System Design Using ispLeverDSP

Technical Support Line 1-800-LATTICE or (408) 826-6002

Web Update To view the most current version of this document, go to www.latticesemi.com.

Lattice Semiconductor Corporation
5555 NE Moore Court
Hillsboro, OR 97124
(503) 268-8000

November 2008

Copyright

Trademarks

Disclaimers

latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of code.
`.` `.` `.`	Omitted lines in code and report examples.
`[ ]`	Optional items in syntax descriptions. In bus specifications, the brackets are required.
`( )`	Grouped items in syntax descriptions.
`{ }`	Repeatable items in syntax descriptions.
`|`	A choice between items in syntax descriptions.

Contents

System Design Using ispLeverDSP

Introduction

This tutorial shows you how to create system designs with LatticeECP-DSP devices, using the Lattice design blocks within the MATLAB® Simulink software. The tutorial addresses designers who are already familiar with system modeling and the Simulink environment as well as those who are new to DSP design and Simulink tools.

Setting Up ispLeverDSP

Before beginning the tutorial, you must have the MATLAB/Simulink software already installed and running with the ispLeverDSP software.

To install the Simulink software, follow the installation instructions that accompany the MATLAB/ Simulink software.

To use ispLeverDSP blocksets, you should manually add ispLeverDSP paths into MATLAB search path list as follows:

1. In the MATLAB startup window, choose **File > Set Path** to open the Set Path dialog box.
2. Add the following paths to the search path list:

   `<ispLEVER_install_path>\ispLeverDSP`
   `<ispLEVER_install_path>\ispFPGA\bin\nt`

Learning Objectives

Lesson 1 of this tutorial briefly explains basic DSP system design and guides you through the creation of a simple FIR filter using ispLeverDSP's Lattice blocks within the Simulink software. If you are already familiar with Simulink and DSP design, you can skip this lesson and go on to the more advanced design of Lesson 2.

After completing this tutorial you should be able to use ispLeverDSP and the Matlab/Simulink software to accomplish the following:

- Create a simple model such as a 2-tap FIR filter
- Define block parameters
- Add Simulink blocks and connect the circuitry
- Simulate the design
- Implement the design
- Develop and simulate a more complex model, such as a numerical controlled oscillator

Accessing Online Help

You can find online help information on any block used in this tutorial by pressing the Help button in the block mask.

To access the ispLeverDSP Help, you need manually copy help files from ispLEVER directory into MATLAB directory as follows:

1. Browse to the <*ispLEVER_install_path*>\ispLeverDSP directory.
2. Highlight and copy the **help** folder.
3. Inside the MATLAB directory, paste the **help** folder on top of the existing help folder.
4. In the Confirm Folder Replace dialog box, choose **Yes to All**.

Prerequisites

Before beginning the tutorial, you must have the Matlab/Simulink software already installed. To install the software, follow the installation instructions that accompany the Matlab Simulink software and the ispLeverDSP software.

You should be familiar with the FPGA design process before beginning the tutorial. To learn about the FPGA design process, see the FPGA and Crossover Design Flow section of the ispLEVER online Help.

Model Examples

A directory of model examples is provided with ispLeverDSP and is available at the following location:

<*ispLEVER_install_path*>\ispLeverDSP**examples**

To view the examples for each of the lessons of this tutorial, open the following files in the examples folder:

- DSP_tutorial1.mdl
- DSP_tutorial2.mdl

Note

You can backup the files into another directory, so that other users can use them later.

Lesson 1: Building a Simple Model

This first lesson briefly explains DSP design and walks you through the basic steps of designing with DSP using the Matlab Simulink software.

DSP Design

The design of a DSP system usually starts with the following:

- High-level models that attempt to model real-world events and effects
- Processing blocks to perform the desired system operation
- Models to manipulate the results and convey the resulting information in a manner that most easily allows you to determine if the system meets your requirements.

As a designer, you would typically modify the system to conform to rules that accommodate real-world devices at real-world system costs. Depending on the targeted hardware, this can involve several different design environments. Often you must modify the design along the way to meet unforeseen constraints. Afterwards you must take this new information back into the original design environment, modify and re-verify the system operation, and then follow the design process back through to the point where the original problem was encountered.

The Simulink Modeling Environment

The Simulink modeling software, in conjunction with ispLeverDSP, allows you to take a real-world application and convert it into a fixed-point system that you can fit into your FPGA. It allows you to complete the entire design process in a single environment. This greatly enhances productivity, reduces errors, and allows more what-if scenarios for quickly determining the optimal solution. It results in faster time to market and a lower system cost. Such an environment is valuable for solving those system design issues that result from incompatible tool flow.

Task 1: Create a New Model

To begin the design process, you will create a blank model window in Simulink and open the Lattice Blockset library.

1. Start the Matlab software, using the instructions in the Mathworks documentation.
2. If you haven't already done so, choose **File > Set Path** to add the following ispLEVER paths to the Set Path dialog box.

   `<ispLEVER_install_path>\ispLeverDSP`
   `<ispLEVER_install_path>\ispFPGA\bin\nt`
3. Choose **File > New > Model** to open a blank model window.
4. Save the model in a directory by choosing **File > Save**. The model must be saved and named before the design can be used.
5. Set MATLAB Current Directory to point to the directory where the design is located.
6. Select the Simulink icon from the toolbar, or type **`simulink`** in the Matlab Command window. This opens the Simulink Library Browser.
7. In the Simulink Library window, double-click **Lattice Blockset** to open the library and display its contents.
8. Position the empty model window so that it is adjacent to the Simulink Library window. Adjust the size of the windows as needed by dragging the sides or corners.

Task 2: Add Lattice Blocks and Define Inputs and Outputs

In order to produce a design that implements your system in a LatticeECP FPGA device, you must include the Lattice Generator block, which serves as a constant source. You must also define the inputs and outputs for the device. You accomplish this by using the Gateway In and Gateway Out blocks.

1. From the Basic Elements, select the **Lattice Generator** block. Holding the left mouse button, drag the block to the blank model window and release the mouse button. The Generator will not be connected to any other blocks but will serve as a constant source block.

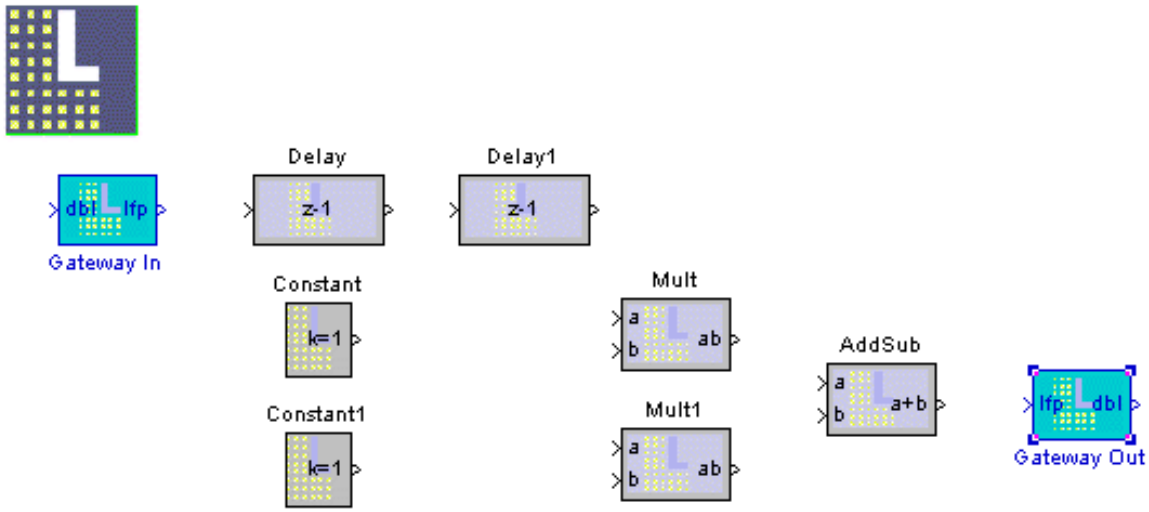
   
2. Select the **Gateway In** block. Holding the left mouse button, drag the block to the model window and release the mouse button.
3. Add the following blocks to the model window following the Lattice Gateway In block:
   - Two Lattice Delay blocks
   - Two Lattice Constant blocks
   - Two Lattice Multiplier (MULT) blocks

- One Lattice Adder/Subtractor (AddSub) block

Note

To copy a selected block inside the model window, hold the Control key as you drag the block to a new position, and then release the mouse button.

4. Add a Lattice Gateway Out following the Adder/Subtractor block. Drag the corner of the window, if needed, to enlarge it to fit the layout. Your model window should resemble the one as follows:

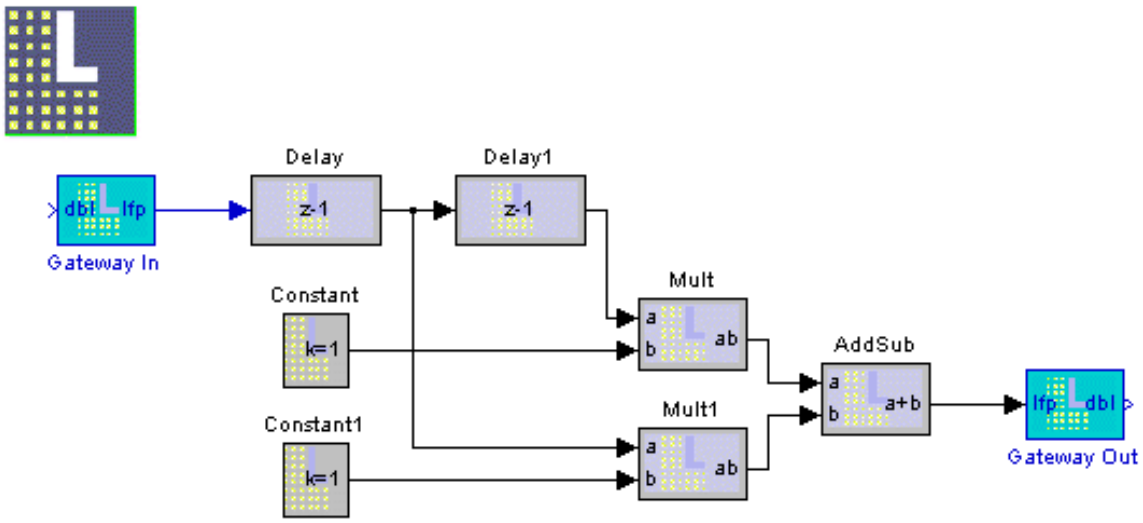


You now have all of the blocks necessary to make a simple 2-tap FIR filter. Any signal that passes through a Gateway In becomes an input to the final FPGA device; any signal that passes through a Gateway Out becomes an output. All blocks that are placed between the Gateway blocks must be Lattice blocks and are targeted to a hardware implementation. Blocks located outside the Gateways can be non-Lattice blocks, since they do not impact the hardware. Such outside blocks can provide stimulus for the hardware and a method for analyzing the results.

Task 3: Connect the Lattice Gateways and Blocks

To connect the blocks in the model window, you can use your mouse to draw a connection from port to port or you can use the Ctrl+select shortcut as follows:

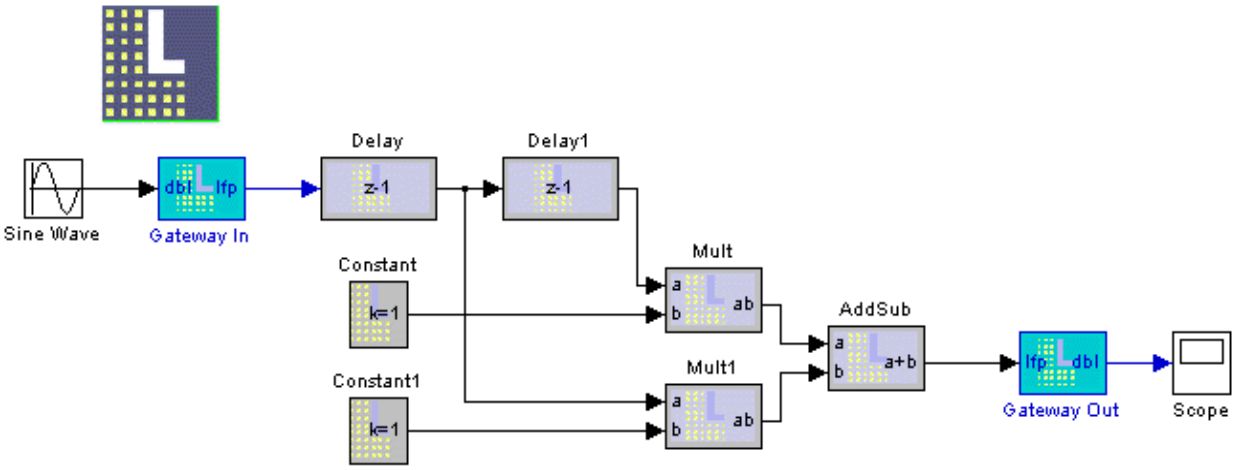
1. In the model window, click the **Lattice Gateway In** block to select it.
2. While holding the **Ctrl** key, select the adjacent Lattice Delay block. The two blocks are now connected.
3. Connect the remaining blocks, as shown in the following image.



Task 4: Add Simulink Blocks

In this task, you will add Simulink blocks outside the Lattice Gateways.

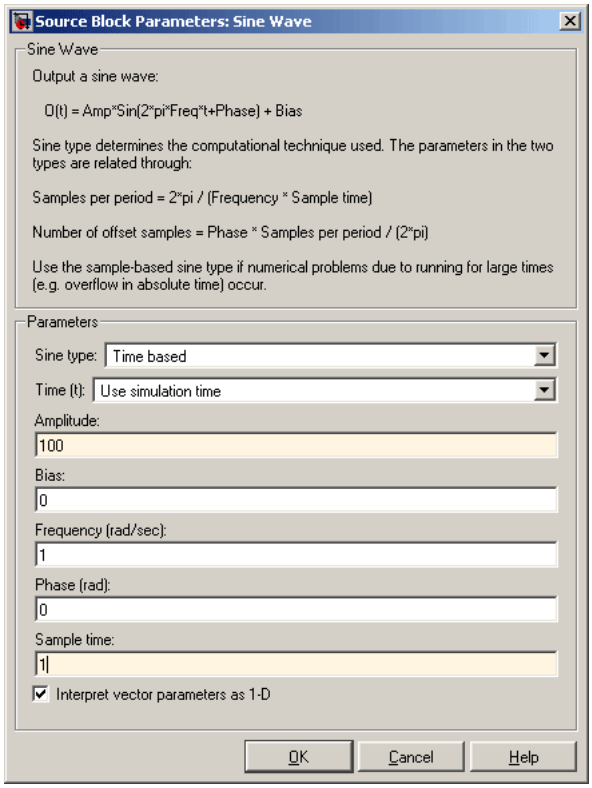
1. In the Simulink Library Browser, open the Simulink > Sources library, and then select **Sine Wave**.
2. Place the Sine Wave in the model window and connect it to the Gateway In block.
3. Open the Simulink > Sinks library, select **Scope**, and connect it to the Gateway Out block. Your layout in the model window should now resemble the following image.



Task 5: Specify Sine Wave Characteristics

In this task, you will set the parameters of the sine wave source, which will provide the stimulus to the system.

1. Double-click the **Sine Wave** block to open the Block Parameters dialog box.
2. Enter **`100`** for Amplitude, 1 for Sample time, and accept the defaults for the other parameters.



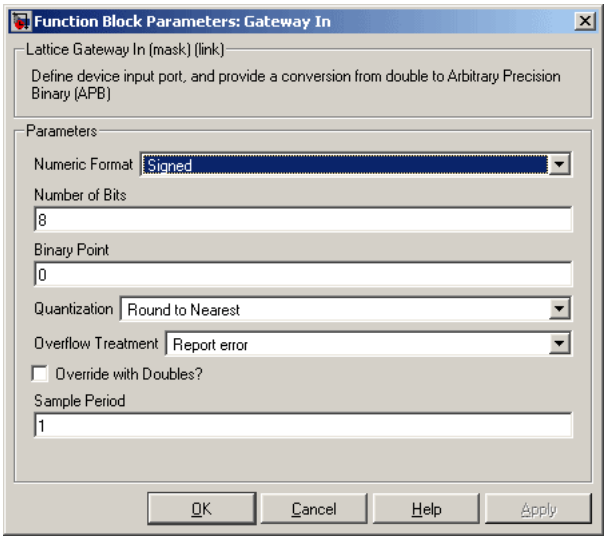
3. Click **OK** to apply the selected parameters and close the dialog box.

Task 6: Define the Precision

To implement a fixed precision system, you must specify the parameters. The Gateway In block can be parameterized to define the precision of each input. Depending on the system, this might be all that is required, since the other blocks inherit this precision by default.

1. Double-click the **Gateway In** block to open the Block Parameters dialog box.

   The default data width and type is 8 bit signed, which is sufficient for the FIR filter.



2. Click **OK** to accept the defaults and close the dialog box.

   Now you will set the coefficients for the FIR filter.

3. Double-click the first Lattice Constant block, and type **1** in the Value box.

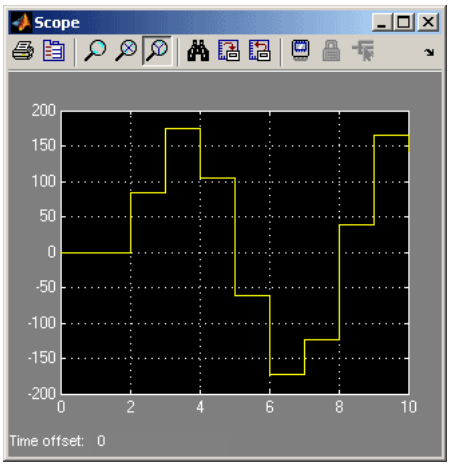
   Since a constant is also an input, it must be set to the desired bit width and numeric format. For your FIR filter, the Value of 1 is sufficient.

4. Click **OK** to apply the parameters and close the dialog box.

5. Double-click the second Lattice Constant block, set the Value to **1**, and click **OK**.

Task 7: Simulate the Design

To simulate the design, choose **Simulation** from the toolbar, and then select **Start**. Double-click the Scope block. A scope window opens and displays the results of the simulation.

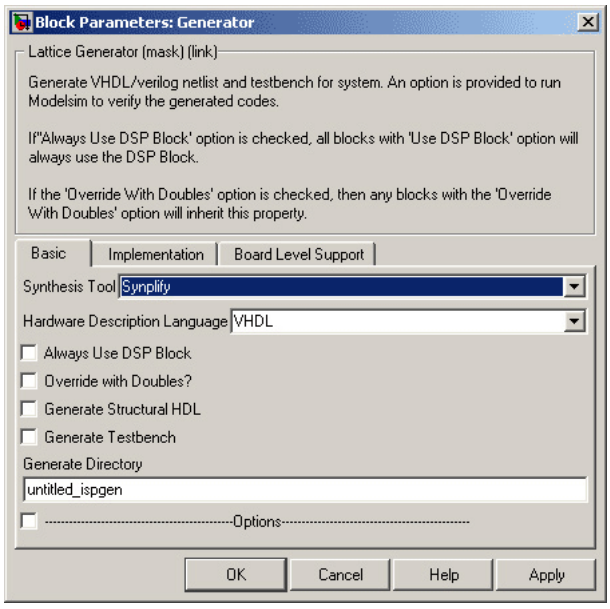


You now have a functional design, complete with a stimulus and a verified response.

Task 8: Generate and Verify the VHDL Code

In this task, you will proceed with the hardware implementation phase.

1. Double-click the **Lattice Generator** block to open the Block Parameters dialog box.



2. In the dialog box, select the checkboxes **Generate Structural HDL** and **Generate Testbench**. Selecting the latter opens up the Simulation Tool menu.

3. From the Simulation Tool drop-down menu, choose **Active-HDL GUI**.

   Notice that the data field for Generate Directory shows “untitled_ispgen,” This is the default name that the Lattice Generator inherits since no other name has been specified. You can modify the directory name or leave it as it is. If you wish to explore other directory naming options, select the Options checkbox.

4. Click **OK** to close the dialog box.
5. Choose **Simulation** from the toolbar and select **Start**.

The simulation runs again. An Active-HDL process starts in the Active-HDL GUI. At the bottom of the console window you should see indications that the simulation has terminated. There should be no error messages prior to this message.

Because of the options selected, after the stimulus and response as seen by the Gateway, a block has been saved from the Simulink simulation. A VHDL netlist has been generated, and a testbench has been created to utilize the stimulus and response information. The simulation was then started and the function of the netlist was compared to that of the Simulink function. No errors were detected. A VHDL implementation of the design has been created and verified.

Target the Design to an FPGA Device

You have now completed the Simulation portion of the design. The next and final steps are to synthesize the VHDL code and run the output through the FPGA place and route process. All of the required code is saved in the directory you chose in the Lattice Generator block menu.

Follow the directions in the ispLEVER instructions to synthesize and place and route the design. Include all files except the testbench file named `<design_name>_tb.vhd`.

Tutorial Example

To view the model example for this lesson, see

<ispLEVER_install_path>\ispLeverDSP\examples**DSP_tutorial1.mdl**

Lesson 2: Building a Complex Model

In this lesson, you will use ispLeverDSP and Lattice library blocks to build and test a Numerically Controlled Oscillator (NCO) within the Matlab Simulink environment. The NCO will be a 1/8 wave type. To successfully complete this lesson, you should be familiar with the Simulink software and the basic concepts of DSP design.

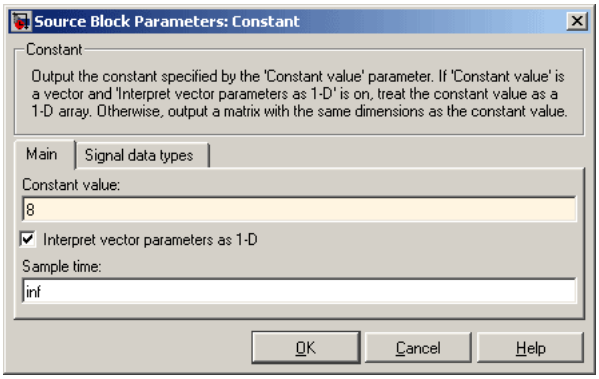
Before you begin this lesson, you must have the Matlab Simulink software installed and running.

Task 1: Begin the Accumulator

In this task, you will add the Lattice blocks to build an accumulator and make the connections in the Simulink design model window.

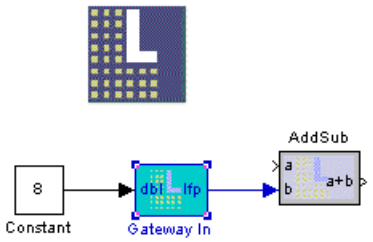
1. In the Simulink Library Browser, choose **File > New > Model** to open a blank model window.
2. Save the model in a directory by choosing **File > Save**. The model must be saved and named before the design can be used.
3. Set MATLAB Current Directory to point to the directory where the design is located.
4. Type **`simulink`** in the Matlab Command window to opens the Simulink Library Browser.
5. In the Simulink Library window, double-click **Lattice Blockset** to open the library and display its contents.
6. Select the **Lattice Generator** block, drag it to the model window, and position it off to the side.

7. In the Browser, select **Sources** from the Simulink library, and drag the Constant block to the model window.
8. Double-click the Constant block to open the Block Parameters dialog box. Enter a value of **8** under Constant value, and select the checkbox **Interpret vector parameters as 1-D**.



9. Click **OK** to apply the parameters and close the dialog box.
10. From the Lattice Blockset, select the **Lattice Gateway In** block and drag it to the model window. Connect its input to the output of the Constant block.

    The Gateway In is necessary for changing the double precision of the Matlab Constant block to fixed-point precision.
11. Double-click the **Gateway In** block to open the Block Parameters dialog box. In the dialog box, set the following parameters:
    - Numeric Format: Unsigned
    - Number of Bits: 12
    - Binary Point: 0
    - Quantization: Round to Nearest
    - Overflow Treatment: Report error
    - Specify 1 for the sample period.
12. Drag the Lattice Adder/Subtractor (AddSub) block to the model window and connect it with the Gateway In output port.



13. Set the parameters for the Lattice Adder/Subtractor block as follows:
    - Mode: Addition
    - Precision: User Defined
    - Output Width: 12
    - Binary Point: 0
    - Numeric Format: Unsigned
    - Quantization: Truncate
    - Overflow Treatment: Wraparound
    - Latency: 1

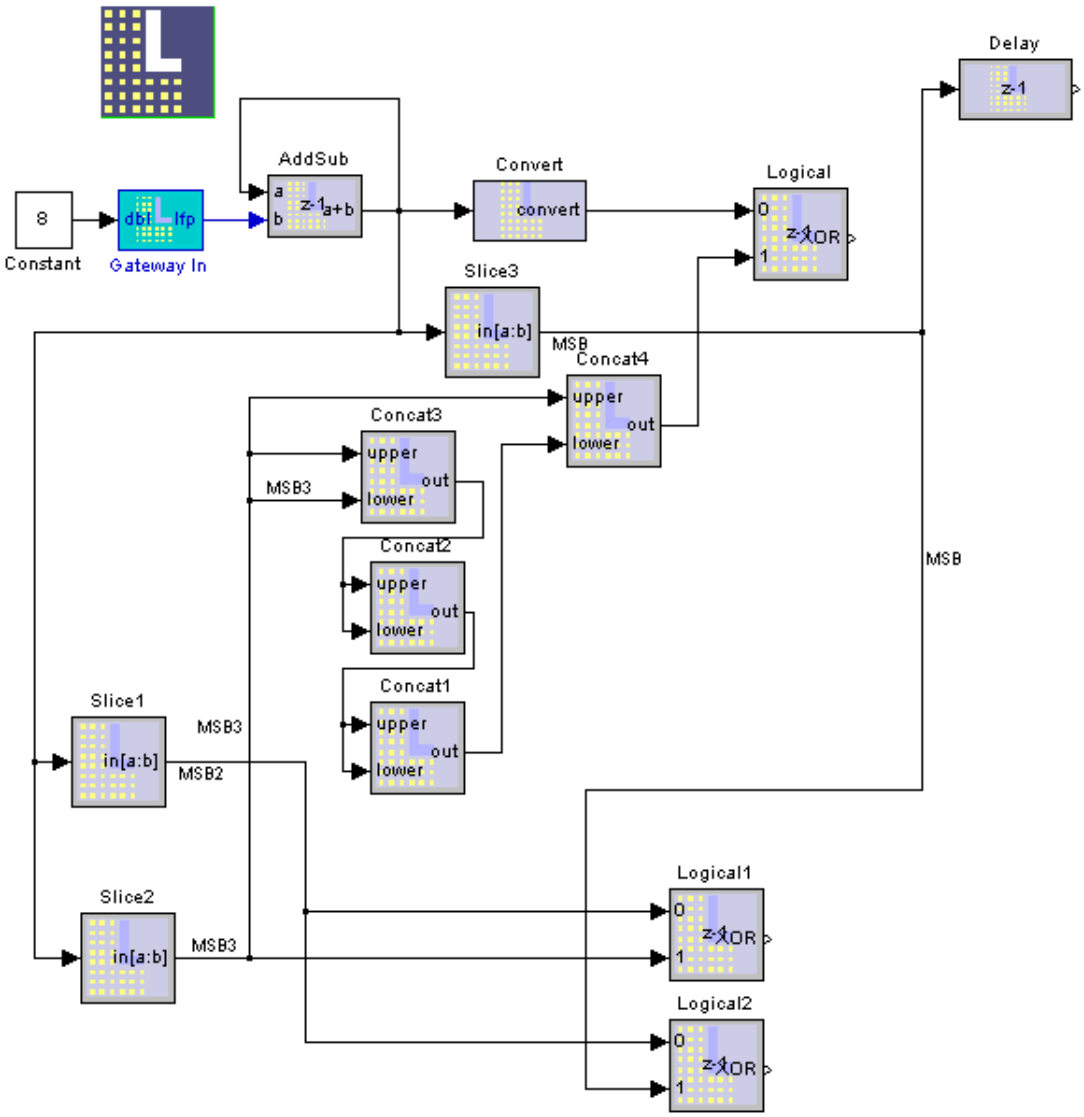
Task 2: Develop the Accumulator with Lattice Blocks

In this task you will add a series of Lattice Slices, Logical blocks, and Bus Concatenation blocks, in addition to one Convert block and one Delay block.

1. Drag a Lattice Convert block to the model window and connect its input port to the output port of the Adder/Subtractor.
2. Set the parameters for the Convert block as follows:

- Precision: User Defined
- Output Width: 9
- Binary Point: 0
- Numeric Format: Unsigned
- Quantization: Round to Nearest
- Overflow Treatment: Wraparound
- Latency: 0

3. Drag a Lattice Delay block to the model window and accept the default parameters.
4. Drag the following blocks to the model window:
   - Three Lattice Logical blocks
   - Four Lattice Bus Concatenation (Concat) blocks
   - Three Lattice Slices
5. Arrange the blocks in the model window and connect them so that the layout resembles the one shown below.



6. To add names on connecting lines, double-click the line and type the name in the text box that appears.
7. Set the parameters for Lattice blocks Logical 1 and Logical 2 as follows:
   - Function: XOR
   - Number of inputs: 2
   - Precision: User Defined
   - Output Width: 1
   - Binary Point: 0
   - Numeric Format: Unsigned
   - Quantization: Round to Nearest
   - Overflow Treatment: Saturate

- Latency: 1

8. For the block Lattice Logical, set the Output Width to 9; for the other parameters, use the same settings as the Logical 1 and Logical 2 blocks.
9. Accept the default parameters for all four Lattice Bus Concatenation blocks.
10. Set the following parameters for all three Lattice Slices:
    - Slice Location: Upper Integer Bits
    - Number of Bits: 1
    - For Lattice Slice 1, set the Offset to 1.
    - For Lattice Slice 2, set the Office to 2.
    - For Lattice Slice 3, set the Offset to 0.

The Accumulator

So far you have set up an accumulator (Adder/Subtractor) with unit delay feedback represented by the Latency of 1. This accumulator has wraparound capability so that the output goes back to almost zero each time maximum accumulation occurs that a 12-bit width can represent.

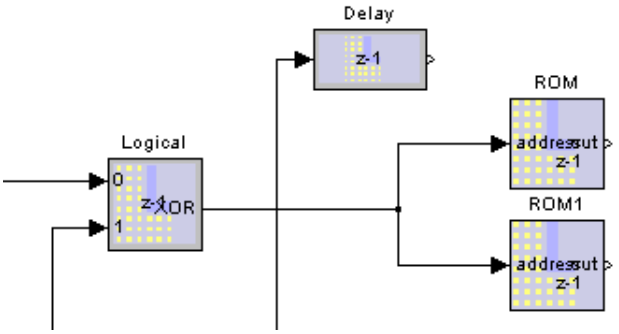
The third MSB from the accumulator's output is stripped off by Slice2 and goes through a sequence of four concatenation blocks that duplicate this MSB to give a 9-bit value of either all 1s or all 0s.

Task 3: Add the Remaining Circuit Blocks and Control Logic

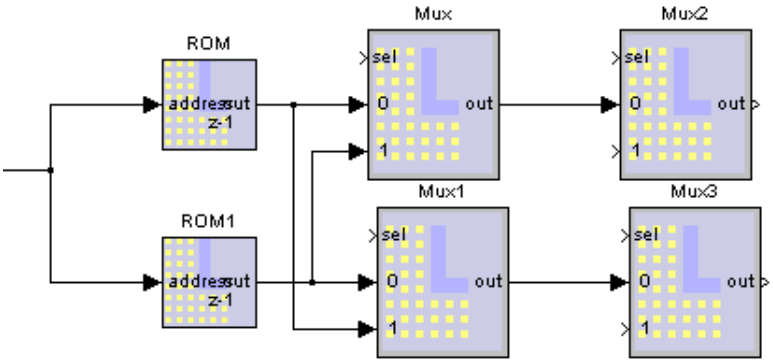
In this task, you will add and connect the remaining circuit blocks, consisting of SIN and COS LUTs and control logic that allows 1/8-wave implementation.

1. From the Lattice Blockset, select the **Lattice ROM** block, drag it to the right of the Lattice Logical block in the model window and duplicate it. Connect the output of the Lattice Logical to the inputs of each of the ROM blocks.

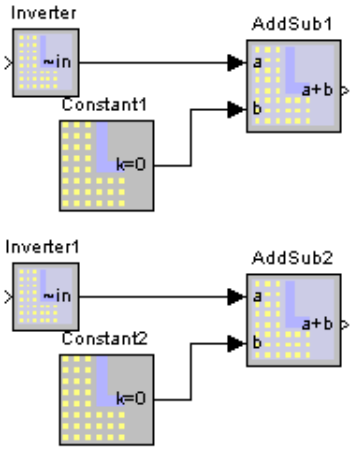
   .



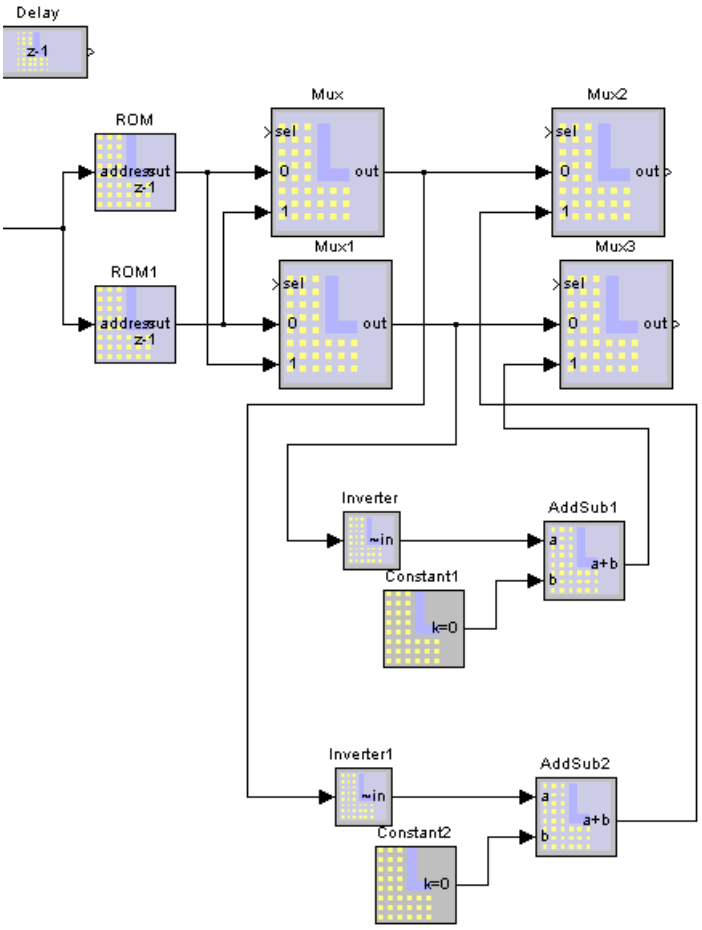
2. Select the Multiplexer (Mux) from the Lattice Blockset, drag it to the right of the Lattice ROM blocks and duplicate it three times. Connect the four multiplexors with the ROM blocks as shown:



3. Select a Lattice Inverter, a Lattice Adder/Subtractor (AddSub), and a Lattice Constant and drag them to the model window beneath the four Lattice Multiplexers. Duplicate the group and make the connections as shown:



4. Connect the Multiplexers with the Inverters and Adder/Subtractor blocks so that this part of the layout in the model window resembles the image below.



Now you will connect the circuits with those you created in Tasks 1 and 2.

5. Connect the two outputs from Lattice Logical 1 to **`sel`** on Multiplexer and Multiplexer1.
6. Connect the output from Lattice Logical 2 to Multiplexer3.
7. Connect the output from Lattice Delay to Multiplexer2.

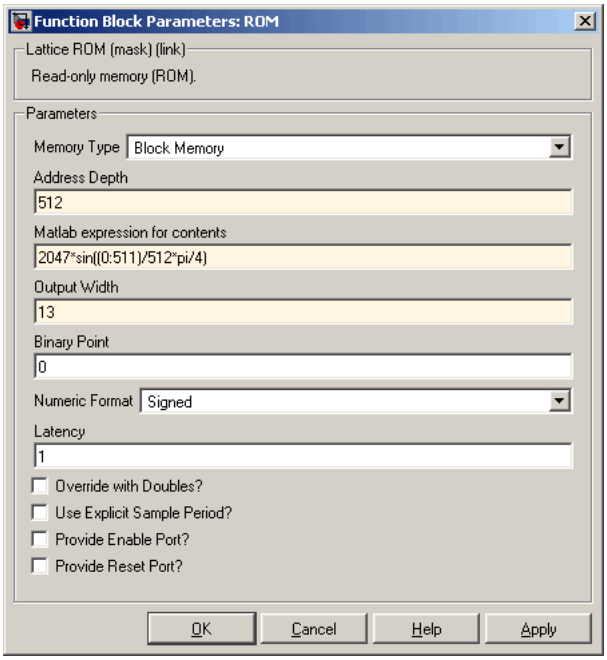
Note

To view these Multiplexer connections, see the full layout on page 21.

Task 4: Set the Block Parameters

In this task you will set the parameters for the blocks that you have added.

1. Set the parameters for the Lattice ROM block as shown in the image below:



2. For the Lattice ROM1 block, use the same parameters as Lattice ROM, but change the trigonometric function to a `cos`.
3. Set the parameters for all four Lattice Multiplexers as follows:
   - Number of inputs: 2
   - Precision: User Defined
   - Output Width: 13
   - Binary Point: 0
   - Numeric Format: Signed
   - Quantization: Round to Nearest
   - Overflow Treatment: Saturate
   - Latency: 0
4. Set the Latency of both Lattice Inverters to 0.
5. Set the parameters for Lattice Adder/Subtractor1 and Adder/Subtractor2 as follows:
   - Mode: Addition
   - Precision: User Defined
   - Output Width: 13
   - Binary Point: 0

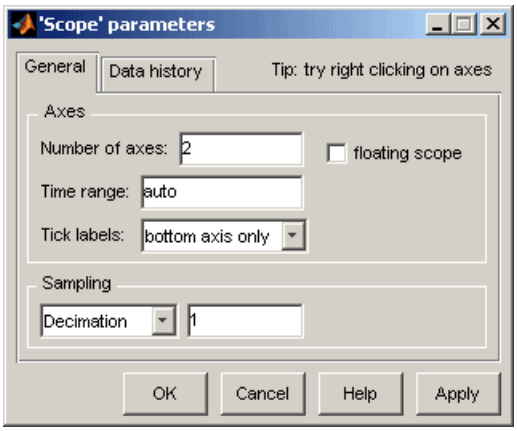
- Numeric Format: Signed
- Quantization: Round to Nearest
- Overflow Treatment: Saturate
- Latency: 0

6. Set the parameters for Lattice Constant Block1 and Constant Block2 as follows:
   - Value: 1
   - Output Width: 12
   - Binary Point: 0
   - Numeric Format: Signed

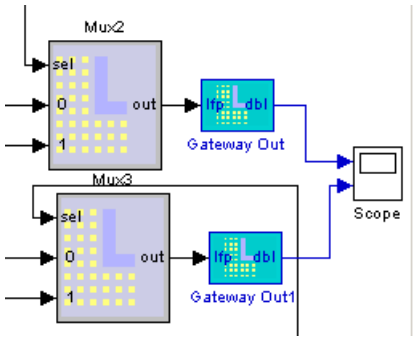
Task 5: Add the Gateways Outs and Scope

The outputs of Multiplexer2 and Multiplexer3 are the SIN and COS waves respectively. You now need to connect these two outputs to a Simulink scope. To do this, you must first convert these two outputs back to a double precision that the sink will accept.

1. Drag over two Lattice Gateway Out blocks and connect each of them to the Multiplexer outputs.
2. From the Simulink > Sinks library, select the **Scope** (not the floating scope) and drag it to the right of the Gateway Output blocks.
3. Double-click the Scope block and click the Parameters button to open the Scope parameter dialog box.
4. Set the general parameters for the scope as shown below:

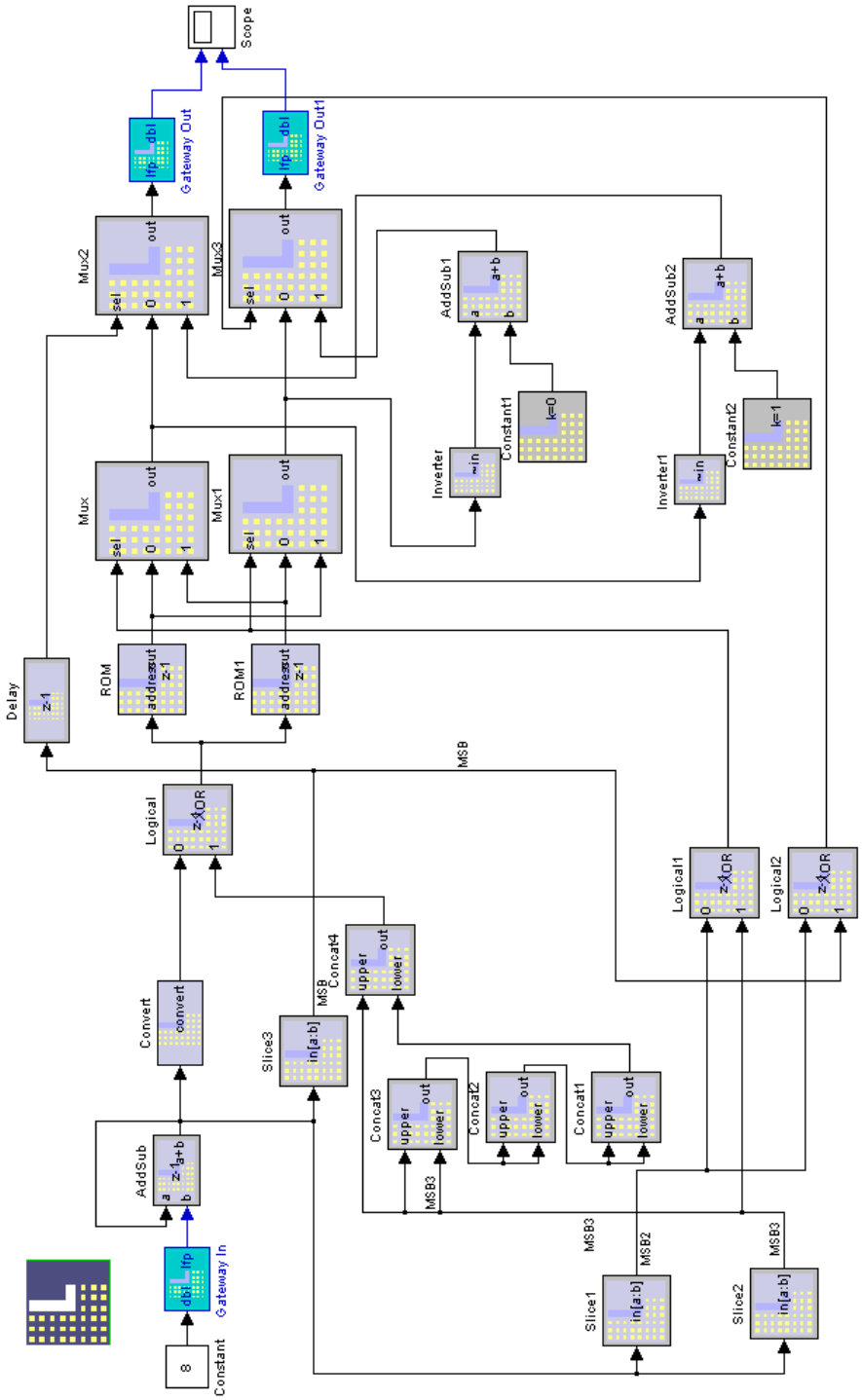


5. Connect the outputs of each of the Lattice Gate Out blocks to the Scope.



Now all the blocks should be in place and connected. Your model output window should resemble the image on page 20.

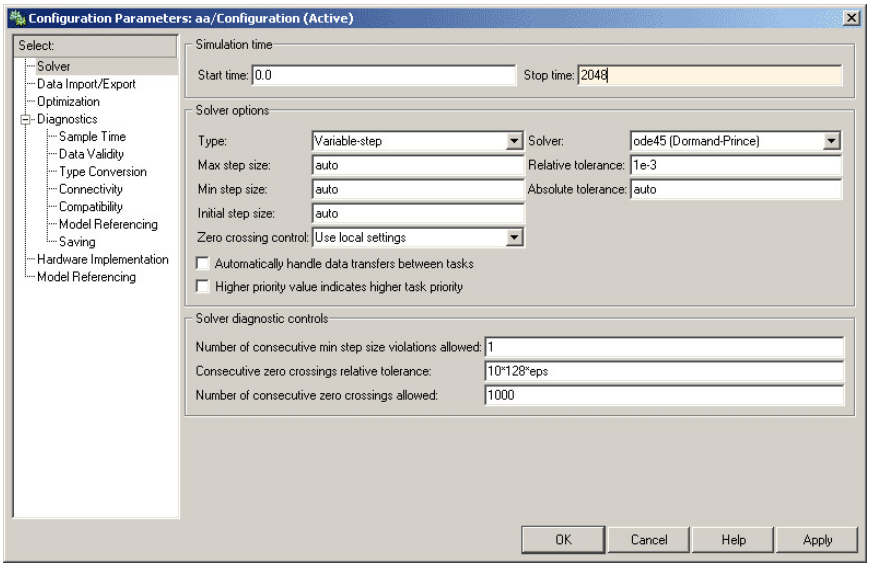
6. Save your work.


Scope
Gateway Out
Gateway Out1
Mux2
Mux3
AddSub1
AddSub2
Constant1
Constant2
Inverter
Inverter1
Mux
Mux1
Delay
ROM
ROM1
MSB
Logical
Logical1
Logical2
Concat4
Convert
Slice3
Concat3
Concat2
Concat1
AddSub
MSB3
MSB2
Slice1
Slice2
Gateway In
Constant

Task 6: Simulate the NCO Model

In this task, you will simulate the NCO model. First, you must tell the system how many points you want to run.

1. Choose **Simulation** from the Toolbar, and then select **Configuration Parameters** to open the dialog box.



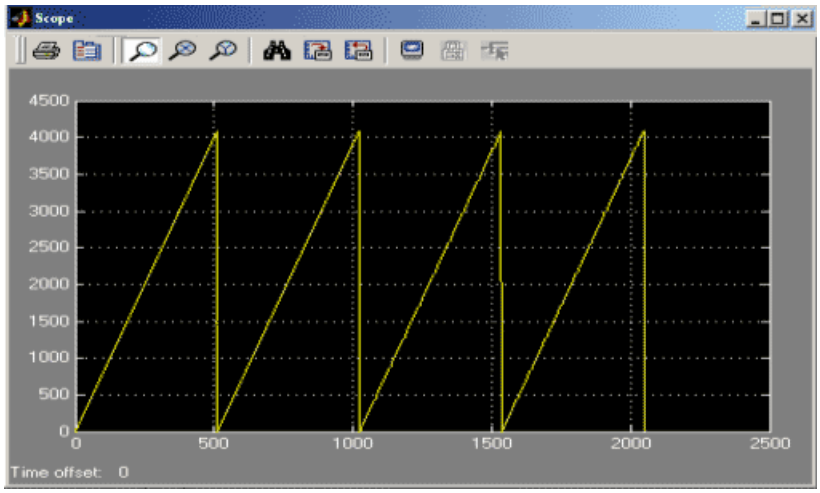
2. Set the Stop time at 2048 and click **OK** to close the dialog box.
3. From the Toolbar of the model window, select **Simulation**, and then click **Start** to begin the simulation.
4. After the compilation has finished, double-click the Scope block to observe the SIN and COS plots.

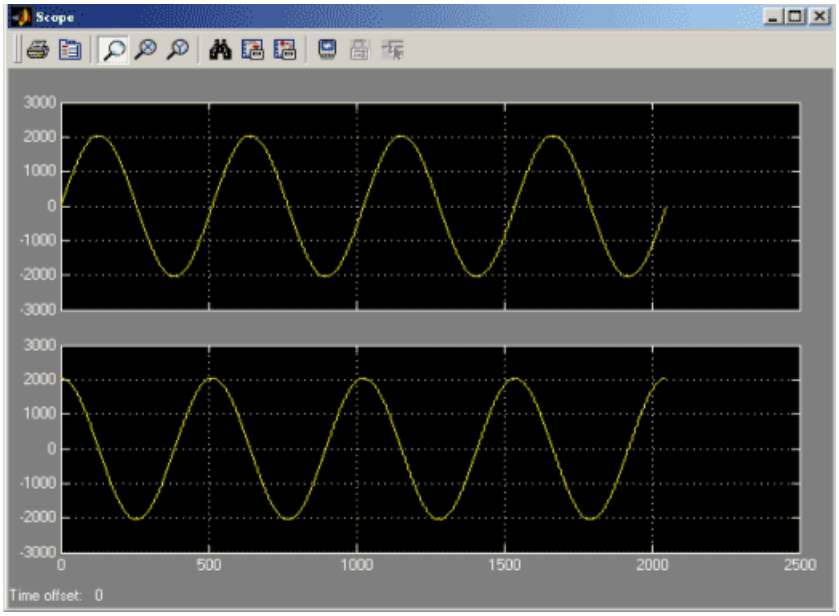
Note

To bring the plots into view, you might need to right-click inside each screen and choose Autoscale.

You can use other scopes, or sinks, from the Simulink library, but remember to connect a Lattice Gateway Out block between the sink's input and the Lattice block you want displayed. For example, placing a scope block on the output of the Accumulator will display the ramp function representing the address to the ROMs.

The following images display the results for the Accumulator's output and the SIN and COS outputs.





Task 7: Create a Subsystem

If you wish to make the system between the Gateway In and Gateway Out blocks shown as a single subsystems block, do the following:

1. Using your mouse, drag a rectangle around the group of blocks you want included as a subsystem. Release the mouse button when all the desired blocks are selected in the rectangle.
2. Choose **Edit > Create Subsystem**. Notice that all the blocks you selected have now been replaced by a subsystem block. If you decide that you do not want this subsystem, choose **Edit > Undo Create Subsystem**.

Note

If you save the model after creating the subsystem, you will not be able to undo the subsystem later.

Further Steps for Advancing the NCO

The following are some of the methods that you can use to advance the NCO.

- Include dithering for the amplitude to reduce spectral spurs or to allow for amplitude modulation.
- Modify the ROMs to implement coarse and fine LUTs.
- If phase modulation is required, place an additional digital mixer before the LUTs.
- For QAM generation, add more circuitry (digital multipliers) on the ROM LUT to perform complex multiplication so that the In-phase output is $I\cos\theta - Q\sin\theta$ and the Quad-phase output is $I\sin\theta + Q\cos\theta$, where *I* and *Q* are from the complex number $I + jQ$.

Tutorial Example

To view the model example file for this lesson, see

<*ispLEVER_install_path*>\ispLeverDSP\examples**DSP_tutorial2.mdl**