Lattice Propel Builder 1.0 User Guide

April 20, 2020

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

Designing with Lattice Propel Builder

Use Lattice Propel Builder to assemble complex system-on-chip modules for use in Lattice FPGAs with an easy drag-and-drop method. With Propel Builder, you simply drag flexible, pre-built modules from an included catalog and drop them into a schematic view, customizing the modules as you go. Nets connecting the modules are simply drawn, dragging lines from one pin to another. Propel Builder also helps you customize address spaces when those are part of the modules, as with a processor.

When complete, the Propel Builder module can be instantiated in your design project.

While Propel Builder is often a good option, note the following limitations:

- Propel Builder can only use IP that come with Propel Builder. This means no PMI and no HDL modules.
- Propel Builder modules cannot be nested. One Propel Builder module cannot include another Propel Builder module.

Lattice Propel Builder Design Flow

Here are the recommended steps for creating a Propel Builder module.

If you are going to include a processor in your design, we recommend creating the processor platform first. This is to minimize waiting for software development. Simulation testing of the platform requires a Lattice system memory initialization file (.mem) that is produced by Lattice Propel SDK.

While building your design, you should occasionally click the Save button. Propel Builder does not automatically save while you are working. To document the design, you can also make a printout of the Schematic view.

To create a Propel Builder module:

1. Start a Propel Builder project. See "Opening a Propel Builder Project" on page 2.
2. Add modules. See "Adding Modules" on page 2.
3. Connect the modules to each other. See "Connecting Modules" on page 11.

   Any time the schematic gets hard to read, you can move the modules or zoom in on a part of the design. See "Adding Modules" on page 2.
4. Manually create top-level ports for signals that go to more than one pin, such as clock and reset signals. See "Creating Top-Level Ports" on page 13.
5. Automatically create the other top-level ports. See "Creating Top-Level Ports" on page 13.
6. Adjust address space if necessary. See "Adjusting Address Spaces" on page 14.
7. Generate the RTL file. See "Generating the RTL File" on page 16.
8. If the Propel Builder project includes a processor, generate the software package for the Lattice Propel SDK software development kit. See "About Software Development with Lattice Propel SDK" on page 16.

Opening a Propel Builder Project

To start a Propel Builder project, do one of the following:

- To start a new project, click + **New Design**. In the Create System Design dialog box, choose the language for the design's RTL file. Then enter a name and location for the project.
- To open an existing project, click **Select from disk**. Then browse for the project's .sbx file, which defines a Propel Builder project.

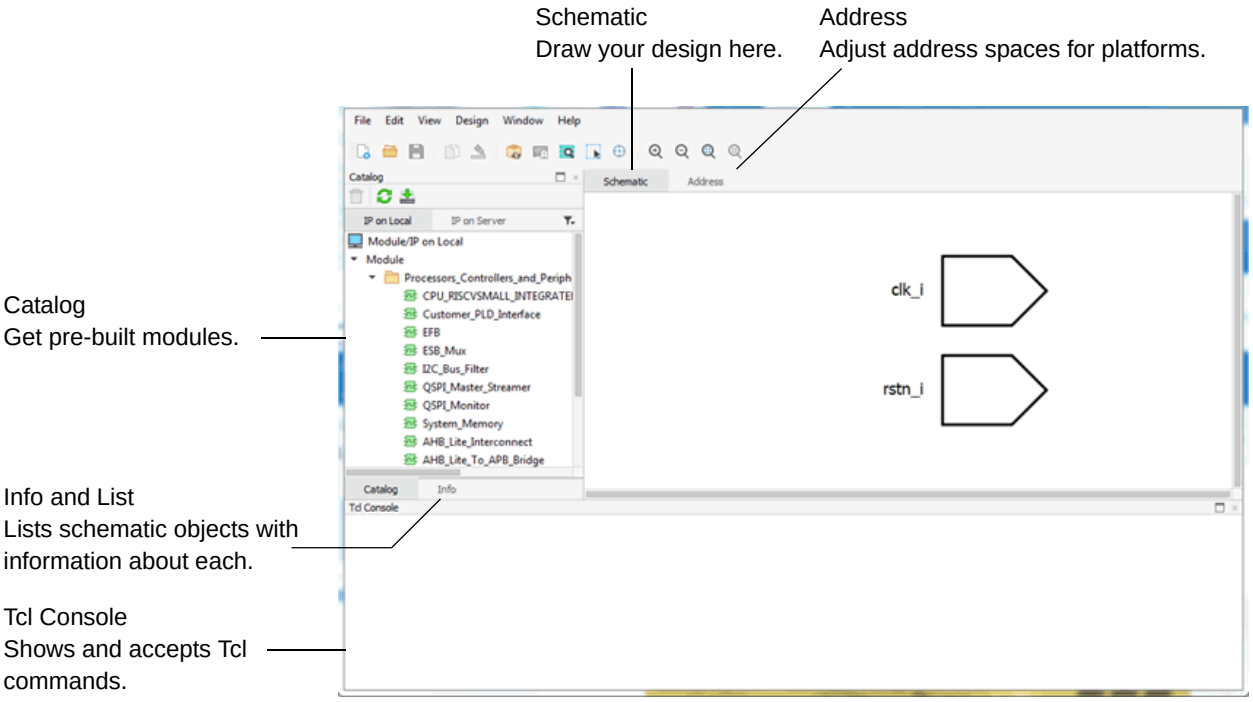
The Propel Builder pane divides into a few views, ready for you to start working on the Propel Builder module, as shown in Figure 1.

Adding Modules

Add modules by dragging them from the Catalog view to the Schematic view.

The Catalog view comes with a large variety of commonly used modules. These are under the IP on Local tab. Click the IP on Server tab to see more-specialized modules that you can download.

Figure 1: Lattice Propel Builder Window



Adding from the Catalog View

Start by dragging in modules and IP from the Catalog view.

To add a module from the Catalog view:

1. In the Catalog view, click the **IP on Module** tab.
2. Expand the folder tree and select the module you want to use. You can filter the list by clicking the Filter button that is to the right of the tab. Search by the bus type or module name.
3. For more information, scroll all the way to the right and hover over the IP. A blue circle with a question mark appears. Click the blue circle.

   A brief description of the module appears. To get more information about this module, click **User Guide** in the description.
4. Drag the module to where you want it in the Schematic view.

   The Module/IP Block Wizard opens.
5. Enter an instance name and browse to a location for the module's files.
6. Click **Next**.
7. Configure the module. These modules can be extensively customized for your design. The options range from setting the width of a data bus to selecting features in a communications protocol.
8. Click **Generate**.

9. In the Check Generated Result page, make sure **Insert to project** (in the lower-left corner) is selected.
10. Click **Finish**.

    The Define Instance dialog box opens with a default name.
11. Change the instance name if desired. Spaces and special characters are not allowed. Underscores (_) are OK.
12. Click **OK**.

    The schematic block for the module appears in the Schematic view. System bus modules are blue.

Downloading from IP on Server

You can use the IP on Server tab of the Catalog view to download and install the latest IP available from Lattice Semiconductor.

To download Lattice IP:

1. In the Catalog view, click the **IP on Server** tab.
2. Click the refresh button to update the list.
3. Expand the folder tree and select the IP you want to download. You can filter the list by clicking the Filter button that is to the right of the tab. Search by the bus type or module name.

   IP that are compatible with your selected device have blue IP icons. IP that are not compatible have blue icons with a yellow triangle .
4. For more information, scroll all the way to the right and hover over the IP. A blue circle with a question mark appears. Click the blue circle.

   A brief description of the module appears. To get more information about this module, click **User Guide** in the description.
5. Click the Install button.

   The downloaded IP is added to the **IP on Local** tab. To use the new IP, see “Adding from the Catalog View” on page 3.

Working with the Schematic View

There are several ways that you can change the Schematic view including an automatic layout to clean up the display. Or you can move, resize, or rename blocks manually. You can also focus the view by highlighting objects and zooming the display in and out.

About the Plus Signs in the Schematic

Often the schematic shows functional groups of pins combined into a bus marked with a large plus sign ⊞. This is a “piniface” type object in the Info view. In the List view, they are listed under BusInterfaces. Some of these groups are complex, such as the AHB Lite buses. Some are simple, such as an interrupt request pin.

These buses can be expanded to see what signals they contain. Simply click the plus sign. The plus sign changes to a negative sign. The bus expands to show the pin names but does not show the nets connected to these pins. The schematic only shows the net connected to the name of the functional group.

The pin names under the pinifaces are meant to identify functions. The actual pin names can be found in the Info view. For example, if you expand both of the IRQ00 and SOFT_INT pinifaces in the CPU module, you see that they both have a pin labeled “IRQ.” But the Info view shows that the real names are irq_i and soft_int_i.

While you can select these individual pins and buses, do not try to connect them individually. Always work with the piniface group as a whole.

You can also see the signals in the List view. Find the bus in the module’s BusInterfaces list and click the triangle next to the name.

To close the expanded bus, click the negative sign. The schematic returns to its previous form.

Selecting Objects

There are several ways to select one or more objects.

To select objects:

- Click on the object in the Schematic or List view.
- Ctrl-click or Shift-click in the Schematic view to select more objects.
- Click the Area_select button. Then click and drag to draw a selection rectangle around modules and ports in the Schematic view. This method only selects modules and ports, not nets. When you are done, click the button again to turn off the Area_select mode.
- Right-click in the Schematic view and choose **Select All**. This command selects all of the objects.
- Press **Ctrl-A**.This command selects all of the objects.

To de-select one object while leaving others selected:

- Ctrl-click or Shift-click on the object in the Schematic view.

Moving Objects

You can rearrange the schematic by dragging objects. But you cannot drag objects just anywhere. Propel Builder has rules for placing objects and adjusts the schematic to keep an organized arrangement. As you drag, Propel Builder provides clues to how the schematic will change but does not immediately move the items.

To move items:

1. Select the desired modules or port (not both).

   You can drag one or more modules at the same time or just one port.

2. Click on one of the selected items and drag to where you want them.

   Notice the clues to how objects will move.

3. Release the mouse button.

   The selected objects move to the specified location or as near as the rules allow. Other objects in the schematic may also move to accommodate the selected objects' new location.

Locating Objects

If your schematic is much bigger than your Schematic view, locating objects can be difficult. Use the Locate Object mode to bring selected objects to the center of the Schematic view.

To locate an object in the Schematic view:

1. Click the Locate Object button so the background turns a darker gray.
2. Select the object in the List view.

   The Schematic view shifts so that the selected object is in the center of the view.

Simplifying the Layout

After connecting and moving modules, the schematic may be hard to read. Propel Builder has an easy way to simplify the layout.

To automatically simplify the layout:

- Right-click anywhere in the Schematic view and choose **Relayout**.

  Propel Builder rearranges the module blocks for a compact arrangement with simpler nets. The result should be easier to analyze.

Copying Modules

You can easily create duplicates of a module.

To copy a module in the Propel Builder project:

- In the Schematic view, right-click on the module and choose **Clone**.

  A copy of the schematic block appears with a new instance name. You can change this name in the Info view. See "Renaming Objects" on page 10.

Restoring Deleted Modules

If you delete a module from the Schematic view, the module's component is still in the List view but with lighter text. You can create a new instance from that component.

To restore a deleted module:

1. In the List view, go to the Components folder.
2. Right-click on the component and choose **Instantiate**.

   The Define Instance dialog box opens.
3. Enter a name for the new instance.
4. Click **OK**.

   The module appears in the Schematic and List views, and the component name is bolded.

Reconfiguring Modules

You can change a module after it's been added to a project.

To reconfigure a module:

1. Double-click the module or right-click the module and choose **Reconfig**.

   The Module/IP Block Wizard opens.
2. Configure and generate the module as usual.

   The schematic block for the module changes to match the new configuration.

Resizing Module Blocks

You can change the size and shape of module blocks by dragging the corners.

To resize a module block:

1. Select the module block.

   The block is highlighted in red with black corners.

2. Click and drag one of the corners to change the size and shape of the block.

3. Release the mouse button.

   Other objects move to make room for the block.

To restore the size of a module block:

- Right-click the module block and choose **Unresize Instance**.

  The block returns to its original size. Propel Builder creates module blocks with a minimum size that can clearly display the pin names.

Zooming

You can zoom within the Schematic view using a variety of methods, including toolbar commands and dragging in the Schematic view.

Toolbar Commands The following commands are available on the toolbar:

 Zoom In – enlarges the view of the entire layout.

 Zoom Out – reduces the view of the entire layout.

 Zoom Fit – reduces or enlarges the entire layout so that it fits inside the window.

 Zoom To – enlarges the size of one or more selected objects on the layout and fills the window with the selection.

Zooming with Function Key Shortcuts The following key combinations enable you to instantly zoom in or out from your keyboard:

- Zoom In – Ctrl++
- Zoom Out – Ctrl+-

Zooming with the Mouse Wheel The mouse wheel gives you finer zoom control, enabling you to zoom in or out in small increments.

1. Place the cursor over the spot that you want to zoom on.
2. While pressing the **Ctrl** key, roll the mouse wheel forward to zoom in and backward to zoom out.

Zooming by Dragging First, make sure that the Area_select button is not selected. Zoom by holding the mouse button and dragging:

- To zoom to fit the window, drag up and to the left. The image adjusts to fill the window.
- To zoom out, drag up and to the right. The distance you drag determines the amount of zoom. 1 means half as big, 2 means ¼ as big, and so on. The image is reduced and centered in the window.
- To zoom in, drag down and to the left. The distance you drag determines the amount of zoom. 1 means twice as big, 2 means four times as big, and so on. The image is enlarged and centered in the window.
- To zoom in on a specific area, start at the upper-left corner of the area that you want to see better and drag to the lower-right corner of the area. The area that you drag across is adjusted to fill the window.

If you frequently want to use Area_select and zoom by dragging, you can have both available at all times. Choose **Edit > Options**. Select **Use mouse right button** and click **OK**. After this, dragging with the left button selects instances and dragging with the right button zooms.

Panning and Scrolling

You can move the schematic image within the Schematic view by panning and scrolling.

To pan the image:

- Hold down the Ctrl key and the left mouse button while dragging the image.

To scroll vertically:

- Rotate the mouse wheel or click in the vertical scroll bar.

To scroll horizontally:

- Hold down the Shift key and rotate the mouse wheel, or click in the horizontal scroll bar.

Showing Connectivity

In a complex design it may help to see how a module is connected.

To show the connectivity of a module:

- Right-click the module and choose **Show connectivity**.

  All nets connected to the module and all pins and ports that those nets connect to are highlighted.

Highlighting

Highlighting makes objects easier to find in a complex schematic by marking them with a bright color. Highlighted objects are not selected for commands. Highlighting does not appear in printouts.

To highlight an object:

- Right-click the object and choose **Highlight**.

To remove highlighting:

- Right-click the object and choose **Highlight** again.

Renaming Objects

You can change the name of an object using the Info view. You cannot change the names of pins inside module blocks. These are set inside the modules.

To change the name of an object:

1. Select the object.

   In the Info view, information about the object, including its name, appears.
2. Change the name.
3. Click **Enter**.

   The name changes in the Schematic and List views.

Printing Schematics

The print function prints the whole schematic on one page.

To print a schematic:

1. Choose **File > Print Preview**.

   The Print Preview window opens.
2. Expand the Print Preview window to the desired size.
3. Click the Page setup button and adjust the paper size and margins, if necessary.
4. Click the Print button.
5. Adjust the printer settings if necessary and click **Print**.

Connecting Modules

Connect the pins of modules to other modules or to top-level ports by dragging a line between them or by selecting connection points. You can also assign a constant value to an input pin or bus.

Propel Builder does not allow obviously inappropriate connections, such as between two output pins or mismatched buses.

Warning

Do not try to connect the individual signals within pinifaces (the plus signs ⊞). Always work with the piniface group as a whole. See "About the Plus Signs in the Schematic" on page 5.

Drawing Nets

You can create a net by drawing a line from one pin or port to another or to an existing net.

To connect by drawing:

1. When you move the cursor to a pin or port, the cursor changes to a pencil icon: . Click and hold while dragging to another pin, port, or net.

   You can release the mouse button once drawing mode starts. You don't need to hold the button the whole time.

   An allowed pin or port shows a green check mark when you hover over it. An allowed net is bolded when you hover over it.

2. Click on the pin, port, or net that you want to connect to.

   If the connection is allowed, a line appears connecting the two objects. Propel Builder creates a path around other objects.

3. If this net is to connect a source to multiple input pins, drag to another pin and click it. Continue this way until the net is complete.

4. When the net is complete, right-click to leave drawing mode.

Selecting Connection Points

You can create a net by selecting the connection points of the net. Connection points can be appropriate pins, ports, or an existing net.

To connect by selecting points:

1. Select the pins, ports, and nets that you want to connect.
2. Right-click one of the selected objects and choose **Connect**.

   If the connection is allowed, a line appears connecting the items. Propel Builder creates a path around other objects.

   If the Connect command is grayed out, the connection is not allowed.

Assigning Constant Values

You can assign constant values to pins or buses.

To assign a constant value to an input pin:

1. Right-click the pin and choose **Assign Constant Value**.

   A small dialog box appears.
2. Enter the desired value. To erase the value and start over, click the X.

   For all buses (more than one pin), the format is hexadecimal. Make sure your value fits the number of pins in the bus. For example, a 3-pin bus can accept 0-7, but not 8.
3. To set the value, click the check mark.

   The value appears over the pin.

Disconnecting

Usually, disconnecting modules just means deleting a net. If the net has multiple branches, you can delete just one branch, leaving the rest of the net intact.

To disconnect one branch of a net:

- Right-click the pin of that branch and choose **Disconnect**.

  The branch going to that pin disappears.

  Note

  If the pin is the signal source for the net, the whole net disappears because a net must have a source.

To disconnect a whole net:

- Right-click the net and choose **Delete**.

  The whole net is deleted.

Creating Top-Level Ports

With multiple modules in a Propel Builder project, you need to create top-level ports. These will be the ports for the complete Propel Builder module.There are two methods: manual and automatic.

For input ports that connect to more than one pin, such as for clock and reset signals, the manual method is more convenient than automatic. Also, if you would want to change the automatically generated port names, you might prefer the manual method.

For other pins, the automatic method is usually preferred. Automatic creates the appropriate port type and connecting net simultaneously. Automatic can also create multiple ports at the same time.

Automatically creating ports is most effective if you do all of the manual module and port connections first. If you then select all the modules, Propel Builder can automatically create ports for all the remaining pins in one step.

Manually Creating Ports

When creating a port manually, you specify all of the port's characteristics, such as name and direction. Ports for pinifaces (pins with a plus sign ⊞) have different characteristics than ports for regular pins and buses, so there are different processes.

After creating the port, you need to manually connect it to the pins of modules.

To manually create a port for a regular pin or bus:

1. Right-click in the Schematic view and choose **Create Port**.
2. In the Create Port dialog box, enter a name for the port.
3. Choose a direction.
4. Choose a type: **Port** for single bit; **PortBus** for a bus (multiple bits).
5. If you chose PortBus, enter the number for the most significant bit (MSB) and least significant bit (LSB). This defines the width of the bus.
6. Click **OK**.

   The port appears to the left (for an input) or right (for an output or inout) of the schematic. You may need to scroll to see it.
7. Connect the port to the module pins. See "Connecting Modules" on page 11.

To manually create a port for a piniface:

1. Right-click in the Schematic view and choose **Create Interface Port**.
2. In the Create Interface Port dialog box, enter a name for the port.
3. Choose a mode: **Master** for an output port, **Slave** for an input port.

4. Choose a type.
5. Click **OK**.

   The port appears to the left (for an input) or right (for an output or inout) of the schematic. You may need to scroll to see it.
6. Connect the port to the module pins. See "Connecting Modules" on page 11.

Automatically Creating Ports

For most of your top-level ports, the simplest method is to have Propel Builder create the ports automatically with the Export command.

To automatically create ports:

1. Select the pins that will connect to the top-level ports. All the pins in a module can be selected by clicking its block. Any pins that are already connected to a net or a constant value will be skipped.
2. Right-click one of the selected pins or modules and choose **Export**.

   The selected pins are extended by lines to new top-level port symbols. The names of the ports and nets are added to the List view. You may need to zoom out or scroll the image to see the new ports.
3. Change port or net names if you want to. See "Renaming Objects" on page 10.

Adjusting Address Spaces

The Address view shows the base address, size of the address segment, and end address for each leaf memory-mapped slave connection in the Propel Builder project. Propel Builder automatically assigns values, but you can change the base addresses. The ranges were set when the modules were configured. The end addresses are calculated.

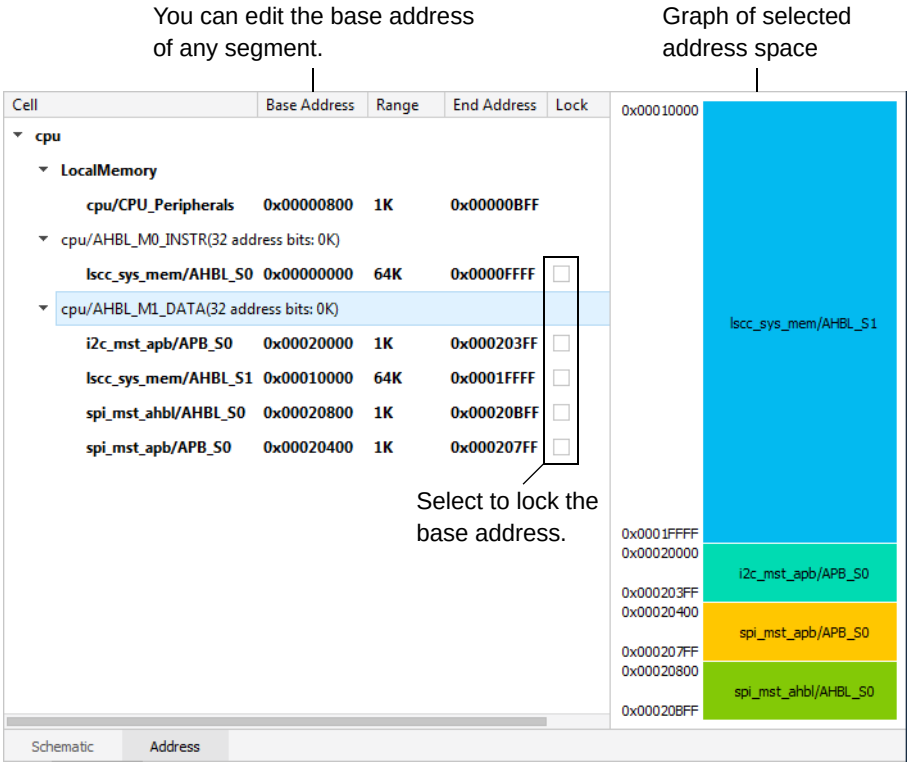
The Lock option on each address space prevents Auto Assign from changing the base address. The Lock option is selected automatically when you manually change the address. To reset the address space, clear the Lock option before clicking the Auto Assign button.

Note that there is no Lock option on LocalMemory. You can always change the base address, but Auto Assign will not reset it to the original value.

To adjust address spaces:

1. Click the Address tab.
2. Set or clear the **Lock** options as desired. If this is the first time adjusting the address spaces, all the Lock options should be cleared.

Figure 2: Address View



3. In Propel Builder's button bar, click the Auto Assign button.

   Default address values appear.

4. Double-click in the base address.

5. Type the new value and press **Enter**. Values must align with 1K boundaries, such as 0x00000400, 0x00000800, and 0x00000C00.

   The end address changes based on the new value. Also, the Lock option is selected.

   If there is a conflict with a related address space, that is shown with red in the graphic.

Validating the Design

You can run design rule checks (DRC) at any time. This checks if there are any illegal connections or overlapping address spaces.

To validate the design:

- Click the Validate Design button.

  The DRC results appear in the Tcl Console.

Generating the RTL File

The final step of creating a Propel Builder module is to generate an .sbx file, which defines a Propel Builder project, the RTL file, and the instantiation templates. The RTL file has the Verilog code for your module. The instantiation templates have Verilog and VHDL code to help instantiate the Propel Builder module in a design.

The Generate button also saves the Propel Builder design and runs design rule checks (DRC).

To generate the module:

1. Click the Generate button.
2. Check the Tcl Console for errors.

About Software Development with Lattice Propel SDK

If your Propel Builder project includes a processor, you will need to create software applications. To help with this, Lattice Semiconductor provides Lattice Propel SDK (Software Development Kit). Lattice Propel SDK is an integrated development environment (IDE) based on the popular Eclipse tools. Propel SDK enables you to develop C or C++ application code that runs on platforms created with Propel Builder, to validate the code on a test board, and to deploy the application to on-chip or flash memory.

Software development should start as soon as possible because simulation testing of the processor platform must wait for the Lattice system memory initialization file (.mem) that is produced by Propel SDK.

For complete information about using Propel SDK, refer to the Help in the Lattice Propel SDK window.

Tcl Commands

You can create your own Tcl scripts to run Lattice Propel Builder. Refer to the following command descriptions. Also refer to the Tcl Console for examples.

sbp_design

The sbp_design command is used for high-level management commands (such as opening and closing) of design files created by Propel Builder.

open Open an existing Propel Builder design for modification.

sbp_design open -name <*design name*> -path <*design path*> [-device <*device name*>]

Example: sbp_design open -name project1 -path project1.sbx

close Close the currently open Propel Builder design.

sbp_design close [-force]

new Create a new Propel Builder design.

sbp_design new -name <*new design name*> -path <*new design path*>

save Save the current design to a file on disk or save as a new file. You can save the design to the project location (Example 1) or to a specific path (Example 2).

sbp_design save [-path <*new design path*>]

Example 1: sbp_design save

Example 2: sbp_design save -path new_design.sbx

drc Run the design rule checker on the Propel Builder design file.

sbp_design drc

generate Generate RTL code to instantiate and connect the IP cores specified in the Propel Builder design file.

sbp_design generate

auto_assign_addresses Automatically assign memory mapped addresses to all the slaves in the system. These addresses should be chosen to avoid slaves with multiple non-contiguous address ranges.

sbp_design auto_assign_addresses

sbp

The sbp command is used to specify connectivity and IP instantiation to the Propel Builder backend.

add_component Instantiate an IP component into the system. Must specify the component VLNV identifier. This will correspond to a component instance

in the IP-XACT design. The example below instantiates an AHB-Lite interconnect component.

sbp_add_component -vlnv <*VLNV*> -name <*instance_name*>

Example: sbp_add_component -vlnv lattice:ip:ahbl_interconnect:1.0 -name ahblite_interconnect

add_port Create a top-level I/O port. Must specify the direction.

sbp_add_port [-from <*bit number*>] [-to <*bit number*>] -direction (in | out | inout) <*port_name*>

add_interface_port Create a top-level bus interface port. Must specify the bus VLNV identifier and either master or slave.

sbp_add_interface_port -direction (master | slave) -vlnv <*VLNV*> <*interface_port_name*>

connect_net Connect all of the specified pins and ports to the same net. The arguments can be pins or ports in the system design. Only one of the arguments can be the driver (output pin or port), driving all other input pins and ports. The example below connects the clk port to all components (assuming component pins are all named clk):

sbp_connect_net [-name <*net name*>] <*pin_port*> <*pin_port*>

Example: sbp_connect_net sbp_get_pins clk {CLOCK_IN}

connect_interface_net Connect a bus interface pin or port to another interface pin or port. This corresponds to the interconnection element in the IP-XACT design.

sbp_connect_interface_net <*pin_port*> <*pin_port*>

connect_constant Connect a constant integer to a pin, pinbus, port, or portbus. To assign to a pin or pinbus, it must be an input pin or pinbus. To assign to a port or portbus, it must be an output port or portbus. If the integer requires multiple bits (is not 0 or 1), then it must be a bus. The Tcl command can be used to assign the same constant to multiple pins, pinbuses, ports, or portbuses at once.

sbp_connect_constant -constant <*integer*> {<*pin_pinbus_port_portbus*>}

Example: sbp_connect_constant -constant 1 {test/i2c_mst_apb/rst_n_i} {test/riscv/clk_i}

disconnect_interface_net Disconnect an interface pin or port from the interface nets that they attached to. Note that any interface pin or interface port can attach to at most one interface net.

sbp_disconnect_interface_net <pin_port> <pin_port>

disconnect_net Disconnect all of the specified input pins and ports from the nets that they attached to. Note that any pin or port can attach to at most one net. Can also be used to disconnect a constant that was connected to a pin or port with connect_constant.

sbp_disconnect_net <*pin0*> <*pin1*> <*port2*>

assign_addr_seg Assigns a memory map between a pair of master and slave interfaces. Range specifies the range of the segment, such as: 32'h0000400, 32'h0001000. Offset specifies the base offset of the range, such as: 32'h0000400.

sbp_assign_addr_seg -offset <*offset*> <*slave connection name*>

Example: sbp_assign_addr_seg -offset 32'h00001000 simple/riscv/AHBL_S00

unassign_addr_seg Unsets the fixed offset flag for a memory map to allow the auto_assign Tcl command to assign the memory map offset.

sbp_unassign_addr_seg <*slave interface name*>

Example: sbp_unassign_addr_seg simple/spi/AHB_S00

assign_local_memory Assigns a base address to a local memory map of a master address space.

sbp_assign_local_memory -offset <*offset*> <*master_addr_space*>

Example: sbp_assign_local_memory -offset 'h0050000 Foundation_SoC/riscv/ahbl_m_data_Address_Space

export_pins Export a list of pins (or interface pins) or all not-yet-connected pins of the components to the design top-level port list. The function will detect whether the arguments are pins or components. Example 1 demonstrates a Tcl command to export the pin init_done. Example 2 demonstrates a Tcl command to export all pins and interfaces of the ddr3 component.

sbp_export_pins <*pin_component*> <*pin_component*>

Example 1: sbp_export_pins {ddr3/init_done}

Example 2: sbp_export_pins {ddr3}

export_interface Exports bus interfaces that are passed as arguments to the Tcl command from the component to the top-level component. The example below exports the AHBL_MASTER bus interface of the RISC-V component to the top-level component:

sbp_export_interfaces <*interface*> <*interface*> <*interface*>

Example: sbp_export_interfaces simple/riscv/AHBL_MASTER

rename Renames objects within the design. The Tcl command takes as input the new name of the object and the current hierarchical name of the given object. The object could be an interface connection, connection, port, interface, or component. The example below demonstrates the changing of the name of a port in the milestone project from CLK to CLOCK.

sbp_rename -name <*new name*> <*object name*>

Example: sbp_rename -name CLOCK milestone/CLK

replace Replaces a component with a new configuration of itself. Then the Tcl command creates the necessary connections that were there for the component of ports, port busses and bus interfaces that still exist. VLNV refers to the newly generated IP. Component name refers to the existing component under re-config (to be replaced). Instance name refers to the name of the reconfigured component.

sbp_replace -vlnv <*VLNV*> -name <*instance*> -component <*component name*>

Example: sbp_replace -vlnv lattice:ip:ahblite_bus_0:1.1 -name ahbl_bus_0 -component simple/ahblite

copy Copy objects (IP instances and nets) from the current or other open sbp designs to the current sbp design. All objects will be postfixed with a postfix String. The example below duplicates the components and connections with new instance names postfixed with X, by calling copy on all the components, ports, and nets. Pins are automatically duplicated as the components being duplicated.

sbp_copy -postfix <*postfixString*> objects

Example: sbp_copy -postfix X $selected_objs

delete Delete objects (IP instances and nets) from the current sbp design. Example 1 demonstrates a Tcl command to delete a port. Example 2 demonstrates a Tcl command to delete a ddr3 component.

Sbp_delete objects -type <*type name*>

Example 1: sbp_delete [sbp_get_ports <clock>] -type port

Example 2: sbp_delete {ddr3} -type component

get_components Get a list of component names that match a pattern string, and/or the components that are associated with an object. The default pattern string is a wildcard “*” that matches all components. The pattern string may consist of string segments and wildcards. The example returns all the component names that contain “interconnect”. The command returns an empty string if no match is found.

sbp_get_components <*component name*>

Example: sbp_get_components {*interconnect*}

get_pins Get a list of pin names that match a pattern string and/or the pins that are associated with an object. The object in the [-from <*objectName*>] option can be a net or a component. The example below gets the clk pin from the interconnect IP.

sbp_get_pins [-from <*objectName*>] [pattern]

Example: sbp_get_pins -from ahblite_interconnect clk

get_interface_pins Get a list of interface names that match a pattern string and/or the interfaces that are associated with an object. The object in the [-from <*objectName*>] option can be an interface net or a component. The example below gets all AHB-Lite slave interface pins from the interconnect IP.

sbp_get_interface_pins [-from <*objectName*>] [pattern]

Example: sbp_get_interface_pins -from ahblite_interconnect S*_AHB

get_ports Get a list of the names of ports that match a pattern string and/or the ports that are associated with an object. The object in the [-from <*objectName*>] option can be a net.

sbp_get_ports [-from <*objectName*>] [pattern]

get_interface_ports Get a list of interface names that match a pattern string and/or the interface ports that are associated with an object. The object in the [-from <*objectName*>] option can be an interface net.

sbp_get_interface_ports [-from <*objectName*>] [pattern]

get_nets Get a list of net names that match a pattern string and/or the nets that are associated with an object. The object in the [-from <*objectName*>] option can be a pin or a port.

sbp_get_nets [-from <*objectName*>] [pattern]

get_interface_nets Get a list of interface net names that match a pattern string and/or the interface nets that are associated with an object. The object in the [-from <*objectName*>] option can be an interface pin or an interface port.

sbp_get_interface_nets [-from <*objectName*>] [pattern]

set_property Set the properties of an input object. The first argument is a list of name value pairs. You can submit changes to many parameters of an IP component at once. Example 1 changes the data width of the RAM block named “ebr_0.” Example 2 changes the number of master interfaces to two and number of slave interface to three on an AHB-Lite interconnect block named “ahbl_interconnect”.

sbp_set_property <*name0 value0 name1 value1* …> object

Example 1: sbp_set_property {datawidth 32} {test/ebr_0}

Example 2: sbp_set_property {NUM_MI 2 NUM_SI 3} test/ahbl_interconnect

get_property Get the property of the object. The example below gets the number of slave interfaces of an AHB-Lite interconnect block named “ahbl_interconnect_0.”

sbp_get_property <*parameter name*> <*object*>

Example: sbp_get_property NUM_SI ahbl_interconnect_0

report_properties Print all the properties and values associated to the type of the object. The example below print outs:

NAME: ahbl_interconnect
NUM_SI: 1
NUM_MI: 2
DATA_WIDTH: 32
ADDRESS_WIDTH: 32

sbp_report_properties object

Example: sbp_report_properties ahbl_interconnect

Revision History

The following table gives the revision history for this document.

Date	Version	Description
4/20/2020	1.0	Initial release.