# Revision Control – Lattice Propel Builder 2026.1

# User Guide

FPGA-UG-02252-1.0

June 2026

## Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

## Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

# Contents



# Figures

# Abbreviations in This Document

A list of abbreviations used in this document.

| Abbreviation | Definition |
|---|---|
| FPGA | Field Programmable Gate Array |
| GUI | Graphical User Interface |
| IP | Intellectual Property |
| SDK | Software Development Kit |
| SoC | System on Chip |
| SVN | Subversion software, an open-source version control system |
| TCL | Tool Command Language |
| VHDL | VHSIC Hardware Description Language |

# 1. Introduction

This user guide introduces how to use revision control for Lattice Propel™ projects. The Lattice Propel design environment supports revision control for your projects. You can access the revision control system to get all the change logs and switch to previous milestones if necessary. Using revision control for Lattice Propel projects, compile time is reduced. Compilation starts only when inputs change.

# 2. Recommended Revision Control Systems

Lattice recommends the following revision control systems for use with Lattice Propel software:

- Git
  Git is an open-source distributed version control system.
- SVN
  The Subversion software is an open-source version control system.
- Perforce
  Perforce is a centralized version control system.

# 3. Lattice Propel Revision Control Strategies

The Lattice Propel design environment works with many revision control systems, and you do not need to consider version compatibility. The Lattice Propel design environment does not directly integrate with any specific revision control system. You can select different revision control systems to work with the Lattice Propel design environment.

Only the necessary files that constitute the project are managed by the revision control system. Other files, including some intermediate files and your configuration files, are only used in your own project and are not submitted to the version control system.

## 3.1. Lattice Propel Project Directory Structure

In the Lattice Propel design environment, place all source code and IP designs on the same disk partition rather than across partitions.

Lattice Propel projects use one directory for all source and project files. After creating a project, the directory structure of the project is shown in Figure 3.1 and Figure 3.2.

All source files are stored in the folder named after the project, such as the hello_world folder shown in Figure 3.1.

```
├── hello_world
│   ├── application           -- Folder containing functional-ready embedded application source
│                                codes(only exists for template SoC with legacy C design flow)
│   │   ├── main.c
│   │   ├── utils.c
│   │   └── utils.h
│   ├── hello_world.layout    -- Created after running “Generate”
│   ├── hello_world.sbx       -- Builder Project file
│   ├── hello_world_tmpl.v    -- Verilog instantiation template top file
│   ├── hello_world_tmpl.vhd  -- VHDL instantiation template top file
│   ├── hello_world.v         -- Verilog file
│   └── lib                   -- IP instances files under the folder
│       └── latticesemi.com
│           ├── ip
│           └── module
├── hello_world.pdc            -- Radiant constraint file
├── sge                        -- The folder created after running “Generate”
│   ├── bsp
│   │   ├── driver             -- Folder containing driver codes from the IP in the platform
│   │   │   ├── gpio
│   │   │   ├── riscv_mc
│   │   │   └── uart
│   │   ├── riscv_errors.h
│   │   └── sys_platform.h
│   ├── cpu0.yaml              -- Processor description file used at debugging time
│   ├── soc_svd                -- System view description file used for peripherals registers
│                                 view at debug perspective.
│   │   └── hello_world.svd
│   └── sys_env.xml            -- System environment file describing aspects of the platform,
│                                 such as memory spaces
├── .socproject                -- Propel project file, internal file
```

**Figure 3.1. Source Files**

If the Propel project has a verification design, the related files are placed in a verification directory, as shown in Figure 3.2.

```
├── verification
    ├── hello_world_v.json      -- File describes current simulator and do file path
    ├── hello_world_v.sbx       -- Verification project file
    └── lib                     -- IP instances files for verification project
        └── latticesemi.com
            └── sim_model
```

**Figure 3.2. Verification-related Files**

If there is an extra C/C++ project, all typical files are shown in Figure 3.3.

```
├── riscv_xx_helloworld
    ├── .settings                   -- C/C++ project setting
    ├── Debug                       -- Project building output files
    ├── Release                     -- Project building output files
    ├── src                         -- Source files including c, head and link files
        ├── bsp                     -- SoC drive files
        ├── main.c                  -- C/C++ project main file
        ├── platform_init.h         -- C/C++ project head file
    ├── .cproject                   -- C/C++ project file
    ├── .project                    -- Project file (link C and SoC project)
    ├── riscv_nano_helloworld.launch  -- C/C++ project launch file
```

**Figure 3.3. Verification-related Files for an Extra C/C++ Project**

## 3.2. Lattice Propel File Types

The project and constraint file types under a Lattice Propel project directory include:

- .socproject file: Propel project file, internal file
- .pdc file: Radiant constraint file
- .rdf file: Radiant project file
- .lpf file: Diamond constraint file
- .ldf file: Diamond project file
- .sty file: Diamond/Radiant strategy file

The basic source or design file types include:

- .v file: Verilog file
- .vhd file: VHDL file
- .sv file: SystemVerilog file
- .sbx file: Builder Project file

## 3.3. Files Committed to Lattice Propel Revision Control Strategies

The following files are committed to revision control strategies of the Lattice Propel design environment:

- The .socproject project file and .pdc/.lpf constraint file
- All the source files under the directory named after the project
- Verification directory files
- All files under the C/C++ project except the Debug and Release folder

For an SoC project named hello_world, the files committed are shown in Figure 3.4.

```
(Propel SoC Project)
├── hello_world
    └── *.*
├── sge
    └── *.*
├── hello_world.pdc
├── hello_world.rdf
├── hello_world1.sty
├── .socproject
└── verification
```

**Figure 3.4. Files Committed for an SoC Project**

For a C/C++ project named riscv_nano_helloworld, the files committed are shown in Figure 3.5.

```
(Propel C/C++ Project)
├── .settings
├── src
├── .cproject
├── .project
└── riscv_nano_helloworld.launch
```

**Figure 3.5. Files Committed for a C/C++ Project**

# 4. Lattice Propel Revision Control Workflows

The Lattice Propel revision control workflows are presented in the following sections.

## 4.1. Lattice Propel Revision Control Normal Workflow

Each developer has a working directory. You can get the latest code from the Lattice Propel version server and develop in your own development environment. Later, you can submit the modified code or new design to the Propel version server if needed.

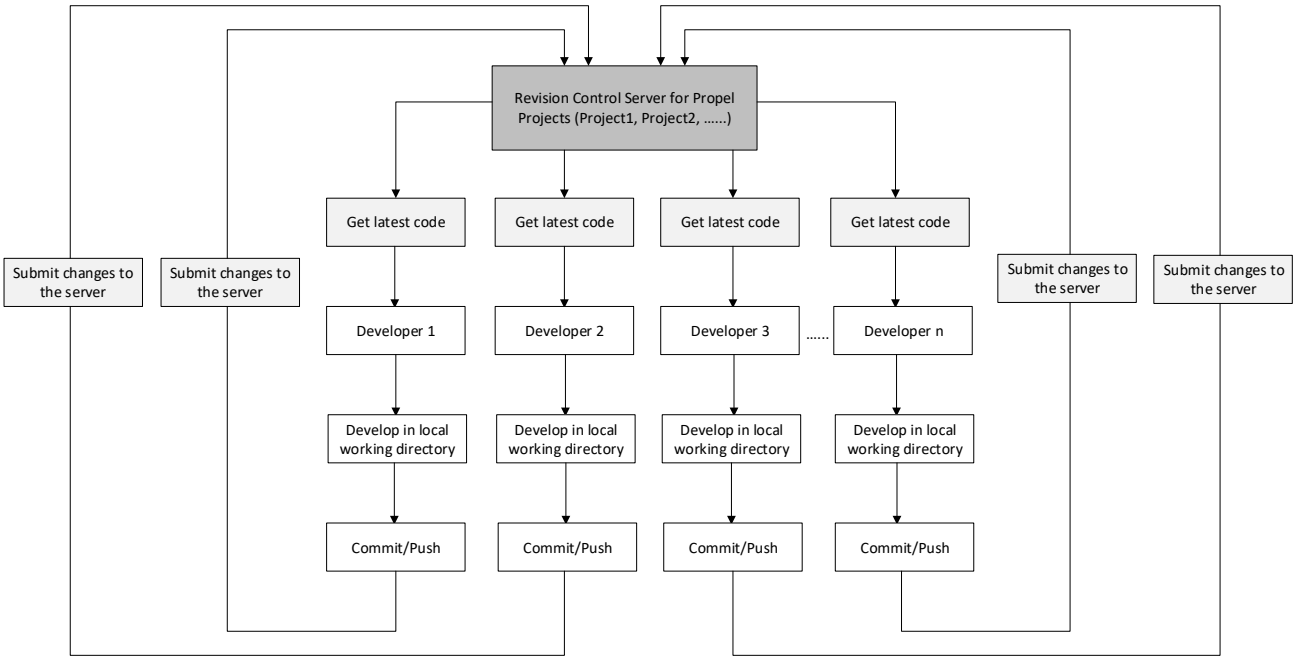


**Figure 4.1. Lattice Propel Revision Control Normal Workflow**

## 4.2. Multiple Developers Working on the Same File Workflow

Most of the time, developers have their own code changes and may modify the same source file or design. If developer A submits a file and developer B has updated the file, then developer A gets an error. At this time, developer A needs to update the latest code locally from the server, integrate it with their own modifications, and submit it again. This is the collaborative operation of multiple developers for revision control.

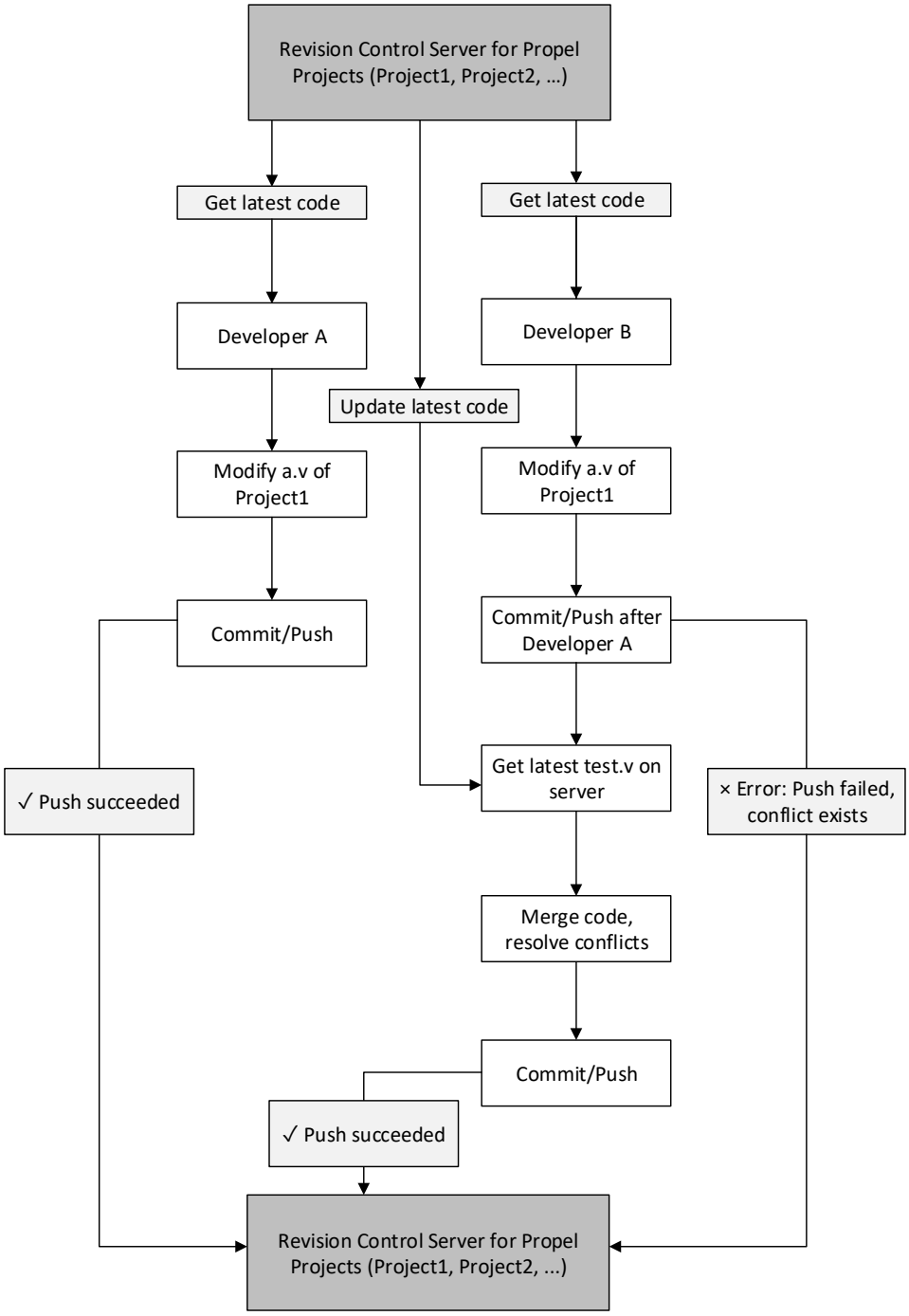


**Figure 4.2. Multiple Developers Working on the Same File Workflow**

## 4.3. Multiple Developers Adding Files Workflow

Developer A may add a new file a.v to the design and submit it together with the Propel project file to the revision control server. Developer B can then pull the latest code from the server, merge the local code, resolve the conflict, and continue the subsequent development.

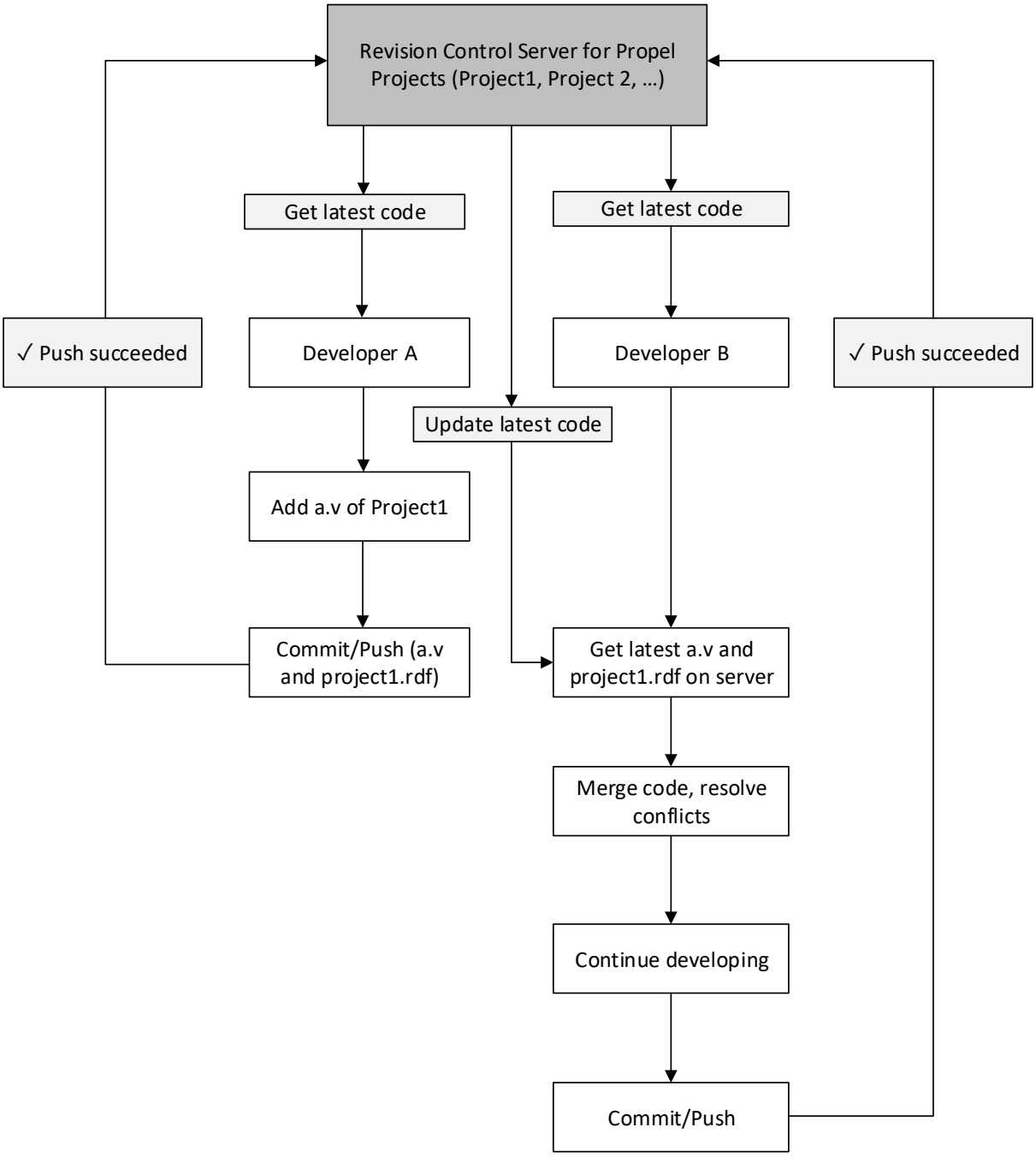


**Figure 4.3. Multiple Developers Adding Files Workflow**

# 5. Reproduce SoC Design Using TCL Script

You can generate a TCL script from the current SoC project, and then set up a new project by launching this TCL script in the new environment.

Any Propel Builder project can be exported as a TCL script. The generated TCL script contains the required TCL commands to recreate the SoC design from scratch, generate all of its IP, and make the required interface and port connections. This script can be used to create a new project from its source project entirely from scratch.

## 5.1. Generate TCL Script for Existing SoC Design

To generate the TCL script:

1. In Lattice Propel Builder, save the current SoC project by clicking the  icon.
2. Type the following command in the TCL console with corresponding options:

   ```
   sbp_design gen_tcl -proj_dir <project directory> -proj_name <project name>
   [-propel_dir <propel directory>] [-o <output name of the generated script>]
   ```

   For example:

   ```
   sbp_design gen_tcl -proj_dir D:/lscc/my_workspace -proj_name test -o newsbx.tcl
   ```

   The output of this command shows the location of the TCL script, as shown in the example below (Figure 5.1):

   ```
   Tcl file C:/lscc/workspace/Example/Example/newsbx.tcl was generated successfully.
   ```

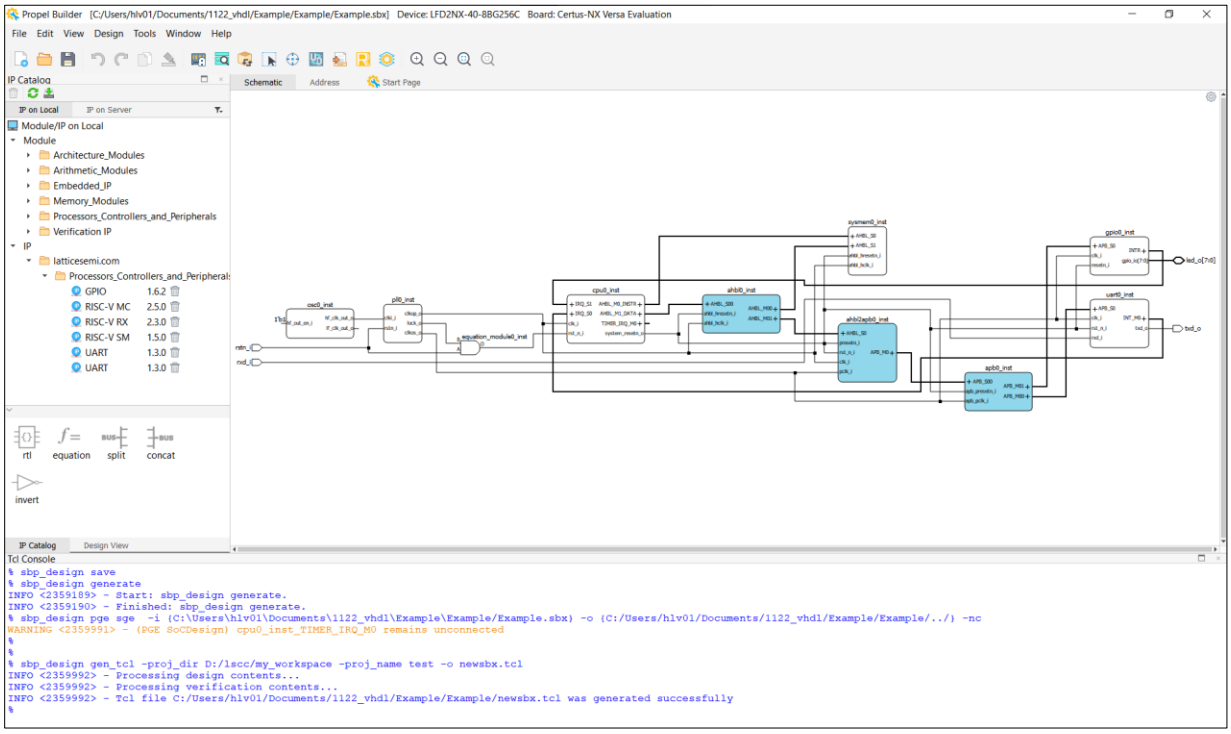


**Figure 5.1. Generate TCL Script for Existing SoC Design**

## 5.2. Reproduce SoC Design

You can use the TCL script generated for the existing SoC project to reproduce a new SoC project.

1. Before launching this TCL script in a new environment, verify that all IPs mentioned in the script are installed in the new environment. The IP names and versions must match.

You can check all IPs from the **IP Catalog** view (Figure 5.2).

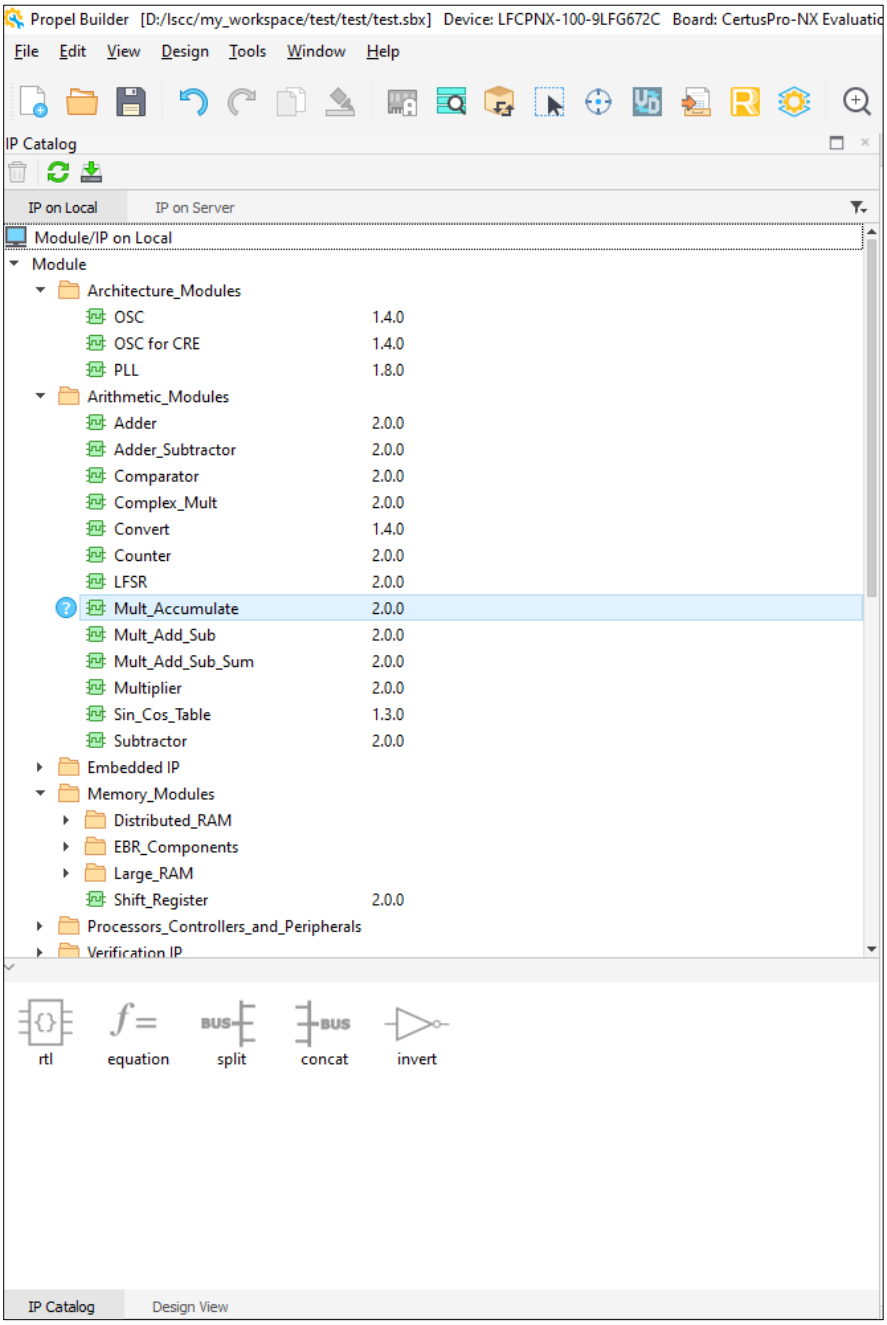


**Figure 5.2. IP Catalog View**

2. Or you can check all IPs by typing the following command in the TCL console (Figure 5.3):
   `ip_catalog_list`

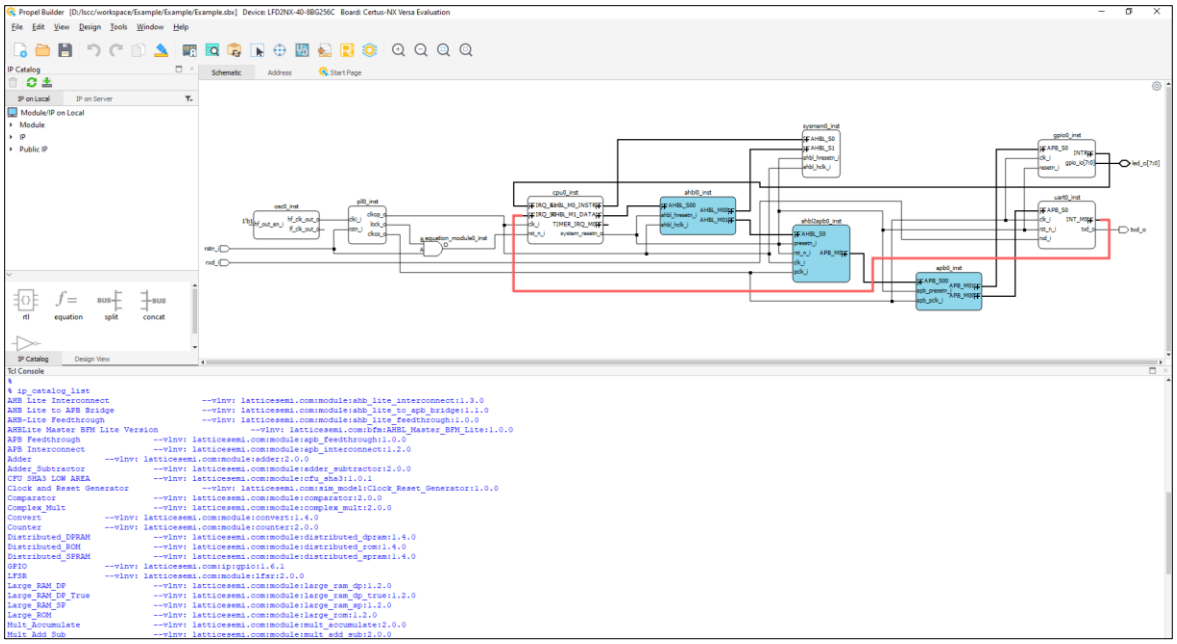

**Figure 5.3. IP Catalog List Command**

3. Fetch the TCL script, and if needed, change the corresponding settings in the script for the new environment:
   - targetDir: The new project directory. verify that this directory exists before launching TCL.
   - projectName: Project name.
   - propelRegenLsccBase: The Propel installation path. Verify that this directory exists.

   **Note:** If there is any file configuration in the sbp_config_ip command, update the file path to the correct path on the local setup. For example, update INIT_FILE in the System Memory IP.
4. To launch this TCL script to create a new project, verify that no design is currently open in Lattice Propel Builder. Type `source <tcl_path>` in the TCL console, as shown in the example below (Figure 5.4):

   ```
   source D:/lscc/workspace/Example/Example/newsbx.tcl
   ```

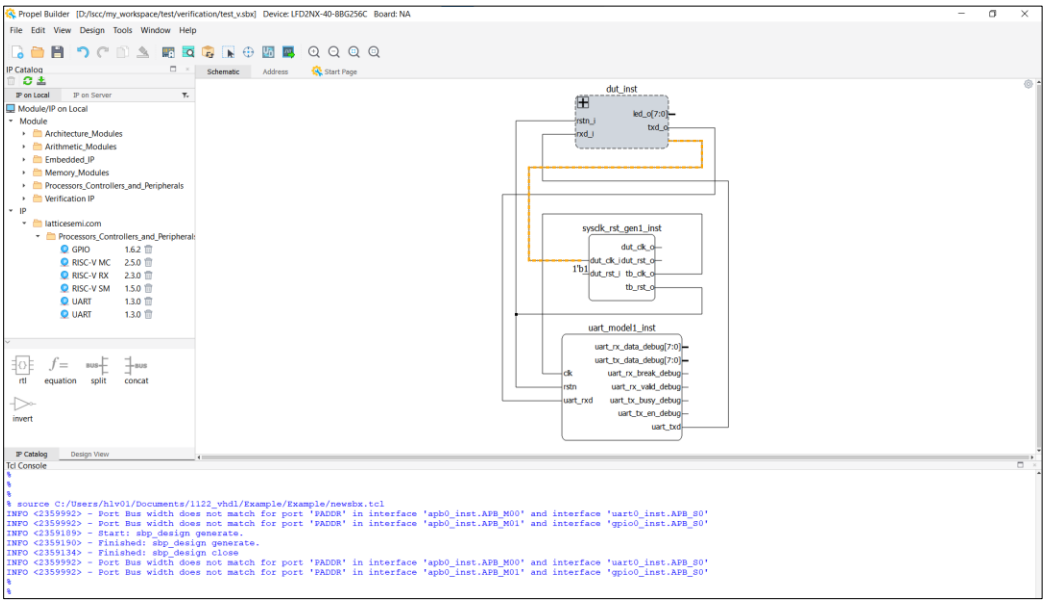

**Figure 5.4. Launch TCL Script to Create New Project**

5. Now you can click the  icon from the Propel Builder toolbar to switch back to the SoC Design project.

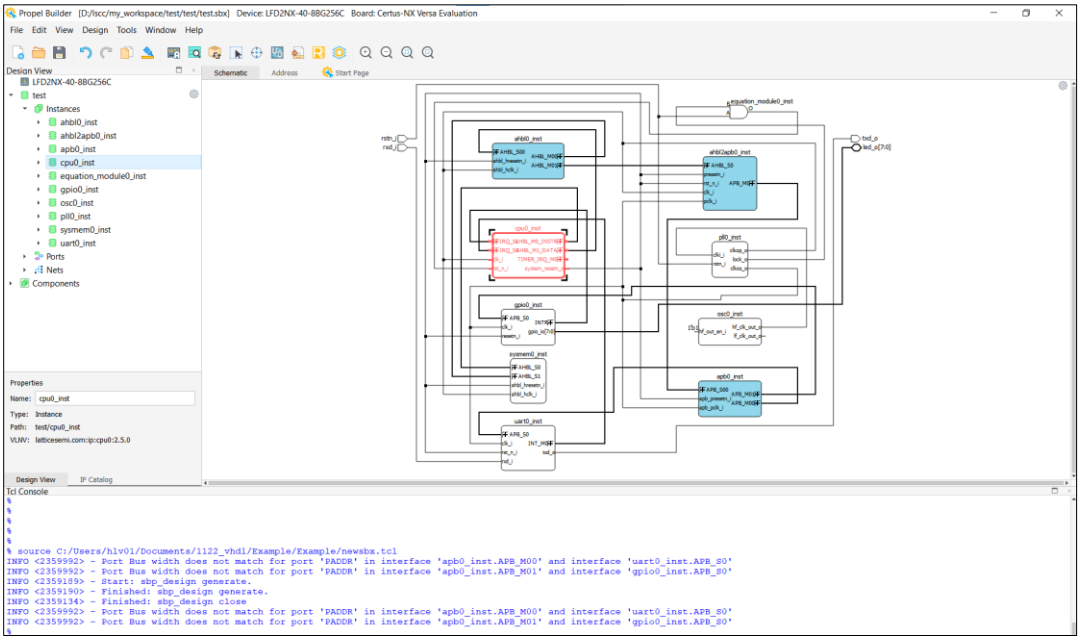


**Figure 5.5. SoC Design View**

6. (Optional) Right-click the Schematic view and select Relayout if the new design is out of proportion (Figure 5.6).

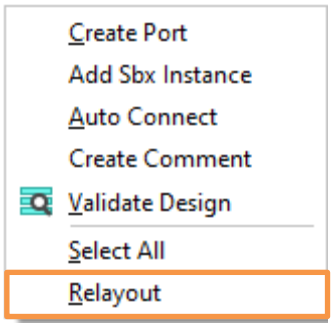


**Figure 5.6. Relayout Option**

7. Add or modify the following files to fully reproduce the original SoC design.

   **For devices supported in Lattice Radiant software:**

   a. If the SoC is at the board level, copy the Lattice Radiant constraint file from the source project (Figure 5.7). The path is `<workspace>/<project_dir>/<project_name>.pdc`.

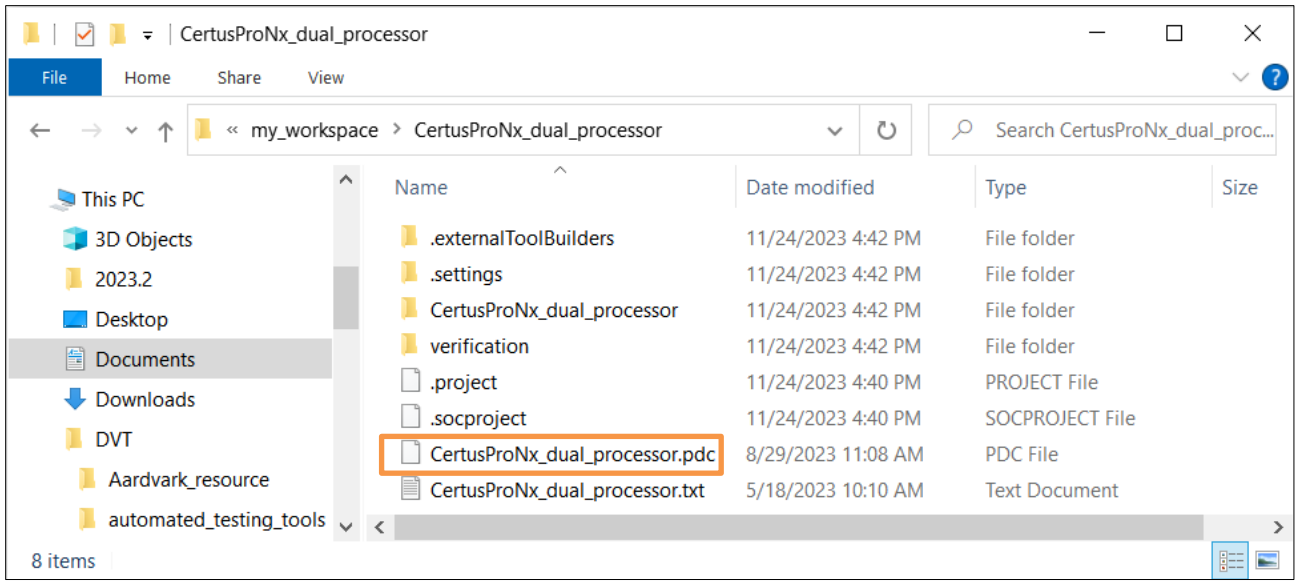


**Figure 5.7. Lattice Radiant Constraint File in Source Project**

b. Rename the copied constraint file with the new project name, as shown in Figure 5.8.

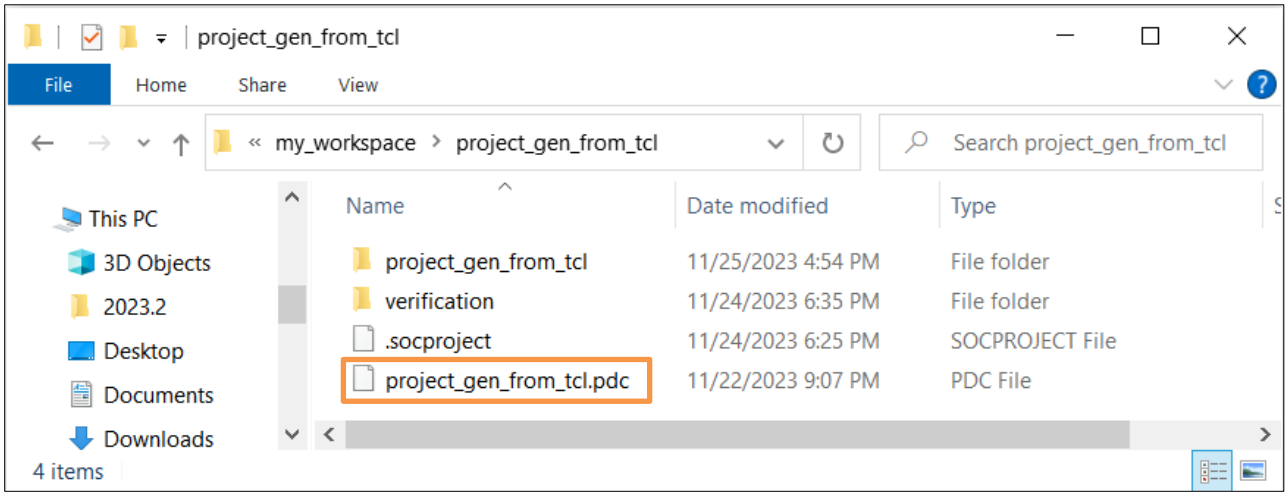


**Figure 5.8. Rename Lattice Radiant Constraint File Copied from Source Project**

c. Open Lattice Radiant software by clicking the icon on the Propel Builder GUI toolbar.

d. Type the following commands in the Radiant TCL console (Figure 5.9).

```
prj_set_strategy_value -strategy Strategy1 par_place_iterator=10
prj_set_strategy_value -strategy Strategy1 par_stop_zero=True
prj_save
```

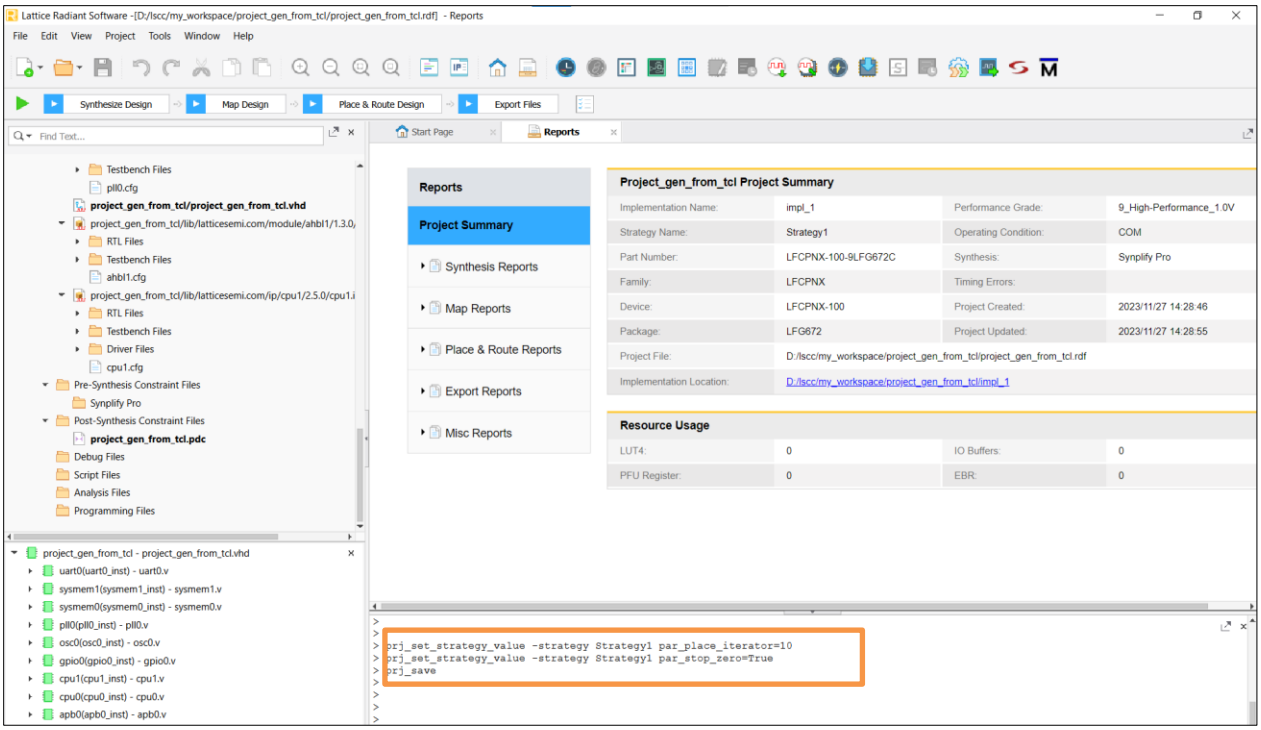


**Figure 5.9. Radiant TCL Console**

e. For some templates, there is an application folder inside the project directory (Figure 5.10). This folder is used for creating a C project in Propel SDK. Copy it from the source project.

The path is `<workspace>/<project_dir>/<project_name>/application`.

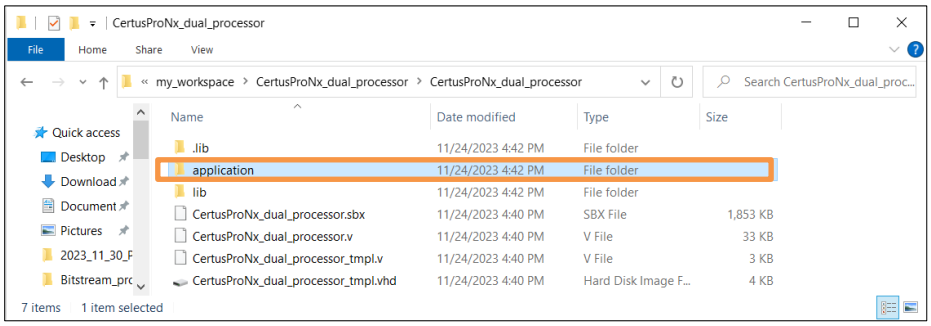


**Figure 5.10. Application Folder in Source Project**

f. Edit the .socproject file to add the ConstraintFile and AddResources entries if there is an application folder in step e, as shown in the example below.

The file path is `<workspace>/<project_dir>/.socproject`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<propelProject>
    <builder-resource>
        <socProject sbxfile="./project_gen_from_tcl/project_gen_from_tcl.sbx" />
        <verifyProject sbxfile="./verification/project_gen_from_tcl_v.sbx" />
        <ConstraintFile>./project_gen_from_tcl.pdc</ConstraintFile>
        <AddResources>
            <AddResource SoC="False">./project_gen_from_tcl/application</AddResource>
        </AddResources>
    </builder-resource>
    <builderInfo version="2023.2" />
</propelProject>
```

**For devices supported in Lattice Diamond software:**

a. Copy top.v from the source project, which is used for the Diamond flow (Figure 5.11).
The file path is `<workspace>/<project_dir>/<project_name>/<project_name>_Top.v`.

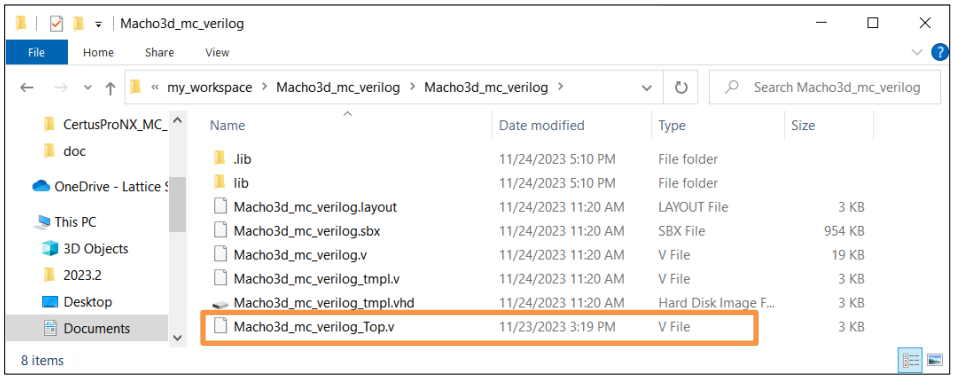


**Figure 5.11. Top.v File in Source Project**

b. Rename the file to `<new_project_name>_Top.v` (Figure 5.12).

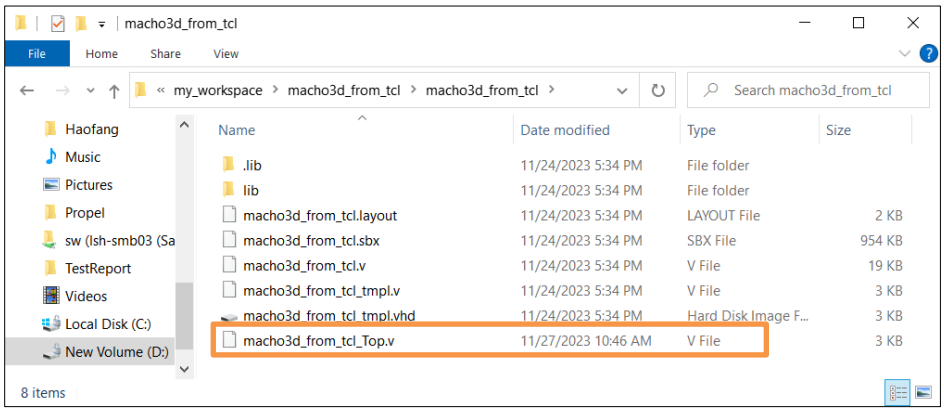


**Figure 5.12. Rename Top File Copied from Source Project**

c. Edit contents of the file, replacing the project name with the new one (Figure 5.13).

```
module HelloWorld_Top (
    input  rstn_i,
    input  rxd_i,
    output txd_o,
    inout [7:0] led_o
);

GSR GSR_INST(.GSR(rstn_i));
wire sys_clk /*synthesis syn_keep = 1*/;
wire esb_oscclk;
OSCJ #(.NOM_FREQ("38.0")) OSCJ (.STDBY(1'b0), .OSC(sys_clk), .SEDSTDBY(),.OSCESB(esb_oscclk));

test_macho3d_mc_tcl HelloWorld_inst (
    .clk_i(sys_clk),
    .rstn_i(rstn_i),
    .rxd_i(rxd_i),
    .txd_o(txd_o),
    .gpio_io(led_o)
);

endmodule
```

**Figure 5.13. Updated Top File in New Project**

d. For a board-level SoC project, copy the Lattice Diamond constraint .lpf file from the source project (Figure 5.14).

   The path is `<workspace>/<project_dir>/<project_name>.lpf`.

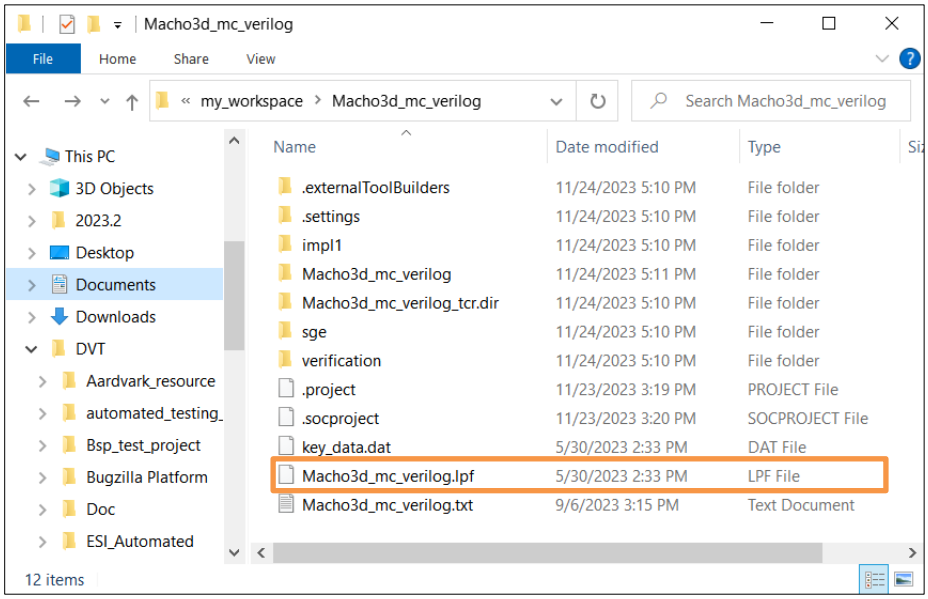


**Figure 5.14. Lattice Diamond Constraint File in Source Project**

e. Rename the file with the new project name (Figure 5.15).

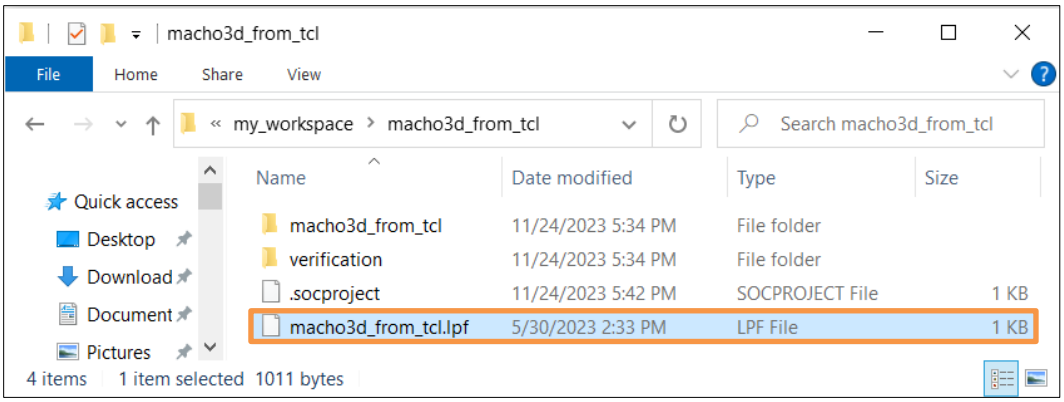


**Figure 5.15. Rename Lattice Diamond Constraint File Copied from Source Project**

f. Open Lattice Diamond software by clicking the  icon in the Propel Builder GUI toolbar. Type the following command in the Diamond TCL console (Figure 5.16):

```
prj_strgy set_value -strategy Strategy1 {par_place_iterator=10}
{par_stop_zero=True}
prj_project save
```

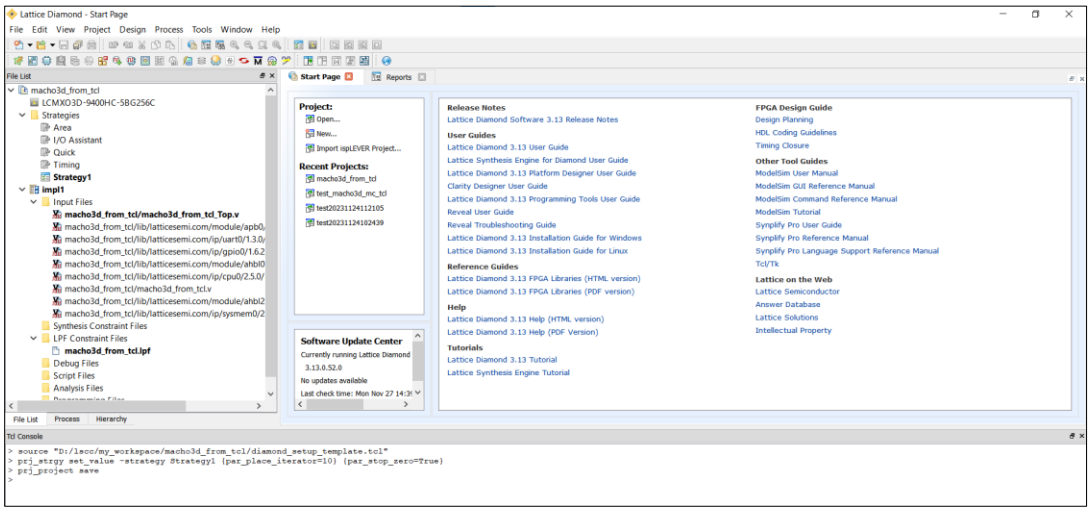


**Figure 5.16. Diamond TCL Console**

g. Check the <workspace>/<project_dir>/<project_name> directory. If there are any other resources, copy them into the same directory under the new project without modification (Figure 5.17).

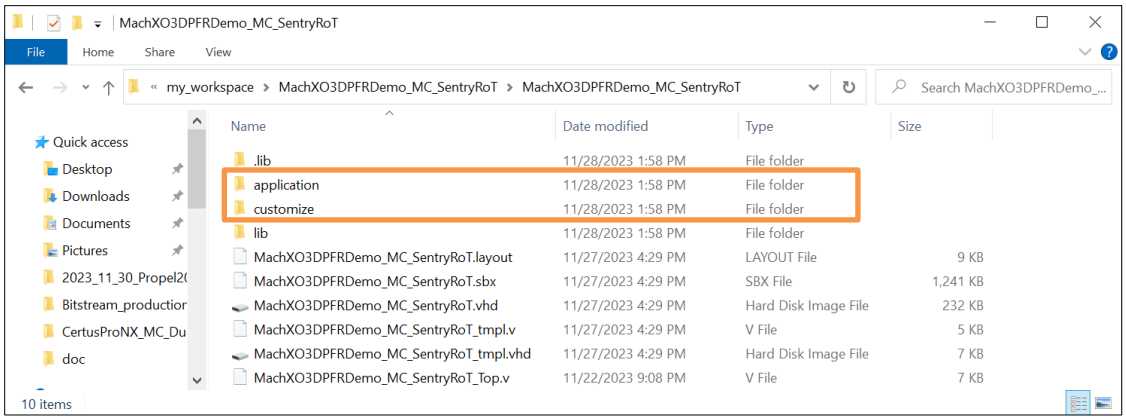


**Figure 5.17. Other Resources in Source Project**

h. Edit the .socproject file to add the ConstraintFile entry. If there are other resources in step g, add them to the AddResources entry, as shown in the example below.

   The file path is `<workspace>/<project_dir>/.socproject`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<propelProject>
    <builder-resource>
        <socProject sbxfile="./macho3d_from_tcl/macho3d_from_tcl.sbx" />
        <verifyProject sbxfile="./verification/macho3d_from_tcl_v.sbx" />
        <ConstraintFile>./macho3d_from_tcl.lpf</ConstraintFile>
        <AddResources>
            <AddResource SoC="False">./macho3d_from_tcl/application</AddResource>
            <AddResource SoC="False">./macho3d_from_tcl/customize</AddResource>
        </AddResources>
    </builder-resource>
    <builderInfo version="2023.2" />
</propelProject>
```

# 6. References

- Lattice Propel 2026.1 SDK User Guide (FPGA-UG-02255)
- Lattice Propel 2026.1 Builder User Guide (FPGA-UG-02254)
- Lattice IP Packager 2026.1 User Guide (FPGA-UG-02253)
- Lattice Propel 2026.1 Installation for Windows User Guide (FPGA-AN-02117)
- Lattice Propel 2026.1 Installation for Linux User Guide (FPGA-AN-02116)

For more information, refer to:

- Lattice Propel software web page
- Lattice Insights for training series and learning plans

# Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

# Revision History

**Revision 1.0, June 2026**

| Section | Change Summary |
|---|---|
| All | Production release. |


www.latticesemi.com