Lattice Synthesis Engine Tutorial

October 2013

Copyright

Trademarks

Disclaimers

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

Learning Objectives **1**

Time to Complete This Tutorial **2**

System Requirements **2**

Accessing Online Help **2**

About the Tutorial Design **2**

Task 1: Specify LSE as the Synthesis Tool **2**
- Opening the Project **2**
- Specifying LSE **3**

Task 2: Adjust the Design Code for LSE **3**
- Inferring RAM **3**
- Inferring I/O **4**

Task 3: Add LSE Constraints **5**

Task 4: Create an SDC File **6**
- Switching to the Modified top.v File **6**
- Creating an SDC File **7**
- Checking the Constraints **9**

Task 5: Set Options for LSE **9**

Task 6: Run Synthesis through the Main Window **10**
- Running Synthesis **10**
- Reading the Synthesis Report **10**
- Checking the Output Files of LSE **11**

Task 7: Merge the .lpf Files **12**

Summary of Accomplishments **13**

Recommended References **13**

Lattice Synthesis Engine Tutorial

This tutorial leads you through the basic steps for using Lattice Synthesis Engine (LSE) as the synthesis tool including:

- Setting up Lattice Diamond™
- Adjusting design source files
- Running synthesis
- Finding the output files and preparing to work with them further

Before starting this tutorial, you should already be familiar with the basic workflow of Diamond.

Learning Objectives

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

- Specify LSE for a project.
- Optimize design code for LSE.
- Get information on LSE attributes and directives and add them to a design.
- Get information on Synopsys Design Constraints (SDC) and create an SDC file for LSE.
- Get information on LSE options and set them in a strategy.
- Run synthesis with LSE using the Diamond main window and read the resulting synthesis report.
- Find the output files of LSE including the timing report and the Verilog netlist that LSE produces for post-synthesis simulation.
- Merge SDC constraints into the active Lattice preference (.lpf) file to be used with the map and place-and-route stages of implementing a design.

Time to Complete This Tutorial

The time to complete this tutorial is about 45 minutes.

System Requirements

The Lattice Diamond software is required to complete the tutorial.

Accessing Online Help

You can find online help information on any tool included in the tutorial at any time by choosing **Help > Lattice Diamond Help** or by pressing **F1**.

About the Tutorial Design

The design in this tutorial is a simple Verilog design targeted to MachXO2™. The design consists of a counter, using the EFB module from IPexpress™, and RAM, using the distributed_SPRAM module. At regular intervals the counter produces a signal that causes the RAM to write from the data port.

Two versions of the source file are available:

- top.v, the initial version that you start the tutorial with
- top_b.v, a version with all the modifications completed

You can study the modified version to see how the modifications should be done or use the modified version instead of entering the modifications yourself.

Task 1: Specify LSE as the Synthesis Tool

We'll start by telling Diamond to use LSE as the synthesis tool for this project. First we'll need to open the project.

Opening the Project

To open the project:

1. If you haven't already, start Diamond by doing one of the following:
   - On Windows, choose **Start > Programs > Lattice Diamond > Lattice Diamond**.
   - On Linux, enter the following on a command line:

<*diamond_install_path*>/bin/lin/**diamond**

2. In Diamond, choose **File > Open >** **Project**.

   The Open Project dialog box opens.

3. Browse to <*install_path*>/docs/tutorial/**LSE_tutor**.
4. Select **LSE_tutor.ldf**.
5. Click **Open**.

Specifying LSE

To specify LSE as the synthesis tool:

1. Choose **Project > Active Implementation > Select Synthesis Tool**.

   The Project Properties dialog box opens with the active implementation selected.

2. In the dialog box, double-click the Synthesis Tool row in the Value column.

   A menu drops down.

3. Choose **Lattice LSE**.
4. Click **OK**.

Task 2: Adjust the Design Code for LSE

You may want to adjust the design source code to get the best results with LSE. In this design, there is a block of RAM that we want LSE to infer rather than using an IPexpress module. We also want to check that I/O will be inferred. If the design were in VHDL, we would also make some other checks of the code because LSE applies the VHDL specification more strictly than some other synthesis tools.

For more information, open the online Help and see Entering the Design > HDL Design Entry > Coding Tips for Lattice Synthesis Engine (LSE).

Inferring RAM

To infer the RAM:

1. Open the File List view. By default this is on the left side of the main window. If the File List view is not open, choose **View > Show Views > File List**.
2. In the File List view, look under Base > Input Files. “Base” is the active (and only) implementation of the design. Double-click **source/top.v**.

   Source Editor opens showing the Verilog source code for the design.

3. Place the cursor at the bottom of the file, below the endmodule statement.
4. Add the code in Figure 1 at the end of the file. (You can copy and paste from the PDF file.)

Figure 1: Simple, Single-Port RAM in Verilog

```
// RAM that can be inferred by LSE.
module RAM_single_port (Address, Data, Clock, WE, ClockEn, Q);
  parameter addr_width = 5;
  parameter data_width = 8;
  input [addr_width-1:0] Address;
  input [data_width-1:0] Data, Q;
  input WE, Clock, ClockEn;
  reg [data_width-1:0] mem [(1<<addr_width)-1:0];
    // Define RAM as an indexed memory array.

  always @(posedge Clock) // Control with a clock edge.
  begin
    if (ClockEn)
      if (WE) // And control with a write enable.
        mem[(Address)] <= Data;
  end
  assign Q = mem[Address];
endmodule
```

This code defines a simple, single-port RAM block in a way that LSE can recognize. The comments point out the keys to making the RAM recognizable:

- Defined as an indexed memory array.
- Controlled with a clock edge.
- Controlled with a write enable.

5. Choose **File >**  **Save top.v**.
6. In the File List view, under Input Files, right-click **source/RAM_single_port/RAM_single_port.ipx**.

   A drop-down menu appears.
7. From the drop-down menu, choose **Exclude from Implementation**.

   The selected file is grayed out. This prevents the RAM module created with IPexpress from being used. The file can be restored later by right-clicking it and choosing **Include in Implementation**.

Inferring I/O

To infer I/O:

1. In Source Editor, find the beginning of the I/O definitions. Look for the following comment, near the beginning of the code:

   // Define bidirectional port for data.

2. If the online Help is not already open, choose **Help > Lattice Diamond Help**.
3. In the online Help, go to **Entering the Design > HDL Design Entry > Coding Tips for Lattice Synthesis Engine (LSE) > Inferring I/O**.
4. In the “Inferring I/O” topic, click **Verilog**.

   The Verilog section expands showing models for different kinds of ports.
5. Compare the bidirectional port definition in top.v to the model in the Help. Does the definition in top.v need to be changed to be inferred by LSE?

   No. The port is good as it is.

Task 3: Add LSE Constraints

While the source file is open we can also add some constraints to control the LSE synthesis process. We’ll set constraints in the source code for the following purposes:

- Make sure the global set/reset (GSR) signal can be used.
- Force the RAM block to use the device’s RAM instead of programmable function units (PFU).

For more information about these and other LSE constraints, open the online Help and see Constraints Reference Guide > Lattice Synthesis Engine (LSE) Constraints > HDL Attributes and Directives.

To add LSE constraints:

1. Find the line starting with “module top.” It’s near the start of the file.
2. Place the cursor just before the semi-colon (;) at the end of the line and type in a space and the following:

   /* synthesis GSR=“ENABLED” */

   The result should look like:

   ```
   module top (addr, data, clk, clock_en, read_en)
   /* synthesis GSR=“ENABLED” */;
   ```

   This constraint enables GSR for the whole top module.
3. Choose **File >**  **Save top.v**.
4. In Source Editor, find the lines that instantiate the RAM:

   ```
   // Instantiate RAM.
   RAM_single_port memory (.Address(addr), .Data(data_in), .Clock(clk),
     .WE(read_write), .ClockEn(clock_en), .Q(data_out));
   ```
5. Place the cursor just before the semi-colon (;) at the end of the line and type in a return and the following:

   /* synthesis syn_ramstyle=”block_ram” */

   The result should look like:

```
// Instantiate RAM.
RAM_single_port memory (.Address(addr), .Data(data_in), .Clock(clk),
  .WE(read_write), .ClockEn(clock_en), .Q(data_out))
  /* synthesis syn_ramstyle="block_ram" */;
```

This constraint forces this instance of the RAM to use the device's dedicated RAM resources.

6. Choose **File >**  **Save top.v**.
7. Close Source Editor by clicking the red box  in its tab.

Task 4: Create an SDC File

A Synopsys Design Constraint (SDC) file can be used to supply timing constraints used in the static timing analysis. Also, these constraints drive optimization of the design if LSE's Optimization Goal is set for timing (as it will be in the next task). After synthesis, the SDC constraints become preferences added to the Lattice preference (.lpf) file. So they'll be available to the map and place-and-route stages too.

We'll add constraints so that static timing analysis assumes:

- A clock signal with our desired frequency of 250 MHz (a period of 4.0 ns)
- Input signals reach the device after a delay of 1.2 ns
- Output signals reach other components on the board after a delay of 1.2 ns

The clock signal will be applied to the design's CLK port. The delays will be relative to the CLK signal. The input and output delays model the interface of the device with other components on the board. We could apply different delays to different signals, but to start we'll just apply the same value to them all.

For more information about these and other SDC constraints, open the online Help and see Constraints Reference Guide > Lattice Synthesis Engine (LSE) Constraints > Synopsys Design Constraints (SDC).

Switching to the Modified top.v File

If you have worked through all of the preceding tasks, your top.v file is ready to compile, which is required for this task. If you have not, there is another file that already has the modifications made.

To use the already modified file:

1. In the File List view, right-click  **source/top.v**.
2. In the drop-down menu, choose **Open Containing Folder**.

   A window opens showing the contents of <*install_path*>/docs/tutorial/LSE_tutor/source.

3. Rename top.v as top_a.v.
4. Rename top_b.v as top.v.

Creating an SDC File

An SDC file is a text file and can be created with any text editor including Diamond's Source Editor, which includes templates for all supported constraints. However, when using LSE, the preferred tool is Diamond's LDC Editor, which this procedure demonstrates.

Note

SDC files that are intended for LSE are called "LDC" files in Diamond and use an .ldc filename extension. This is to clearly identify them from the .sdc files created for other tools such as Synplify Pro.

With LDC Editor you don't have to know the syntax for the constraints. Instead, you specify the kind of constraint and parameter values in a spreadsheet format with each row representing one constraint. In most cases, values are chosen from drop-down menus. However, you still need to be familiar with the kind of constraints available and the functions of the parameters.

To open a new .ldc file:

1. Choose **File > New >**  **File**.

   The New File dialog box opens.
2. In the Source Files box, select  **LDC Files**.
3. Enter a base name for the new file, such as **timing**. Do not add an extension. That's included automatically.

   The rest of the fields are good as they are so don't change them. They specify that the new .ldc file be placed in the project folder and be part of the active implementation.
4. Click **New**.

   LDC Editor opens with an empty timing.ldc file. Also, in the File List view, the .ldc file is added to the list of Synthesis Constraint Files for the Base implementation.

   As LDC Editor opens it runs a preliminary compile of the design. This is not synthesizing the design but creating a netlist for use in the editor. With the tutorial's design, compiling takes a few seconds. For a large design, compiling would take several minutes.

To specify a clock signal:

1. Click the **Clocks** tab.

   Use the Clocks tab for the create_clock constraint.
2. Double-click in the cell under Source.

3. In the drop-down menu, choose **clk**.
4. Click in the Clock Name column and type a name for the clock signal, such as **Clock**.
5. Click in the Period(ns) column and type **4**.

To specify delays for input signals:

1. Click the **Inputs/Outputs** tab.

   Use the Inputs/Outputs tab for the set_input_delay and set_output_delay constraints.

   The Delay Paths tab is for the set_max_delay, set_multicycle_path, and set_false_path constraints. But we will not be using them in this tutorial.
2. Double-click in the Port column.

   The Port cell changes to a drop-down menu.
3. Choose **[all_inputs]**.

   “Input” appears in the Type column.
4. Double-click in the Clock column.

   The Clock cell changes to a drop-down menu.
5. Choose **clk**.
6. Click in the Value(ns) column and type **1.2**.

To specify delays for output signals:

1. In the next row of the Inputs/Outputs tab, double-click in the Port column.

   The Port cell changes to a drop-down menu.
2. Choose **[all_outputs]**.

   A warning icon appears in the Port cell and “Input” appears in the Type column.
3. Double-click in the Type column.

   The Type cell changes to a drop-down menu.
4. Choose **Output** and then click anywhere.

   The warning icon disappears.
5. Double-click in the Clock column.

   The Clock cell changes to a drop-down menu.
6. Choose **clk**.
7. Click in the Value(ns) column and type **1.2**.
8. Choose **File >** **Save timing.ldc**.
9. Close LDC Editor by clicking the red box in its tab.

In the drop-down menu, the file name is in bold text to indicate that the file is active and will be used when synthesis is run.

Checking the Constraints

After creating the .ldc file, you can see, and modify, the actual constraint statements using Source Editor or any other text editor.

To check the constraint code:

1. In the File List view, right-click **timing.ldc**.
2. In the drop-down menu, choose **Open With**.
3. In the Open With dialog box, select **Source Editor**.

   LDC files can also be created with Source Editor or any text editor. You can change the default editor for .ldc files in this dialog box. For now we'll leave the default alone and just take a look at the code that LDC Editor created.
4. Click **OK**.

   Source Editor opens with the timing.ldc file.
5. You should see the following:

   ```
   create_clock -period 4.000000 -name Clock [get_ports clk ]
   set_input_delay 1.200000 -clock [ get_ports clk ] [all_inputs]
   set_output_delay 1.200000 -clock [ get_ports clk ] [all_outputs]
   ```

   If you don't, you can modify the code in Source Editor.
6. Close Source Editor by clicking the red box in its tab.

Task 5: Set Options for LSE

One of the things to do before running synthesis is to set options for the synthesis tool. Usually the default settings are good, at least to start. So we'll change just a few options to set:

- Optimization efforts that focus on timing goals
- Producing a logical preference file based on the SDC file

For more information about these and other strategy options for LSE, open the online Help and see Strategy Reference Guide > LSE Options. Also, for information on optimizing a strategy for area or timing, see Implementing the Design > Synthesizing the Design > Optimizing LSE for Area and Speed.

To set LSE options:

1. Choose **Project > Active Strategy > LSE Settings**.

   The Strategies dialog box opens showing the list of LSE options.
2. Select **Optimization Goal**.

   At the bottom of the dialog box a brief description of the option appears.
3. Press **F1** to open the help with a longer description of the option (and all the other options as well).

4. Double-click in the Value column.

   The Value cell changes to a drop-down menu.

5. Choose **Timing**.
6. Select **Output Preference File** and double-click in the Value column.

   The Value cell changes to a drop-down menu.

7. Choose **True**.
8. Click **OK**.

Task 6: Run Synthesis through the Main Window

At this point you are ready to run synthesis with LSE. Afterward we'll examine the LSE report and merge LSE's .lpf file into the project's .lpf file.

Running Synthesis

To run synthesis:

1. Go to the Process view in Diamond. By default this is on the left side of the main window. If the Process view is not open, choose **View > Show Views > Process**.
2. Notice that under Synthesize Design it says "Lattice Synthesis Engine" and that the Translate Design stage is gone. LSE combines the synthesis and translate stages.
3. Double-click **Synthesize Design**.
4. When synthesis completes, check the Error and Warning tabs of the Output view. You may have warnings but there should be no errors.

Reading the Synthesis Report

To read the synthesis report:

1. If the Reports view is not already open, choose **View >**  **Reports**.
2. In the Reports view, click  **Lattice LSE**. (It's under Process Reports.)

   The "Synthesis and Ngdbuild Report" appears.

3. Double-click  **Lattice LSE** to expand the hierarchy tree. These are actually links to different sections of the report.

   We'll skip the "Synthesis Options" section, which details the target device and the synthesis options.

4. Click **Compile Design**.

The display jumps down to the “Compile Design” section, which summarizes the steps taken while synthesizing the design including output file names. This section also includes the area and clock reports.

5. Scroll down to the “Area Report” section. Check how the design will use the device resources. This is just a top-level summary. A more detailed report is also available in the implementation folder. It’s described in the next section.
6. Scroll down to the “Clock Report” section. See how it details the clock nets. You can also see how heavily loaded are the nets with the highest fanout.
7. Go to the “Timing Report Summary” section of the report. This section summarizes how well the synthesized design meets the timing constraints.

   For more details, see the full timing report: top_lse.twr in the Base implementation folder. To access the report, see the next section.

Checking the Output Files of LSE

You can take a closer look at the files produced by LSE if you go to the implementation folder:

1. In the File List view, right-click **Base**.
2. In the drop-down menu, choose **Open Containing Folder**.

   A window opens showing the contents of <*install_path*>/docs/tutorial/LSE_tutor/Base.

This folder holds the files specifically associated with the Base implementation. The files produced by running LSE include:

- automake.log, a record of messages from LSE as it was running; the same information that is in the Output view
- LSE_tutor_Base.arearep, the area, or resources, report that is summarized in the synthesis report
- LSE_tutor_Base.lpf, the Lattice preference (.lpf) file based on the .ldc file
- LSE_tutor_Base.lsedata, lists of pins and nets for each module
- LSE_tutor_Base.ngd, the native generic database that will be used by the Map Design stage of implementing the design
- LSE_tutor_Base_lattice.synproj, the arguments used by the `synthesis -f` command, which runs LSE
- synthesis.log, a record of messages from LSE as it was running; basically the same information that is in the synthesis report
- synthesis_lse.html, the synthesis report
- top_drc.log, results of the design rule check, which are also shown in the Output view
- top_for_lpf.sdc, an .sdc file based on the .ldc file

- top_lse.twr, the timing report that is summarized in the synthesis report
- top_prim.v, the Verilog netlist that can be used by a simulator

Four files are of special interest.

Open LSE_tutor_Base.arearep with a text editor. You will see many more details on how the synthesized design uses the device's resources. Besides resources used by the total design, the report also shows resources used by each module.

Open top_lse.twr with a text editor. You will see many more details on how well the synthesized design meets the timing constraints.

Open top_prim.v with a text editor or a Verilog simulator. You will see how the design was expanded and implemented by LSE. The <*top_module*>_prim.v file produced by LSE can be used in a simulator to further test the synthesized design before going on with the rest of the implementation process.

For using LSE_tutor_Base.lpf, go to the next task.

Task 7: Merge the .lpf Files

LSE produces a Lattice preference (.lpf) file based on the .ldc file. This .lpf file has preferences equivalent to the .ldc file's constraints. If these preferences are added to the active .lpf, created while developing the design, they will also affect the map and place-and-route stages of design implementation.

The file created by LSE can be recognized by the form of its file name: <*project*>_<*implementation*>_lse.lpf.

You can use any text editor to modify the .lpf file, but the following procedure uses Diamond's Source Editor.

To merge LSE's .lpf into the active .lpf:

1. Go to the File List view.
2. Under Base > LPF Constraint Files, double-click **LSE_tutor.lpf**.

   Source Editor opens with the .lpf file. This is a previously created .lpf file. The first two lines with BLOCK preferences were added by default. The third line was added manually.
3. In the main window, choose **File > Open > File**.
4. In the Open File dialog box, in the "Files of type" drop-down menu at the bottom, choose **Preference Files**.
5. Browse to the **Base** implementation folder and select **LSE_tutor_Base.lpf**.
6. Click **Open**.

   Source Editor opens a second view with the .lpf file. This is the .lpf created by LSE. Besides the preferences based on the SDC constraints, the .lpf

file automatically includes the same BLOCK preferences seen in the active .lpf.

7. Choose **Window >**  **Split Tab Group**.

   The tools area of the main window splits into two groups. You should have LSE_tutor.lpf on one side and LSE_tutor_Base.lpf on the other side. If they are both on the same side, select the tab of one and drag it to the other side.

8. In LSE_tutor_Base.lpf, select everything from line 3 down. This is everything except the first two BLOCK preferences.
9. Press Ctrl-c to copy the selected preferences.
10. Go to LSE_tutor.lpf and place the cursor at the bottom of the file, in a blank line beneath all the text.
11. Press Ctrl-v to paste in the new preferences.

    Note

    If the active .lpf file already has preferences from a previous run of LSE, paste the new preferences over the old version. Do not create multiple copies of the same preferences.

12. Save the LSE_tutor.lpf file and close both files.

Summary of Accomplishments

You have completed the *Lattice Synthesis Engine Tutorial* tutorial. In this tutorial, you have learned how to:

- Specify LSE for a project
- Adjust the design code for LSE
- Set constraints for LSE
- Adjust LSE options in a strategy
- Run synthesis with LSE using the Diamond main window
- Find the output files of LSE
- Merge the .lpf files

Recommended References

You can find additional information on the subjects covered by this tutorial in the online Help:

- Managing Projects > Setting Options for Synthesis and Simulation
- Strategy Reference Guide > LSE Options
- Entering the Design > HDL Design Entry > Coding Tips for Lattice Synthesis Engine (LSE)

- Constraints Reference Guide > Lattice Synthesis Engine (LSE) Constraints
- Implementing the Design > Synthesizing the Design