DSP: Using Upsampling and Downsampling for Color Space Conversion

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

November 2008

Copyright

Copyright © 2008 Lattice Semiconductor Corporation.

This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Lattice Semiconductor Corporation.

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, E2CMOS, Extreme Performance, flexiMAC, flexiPCS, FreedomChip, GAL, GDX, Generic Array Logic, HDL Explorer, IPexpress, ISP, ispATE, ispClock, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDXV, ispGDX2, ispGENERATOR, ispJTAG, ispLEVER, ispLeverCORE, ispLSI, ispMACH, ispPAC, ispTRACY, ispTURBO, ispVIRTUAL MACHINE, ispVM, ispXP, ispXPGA, ispXPLD, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeECP2, LatticeECP2M, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, MACH, MachXO, MACO, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, PURESPEED, Reveal, Silicon Forest, Speedlocked, Speed Locking, SuperBIG, SuperCOOL, SuperFAST, SuperWIDE, sysCLOCK, sysCONFIG, sysDSP, sysHSI, sysI/O, sysMEM, The Simple Machine for Complex Design, TransFR, UltraMOS, and specific product designations are either registered trademarks or trademarks of Lattice Semiconductor Corporation or its subsidiaries in the United States and/or other countries. ISP, Bringing the Best Together, and More of the Best are service marks of Lattice Semiconductor Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS “AS IS” WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE SEMICONDUCTOR CORPORATION (LSC) OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF LSC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

LSC may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. LSC makes no commitment to update this documentation. LSC reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. LSC recommends its customers obtain the

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

DSP: Using Upsampling and Downsampling for Color Space Conversion

This tutorial demonstrates the use of the Upsample and Downsample blocks in the ispLeverDSP blockset for MATLAB/Simulink.

Learning Objectives

When you have completed this tutorial, you should be able to do the following:

- Serialize data using the Downsample block
- Convert serial data to parallel form using the Upsample block
- Compare converted data to the original to insure that the conversion was successful
- Create and use subsystems
- Parameterize subsystems
- Use libraries to simplify design reuse

Time to Complete This Tutorial

The time to complete this tutorial is about 45 minutes.

System Requirements

One of the following software configurations is required to complete this tutorial:

- ispLEVER software with latest ispLeverDSP MATLAB/Simulink blockset installed.
- Active license for the MathWorks MATLAB/Simulink software.

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**.

About the Tutorial Design

It is assumed that you are familiar with ispLeverDSP and MATLAB/Simulink.

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 ispLEVER software.

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

Model Examples

The tutorial uses the files in the following directory:

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

The tutorial uses the following files:

- ColorSpaceConverter_tutorial.mdl
- ColorSpaceConversion_lib.mdl
- ColorSpaceConverter_Load.m

- ColorSpaceConverter_Plot.m

These files along with an image file should be available in the same directory to run this design.

Note

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

Task 1: Open the Example File

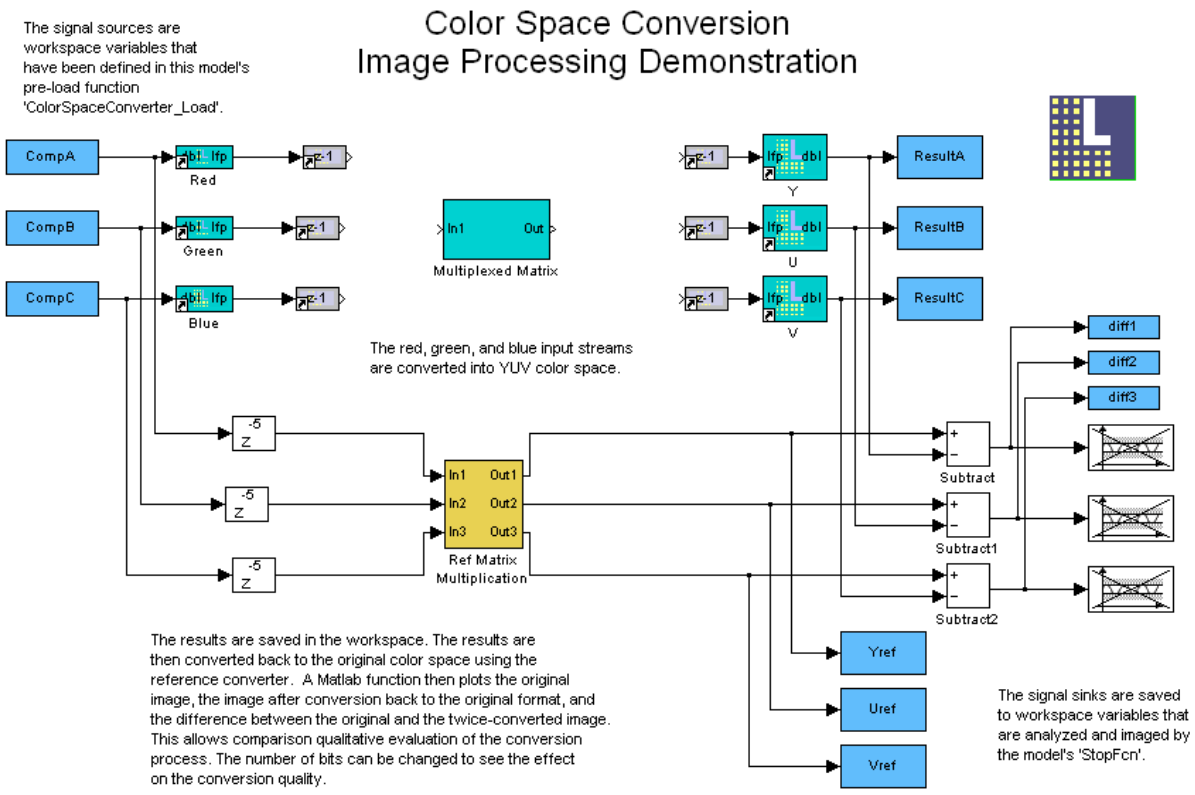
We now start the tutorial by opening the example model file.

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. From the MATLAB startup window, set the Current Directory to your working directory. For example:

   `<ispLEVER_install_path>\ispLeverDSP\examples\csc_tutorial`
4. Choose **File > New > Model** to open a blank model window.
5. Select the Simulink icon from the toolbar, or type **simulink** in the MATLAB Command window. This opens the Simulink Library Browser.
6. Choose **File > Open**, and open the **ColorSpaceConverter_tutorial.mdl** file in the **csc_tutorial** directory.
7. You will be asked to select an image file. This is the image file that will be used for the color space conversion test. You can use the image file **ColorSpaceConverter_image.jpg** in the tutorial directory. Or you can type in or browse to the directory and select another image file.

The model opens and should appear as follows:



Task 2: Add Blocks to the Model

This model contains a top-level testbench and an instance of a 3x3 matrix multiplier that will be used for the color space conversion. We will add blocks to this model to perform a conversion from a parallel data format to a serial form, and also to convert the serial output the matrix multiplier back to a parallel form.

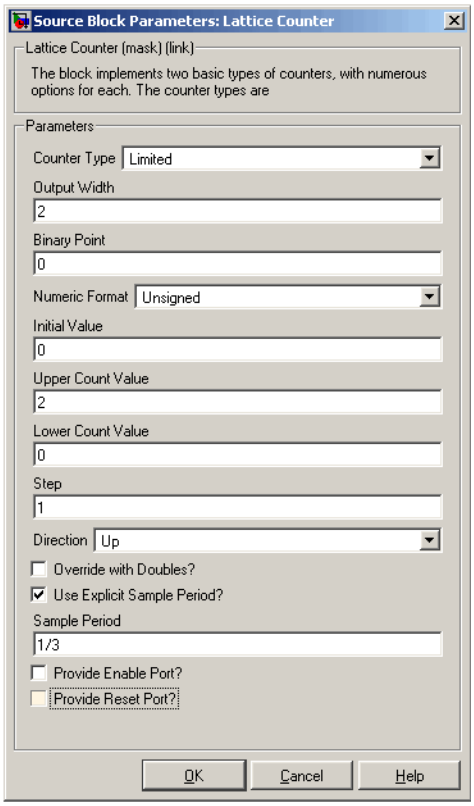
1. From the Lattice blockset, select the following blocks and drop them into the model outside of the existing blocks:
   - Upsample
   - Downsample
   - Delay
   - Counter
   - Mux

Task 3: Upsample the Inputs

We will upsample the inputs to the color space converter by 3 times, and then use a counter and a multiplexer to serialize the data. The output of the color space converter will then be downsampled to de-serialize the output.

1. Open the **Upsample** block mask and set the upsample multiple to **3**; all other parameters are left at their default values.
2. Copy the upsample block 2 times.
3. Open the **Counter** mask and change its parameters as follows:
   a. Modify the **Output Width** to be **2**,
   b. Change the **Upper Count Value** to **2**
   c. Select **Use Explicit Sample Period** and enter **1/3** in **Sample Period**.

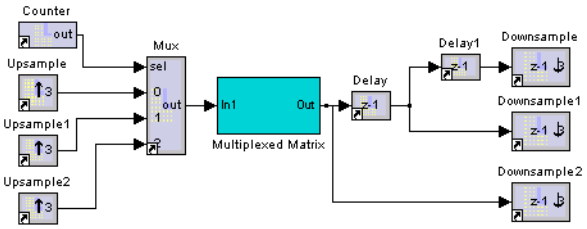
   The modified Counter mask looks like the following:



4. Open the **Mux** mask, and set it for three inputs.
5. Connect the counter to the Mux select, and connect each of the three Upsample block outputs to each mux data input.
6. Drag the Multiplexed Matrix block over to the added blocks, and connect its input to the mux output.
7. Make a copy of the delay block and connect them serially to the matrix output.

8. Open the **Downsample** block and set the downsample multiple to **3** and the Latency to **1**. Make two copies of this block.
9. Connect the inputs of each block to the output of the matrix and the two delay block outputs. You can resize the blocks and turn off some of the block labels to make it easier to read if desired.

   The added blocks should look something like the following:



Task 4: Make a Subsystem

Now we will make a subsystem out of this group of blocks.

1. Select all of the newly-added blocks and select **Create Subsystem** from the edit menu. The blocks will be replaced by a single subsystem block.
2. Double-click on this **Subsystem** block to view the original constituent blocks in a new window. Port connections have been added to all unconnected inputs and outputs. You can edit the port names to something more meaningful if desired.
3. Make sure the first input port (**In1**) is connected to the first Mux input through an Upsample block. The second input port (**In2**) is connected to the second Mux input through an Upsample block, and so forth.
4. Drag this block back to where the original Multiplexed Matrix block was, and connect the three inputs to the block.
5. Connect the three outputs to the delay blocks that produce the outputs. The structure of the design is now complete.

Task 5: Parameterize the Subsystem

The next step is to parameterize the newly created subsystem to allow the color space converter coefficients to be entered.

1. Right click on the Subsystem block and select **Mask Subsystem...**.
2. Select the parameters tab and then click the Add button (top left-most icon).
3. Enter into the Prompt field the text **Coefficients**. This is the text that will show up in the mask prompt.
4. Enter into the Variable field the text **K**. This is the name of the variable that will hold the coefficient information. The Type field will remain at the default setting of **Edit**. When you finish, click **OK**.

5. Double-click on the subsystem box once again, and you will see the mask with the parameters ready to be filled in.
6. Enter as the coefficient parameter the following:

   [0.299 0.587 0.114; -0.147 -0.289 0.436; 0.615 -0.515 -0.100]

   This is the coefficient set for a color space conversion from RGB to YUV.

Task 6: Simulate the Design

The design is now complete and can be simulated.

When the simulation completes, a MATLAB script will run and convert the results back to RGB using a floating-point conversion and then display the resulting image alongside the original. Also, the difference between the two images will be computed and multiplied by 16, and the result will be displayed as well. If everything has been done correctly, the two images will look almost identical, and the difference image will be mostly gray.

Summary

You have completed the “DSP: Using Upsampling and Downsampling for Color Space Conversion” tutorial. In this tutorial, you have learned how to do the following:

- Serialize data using the Downsample block
- Convert serial data to parallel form using the Upsample block
- Compare converted data to the original to insure that the conversion was successful
- Create and use subsystems
- Parameterize subsystems
- Use libraries to simplify design reuse