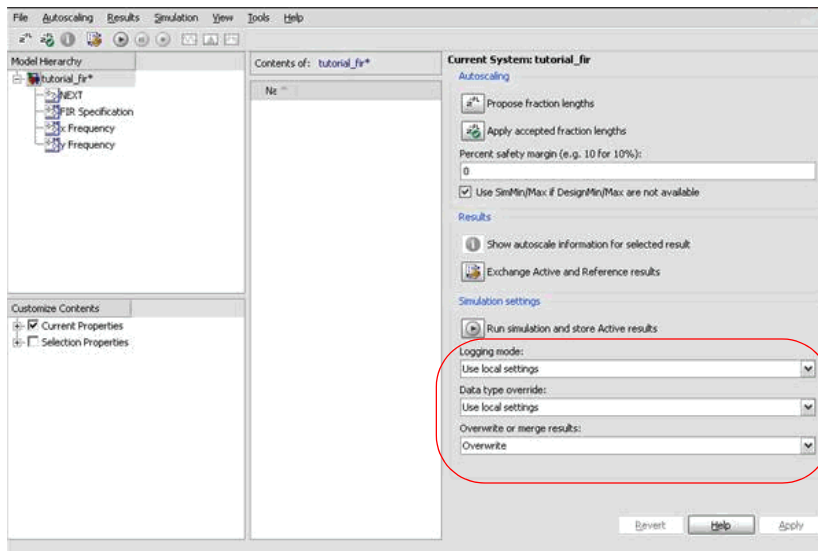
2. View the settings by double-clicking Fixed Point Management to open the Simulink Fixed-Point Settings toolbox. This toolbox provides control over the accuracy for individual levels or blocks in a hierarchy. Note the following settings:
 - Select Current System is set to tutorial_fir, because this is a small design and so that the settings apply to all blocks.
 - Logging mode is set to Use local settings.
 - Data type override is set to Use local settings.
 - Close the window.

Note: Overflow logging is only supported with MATLAB 2008A and later; it is not supported with MATLAB 2007A or 2007B.

The MATLAB interface varies, depending on which version you are using. The following shows the relevant portion of the toolbox as it appears in MATLAB 2007B:

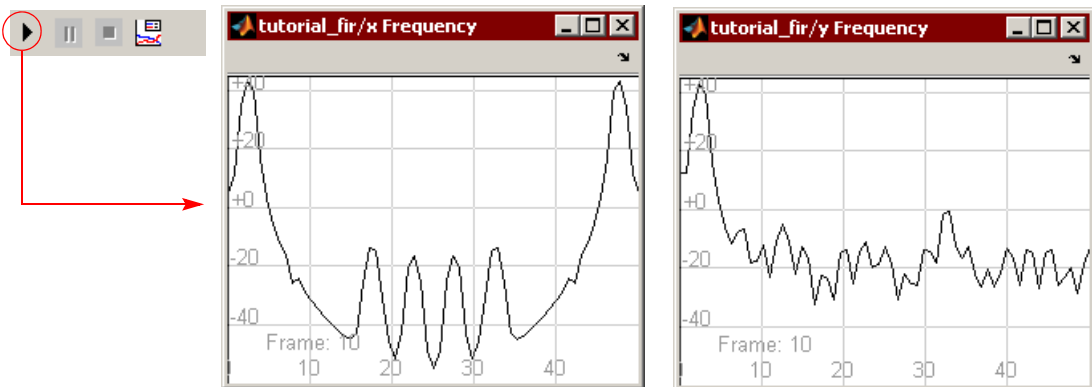


The following shows the MATLAB 2008A toolbox:



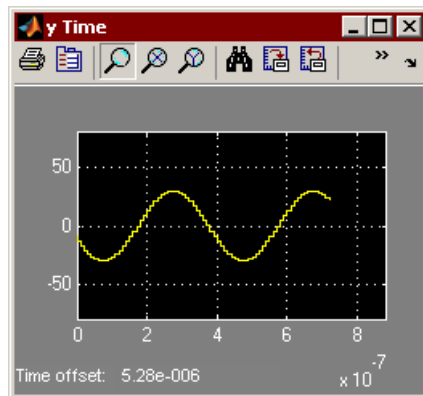
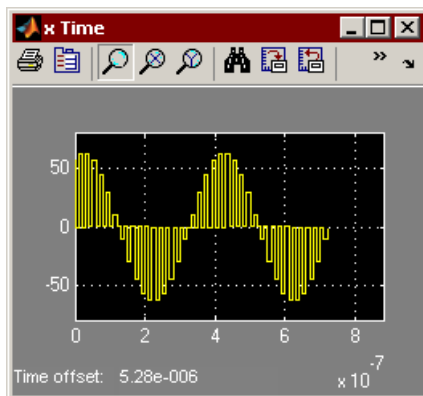
- Click the right arrow in the toolbar of the model window to simulate the design with the fixed-point settings.

You get the results shown below. The input scope (x Frequency) shows low and high frequency spikes. For y Frequency, the high frequency has been filtered.



- Double-click x Time and y Time and view the waveforms.

The input (x) scope shows a low-frequency signal superimposed on a high-frequency carrier, and the output scope shows the filtered low-frequency signal.



5. Close the toolbox window and scopes, and double-click Next in the model window.

Explore Quantization Effects

The following describes how the demo analyzes quantization effects, using floating-point simulation.

- [Running Floating-Point Simulation, on page 8-20](#)
- [Analyzing the Impact of Quantization, on page 8-21](#)

Running Floating-Point Simulation

When you click Next, the following are displayed:

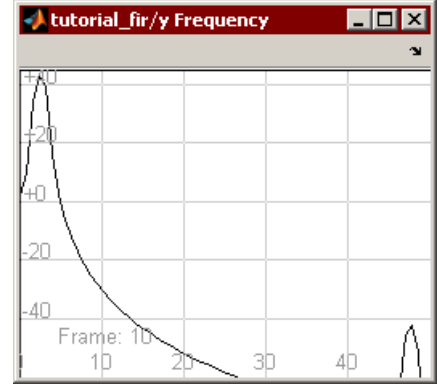
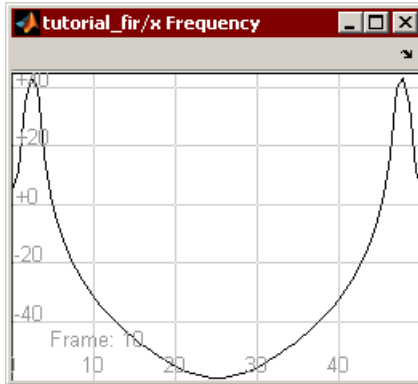
- The Fixed-Point Settings toolbox reopens with new settings.
- The scopes reopen with new data.

The demo automatically overrides the fixed-point settings with the floating-point format. Using floating-point settings ensures that simulation validates the algorithm with full-accuracy calculations. The Synplify DSP tool makes it easy to do this without changing your design by allowing the Simulink floating-point override to propagate through the design subsystems automatically. The following description describes how the demo overrides the original settings with the floating-point format.

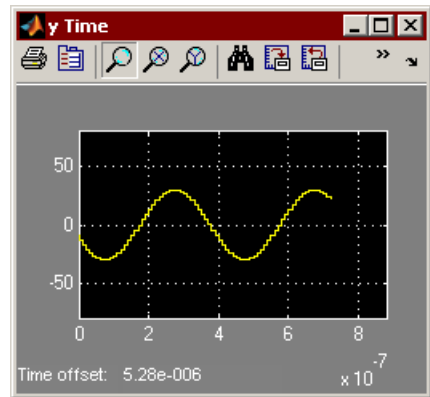
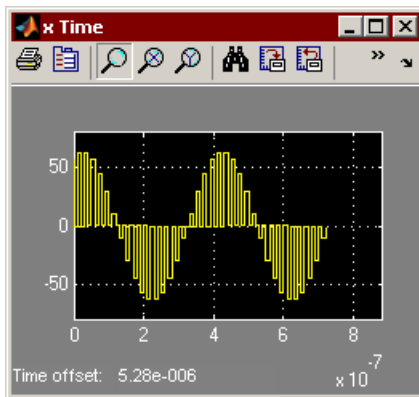
1. In the Fixed-Point Settings toolbox, note the following:
 - Data type override is set to Scaled Doubles.
 - Logging Mode is set to Overflow Only (MATLAB 2008A and 2008B only). This mode is not supported in MATLAB 2007A or 2007B.

This removes quantization and lets you verify the algorithm. You can use this technique to identify quantization effects. Logging the overflow events also lets you explore the effects of quantization.

2. View the results.
 - The demo shows the scope results after a full-accuracy simulation. It runs full floating-point simulation, overriding the previous fixed-point settings. The spectrum scope waveforms have one waveform superimposed over the other. If the spectrum waveforms exceed the graphs, use Axes->Autoscale to fit them.



- The demo shows the following result for the time domain scopes:.



3. Double-click Next.

Analyzing the Impact of Quantization

When you double-click Next after the demo runs floating-point simulation, the following changes occur:

- The FIR dialog box opens with new settings.
- The Fixed-Point Settings toolbox displays new settings.
- The input and output frequency scopes are updated with new data.

- The input and output time scopes show new data.

At this point, the demo deliberately “breaks” the algorithm. This lets you see the process used to analyze the effect of quantization and isolate any problems that might occur. The following steps describe the process.

1. In the FIR dialog box, note the following changes:

- Coefficient fraction length is reduced.
- Output word length has changed.
- Output fraction length has changed.

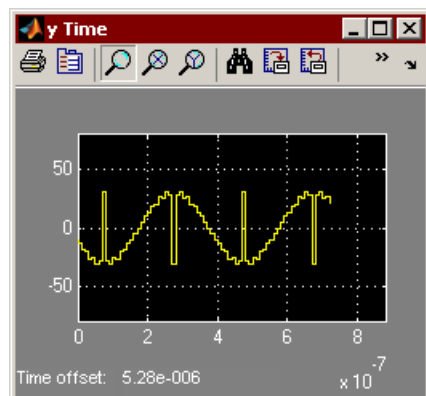
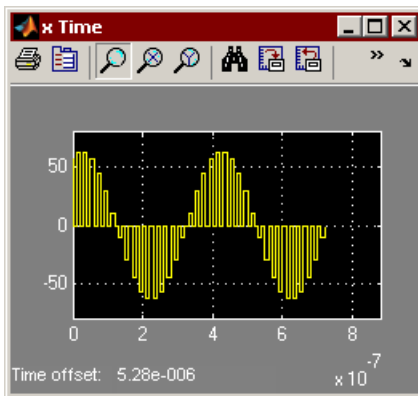
These settings help isolate and analyze quantization problems by deliberately breaking the algorithm.

2. In the Fixed-Point Settings toolbox, note the following changes:

- Logging Mode is set to Overflow Only and the amount of overflow is logged. This mode is not supported in MATLAB 2007A or 2007B.
- Data type override is set to Use Local Settings.

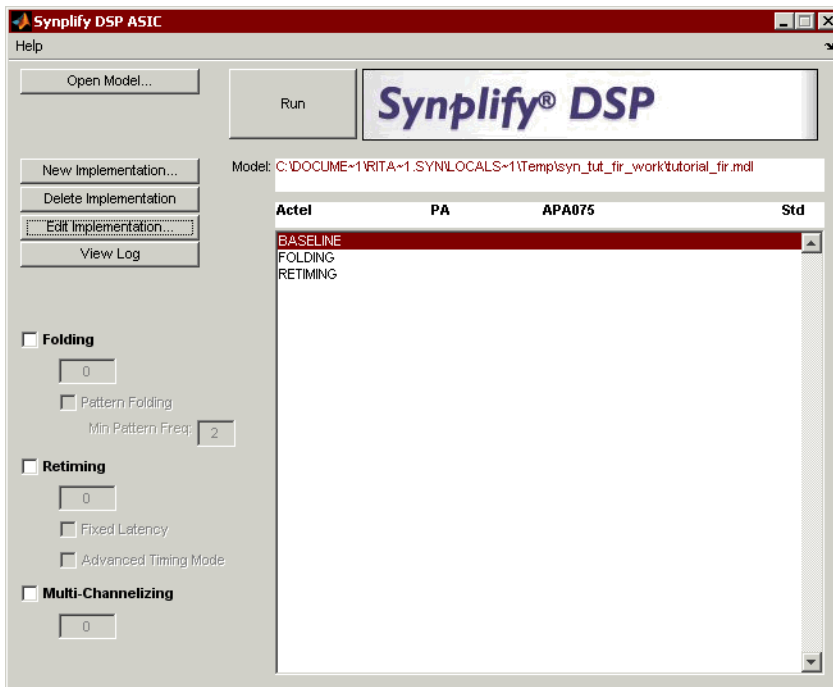
3. Check the results.

- The time scopes show how the quantization affects the output.



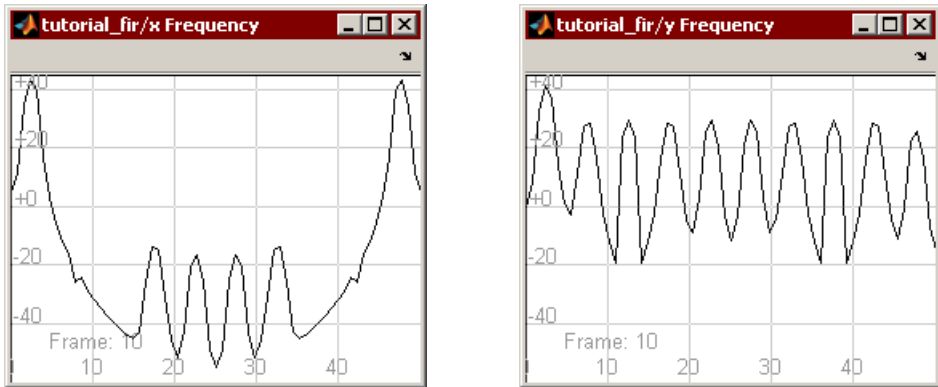
- The frequency scopes reflect similar results.

The demo does this automatically and displays a window, customized for the target technology you chose at the beginning of the tutorial. The following figure shows the window with an Actel target.

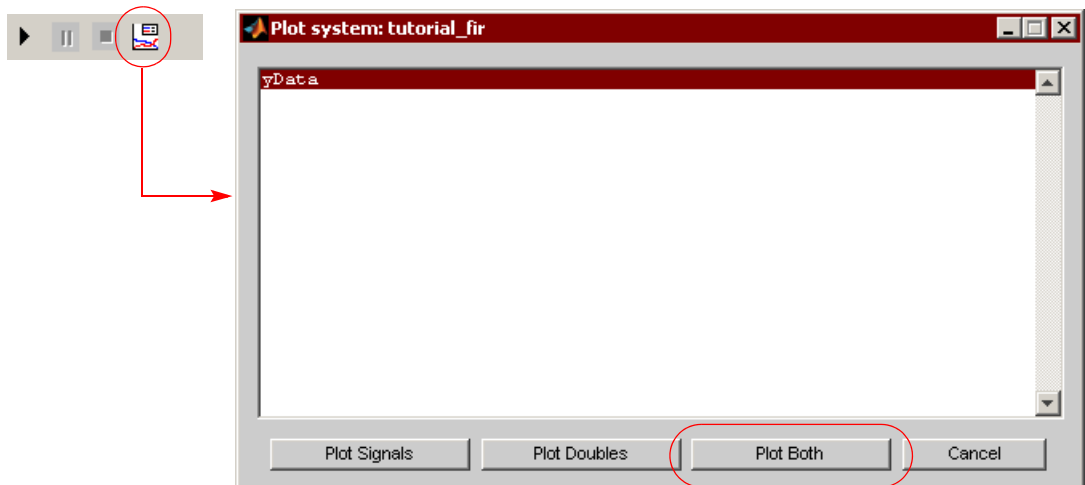


It also shows three implementations (Baseline, Folding, and Retiming) it has created. Each implementation explores different optimization strategies for the same design and stores it in a separate implementation. The implementation is a subdirectory, parallel with the .mdl file associated with the design, and contains any files generated for that particular implementation. The following steps describe the process that the demo ran through automatically.

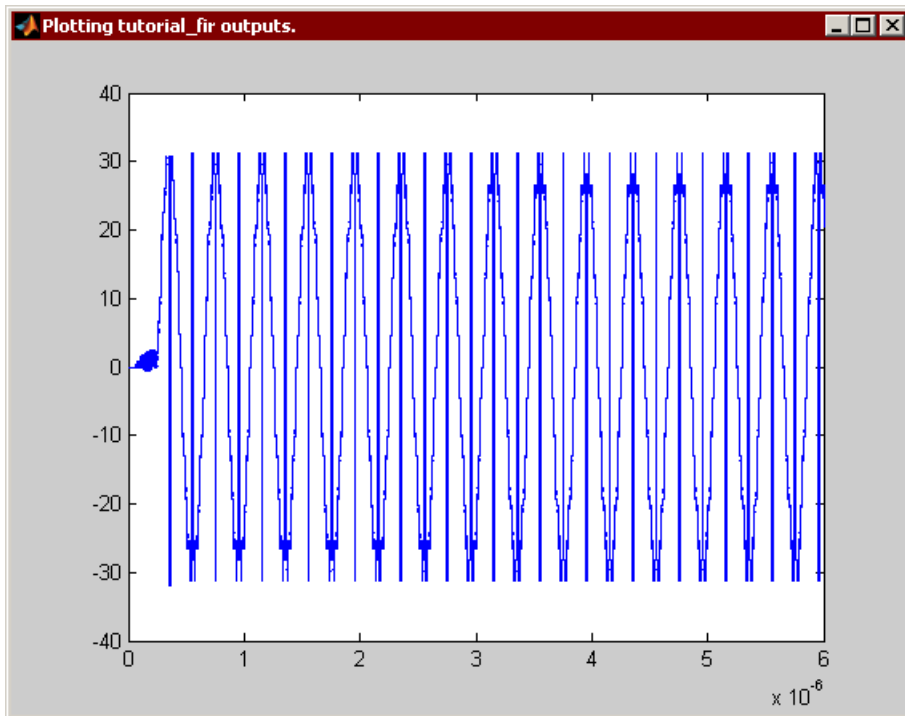
- The demo first set up the implementation and implementation options. You can review the steps by doing the following.
 - Select **BASELINE** and then click **Edit Implementation**. If you were trying to create a new implementation, you would click **New Implementation**. Either of these actions opens the **Implementation Options** dialog box, where you can set options specific to that implementation.
 - Check the settings. The following shows the settings for the Actel demo, so an Actel part and technology is selected on the **Target Options**



4. Plot and compare the waveforms. In MATLAB 2008A, you have to store the signals as reference signals to compare them.



- Check the plotted waveforms.



5. Double-click Next.

Synthesize Optimized Architectures

For more information about the next stages in the flow, see the following:

- [Run DSP Synthesis, on page 8-25](#)

- [Verify RTL, on page 8-29](#)

The tutorial skips this optional step, but you can refer to [Verifying the RTL with a Test Bench, on page 2-76](#) for a detailed procedure.

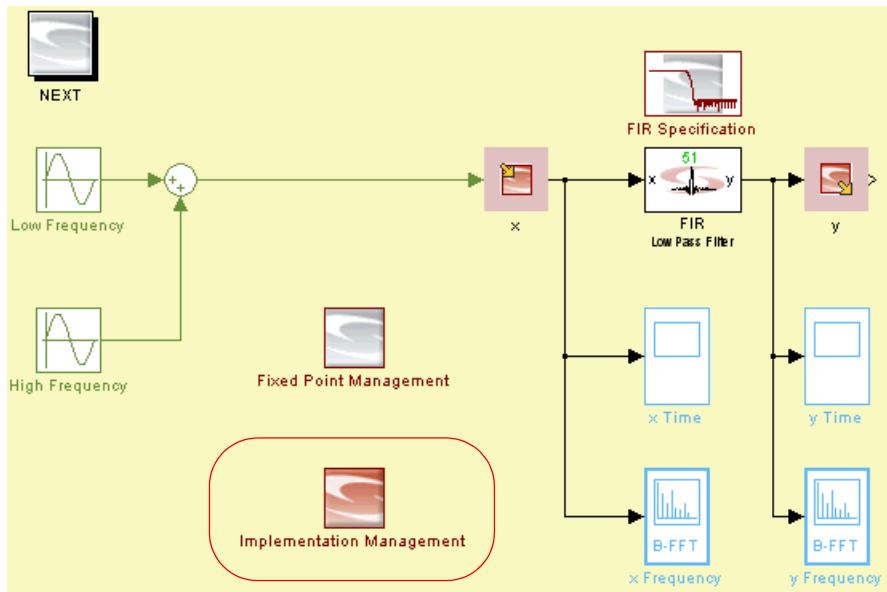
Run DSP Synthesis

When you double-click Next after exploring quantization effects, the demo resets the fixed-point settings and runs synthesis. You see the following changes:

- The model window now includes the SynDSPTool block.
- The Synplify DSP FPGA window (for DSP synthesis) is open.
- The Synplify Pro UI (for logic synthesis) is open.

You manage optimization strategies with the SynDSPTool block. The following steps describes how the demo instantiates and uses this block and then runs DSP synthesis.

1. The model window shows an instance of the SynDSPTool block from the top-level Synplify DSP library. In the demo, it is renamed Implementation Management.

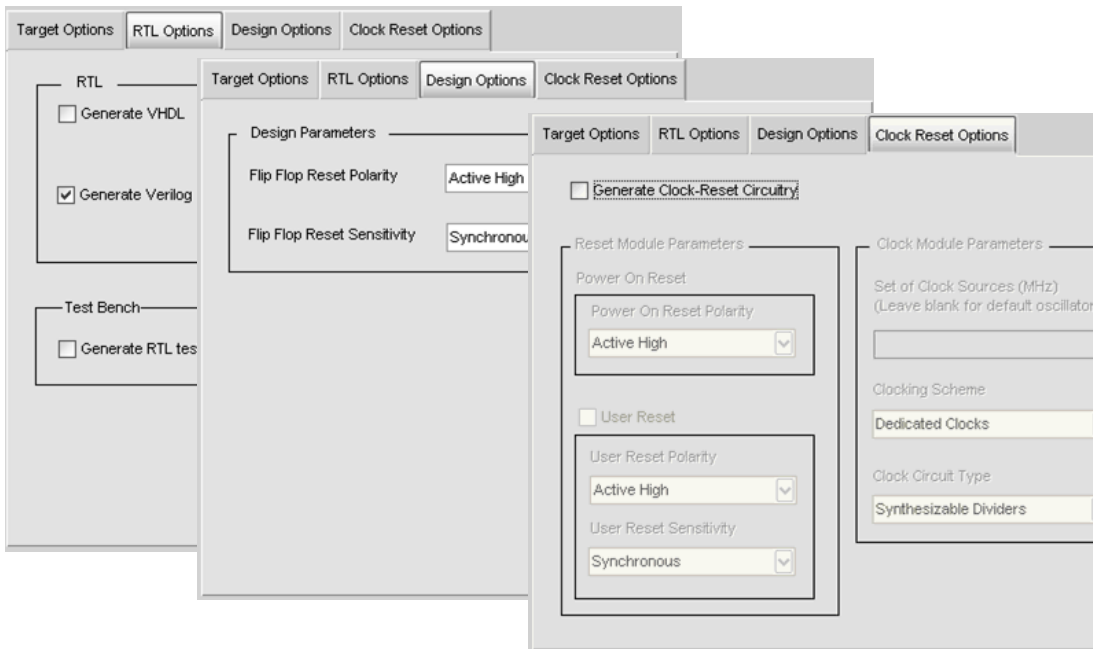


2. Double-clicking Implementation Management opens the Synplify DSP FPGA window.

tab. The selected target is also reflected in the Synplify DSP window, just above the implementations.



- Check the settings on the other tabs:

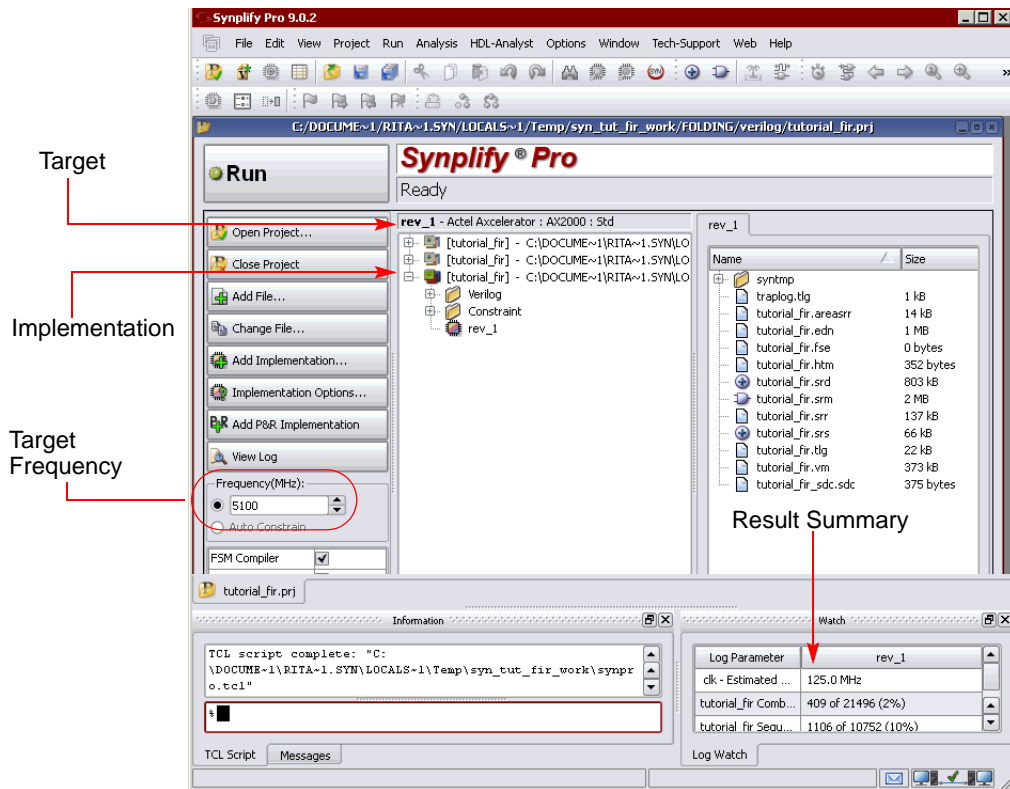


- Click Cancel to close the dialog box.
- Return to the Synplify DSP window and note that no optimizations, like retiming and folding, have been enabled for this implementation.

The other implementations have different settings, which we will explore later. The details are described in [Refine Optimizations, on page 8-31](#).

4. Next, the demo automatically runs DSP synthesis and generates output files. You do not need to do this because this has already been done, but to replicate this step manually, you would select BASELINE in the Synplify DSP window and click Run.
5. Click View Log in the Synplify DSP FPGA window to see a summary of the DSP synthesis run. Close the log window.

The next step, to verify the RTL, is optional, and this tutorial does not do this, but goes on to logic synthesis ([Run Logic Synthesis, on page 8-29](#)).



2. View the implementation.
 - Open the RTL view by clicking on the icon.

Verify RTL

This is an optional step, and the demo does not include it. For a detailed procedure for verifying the RTL, see [Verifying the RTL with a Test Bench, on page 2-76](#).

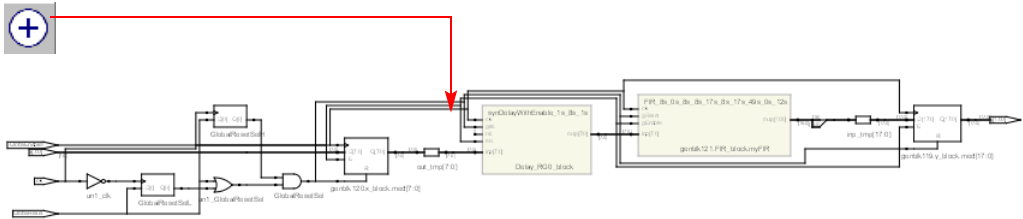
Run Logic Synthesis


After DSP synthesis, the demo automatically starts Synplify Pro and runs logic synthesis on the design. It displays the Synplify Pro window with the three implementations and their results. This section walks through the procedure step-by-step, using the BASELINE implementation.

As a result of DSP synthesis, the following files are generated for logic synthesis in the <design_implementation>/vhdl or verilog subdirectory:

File	Description
<design>.sdc	Synopsys FPGA Design Constraints generated for the design.
<design>.prj	Synopsys Project File generated for the design.
<design>.vhd or .v	The RTL associated with the design.

1. The demo automatically ran Synplify Pro. Examine the results of logic synthesis for the BASELINE implementation by doing the following:
 - In the Synplify Pro project window, select the BASELINE implementation.
 - Note that logic synthesis was run with the same FPGA target you selected. This figure shows an Actel implementation.



- Push down into the FIR module by clicking the  icon and selecting the FIR. View the implemented architecture.

The structure reflects a transposed implementation of the FIR filter: the input goes to different multipliers with each multiplier feeding two different adders (this is a linear phase filter with symmetric coefficients, and the identical coefficients share a multiplier). The adders are registered and accumulated for the final result.

- Close the RTL view.

3. Return to the main Synplify Pro window and check the results summary in the Log Watch window at the lower right. Compare the results to the target frequency.

Note that the results documented here may vary from your results if you used another target or another version of Synplify Pro. The other implementations in the demo illustrate how you can use Synplify DSP optimizations to produce better logic synthesis results. See [Refine Optimizations, on page 8-31](#) for details.

Refine Optimizations

The demo uses the other implementations to illustrate optimization strategies. In your design cycle, you can iterate with different implementations to fine-tune your design or try out different options and strategies.

This section describes the optimization strategies available and then walks you through using some techniques to improve performance and area optimization in the tutorial design:

- [Optimization Strategies, on page 8-32](#)
- [Using Retiming for Performance, on page 8-33](#)
- [Using Folding to Decrease Area, on page 8-34](#)

Optimization Strategies

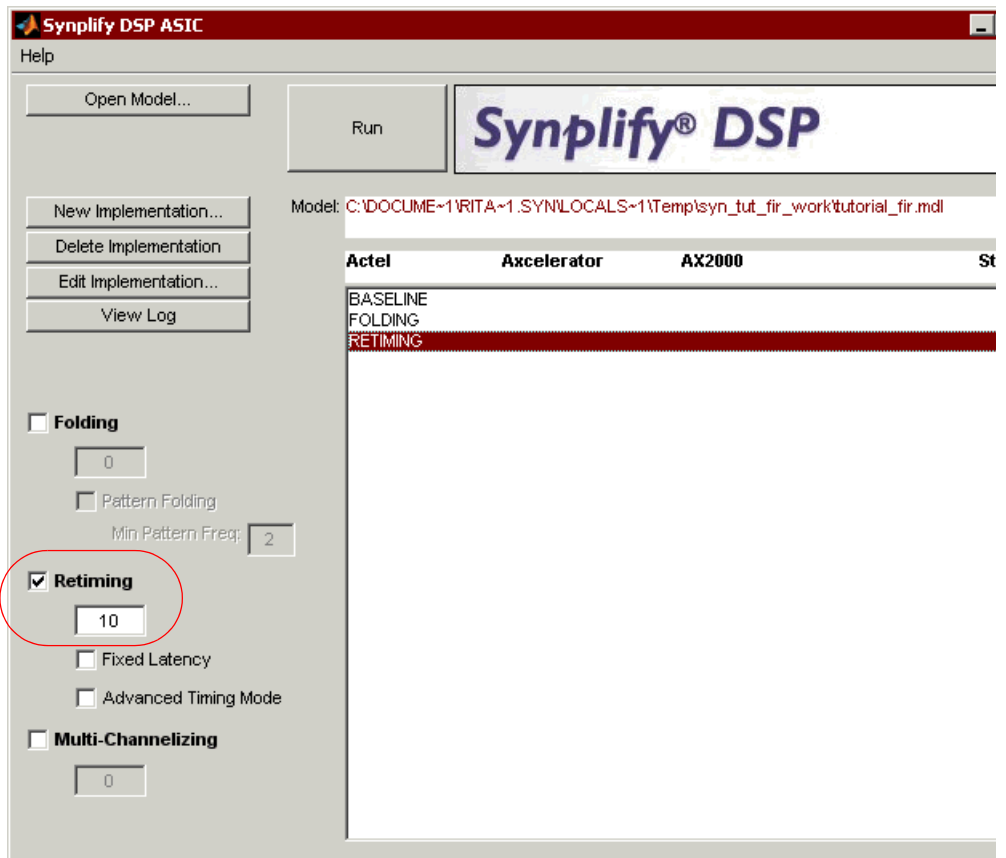
The Synplify DSP software offers the following optimization strategies:

- **Retiming**
Moves existing registers from non-critical to critical performance situations. Optional extra latency for the complete block adds extra register resources for pipeline insertion. The tutorial illustrates this technique in [Using Retiming for Performance, on page 8-33](#).
- **Multi-Channelization**
Multiple data streams share hardware for area optimization. This strategy requires the physical clock for the implementation to accommodate a clock rate equivalent to the sample rate of the individual data streams multiplied by the number of streams sharing the hardware. The tutorial does not illustrate this, but you can refer to [Optimizing with Multichannelization, on page 2-70](#).
- **Folding**
A single data stream shares hardware for area optimization. This strategy requires the physical clock for the implementation to accommodate a clock rate equivalent to the sample rate of the data stream multiplied by the requested folding factor. Folding requires retiming (to bring registers to the folding boundaries). The tutorial illustrates this technique in [Using Folding to Decrease Area, on page 8-34](#).

Using Retiming for Performance

The following procedure shows you how the demo used retiming to improve performance. It automatically created an implementation called RETIMNG

1. Return to the Synplify DSP window and select the RETIMNG implementation.
2. Note the following:
 - The RETIMING option is set. The following figure shows the Actel implementation.
 - Click View Log and check the file. You see that DSP synthesis was run with this option on and the specified number of latency cycles.



3. Go to the Synplify Pro view and select the Retiming implementation in that window.

The window is updated with the relevant data after the logic synthesis run for this implementation.

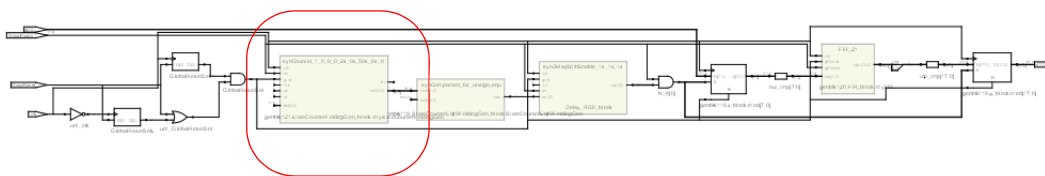
4. Check the following:
 - Check the Log Watch window in the lower right. You see that timing frequency has improved from the BASELINE implementation.
 - When you examine the architecture in the RTL view (see [Run Logic Synthesis, on page 8-29](#) for details), you see that the structure still reflects a direct-form, transposed implementation of the FIR filter. The input of the filter and the outputs of the multipliers are now all registered, and this results in improved timing performance.

Using Folding to Decrease Area

To deal with area challenges, use folding. Folding executes the hardware with the physical clock running at a multiple of the sample clock.

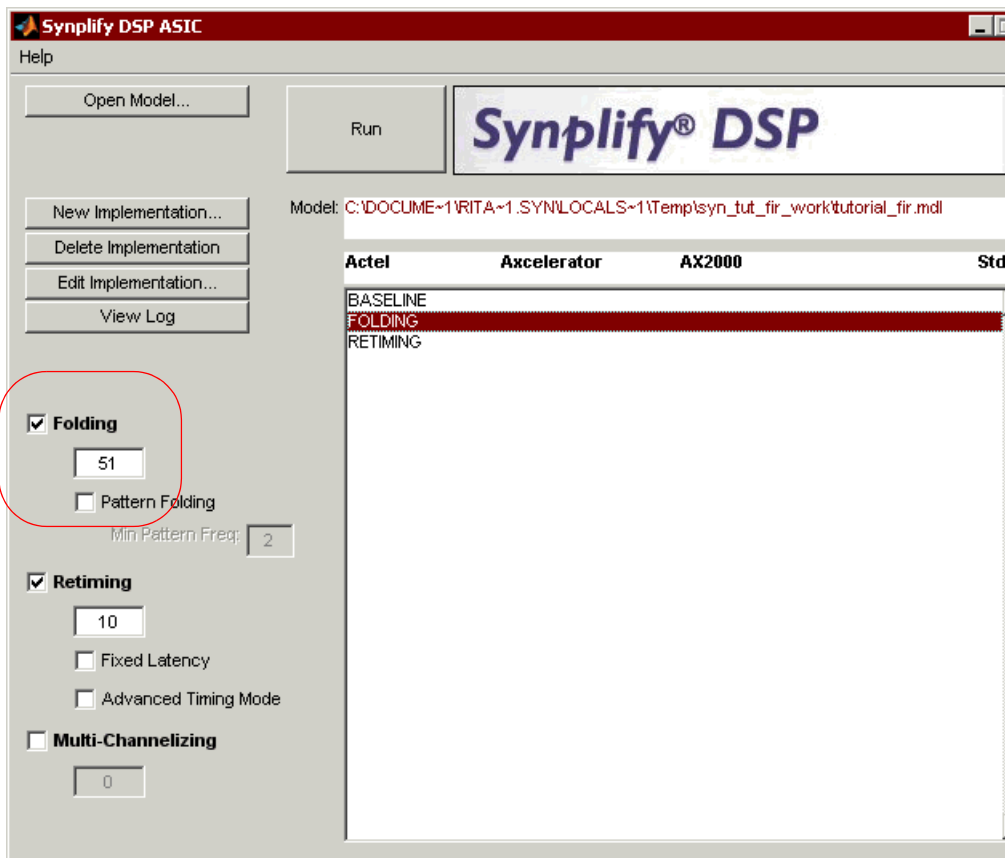
1. Return to the Synplify DSP window and select the FOLDING implementation. The following figure shows the Actel implementation.

- When you check the Log Watch window in the lower right, you see that the resources (number of cells) has been significantly reduced, compared to the BASELINE and FOLDING implementations.
- When you examine the architecture in the RTL view (see [Run Logic Synthesis, on page 8-29](#) for details), you see that the structure still reflects a direct-form, transposed implementation of the FIR filter, but it now includes a counter, to manage the multiplexers over the shared resources.



This illustrates how resources are shared and implemented efficiently by the folding optimization. You can see the addressing logic for the coefficient ROM and the input RAM data storage. With this optimized architecture, not only does the design meet the target performance, but the area is substantially reduced too.

Now that you have completed the tutorial, you are familiar with the design flow, and can use Synplify DSP for your own designs.



2. Note the following:
 - The Retiming and Folding options are both enabled. Selecting Folding automatically enables Retiming.
 - Click View Log and check the file. You see that DSP synthesis was run with a folding factor of 51. This specifies that the physical clock can run 51 times faster than the sample clock to enable resource sharing.
3. Go to the Synplify Pro view and select the Folding implementation in that window.

The window is updated with the relevant data after the logic synthesis run for this implementation.
4. Check the following: