Lattice Propel 1.0

User Guide

FPGA-UG-02110-1.0

May 2020

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 and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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.

Contents

Figures

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
BSP	Board Support Package, the layer of software containing hardware-specific drivers and libraries to function in a particular hardware environment.
DUT	Design Under Test.
SDK	Embedded System Design and Develop Kit. A set of software development tools that allows the creation of applications for software package on the Lattice embedded platform.
IDE	Integrated Development Environment.
Workspace	The directory where stores your work, it is used as the default content area for your projects as well as for holding any required metadata.
Workbench	Refers to the desktop development environment in Eclipse IDE platform.
Programmer	A tool can program Lattice FPGA SRAM and external SPI Flash through various interfaces, such as JTAG, SPI, and I^2C.
Perspective	A group of views and editors in the Workbench window.
RISC-V	A free and open instruction set architecture (ISA) enabling a new era of processor innovation through open standard collaboration.
SoC	System on Chip. An integrated circuit that integrates all components of a computer or other electronic systems.
SRAM	Static Random Access Memory.
UFM	User Flash Memory.

1. Introduction

Lattice Propel™ 1.0 is a complete set of graphical and command-line tools to create, analyze, compile, and debug both FPGA-based hardware and software processor systems.

1.1. Purpose

Embedded system solutions take an important role in FPGA system design allowing you to develop software for a processor in an FPGA device. It provides flexibility for you to control various peripherals from a system bus.

To develop an embedded system on an FPGA, you need to design the System-on-Chip (SoC) with an embedded processor and develop system software on the processor. Lattice Propel™ helps you develop your system with a RISC-V processor, peripheral IP, and a set of tools.

The purpose of this document is to introduce Lattice Propel development suite and flow to help you quickly get started to build a small demo system. You can find recommended flows of using Lattice Propel in this document as well.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice MachXO3D FPGA devices. The technical guidelines assume readers have expertise in the embedded system area and FPGA technologies.

2. Lattice Propel Development Suite

Lattice Propel development suite includes:

- an integrated development environment (IDE), which is the framework of Propel;
- Lattice Propel Builder, which is for SoC design;
- Lattice Propel SDK, which is for system software development.

2.1. Eclipse IDE

Eclipse IDE provides the Propel development suite a platform to manage the SoC project and the Embedded C Project in the same workspace.

The SoC project, which extends from the Propel Builder project, provides easy interaction with other Lattice design tools, such as Lattice Diamond 3.11 within Propel.

The Embedded C Project provides a platform for developing or debugging application code within Eclipse IDE. The project can be created directly from the SoC project with a pre-set Board Support Package (BSP) and applications by using Propel development suite.

2.2. Lattice Propel Builder

Lattice Propel Builder allows you to assemble the larger functional blocks of the design hierarchy. Propel Builder enables you to instantiate modules and IP from the IP Catalog in a schematic view, and can easily connect the modules. Propel Builder also helps you customize address spaces within modules, such as a processor.

In Propel development suite, Propel Builder is used to create a microprocessor integrated platform for both hardware and software development.

2.3. Lattice Propel SDK

Lattice Propel SDK is based on Eclipse Embedded CDT (C/C++ Development Tools). It allows you to create, build, and debug software application projects that drive the platform within the Eclipse framework.

The main features are:

- Create, build, debug, or manage embedded applications for Lattice RISC-V CPU/SoC solution.
- Provide extra build steps to generate the binary and memory files required for deployment.
- Build using the latest industry standard open source components and tools for RISC-V firmware development and debugging.
- Support Newlib (and nano) for RISC-V and provide lightweight standard library implementation.
- Provide fully-configurable toolchain definitions.

3. Lattice Propel Tool Flows

The Propel tool flows including SoC project design flow, C project design flow, system simulation flow, and programming and On-Chip-Debugging (OCD) debugging flow, are discussed in detail in the following sections.

3.1. Propel Environment

After installing Lattice Propel 1.0, you can launch Propel from the desktop shortcut icon or from the Start menu. When Propel is invoked, a dialog (Figure 3.1) pops up. You can browse to select where to locate the workspace. For normal needs, just click **Launch** to pick the default location and continue running Propel.

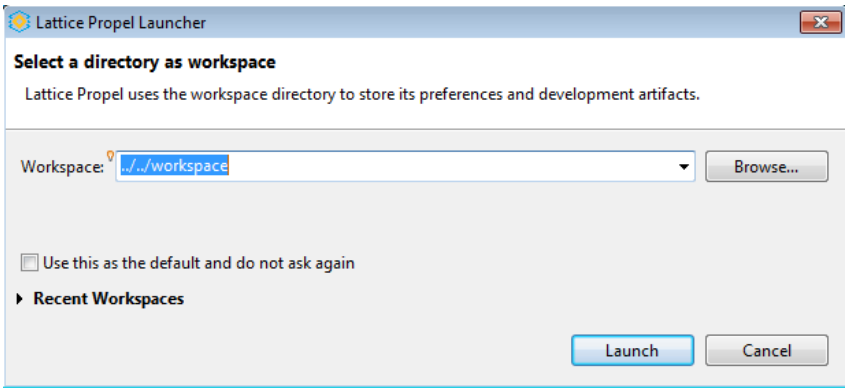


Figure 3.1. Select Workspace Dialog

After the workspace location is chosen, a single workbench window is displayed using default C perspective. The default C perspective contains following five functional areas (Figure 3.2).

1. Menu bar and Toolbar.
2. Project Explorer view: displays projects in the workspace.
3. Editor view: provides capability of editing source files.
4. Outline view: displays an outline of a file that is currently open in the editor area.
5. Problem view, Tasks view, Console view, Properties view, and Terminal view.

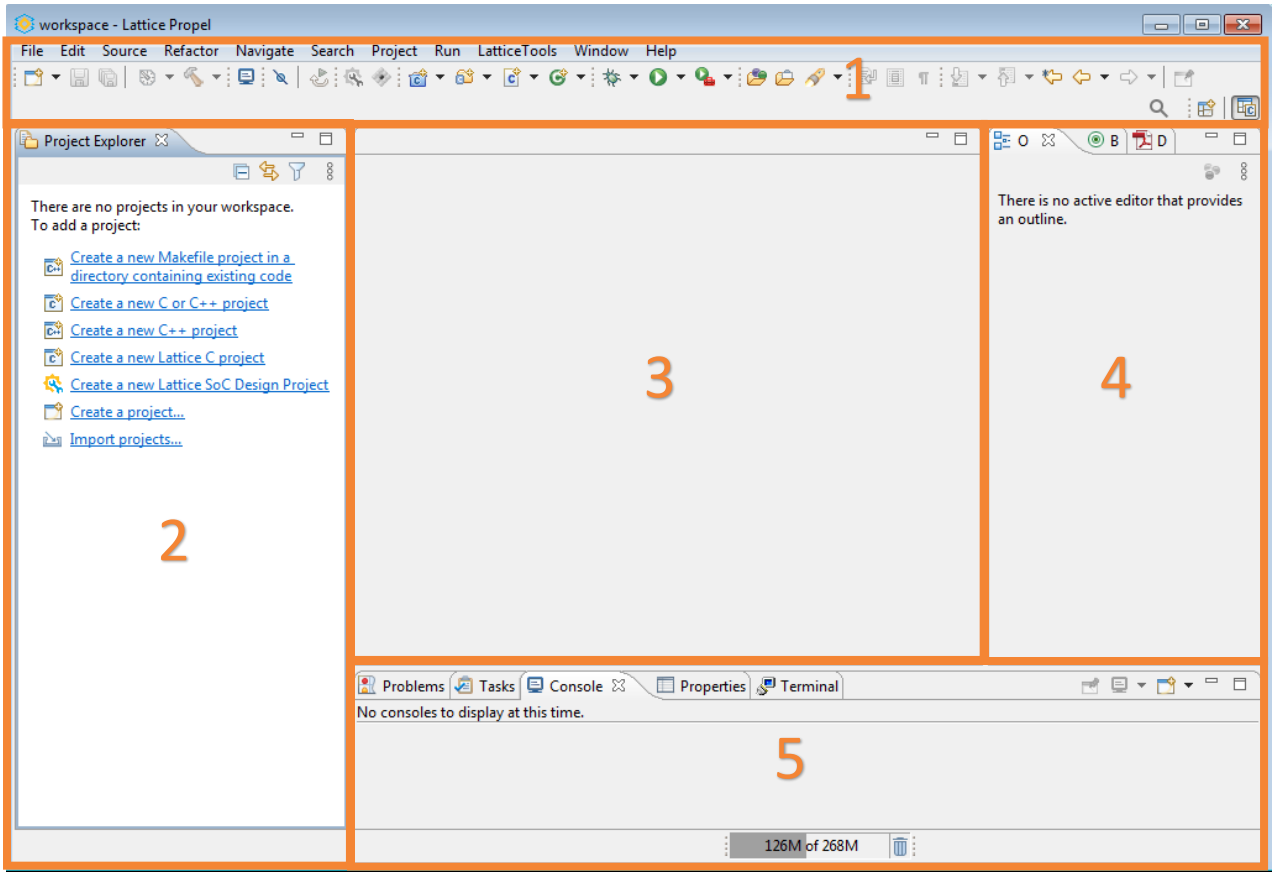


Figure 3.2. Propel Workbench Window

3.2. SoC Project Design Flow

A new SoC design project including a Propel Builder design can be started from the Lattice Propel sets. Follow the steps below to create a new SoC design project.

3.2.1. Creating a SoC Design Project

To start a Lattice SoC Design Project from Propel:

1. Choose **File** > **New** > **Lattice SoC Design Project**. The Create SoC project wizard opens (Figure 3.3).

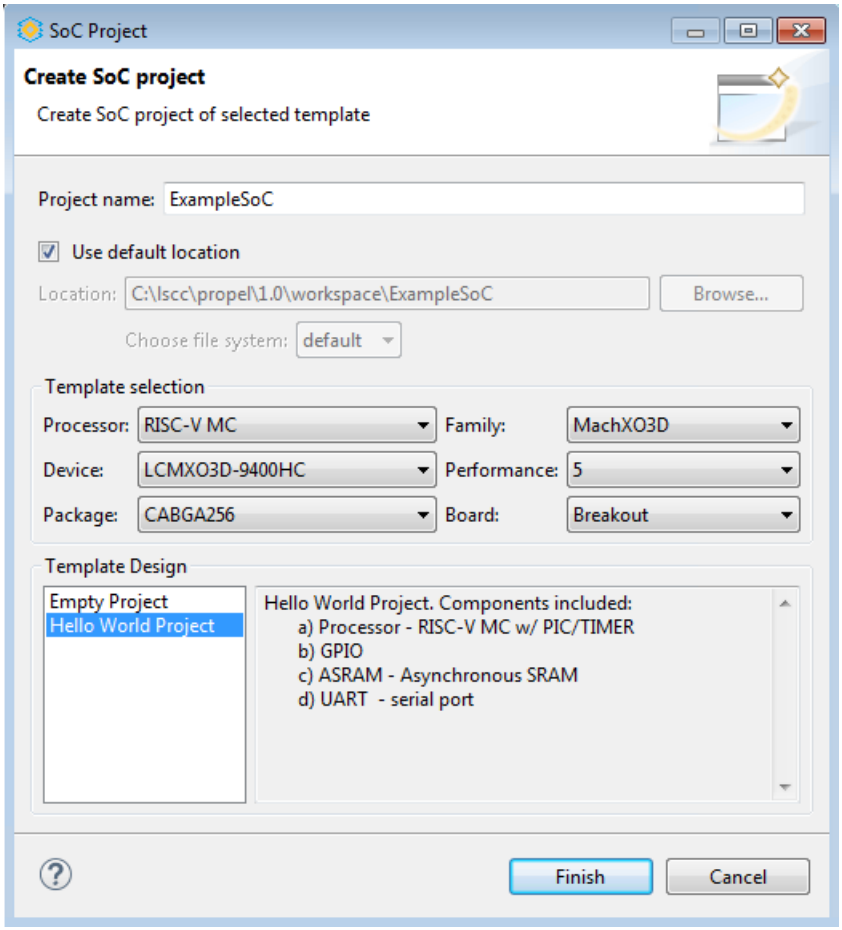


Figure 3.3. Create SoC Project Wizard

2. Enter a project name. Do not include periods, colons, or spaces in the project name.
3. (Optional) To change the default location, clear the **Use default location** option and browse to another location. Choose a file system.
4. Select a desired platform template design. In particular, select empty project for building system from scratch.
5. Click **Finish**.

   The SoC design project is created in workbench, and its design is opened and displayed in Propel Builder (Figure 3.6).

3.2.2. Open Design in Propel Builder

Within an SoC project, there is a Propel builder design.

To open Propel Builder for an SoC project:

1. In the **Project Explorer** view, select an SoC project.
2. Open the SoC project in one of the following three ways:

   - Choose **LatticeTools >** **OpenDesign** (Figure 3.4) from Propel Builder.

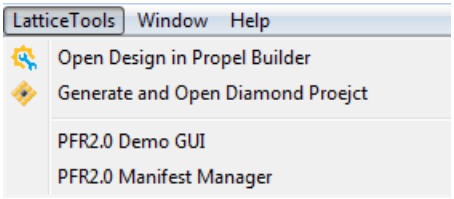


Figure 3.4. LatticeTools Menu

   - Click the **Propel Builder** icon on the toolbar.
   - Right-click the SoC project from **Project Explorer**. Choose **Open Design In** > **Propel Builder** from the popup menu (Figure 3.5).

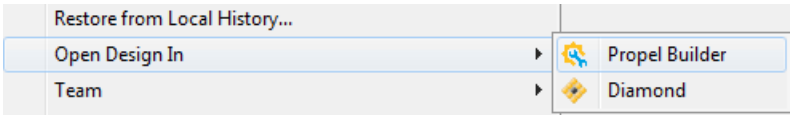


Figure 3.5. Project Explorer Popup Menu

3. SoC Design is opened and displayed in Propel Builder (Figure 3.6).

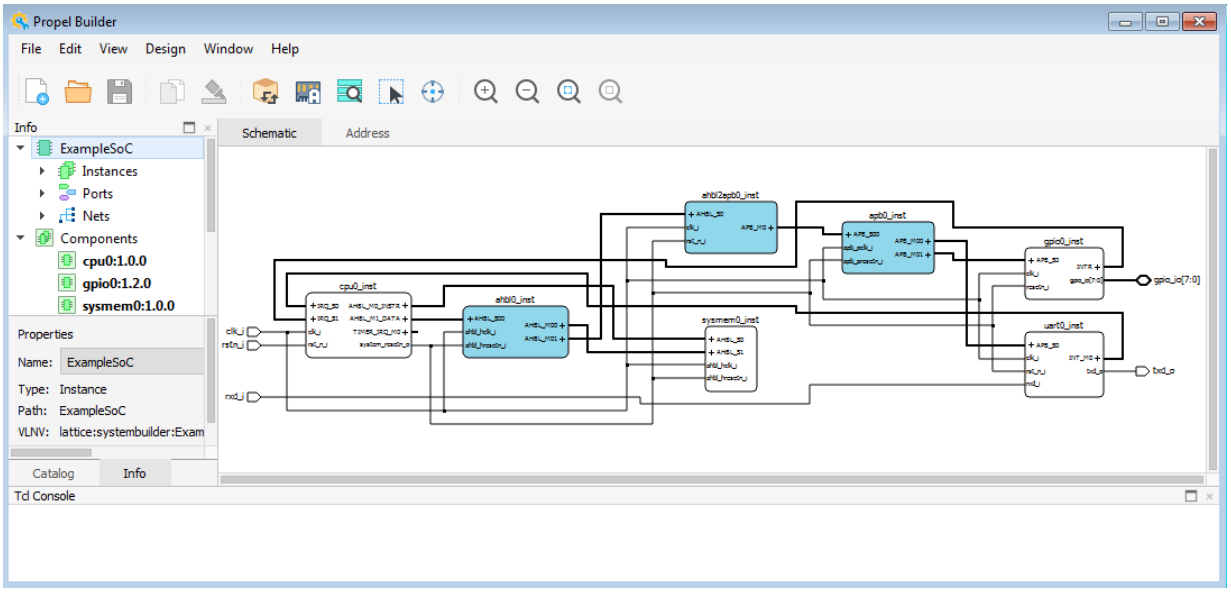


Figure 3.6. Propel Builder Window

4. (Optional) Modify the design in Propel Builder as desired. Most of the templates include a functional-ready SoC design.

   Note: You can only create an SoC design using **Empty Project** template inside the Propel Builder. Refer to the Propel Builder online help for more details on how to create an SoC design using Empty Project template.

3.2.3. Open Design in Diamond

Within an SoC project, you can create a Diamond project including a Propel Builder design and then open the SoC project in Diamond.

To open Diamond for an SoC project from Propel:

1. In the **Project Explorer** view, select an SoC project.
2. Open the SoC project in one of the following ways:
   - Choose **LatticeTools > Generate** and Open Diamond Project.
   - Click the **Diamond** icon from the toolbar.
   - Right-click an SoC project from the **Project Explorer**. Choose **Open Design In > Diamond** from the right-click menu.
3. Diamond project for SoC is generated at background and is launched (Figure 3.7).

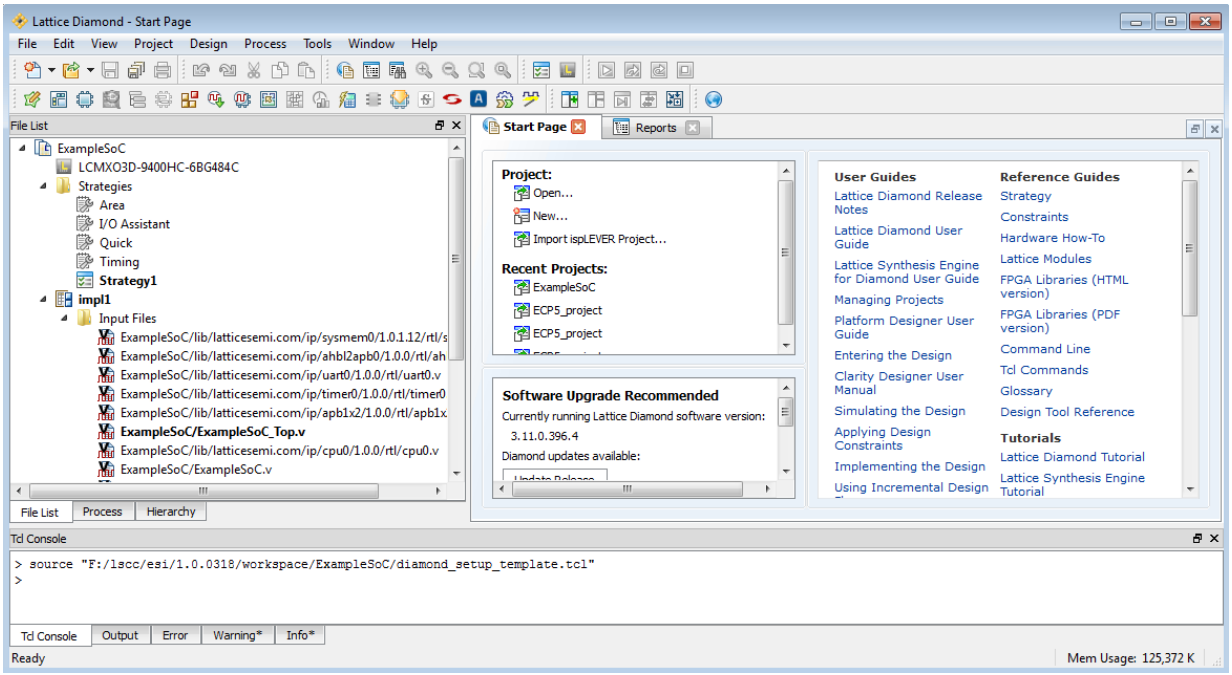


Figure 3.7. Diamond Project

4. (Optional) From the **File List** view of Diamond:
   - modify the top-level RTL file (<*proj_name*>_Top.v) to match the SoC design, presupposition of which is that there is a top-level RTL file in your SoC design; or
   - create a top-level RTL file (<proj_name>_Top.v) to match the SoC design, if the SoC design is created from an Empty Project template and there is no top-level RTL file in your SoC design.
5. (Optional) Modify LPF constraint file (<*proj_name*>.lpf) to match the SoC design, if you have modified the SoC design.

   Note: This step is a must to the SoC design that is created from Empty Project template.

6. Switch to **Process** view of the Diamond project (Figure 3.8). Make sure at least one programming file is checked in the **Export Files** section.

   Note: Available programming file can be different for specific device includes,

   - JEDEC File — JEDEC is the industry standard for PLD formats for programming the device. This option is applicable only to non-volatile FPGAs such as LatticeXP, LatticeXP2, MachXO, MachXO2, MachXO3D, and MachXO3L.
   - Bitstream File — a configuration bitstream (bit images) file contains all of the design configuration information that defines the internal logic and interconnections of the FPGA, as well as device specific information from other files. This option is applicable only to volatile SRAM-based FPGAs, such as a LatticeECP/2/M, LatticeECP3, or LatticeSC/M device.
   - PROM File — an output file in one of several programmable read-only memory (PROM) file formats. This option is applicable only to volatile SRAM-based FPGAs, such as a LatticeECP/2/M, LatticeECP3, or LatticeSC/M device.

7. Choose **Process >** **Run**.

   The Programming file is generated. It can be used in the Diamond Programmer.

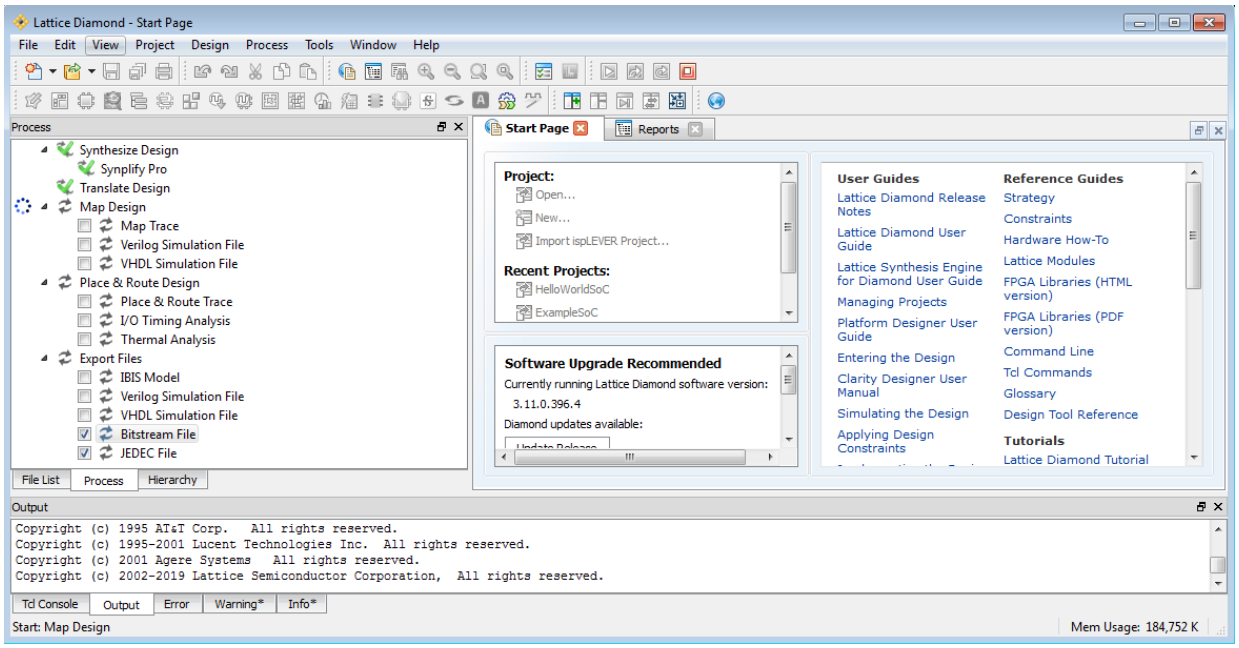


Figure 3.8. Generate Programming File

3.2.4. Generating System Environment by Building Project

System environment package including the system environment file and the BSP package is required for the embedded C project.

To generate system environment package from Propel:

1. In the **Project Explorer** view, select an SoC project.
2. Choose **Project** > **Build Project**.
3. Check the building result in the **Console** view (Figure 3.9).

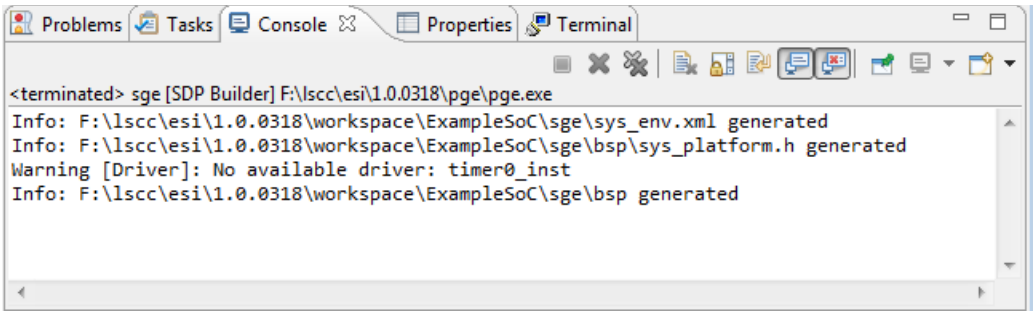


Figure 3.9. Build Result of SoC Project

3.2.5. About SoC Design Project

The SoC project created starts with a functional-ready SoC design and a default simulation environment. In the Project Explorer view, open an SoC project folder and all its sub-folders. The project contains (Figure 3.10):

- *<proj_name>*: Folder contains a Propel Builder design including the .sbx file.
- *<proj_name>*/application: Folder contains functional-ready embedded application source codes.
- Impl1: Folder contains the implementation of Diamond project.
- sge: Folder contains generated package necessary for creating C project.
- sim: Folder contains the default simulation environment.
- *<proj_name>*.ldf: Diamond project file.
- *<proj_name>*.lpf: Diamond project logical preference file.
- *<proj_name>*.txt: Description file from the template.

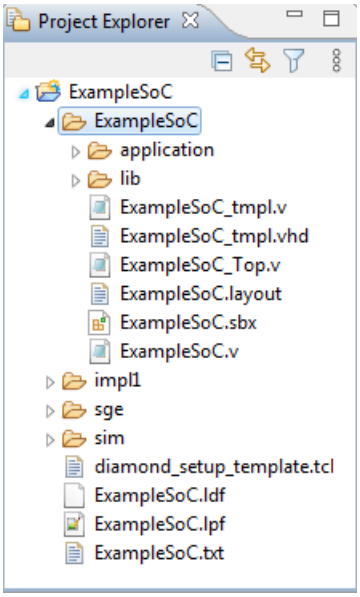


Figure 3.10. Contents of SoC Project

3.3. C Project Design Flow

3.3.1. Creating a C Project

To start a Lattice C Project from Propel:

1. Choose **File** > **New** > **Lattice C Project**.

   The C Project wizard opens with the **Load system and bsp** page (Figure 3.11).

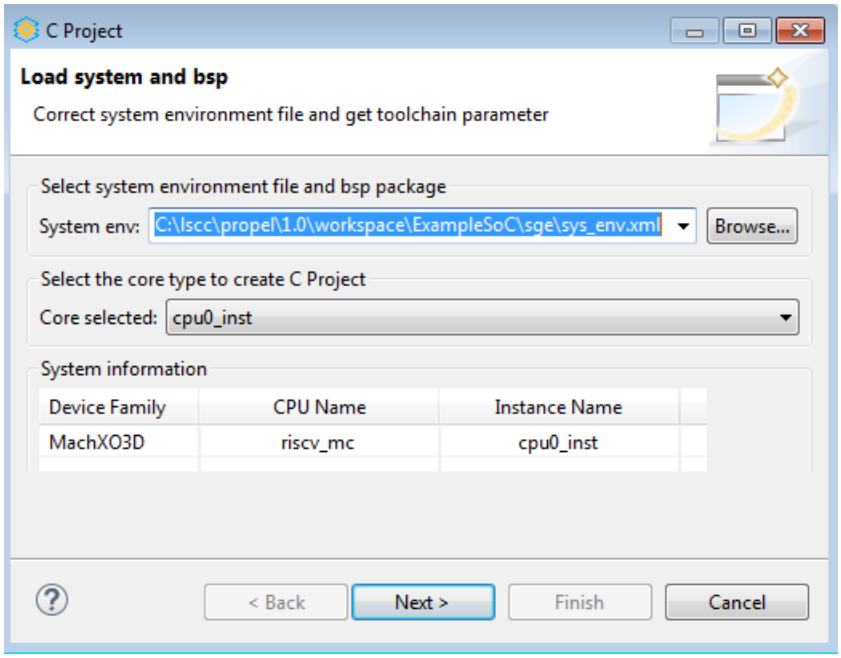


Figure 3.11. Load System and BSP Dialog

2. Browse to the SoC project folder and select the system environment file: sys_env.xml.

   All system environment files available in current workspace can be selected from the **System env:** drop-down menu.

3. If the platform has more than one processor, choose one core.
4. Click **Next**.

   The **C Project** dialog opens (Figure 3.12).

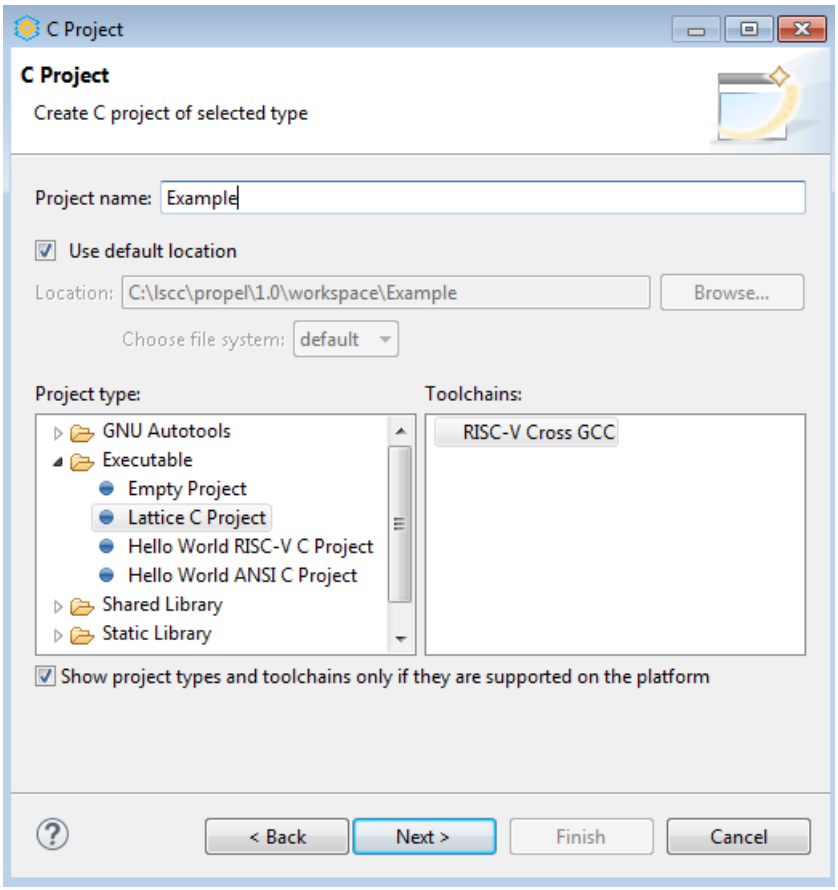


Figure 3.12. Create C Project Dialog

5. Enter a project name. Suggest not using periods, colons, or spaces in your project name. Though spaces are allowed, they may cause certain issue with some tools.
6. By default, the “Use default location” option is checked. The default file system is selected automatically. Suggest using default location, unless you have special need to a special location.
7. Select a project type. Select a toolchain. Normal choice is **Lattice C Project** and **RISC-V Cross GCC**.
8. Click **Next**.

   The **Lattice toolchain setting** dialog opens (Figure 3.13).

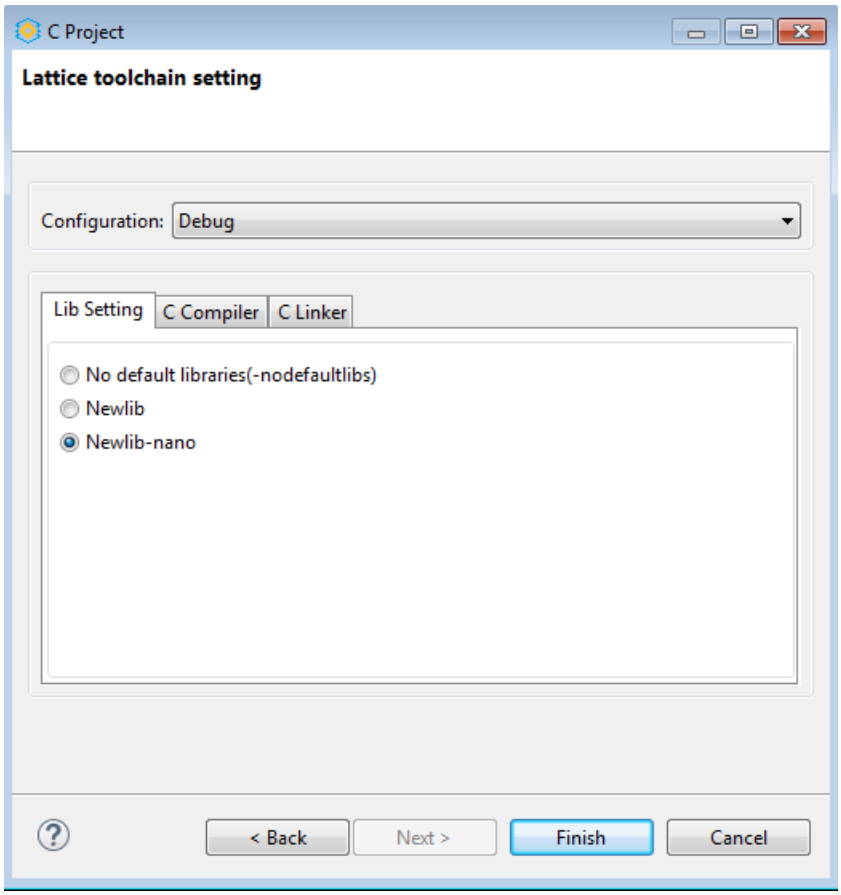


Figure 3.13. Lattice Toolchain Setting Dialog

9. By default, two toolchain configuration modes, **Debug** and **Release**, can be chosen from the Configuration drop-down menu. You can modify library, compiler, and linker options for each configuration.
10. Click **Finish**.

    The C project is created and is displayed using the C/C++ perspective. A perspective is a collection of tool views for a particular purpose. The C/C++ perspective is for creating C/C++ programs.

3.3.2. Building a C Project

To build a C project in Propel:

1. In the **Project Explorer** view, select a C project.
2. Follow steps below if you want to change the active build configuration:
   a. Choose from menu **Project > Build Configurations > Manage…**. Or, click the Configuration icon on the toolbar.
   b. The Manage Configurations dialog opens (Figure 3.14) for selecting active configuration. By default, a Debug configuration creates executables contain additional debug information that lets the debugger make direct associations between the source code and the binaries generated from the original source. A Release configuration provides the tools with options set to create an application with the best performance.

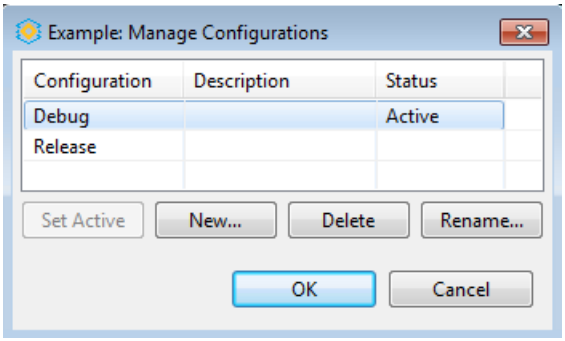


Figure 3.14. Manage Configurations Dialog

3. Choose **Project** > **Build Project**. Or, click the Build icon on the toolbar.
4. The results of the build command are displayed in the **Console** view (Figure 3.15).

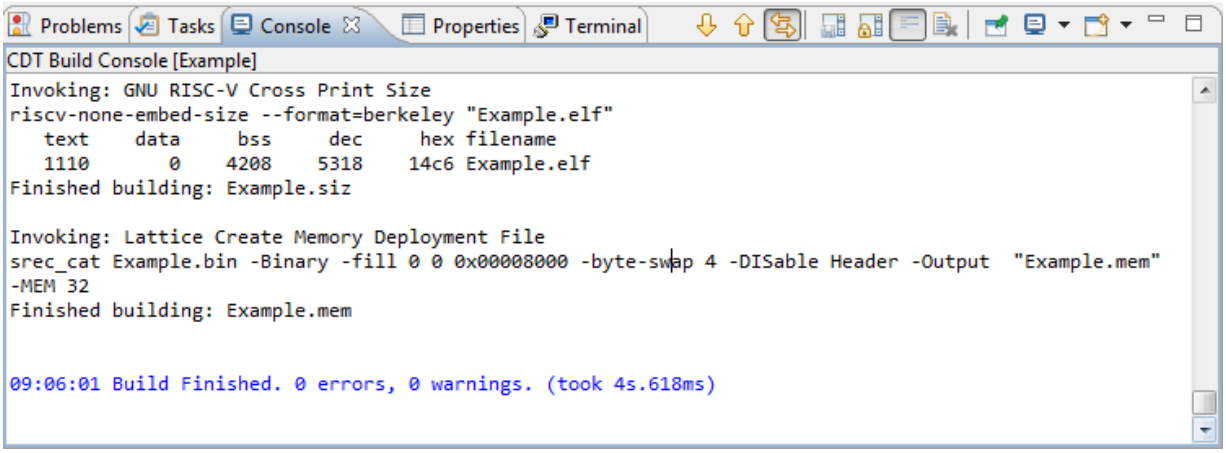


Figure 3.15. Build Result of C Project

3.3.3. About Lattice C Project

The Lattice C project starts with source code already. In the Project Explorer view, open a C project folder and all its sub-folders. The project contains:

- src/bsp/driver: Folder contains driver codes from the IP in the platform.
- src/bsp/sys_platform.h: Header file that defines DEVICE_FAMILY (the Lattice FPGA), address mapping, and any IP parameters that can be used by the drivers.
- src/main.c: Beginning of a C file with the beginnings of system_init and main routines.
- src/cpu0.yaml: Processor description file used at debugging time.
- src/linker.ld: Generated linker script file.
- src/sys_env.xml: System environment file describing aspects of the platform, such as memory spaces.

After build project, the build output locates at each build configuration folder, such as **Debug**; **Release** (Figure 3.16). In each build output folder, it contains:

- <*proj_name*>.elf: Executable file used in on-chip debugging.
- <*proj_name*>.bin: Binary file used in deploying the application to flash memory.
- <*proj_name*>.lst: Extended listing file generated by tool objdump.
- <*proj_name*>.map: Linker map file.
- <*proj_name*>.mem: Lattice system memory initialization file used in System_Memory IP.

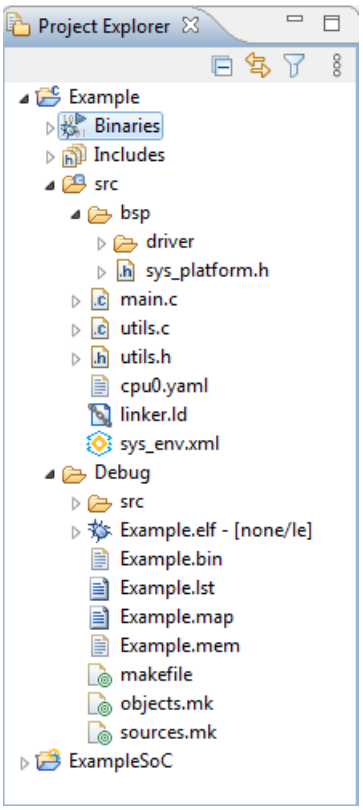


Figure 3.16. Contents of C Project

3.3.4. Writing Code

Lattice Propel is based on Eclipse IDE. You can write application code following the process and usage of the same tools as any in Eclipse IDE. You can get more details regarding Eclipse IDE from either the Propel online help.

For writing code, Lattice Propel SDK provides two extra aids:

- Lattice System Platform: An overview of the processor platform can be displayed (Figure 3.17).
- Linker Editor: An overview of the memory regions of linker script can be displayed. You can modify key linker parameters via graphical interface (Figure 3.18).

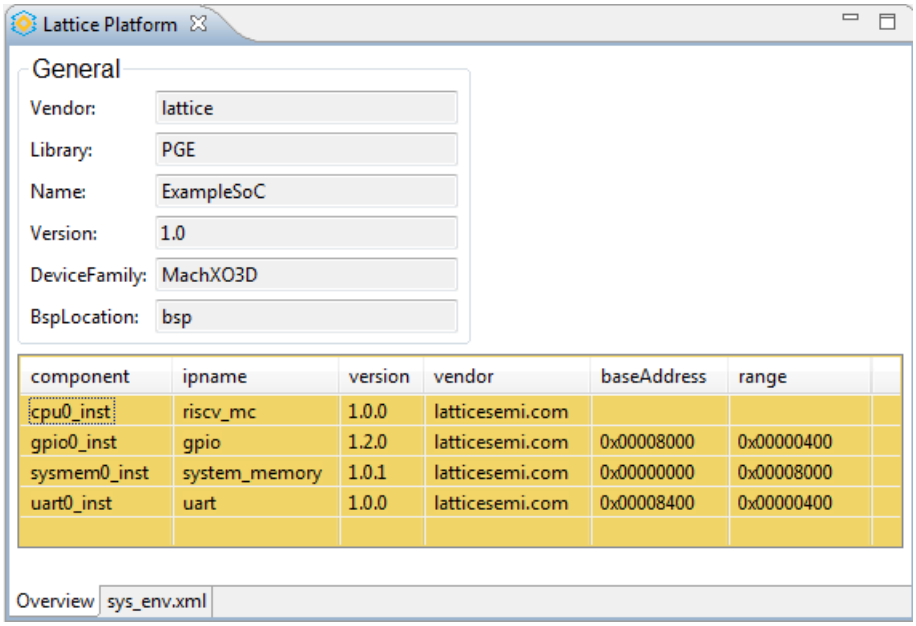


Figure 3.17. Lattice System Platform

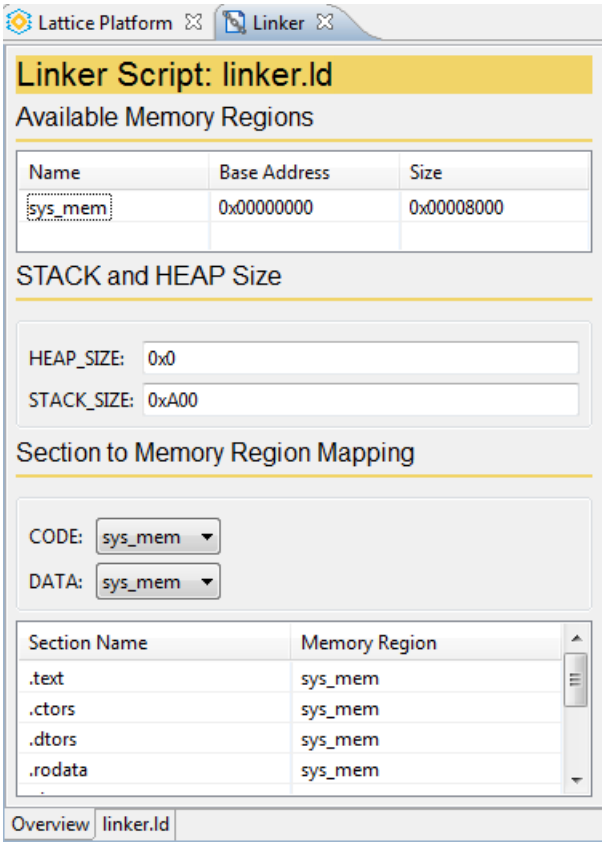


Figure 3.18. Linker Editor

3.3.5. Advanced Tool Chain Setting

Follow the process below to modify the tool chain settings of a C project.

To change tool chain setting in Project Properties in Propel:

1. In the **Project Explorer** view of Propel, select a C project.
2. Choose **Project** > **Properties.** The Properties for current project opens (Figure 3.19).
3. Select **Settings** of **C/C++ Build** category from the left pane. Select the **Tool Settings** tab.

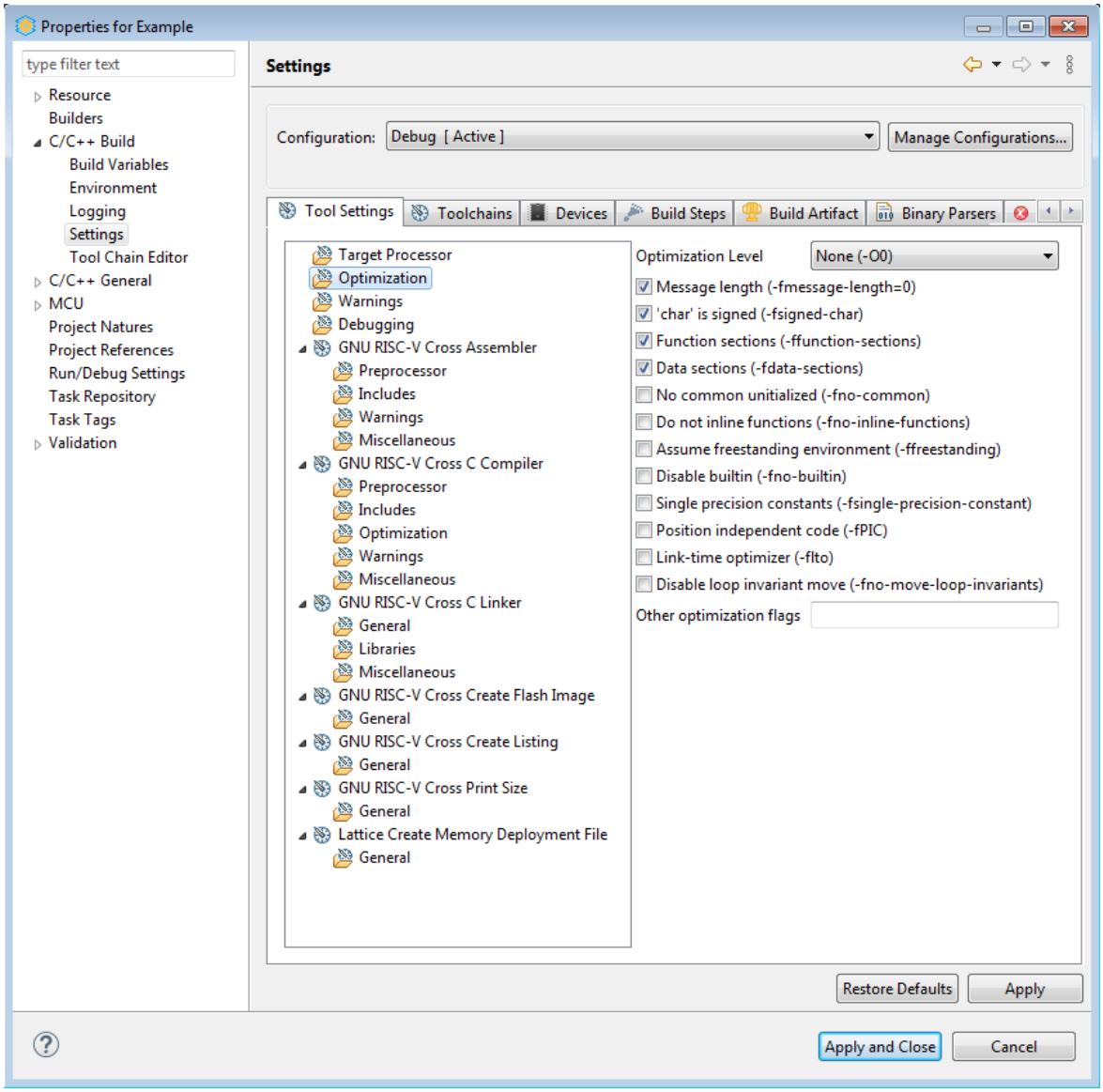


Figure 3.19. Properties of C Project

4. Customize the tools and tool options. All your customization will be resulted in the build configuration in the **Tool Settings** properties tab. The build configuration is used during your C project building.

   Note: Setting for each configuration, **Debug** or **Release**, is independent.

3.4. System Simulation Flow

The SoC Project created from template has a default simulation environment for you to setup and start functional simulation. It is generated automatically along with the SoC project creation. You can use it as a start point and customize accordingly.

The default simulation environment is with the following features:

- Provides similar user experience as real board-level debugging, such as for Hello World SoC, key components including RISC-V MC, System memory and UART, is included.
- Can simulate user-modified template SoC with extended HDL designs.
- Can simulate the whole system using real C projects as stimulus with some necessary modification with all the details for debugging.
- Supports user extension with a friendly and flexible approach.

3.4.1. Launch Simulation

To launch simulation:

1. In Propel Builder, update the sbx file and IPs to enable simulation features.
   - Enable checkbox **Simulation Mode** for *RISC-V MC* IP module from the **General** area (Figure 3.20).

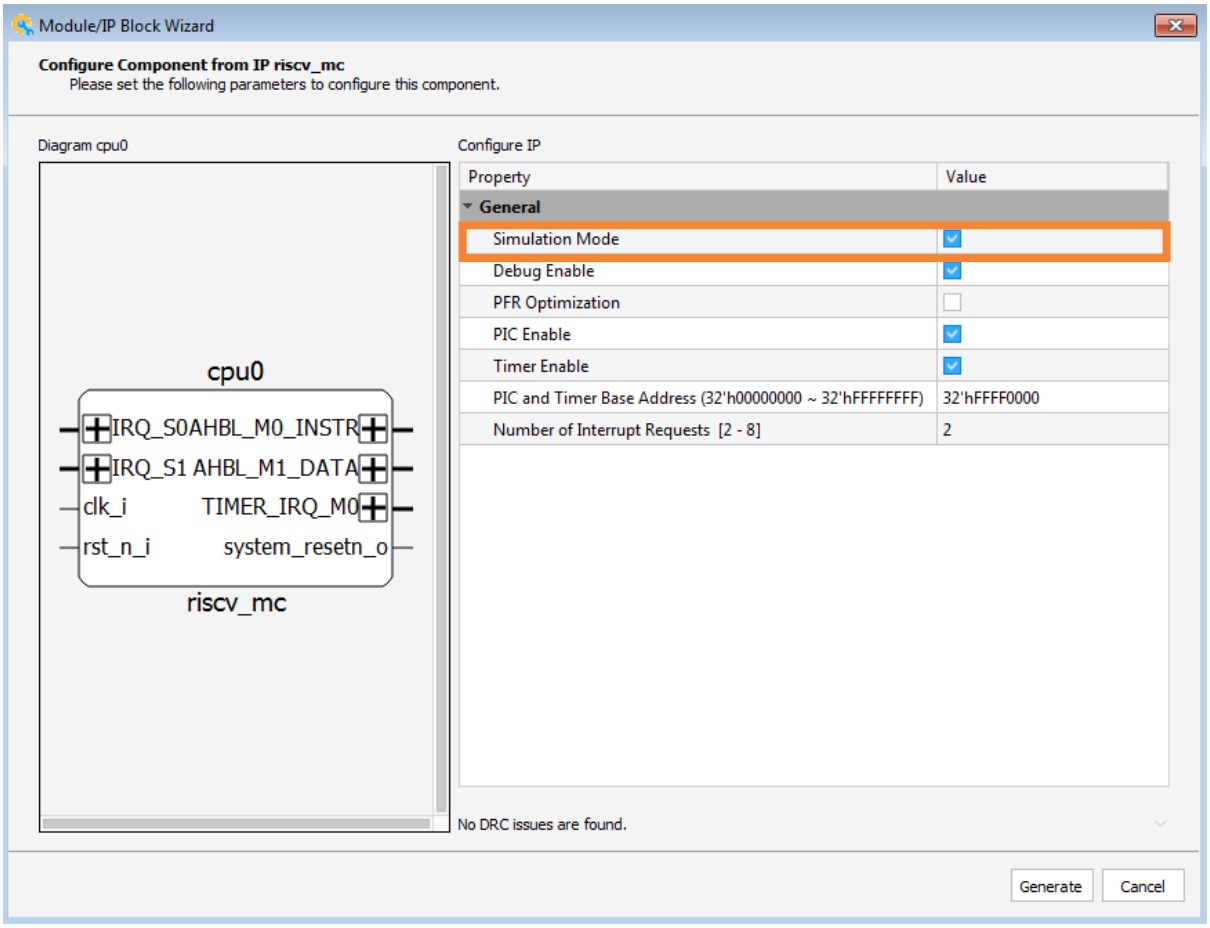


Figure 3.20. Configure Module RISC-V MC

   - Enable checkbox **Initialize Memory** for *system_memory* IP module from the **Initialization** area of the **General** tab, and set **Initialization File** generated from corresponding C project (Figure 3.21).

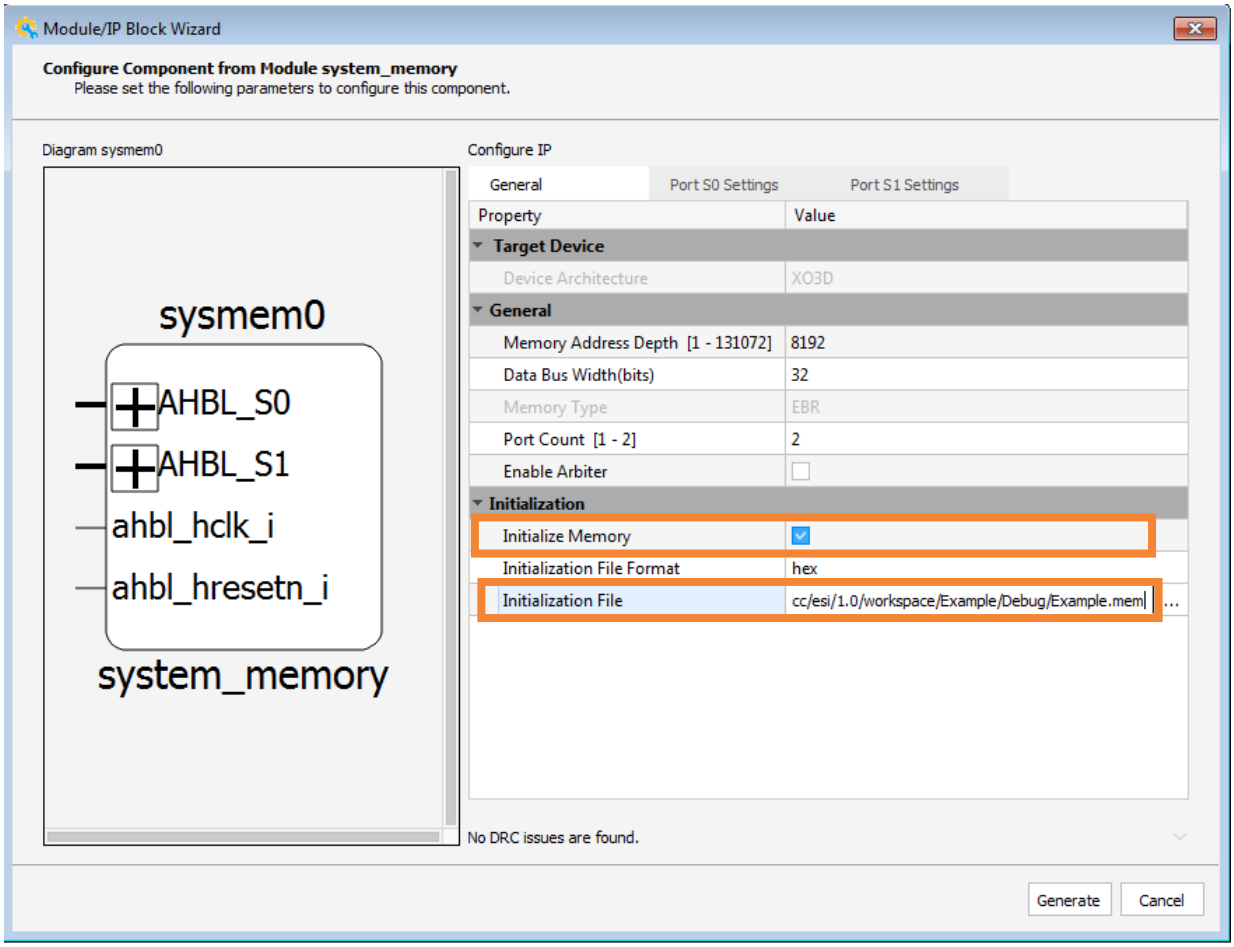


Figure 3.21. Configure Module System Memory

2. In Propel, update the .do file and file list .f file.
   - From the Project Explorer pane on the left, double-click asim.do from the models subfolder under the sim folder. The asim.do file is opened in a text editor on the right pane. Change the variable *Prj_name* to the current project name in the asim.do (Figure 3.22.) file. Save the file.
   - If there are more HDLs added based on SoC template, flist.f is supposed to be updated accordingly.

   You can check readme.txt under sim folder for template specific information.

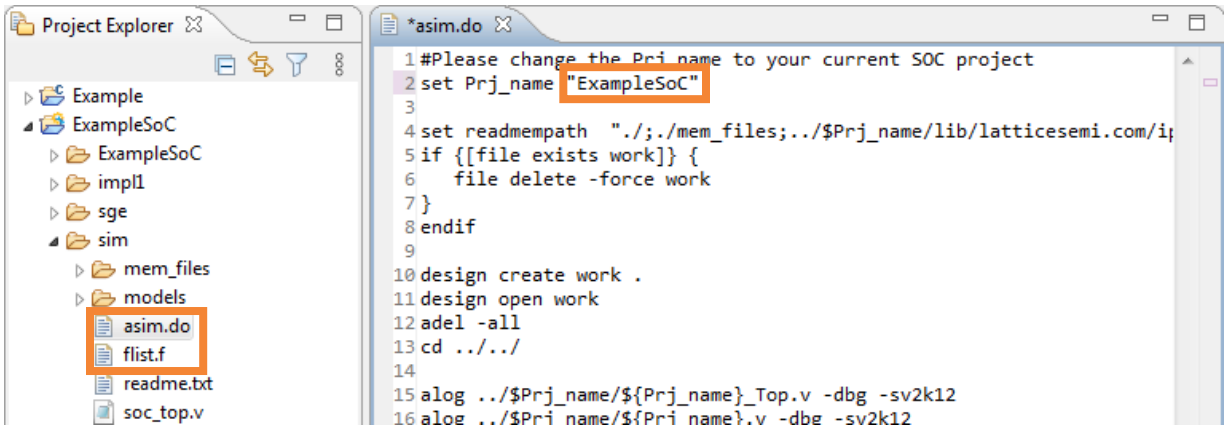


Figure 3.22. Update asim.do

3. In Active-HDL Lattice Edition, launch the simulation.
   - Open Active-HDL Lattice Edition. Change directory and browse to <*soc_project_path*/sim> folder and run “do asim.do”.
   - Check the waveform. Following is the waveform (Figure 3.23) generated from Hello World project.

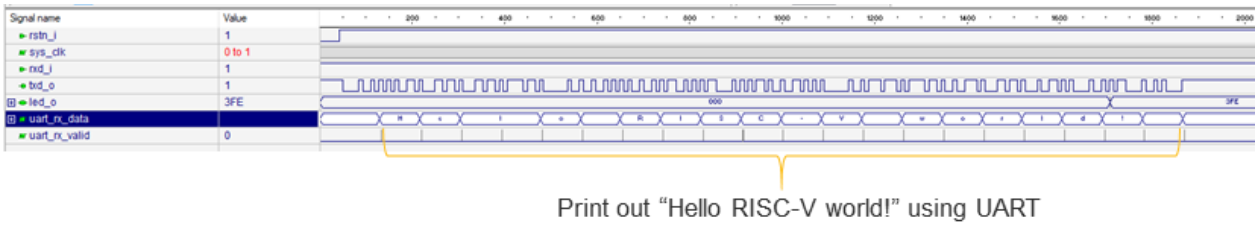


Figure 3.23. Waveform of Hello World Project

3.4.2. Simulation Details

The default simulation environment is located at sim folder of created SoC Project in Propel. It contains:

- soc_top.v – top-level testbench
- asim.do – tcl script for ActiveHDL
- flist.f – file list for current SoC project
- readme.txt – detailed description for the design specific information, tips, notes, etc.
- <models> – all the simulation models used will be put in this folder
- <mem_files> - mem files used for this SoC project

3.4.2.1. About Hello World Project

The Hello World Project is designed to print some strings using a top-level UART port. The testbench instantiates a UART slave to receive these data. Each byte (refer to uart_rx_data shown in Figure 3.23) can be checked using ASCII format. Meanwhile, this UART Slave also supports sending the data to Design Under Test (DUT). The data is stored in a text file and enabled by STIMULUS_GEN parameter.

3.5. Programming and On-Chip-Debugging Flow

This section describes the process of testing and debugging application code on the actual hardware including the Lattice FPGA with the hardware design installed. Debugging with Propel SDK follows the same process and uses of the same tools of any of those in Eclipse IDE.

Before debugging, download the hardware design created from Diamond Programmer. Refer to the User Guide of the specific evaluation board for more details on the evaluation board.

3.5.1. Creating a Debug Launch Configuration

To debug a program, a debug launch configuration must be created. Most of the settings for a debug launch configuration can be automatically entered. Only a few settings need to be manually configured.

To create a debug launch configuration:

1. In the **Project Explorer** view of Propel, select a C project.
2. Build the project and ensure the executable file is available. Refer to the Building a C Project section for details on the process.
3. Choose **Run** > **Debug Configurations…**

   The **Debug Configurations** dialog opens (Figure 3.24).
4. Double-click **GDB OpenOCD Debugging** to create a new launch configuration.

A multi-tab page is displayed. The Main tab should already be filled in with the project name, application file name, and location.

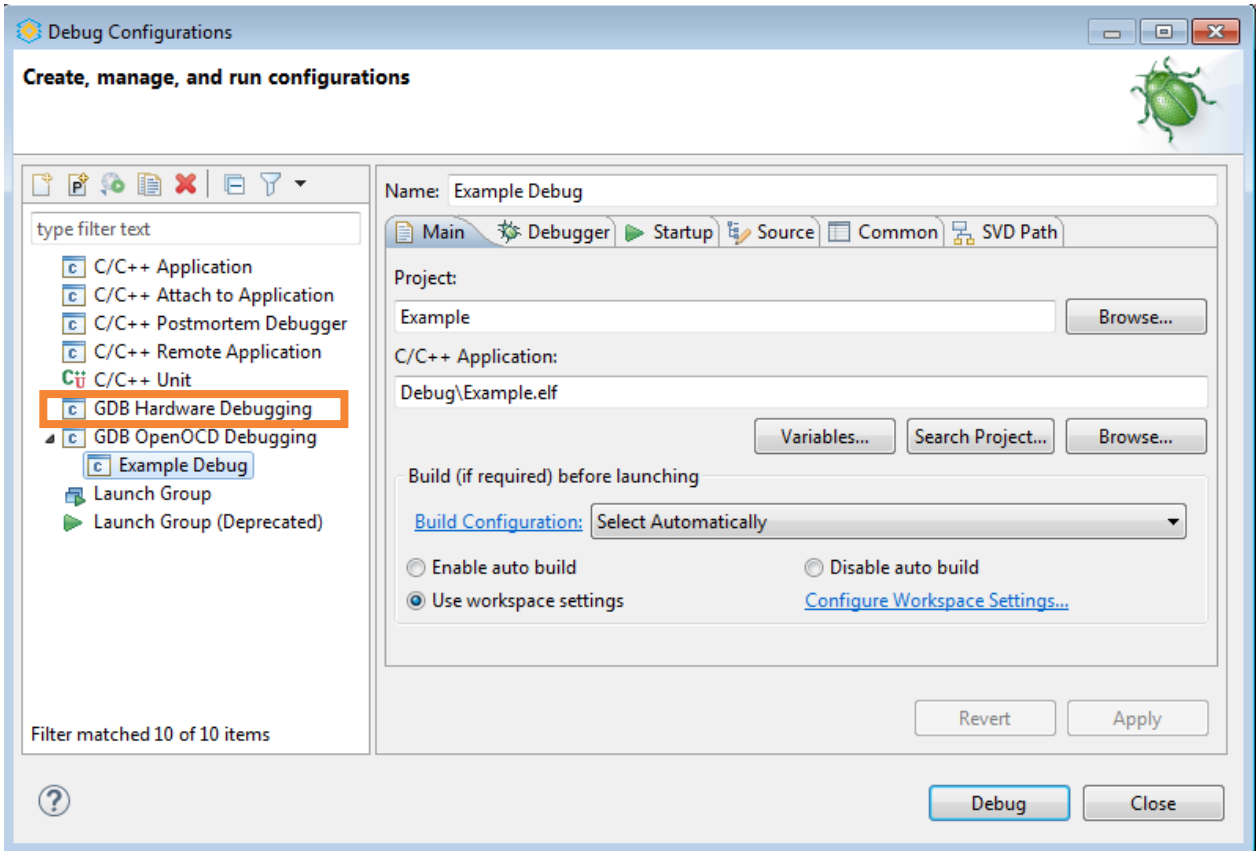


Figure 3.24. Debug Configurations Page

5. Select the **Debugger** tab (Figure 3.25). It is critical that the **Config options** field contains the correct command line options to be passed to OpenOCD.

   -c "set port FTUSB-0" is required for the selection connected to Lattice cable. The default value *"set port FTUSB-0"* is suitable for using one Lattice cable case. If more than one cable is connected, you need update the port value according to system configuration.

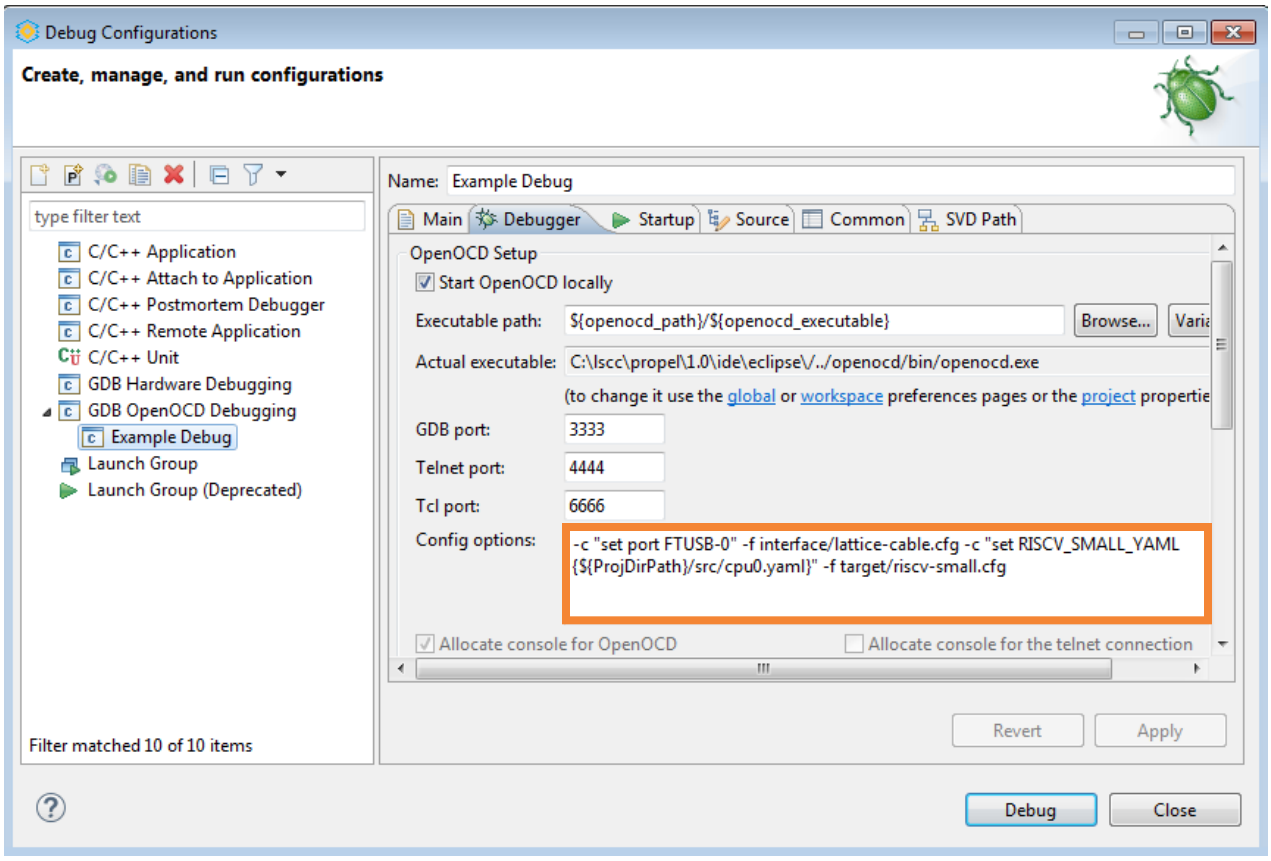


Figure 3.25 Debugger Tab of Debug Configurations

6. (Optional) Select the **Common** tab (Figure 3.26). The **Save as** > **Local file** option is selected by default. This causes the debug launch configuration to be saved into the workspace.

   Change setting the **Save as** field to **Shared file** to let the debug launch configuration saved into the project that aids the project portability.

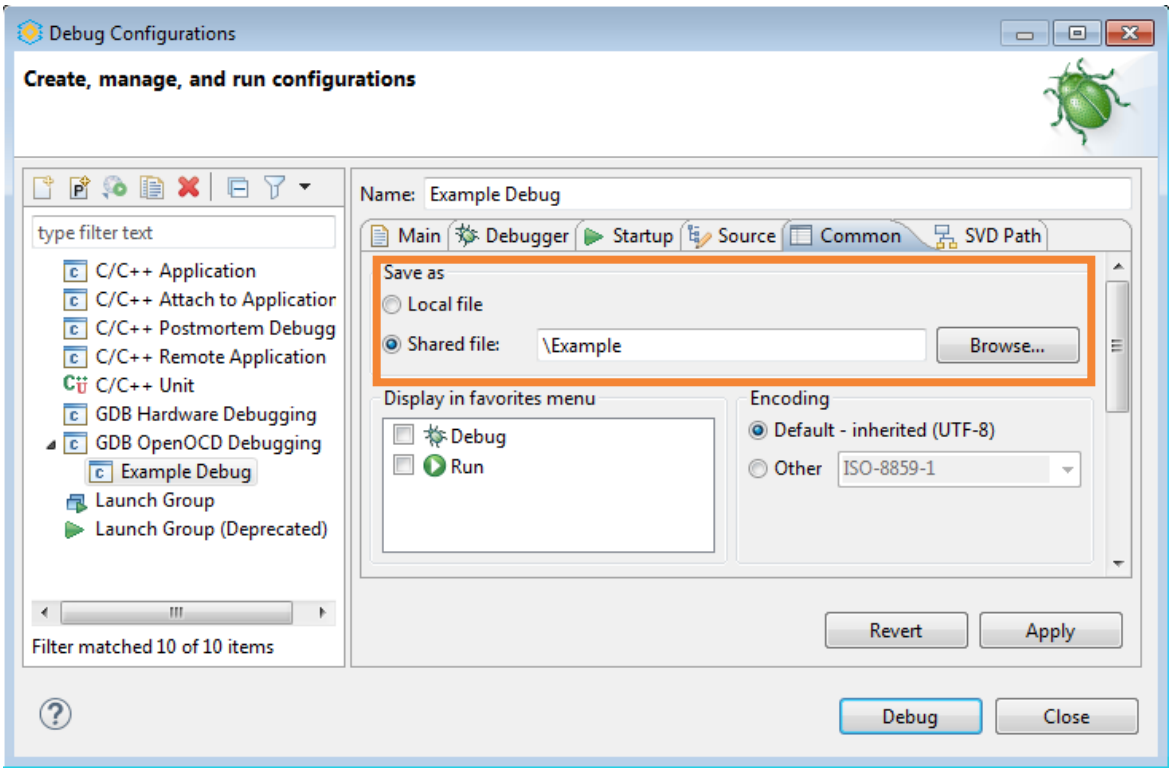


Figure 3.26 Common Tab of Debug Configurations

7. Remain settings as default. Do not change the settings unless necessary, or unless you understand what effect these changes may have.
8. Click **Apply** to keep the current settings.
9. Click **Close**.

3.5.2. Starting a Debug Session

Before starting a debug session, be sure that:

- Lattice cable is connected to the computer.
- The target device is power ON.
- The hardware design has a debug enabled processor module and already programming into the target device.

With the above steps completed properly, follow the steps below to start the debug session from Propel.

1. Choose **Run** > **Debug Configurations…**
2. If necessary, expand the **GDB OpenOCD Debugging** group.
3. Select the newly-defined configuration.

4. Click the **Debug** button (Figure 3.24).

   Alternatively, for later sessions, use the Debug icon on the toolbar. Do not click the Debug icon directly. Instead, click the down arrow beside the Debug icon. Select the desired debug configuration from the drop-down menu (Figure 3.27).

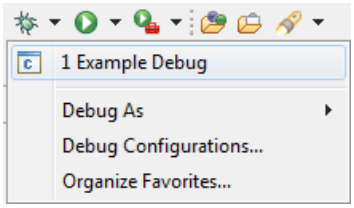


Figure 3.27. Debug Icon on Toolbar

5. Wait for a few seconds for switching to debug perspective, starting the server, connecting to the target device, starting the gdb client, downloading the application and starting the debugging session.
6. The Propel Window displays as shown in Figure 3.28. The execution stops right at the beginning of the *main()* function.

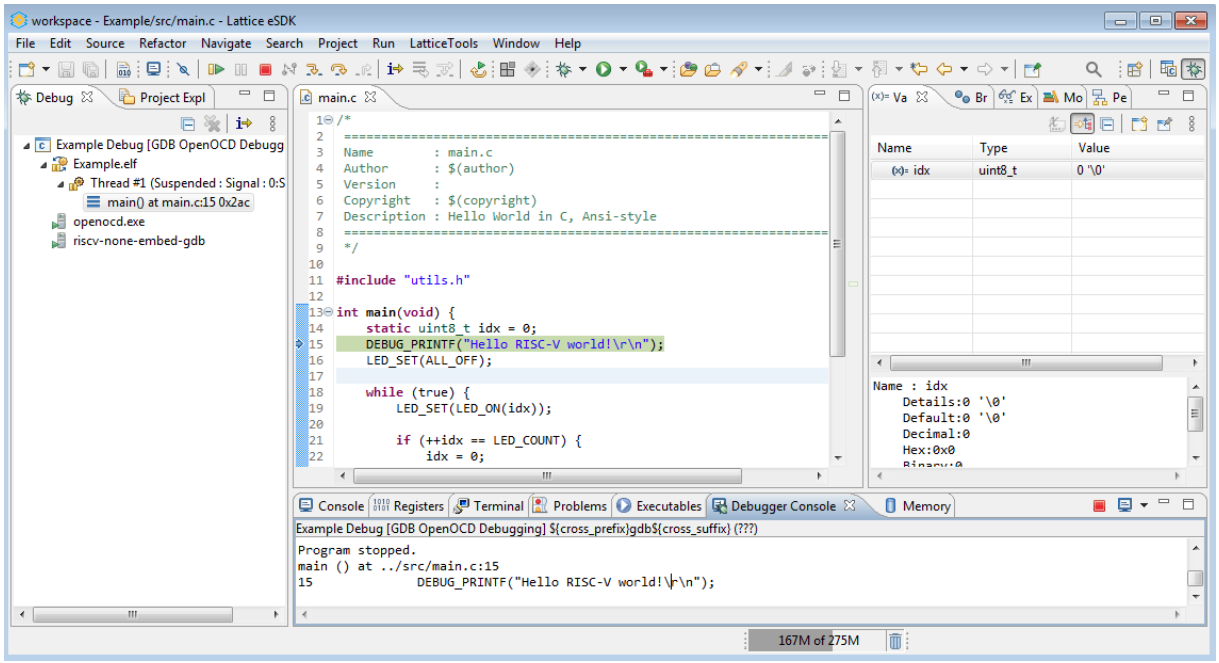


Figure 3.28. Debug Perspective

3.5.3. Serial Terminal Tool

Serial port communication is frequently used in microcontroller debugging time. Propel SDK provides a built-in terminal tool including serial support for debugging.

To launch a serial terminal:

1. Find the **Terminal** view nested to the **Console** view. If not found, re-open from **Window** > **Show View** > **Terminal**.
2. In the **Terminal** view, click the **Open a Terminal** icon . The **Launch Terminal** dialog opens (Figure 3.29).
3. Choose the **Serial Terminal** and configure the **Serial port** with **Baud rate**.

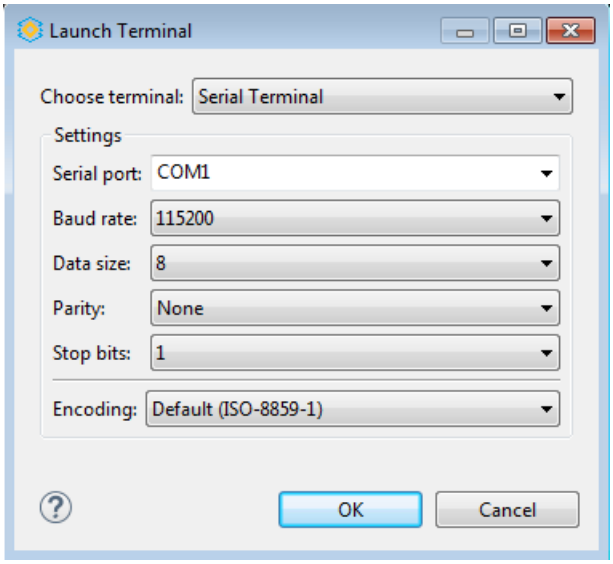


Figure 3.29. Launch Terminal Dialog

4. Click **OK**. A connection opens.
5. (Optional) Click the **Toggle Command Input** icon that adds an edit box to enter text (Figure 3.30).

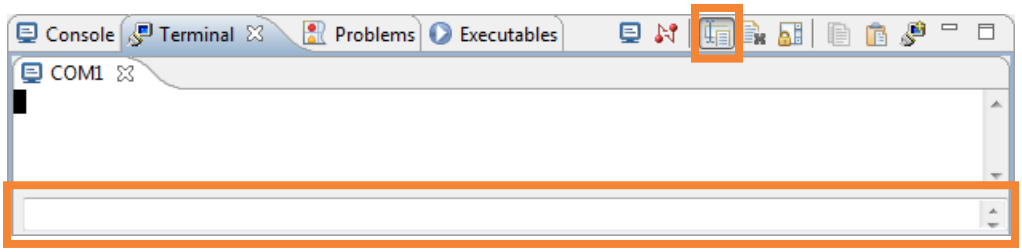


Figure 3.30. Terminal View

4. Propel Tutorial – Hello World

This “Hello World” project can be found from the template. The “Hello World” project provides a template of using hardware and software design with the minimal resource required.

Following this tutorial, you can easily create a hardware and software project. After that you can run the project on your evaluation board.

4.1. Creating SoC Design Project and Preparing Hardware Design

To start an SoC Design Project from Propel:

1. Choose **File** > **New** > **Lattice SoC Design Project**.
2. The SoC project wizard opens (Figure 4.1). Enter a project name, such as **HelloWorldSoC**. Select the **Hello World Project** template.
3. Click **Finish.** An SoC project is created.

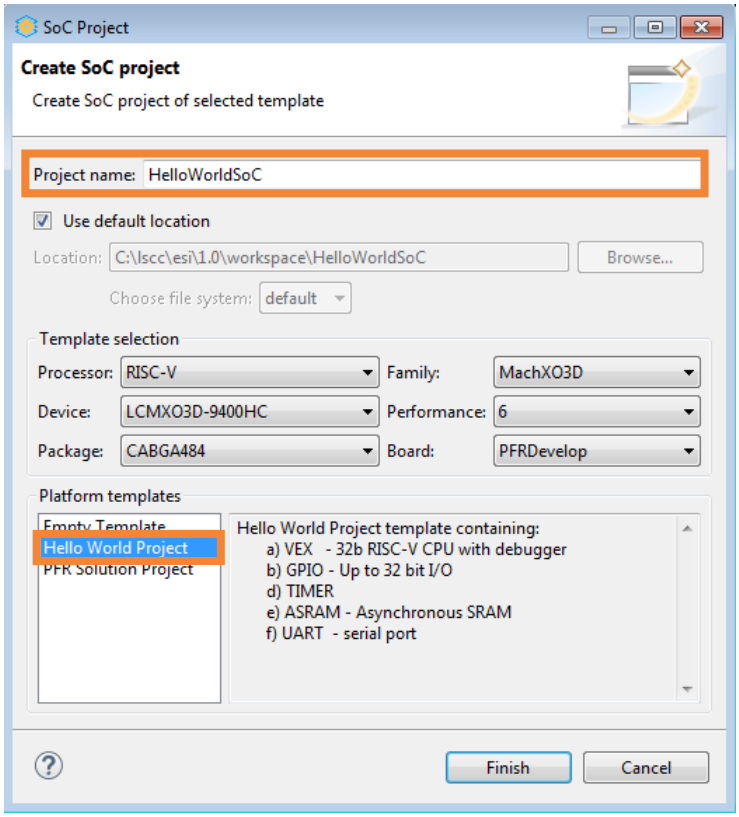


Figure 4.1. Create SoC Project Wizard

4. The created SoC project can be found in the workbench. Its design is opened and displayed in Propel Builder for review (Figure 4.2).

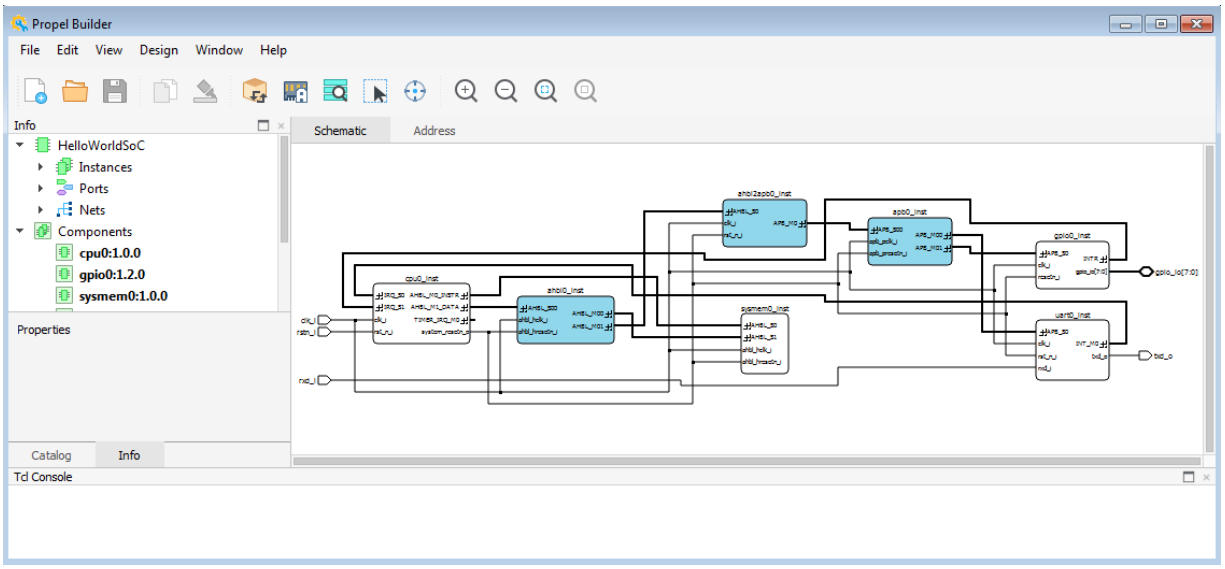


Figure 4.2. Propel Builder Window

4.2. Launching Lattice Diamond

Launch Lattice Diamond from the created SoC project. To do that:

1. In Propel **Project Explorer** view, select the SoC project *HelloWorldSoC*.
2. Click the Diamond icon on the toolbar. A Diamond project is created and thus opened automatically in **Lattice Diamond**.
3. Switch to **Process** view of the Diamond project and make sure **Bitstream File** or **JEDEC File** is checked in the **Export Files** section (Figure 4.3).
4. Choose **Process** > **Run**. Wait for generating programming file successfully. You can see a green checkmark before each successfully completed process.

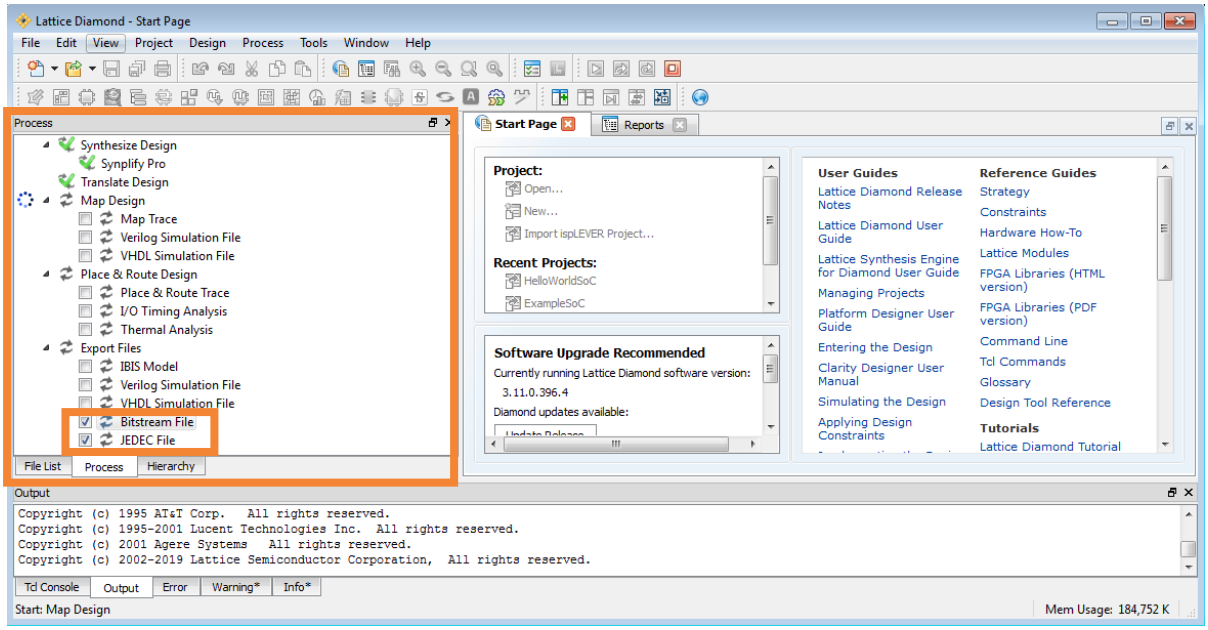


Figure 4.3. Generate Programming File

4.3. Programming the Target Device

Once the programming file is exported successfully in last section, it is ready to program the target device. Make sure the evaluation board is powered ON and connect correctly to host PC before performing the following procedure.

1. Click the **Programmer** icon on the toolbar of the Lattice Diamond Project Explorer.
2. The **Programmer: Getting Started** dialog pops up (Figure 4.4). Click **OK**.

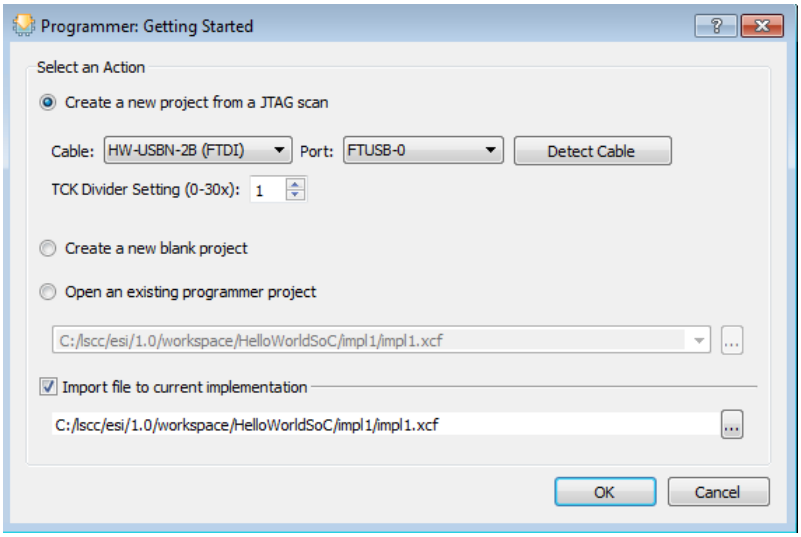
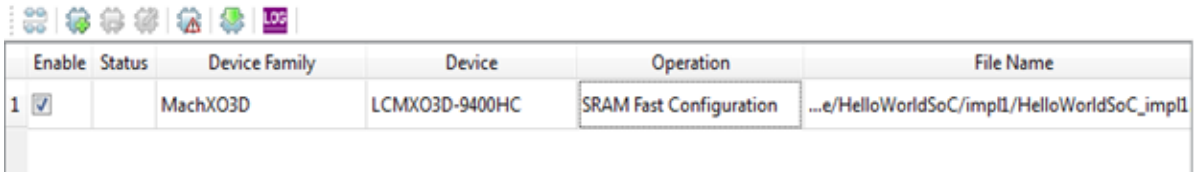


Figure 4.4. Programmer Getting Started Dialog

3. Review the **Device Family**, **Device**, **Operation**, and **File Name** in the Programmer window (as shown below).



4. Click the Program icon to download the programming data file to the device.

4.4. Creating Hello World C Project

Creating C project requires a system environment from SoC project as input.

1. In Propel **Project Explorer** view, select the SoC project *HelloWorldSoC*.
2. Choose **Project** > **Build Project**.

   System environment of the select SoC project is generated under the SoC project folder.

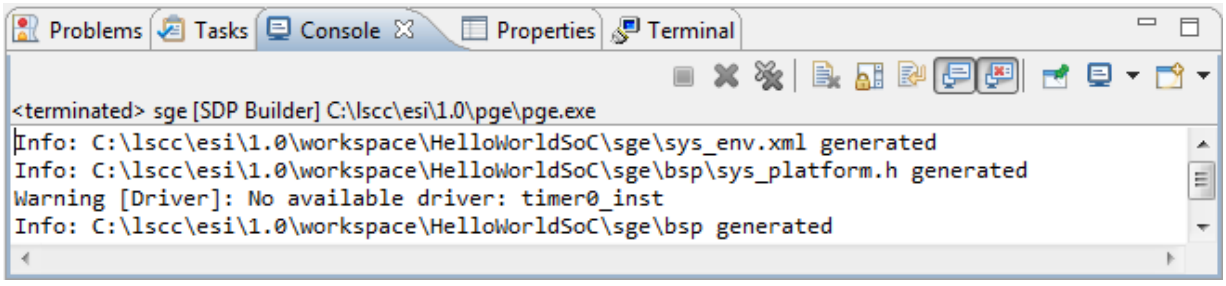


Figure 4.5 Build Result of HelloWorld SoC Project

3. Choose File > New > Lattice C Project.
   The C Project wizard opens with **Load system and bsp** page (Figure 4.6.).

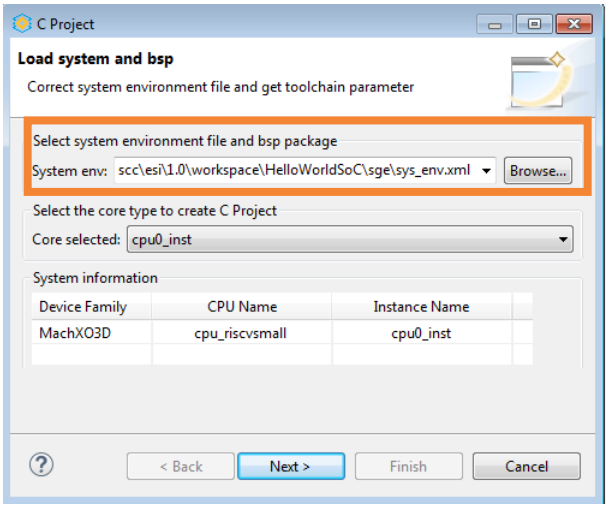


Figure 4.6. Load System and BSP Page

4. Select the system environment file just generated (Figure 4.6.). Click **Next**.
5. Enter project name “HelloWorld” (Figure 4.7). Click **Next**. Then click **Finish**.
   The C project is created and displayed in workbench.

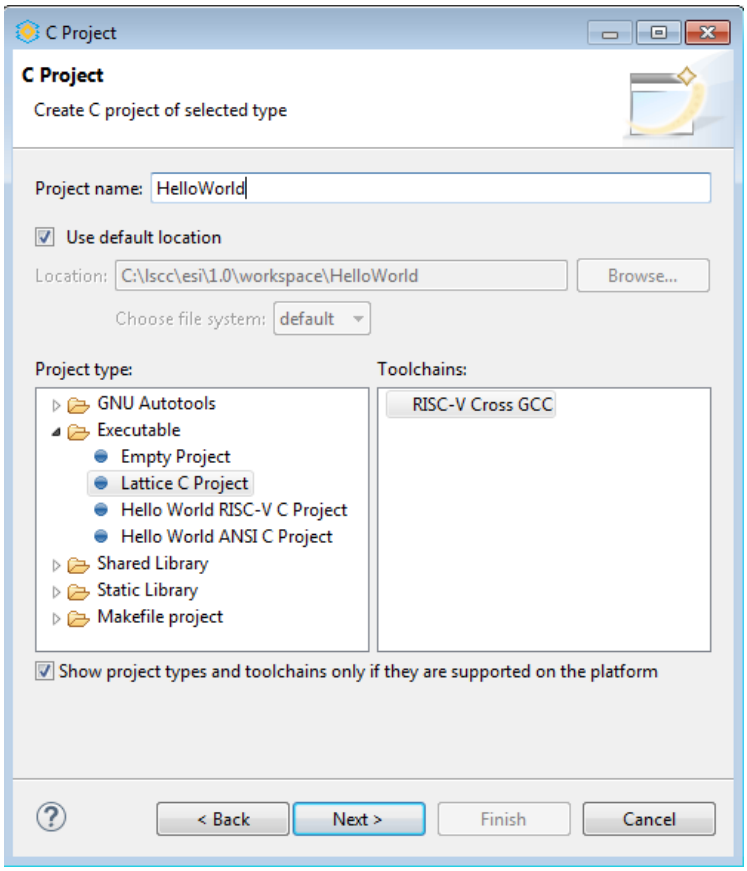


Figure 4.7 Create C Project Page

6. In the Project Explorer view, select the C project “HelloWorld”.
7. Choose Project > Build Project.
8. Check the build result from the Console view (Figure 4.8).

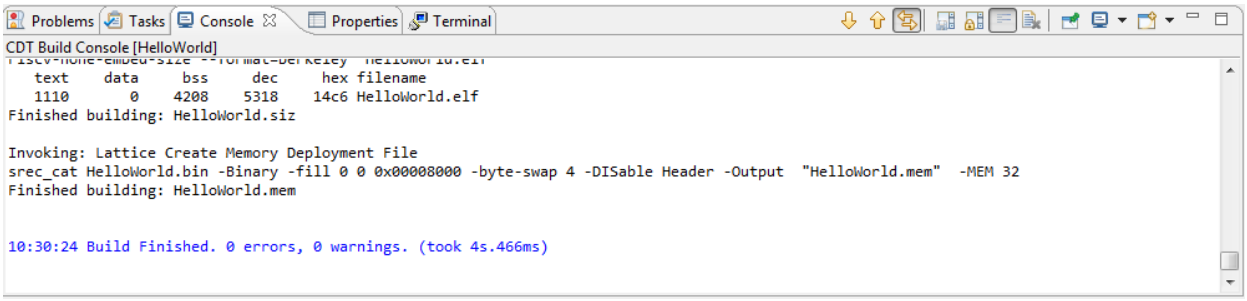


Figure 4.8 Build Result of HelloWorld C Project

4.5. Running demo on MachXO3D Breakout Board – Hello world

1. Find the **Terminal** view nest to the **Console** view. If not found, re-open from **Window** > **Show View** > **Terminal**.
2. In the **Terminal** view, click the **Open a Terminal** icon .
3. Choose the **Serial Terminal** and configure the **Serial port** with **Baud rate** 115200 (Figure 4.9).

   Note: Serial port number depends on specific PC.

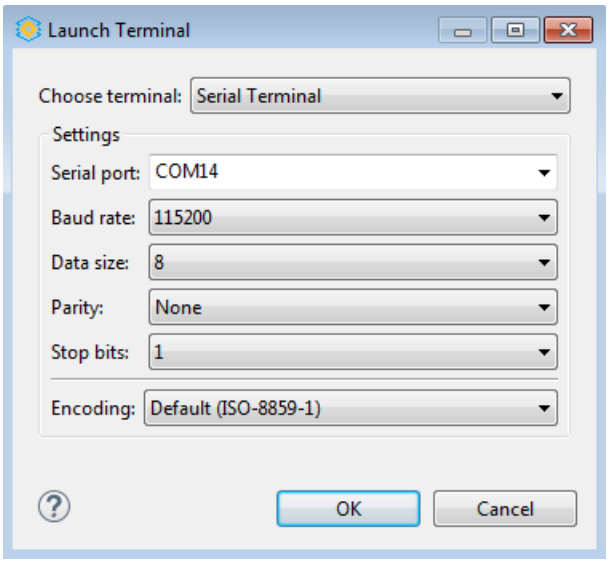


Figure 4.9 Launch Terminal Dialog

4. Click **OK**. A serial connection communicate with UART is ready.
5. In the **Project Explorer** view, select C project “HelloWorld”.
6. Choose **Run** > **Debug Configurations…**
7. Double-click **GDB OpenOCD Debugging** to create a new launch configuration (Figure 4.10).
8. Click **Debug** button.

   Wait for a few seconds for switching to debug perspective, to start the server, and allow it to connect to the target device, start the gdb client, download the application, and then start the debugging session.

   Note: This demo uses default debug configuration options.

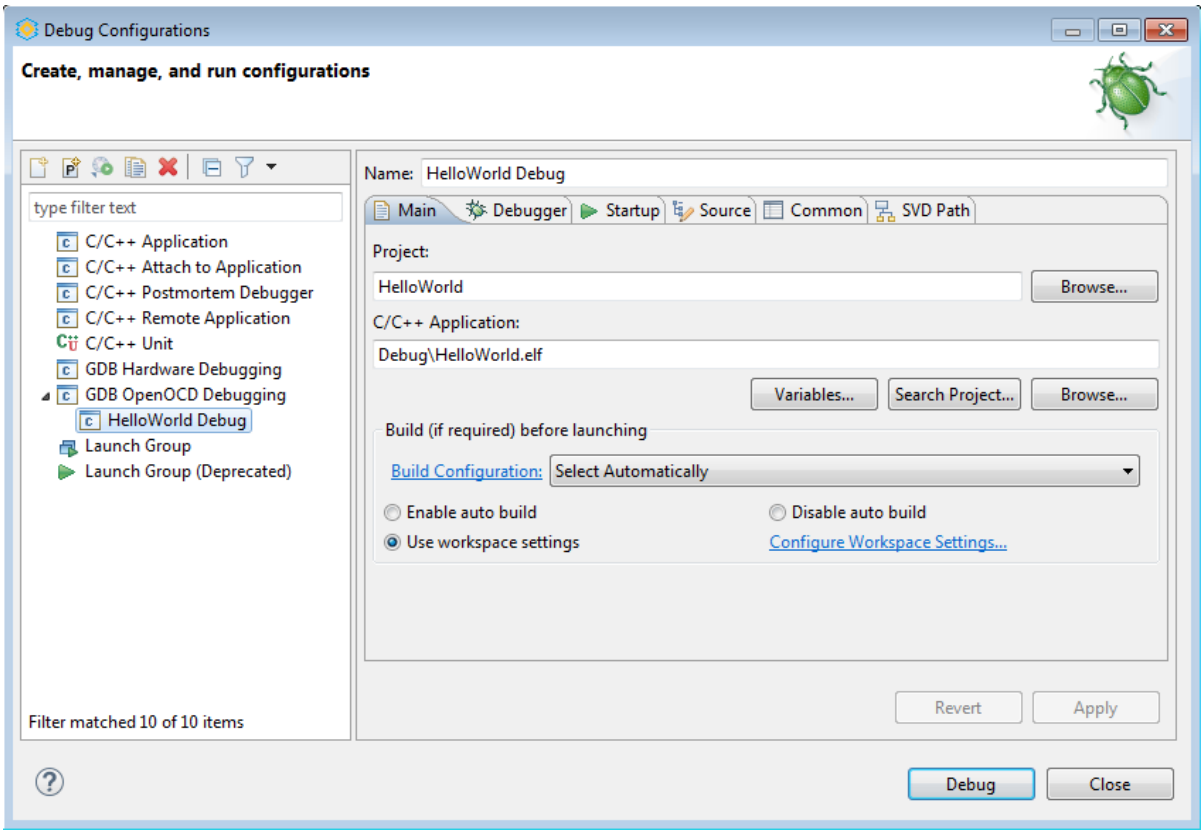


Figure 4.10 Debug Configurations Page

9. Click the Resume icon on the toolbar. The serial terminal outputs “Hello RISC-V world!” (Figure 4.11).

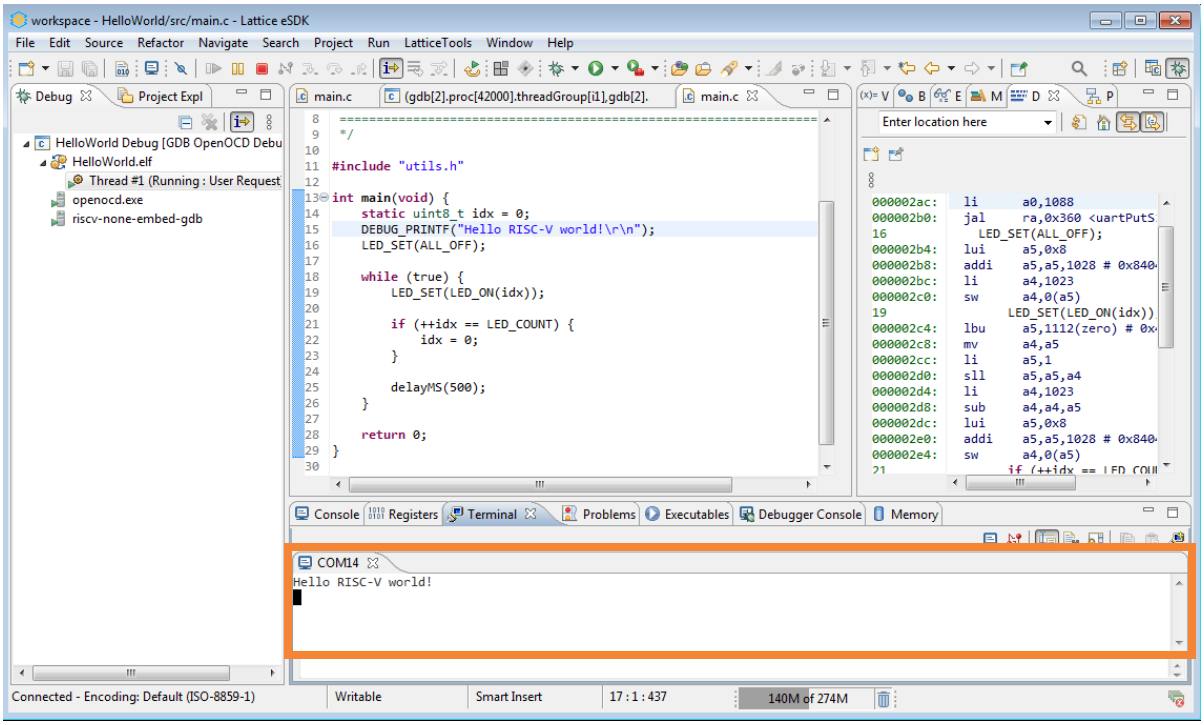


Figure 4.11. Run Result of Hello World Project

References

- MachXO3D Family Data Sheet (FPGA-DS-02026)
- MachXO3D Programming and Configuration Usage Guide (FPGA-TN-02069)
- MachXO3D Breakout Board User Guide (FPGA-UG-02084)

Technical Support Assistance

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

Revision History

Revision 1.0, May 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com