Lattice Nexus Device Multi-Boot Reference Design User Guide

Reference Design

FPGA-RD-02294-1.0

July 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

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
FPGA	Field Programmable Gate Array
IP	Intellectual Property
LED	Light-Emitting Diode
LMMI	Lattice Memory Mapped Interface
RTL	Register Transfer Level
SPI	Serial Peripheral Interface
USB	Universal Serial Bus

1. Introduction

This reference design showcases the Multi-Boot mode supported in Lattice Nexus™ devices. The Multi-Boot mode supports booting from up to six patterns that reside in an external SPI flash device, and up to three patterns for MachXO5™-NX internal flash memory. The patterns include a primary pattern, a golden pattern, and up to four alternate patterns, designated as Alternate Pattern 1 to Alternate Pattern 4.

The device boots by loading the primary pattern from the internal or external flash, depending on the device family. In the static mode, when the PROGRAMN pin is toggled or a REFRESH command is received, Alternate Pattern 1 is always loaded. Each subsequent PROGRAMN/REFRESH event loads the next pattern defined in the Multi-Boot configuration. If loading of the primary pattern or any alternate pattern fails, the device attempts to load the golden pattern. The bitstream pattern sequence, target address of the golden pattern, and target addresses of the alternate patterns are defined during the Multi-Boot configuration process in the Lattice Radiant™ Deployment Tool.

Using the MULTIBOOT primitive allows the device to operate in the dynamic mode. It allows the system to dynamically switch to any of the alternate patterns after the device boots up from the primary pattern while still being protected by a golden pattern. This reference design implements the dynamic mode, which allows the system to dynamically switch between two and three bitstream patterns using the MULTIBOOT primitive. By using Multi-Boot mode, you can combine all the bitstream patterns into a single bitstream image and store it in a single external SPI flash device. This solution decreases cost, reduces board space, and simplifies field upgrades. Note that this reference design is developed using the Lattice CrossLink™-NX device, but the design can be ported to other Nexus devices.

Refer to Multi-Boot User Guide for Nexus Platform (FPGA-TN-02145) for more information.

1.1. Quick Facts

Download the reference design files from the Lattice reference design web page.

Table 1.1. Summary of the Reference Design

General	Target Devices	CrossLink-NX (LIFCL-40-8BG400C)
	Source code format	Verilog
Simulation	Functional simulation	Not performed
	Timing simulation	Not performed
	Test bench	Not available
	Test bench format	Not available
Software Requirements	Software tool and version	2023.2 SP1
	IP version (if applicable)	OSC 1.4.0
Hardware Requirements	Board	CrossLink-NX Evaluation Board
	Cable	USB-A to Mini-B Programming Cable

1.2. Features

A key feature of the Multi-Boot reference design is that it allows the system to dynamically switch between two and three bitstream patterns using the MULTIBOOT primitive while still being protected with a golden pattern.

1.3. Naming Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.3.2. Signal Names

- _*n* are active low, asserted when the value is logic 0.
- _*i* are input signals.
- _*o* are output signals.

2. Directory Structure and Files

Figure 2.1 shows the directory structure.

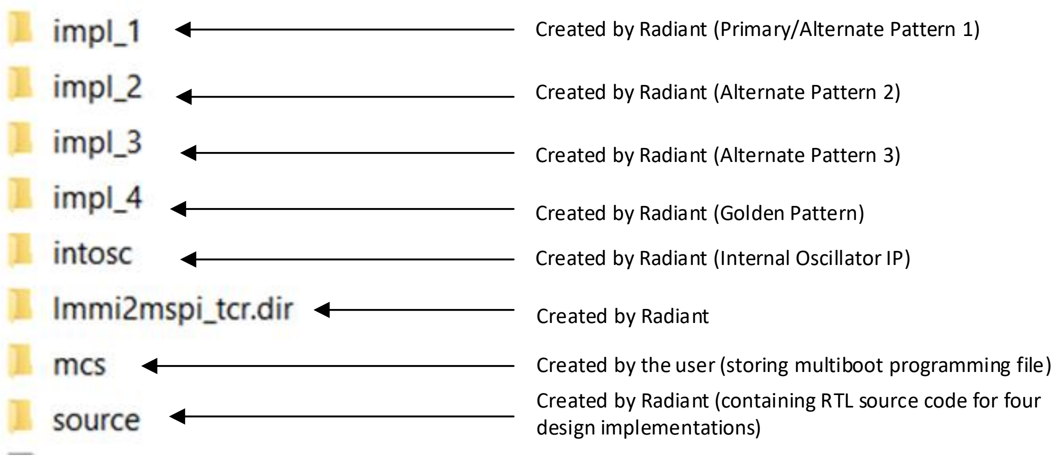

Figure 2.1. Directory Structure

Table 2.1 shows the list of files included in the reference design package.

Table 2.1. File List

Attribute	Description
<Component name>.ipx	This file contains the information on the files associated with the generated IP.
<Component name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact: component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	This file provides the synthesis closed box.
misc/<Component name>_tmpl.v misc/<Component name>_tmpl.vhd	These files provide instance templates for the module.

3. Functional Description

Figure 3.1 shows the top-level block diagram of the reference design. The blocks shown in Figure 3.1 are the fundamental blocks that appear in all four design implementations for Multi-Boot operations.

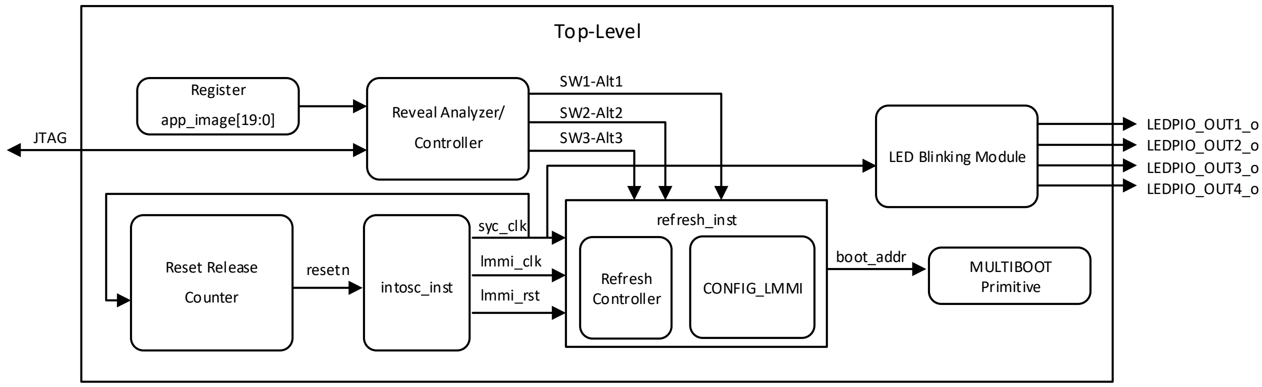

Figure 3.1. Reference Design Block Diagram

3.1. Design Components

The Nexus Multi-Boot reference design includes the following blocks:

- Reset Release Counter: Implements a 16-bit counter to control the reset of intosc_inst and refresh_inst blocks. It holds the intosc_inst and refresh_inst blocks in reset after the device is configured. Then, it releases these blocks from reset after the counter value reaches 32768.
- intosc_inst: This is an oscillator module. It generates clock sources (sys_clk and lmmi_clk) to the refresh controller and CONFIG_LMMI blocks in the refresh_inst block and the LED blinking module. Both sys_clk and lmmi_clk clocks are running at 50 MHz. Refer to Nexus OSC Module - Lattice Radiant Software (FPGA-IPUG-02065) for more information.
- LED blinking module: Implements a counter to blink the four LEDs on board once the device is configured successfully.
- refresh_inst: Consists of refresh controller and CONFIG_LMMI blocks.
    - CONFIG_LMMI: LMMI (Lattice Memory Mapped Interface) interface to the configuration block. Refer to the CONFIG_LMMI page in Lattice Radiant Software Help for more information.
    - Refresh controller: Implements a state machine controller to send the necessary commands to the CONFIG_LMMI block, and the boot address to the MULTIBOOT primitive to boot the desired alternate pattern stored in an external SPI flash. The refresh controller performs the following sequences. See Figure 3.2 for more details.
        - Send 32-bit boot address to MULTIBOOT primitive.
        - Execute ISC_ENABLE_X – Similar to ISC_ENABLE. However, this command puts the device into the Transparent mode. Executing this command is essential to enable the device to execute the next command, which is the LSC_PROG_CNTRL0 command.
        - Execute LSC_PROG_CNTRL0 – Set the SPIM bit in Control Register 0 to 1. When this bit is set to 1 and once the REFRESH command is executed, it enables the device to boot from the image stored in an external SPI flash according to the boot address sent to the MULTIBOOT primitive.
        - Execute ISC_DISABLE – Exit the Transparent mode.
        - Execute LSC_REFRESH – Equivalent to toggling the PROGRAMN pin. Once this command is executed, the device starts to load the desired alternate pattern from the external SPI flash according to the boot address sent to the MULTIBOOT primitive. If loading of the image fails, the device falls back to load the golden pattern.

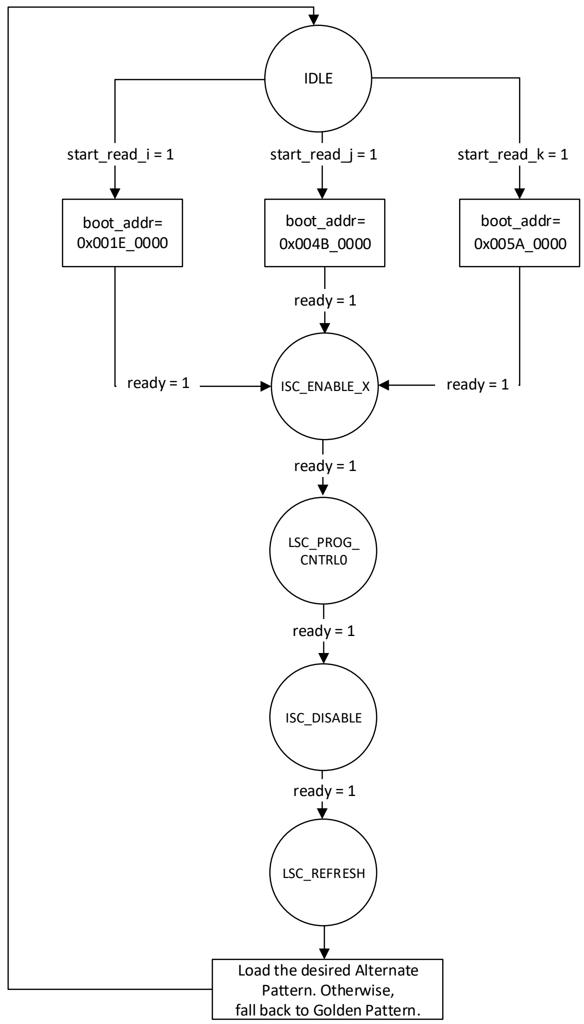

Figure 3.2. State Diagram of Refresh Controller Instance

- MULTIBOOT primitive: A wrapper for the interface to perform the Multi-Boot functionality. It enables the booting to load the desired alternate pattern by sending the REFRESH command to the CONFIG_LMMI block. The AUTOREBOOT port is an unused input port. It is recommended to tie it to 0. Refer to the MULTIBOOT primitive page in Lattice Radiant Software Help for more information.
- Reveal Analyzer/Controller: Reveal Analyzer is used to determine which alternate pattern is running on the device. Reveal Controller is used to implement three virtual switches (Figure 3.3). You can load the desired alternate pattern by toggling the virtual switch. Refer to the following for more details.
  - SW1-Alt1: To load Alternate Pattern 1 from address 0x001E_0000
  - SW2-Alt2: To load Alternate Pattern 2 from address 0x004B_0000
  - SW3-Alt3: To load Alternate Pattern 3 from address 0x005A_0000

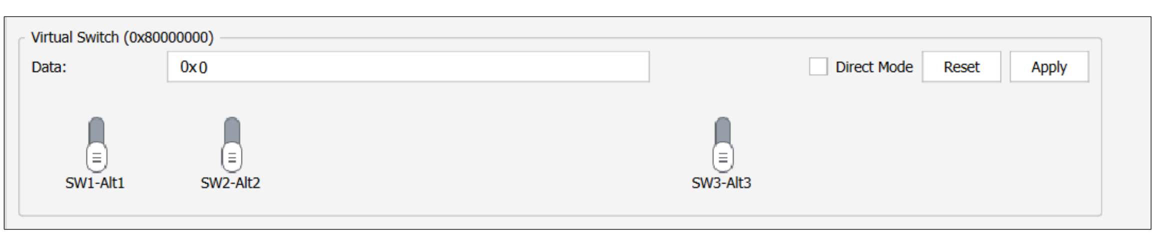

Figure 3.3. Three Virtual Switches

4. Reference Design Parameter Description

The Multi-Boot reference design includes the parameters shown in Table 4.1. You can modify the parameters defined in the top.v file.

Table 4.1. Parameters in <FILE NAME>

Parameter	Default Value	Description
boot_addr_1	32'h001E_0000	Boot address for Alternate Pattern 1
boot_addr_2	32'h004B_0000	Boot address for Alternate Pattern 2
boot_addr_3	32'h005A_0000	Boot address for Alternate Pattern 3

5. Signal Description

The input/output interface signals for the top.v are shown in Table 5.1.

Table 5.1. Primary I/O

Port Name	I/O	Width	Description
LEDPIO_OUT1_o	Out	1	Output to blink LED D3
LEDPIO_OUT2_o	Out	1	Output to blink LED D4
LEDPIO_OUT3_o	Out	1	Output to blink LED D5
LEDPIO_OUT4_o	Out	1	Output to blink LED D6

6. Running the Reference Design

This section describes how to run the Multi-Boot reference design using the Lattice Radiant software. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

The project consists of four design implementations that can be used to distinguish the pattern running on the device. They differ in a unique 20-bit apps_image register value and the output port(s) to blink LED(s) on board. Below are the apps_image values for each design implementation.

- impl_1: apps_image=0xcafe1 and output port to blink one LED (D3)
- impl_2: apps_image=0xcafe2 and output ports to blink two LEDs (D3 and D4)
- impl_3: apps_image=0xcafe3 and output ports to blink three LEDs (D3, D4, and D5)
- impl_4: apps_image=0xcafe4 and output ports to blink four LEDs (D3, D4, D5, and D6)

6.1. Compiling the Reference Design

The reference design file includes the pre-compiled files and bitstreams (.bit and .mcs files) for you to start quickly. However, you can recompile the reference design in a newer version of the Lattice Radiant software or after adding/modifying the user logic if you choose to do so. To do that, you can set the design implementation as the active implementation, make the desired changes, and recompile the design through the standard compilation flow.

Note: .mcs files are the data record files that are in the format commonly known as Intel Hex, Motorola Hex, or Extended Tektronix Hex. They are also known as addressed record files. The advantages include small size and printability, which is good for record-keeping. This type of file is not directly consumable by the utilities supporting it.

6.2. Generating the Bitstream File

This section provides the procedure of creating your FPGA bitstream file using the Lattice Radiant software.
To create the FPGA bitstream file, follow the steps below.

1. Open the Lattice Radiant software, as shown in Figure 6.1.

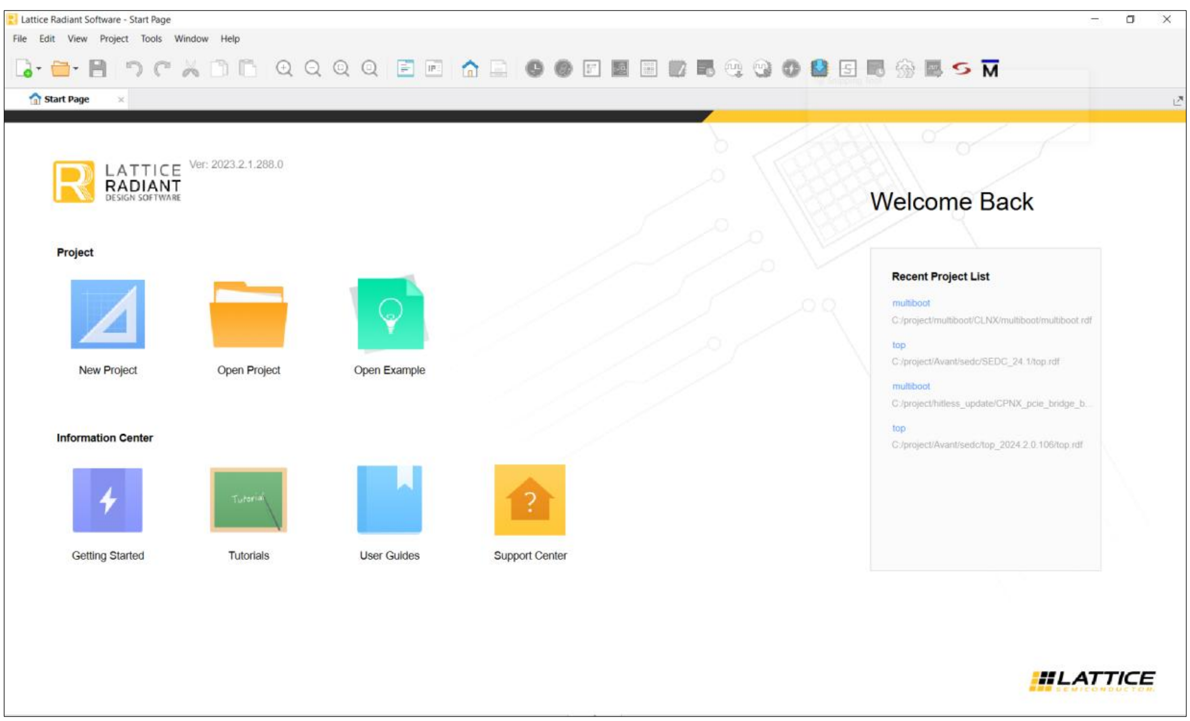

Figure 6.1. Lattice Radiant Software

2. Click **File > Open Project** and from the project database, browse to the reference design project, the multiboot folder, and open the Radiant project file (multiboot.rdf), as shown in Figure 6.2.

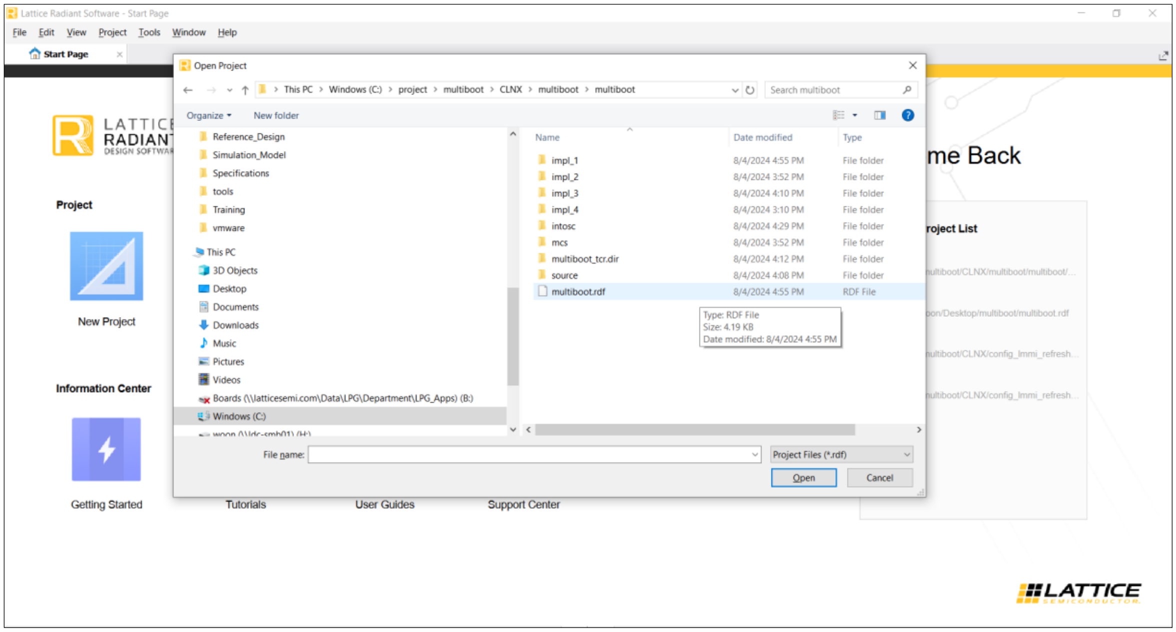

Figure 6.2. Open Project File

3. Click **Export Files** to generate the bit file (Figure 6.3). View the log message in the Export Reports folder for the generated bitstream. Once the compilation is successful, the generated bit file is in the project implementation folder, for example, multiboot\impl_1\multiboot_impl_1.bit. Repeat this step for other design implementations if you need to recompile the design.

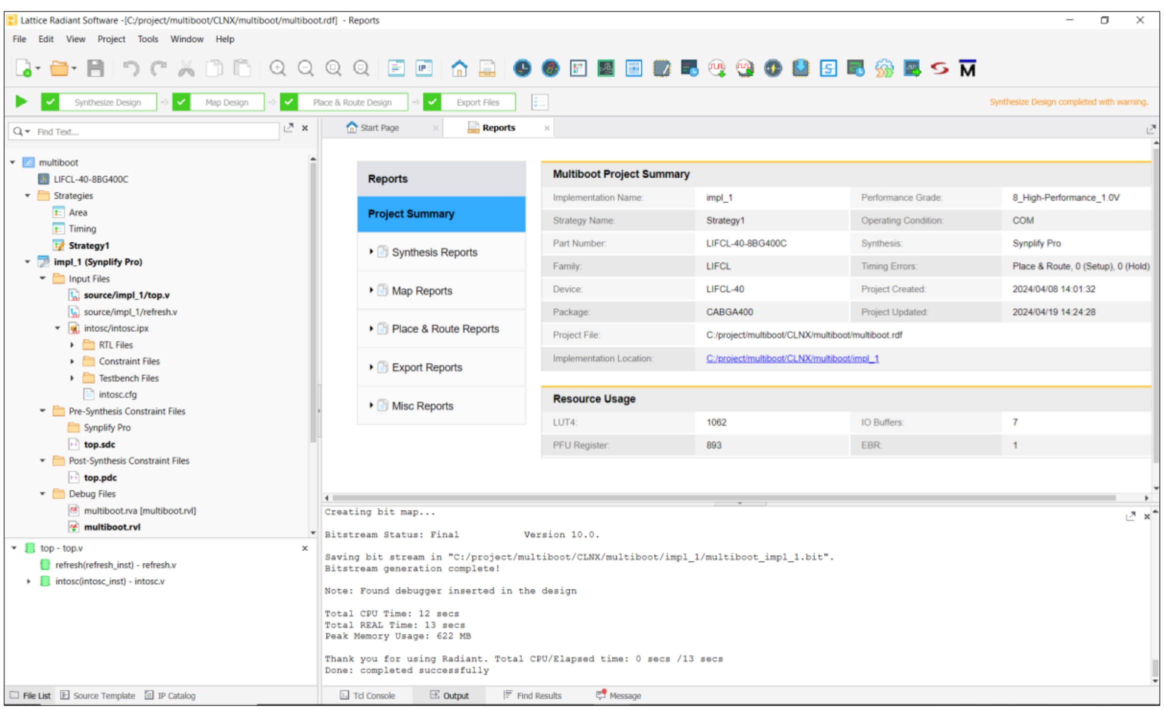

Figure 6.3. Generated Bitstream Log

6.3. Generating Initial Multi-Boot Programming File

This section describes how to generate the initial Multi-Boot programming file.
Note: You can use the provided Multi-Boot programming file located in \Desktop\Multi Boot Tutorial\mcs\multiboot.mcs directory and skip these steps.

Follow the steps below to generate the single Multi-Boot programming file (.mcs).

1. Launch the Radiant Deployment Tool from Radiant Programmer by clicking **Tools** > **Deployment Tool** (Figure 6.4).

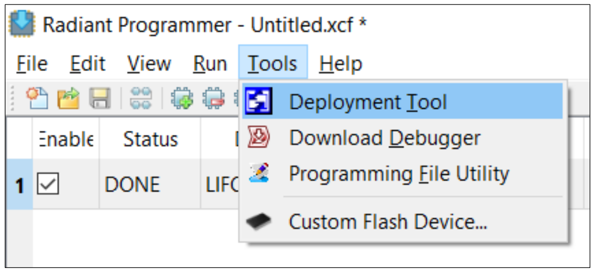

Figure 6.4. Launch Radiant Deployment Tool from Radiant Programmer

2. In the Radiant Deployment Tool window, select External Memory for **Function Type** and Advanced SPI Flash for **Output File Type**. Click **OK** (Figure 6.5).

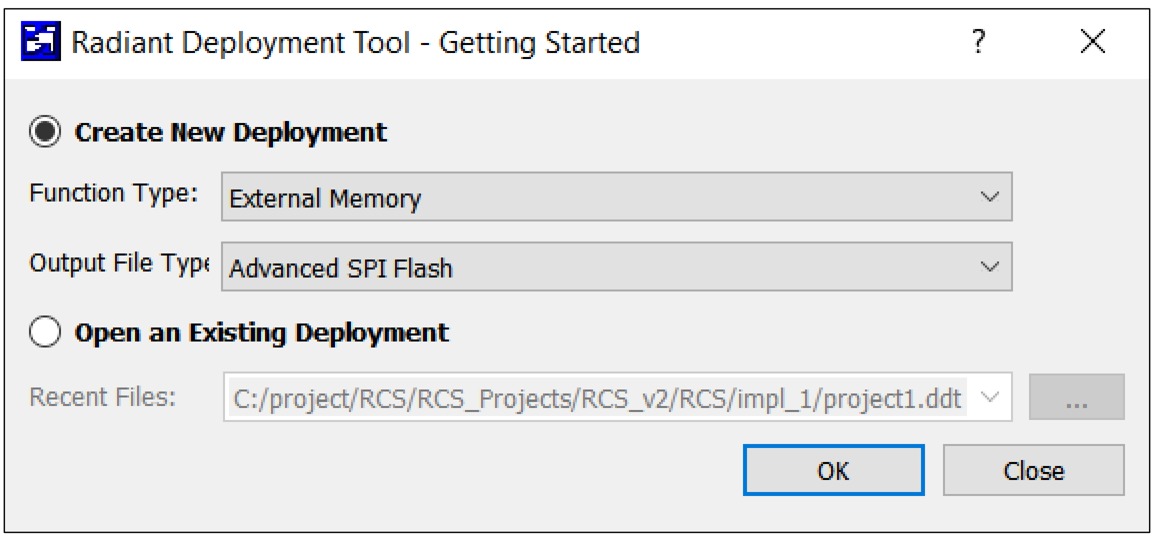

Figure 6.5. Radiant Deployment Tool – Getting Started

3. A four-step wizard opens, showing: External Memory: Advanced SPI Flash.
   a. In Step 1 of 4: Select Input File(s), select a .bit file for **File Name** and click **Next** (Figure 6.6). This is the Primary Image.

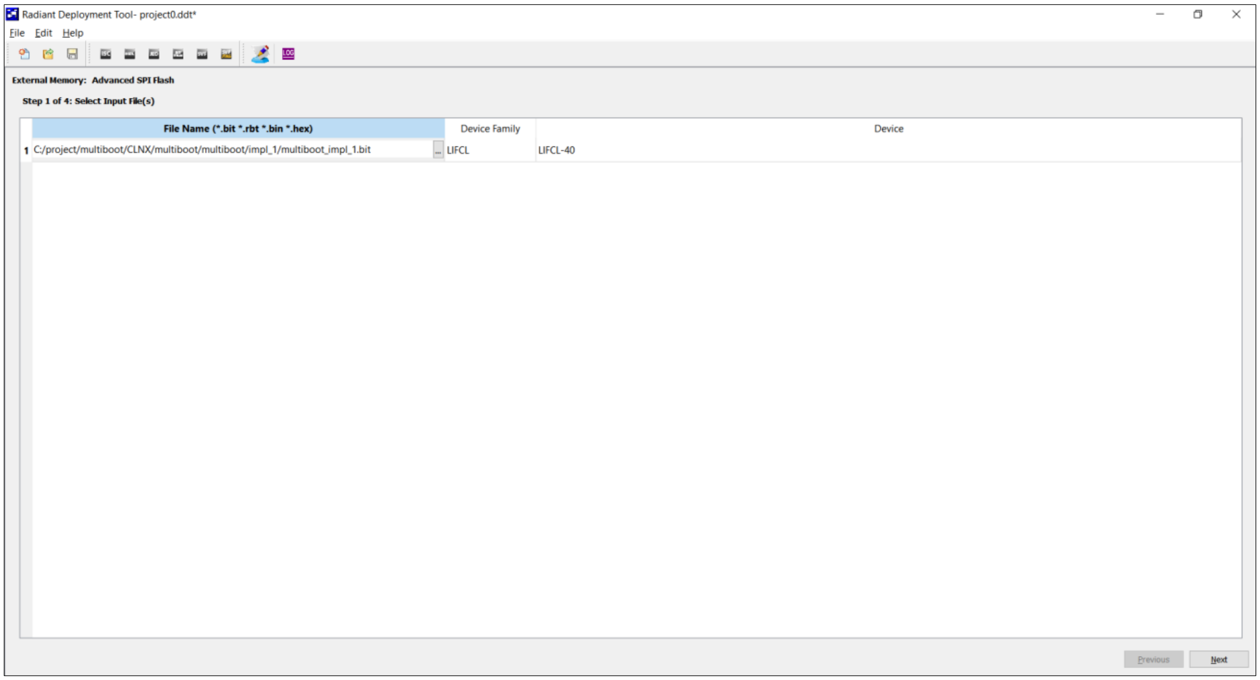

Figure 6.6. Step 1 of 4: Select Input File(s)

b. In Step 2 of 4: Advanced SPI Flash Option (Options tab), select the following options (Figure 6.7):
- Output Format: Intel Hex
- SPI Flash Size(Mb): 128
- SPI Flash Read Mode: Standard Read

Figure 6.7. Options Tab of Step 2 of 4: Advanced SPI Flash Option

c. In Step 2 of 4: Advanced SPI Flash Option (Multiple Boot tab), select the following options (Figure 6.8):
- Check Multiple Boot.
- Golden Pattern:
  Select the multiboot_impl_4.bit file in the multiboot/impl_4 folder.
  Starting Address: 0x000F0000.
- Number of Alternate Patterns: 3 – to program 3 alternative images in the external SPI flash.
- Alternate Pattern 1:
  Select the multiboot_impl_1.bit file in the multiboot/impl_1 folder.
  Starting Address: 0x001E0000 – make sure the value is the same as the value defined for parameter boot_addr_1 in top.v.
  Next Alternate Pattern to Configure: Alternate Pattern 2
  Note: You can select any available option because it no longer affects the next pattern to load once the MULTIBOOT primitive is instantiated in your design.
- Alternate Pattern 2:
  Select the multiboot_impl_2.bit file in the multiboot/impl_2 folder.
  Starting Address: 0x004B0000 – make sure the value is the same as the value defined for parameter boot_addr_2 in top.v.
  Next Alternate Pattern to Configure: Alternate Pattern 3
  Note: You can simply select any available option because it no longer affects the next pattern to load once the MULTIBOOT primitive is instantiated in your design.
- Alternate Pattern 3:
  Select the multiboot_impl_3.bit file in the multiboot/impl_3 folder.
  Starting Address: 0x005A0000 – make sure the value is the same as the value defined for parameter boot_addr_3 in top.v.
  Next Alternate Pattern to Configure: Primary Pattern
  Note: You can simply select any available option because it no longer affects the next pattern to load once the MULTIBOOT primitive is instantiated in your design.
- Click **Next**.

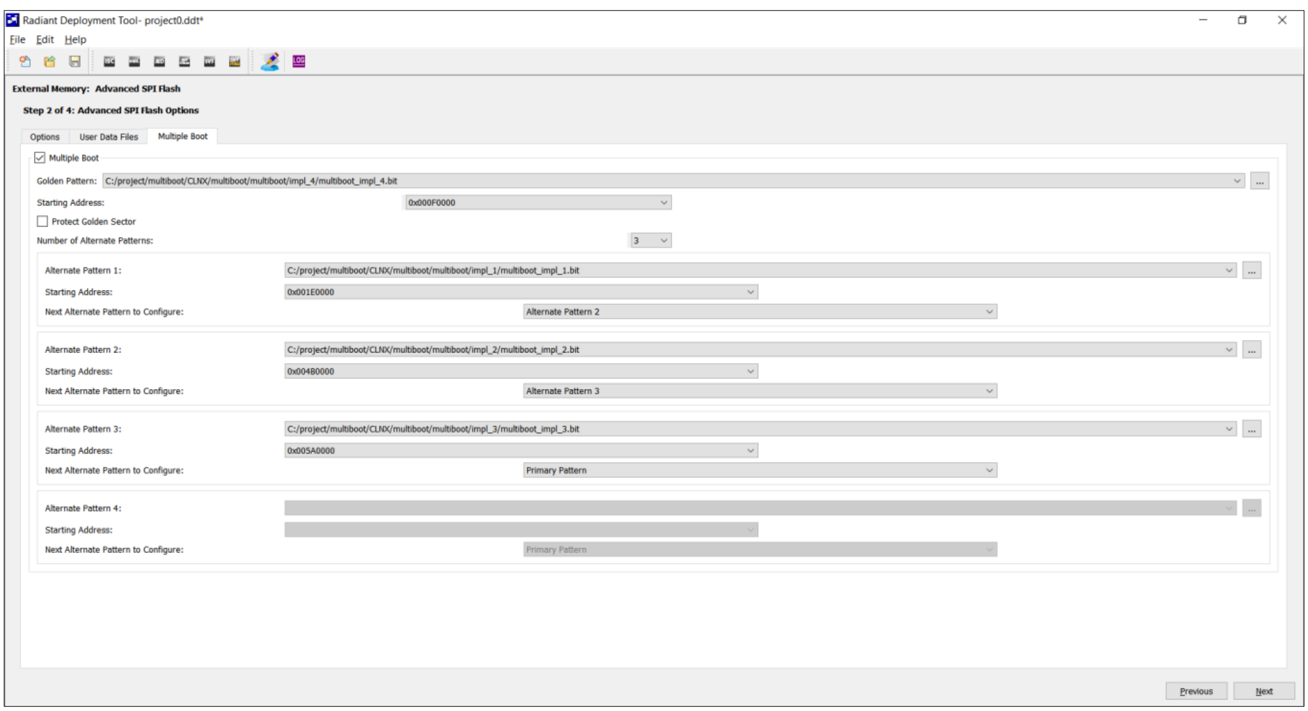

Figure 6.8. Multiple Boot Tab of Step 2 of 4: Advanced SPI Flash Option

d. In Step 3 of 4: Select Output File(s)
Browse to the location to store the .mcs file in the Output File1 field and click **Next** (Figure 6.9).

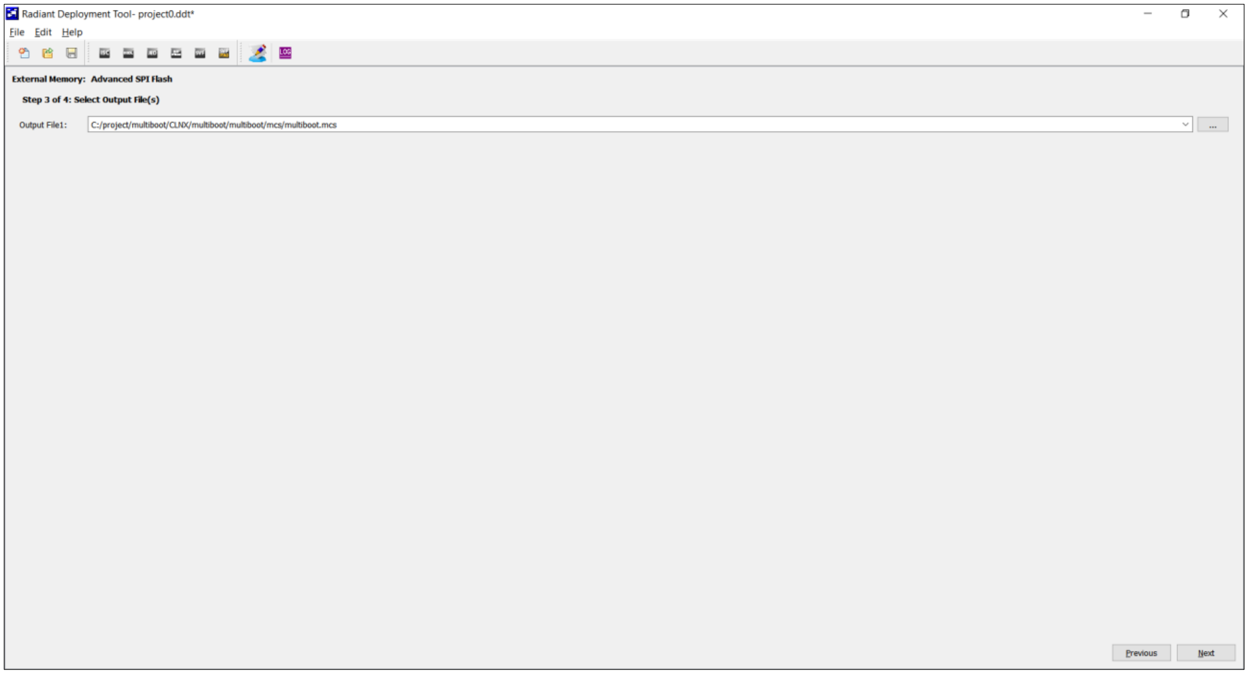

Figure 6.9. Step 3 of 4: Select Output File (s)

e. In Step 4 of 4: Generate Deployment
Click **Generate** to generate the .mcs file. You can close the window when the Deployment Tool prompts Lattice Deployment Tool has exited successfully (Figure 6.10).

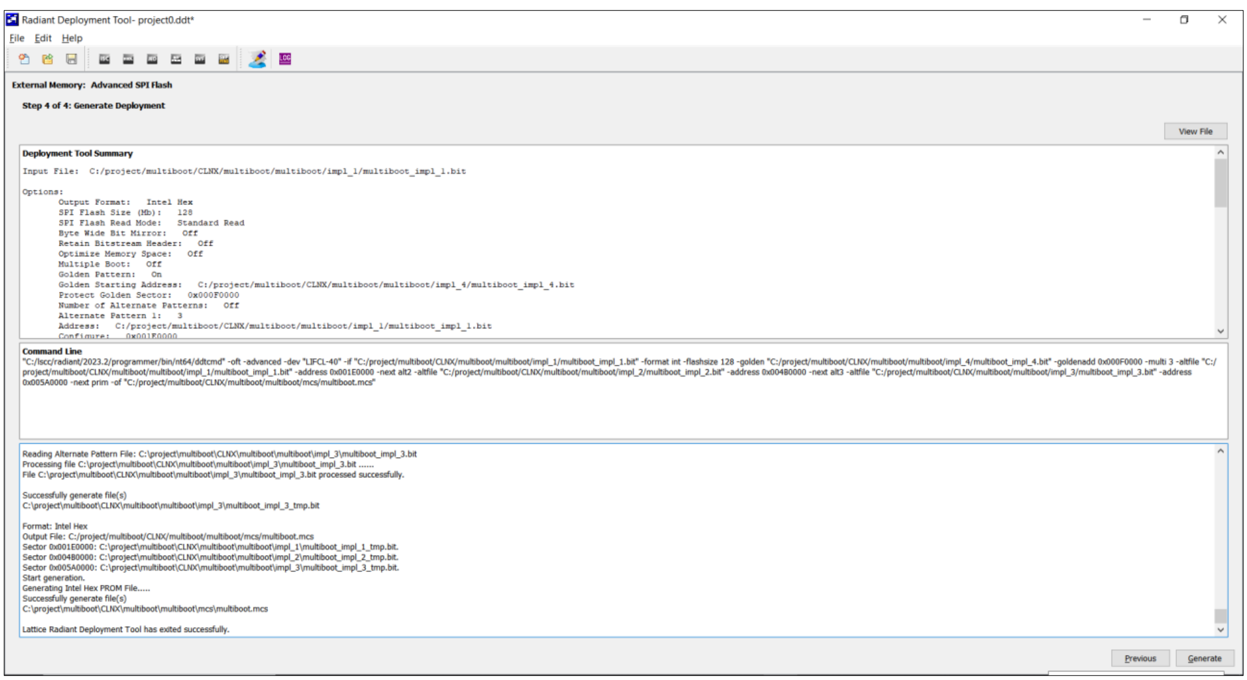

Figure 6.10. Step 4 of 4: Generate Deployment

7. Implementing the Reference Design on Board

7.1. Hardware Requirements

- CrossLink-NX Evaluation Board
- USB-A to Mini-B Programming Cable

7.2. Programming .mcs into External SPI Flash

1. Launch Radiant Programmer and select **Edit** > **Device Properties**.
2. Select the device properties settings, as shown in Figure 7.1. Click **OK**.
3. Click **Program Device** to start programming the board.

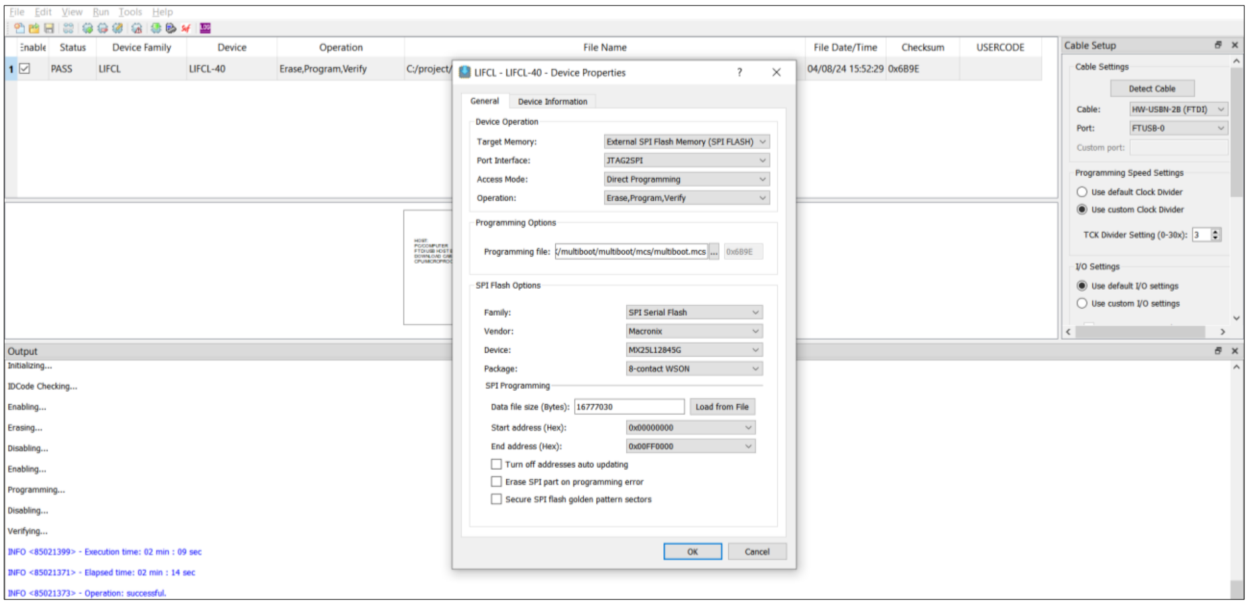

Figure 7.1. Device Properties Window

7.3. Running the Design

7.3.1. Performing Multi-Boot – without Corrupted Pattern

1. The device is configured with Primary Image after the device is powered on, and LED D3 blinks once the device is configured successfully. To confirm this:
   a. Open the multiboot.rdf project located in the multiboot project directory in the Lattice Radiant software.
   b. Launch and run Reveal Analyzer by double-clicking the **multiboot.rva** file (Figure 7.2).

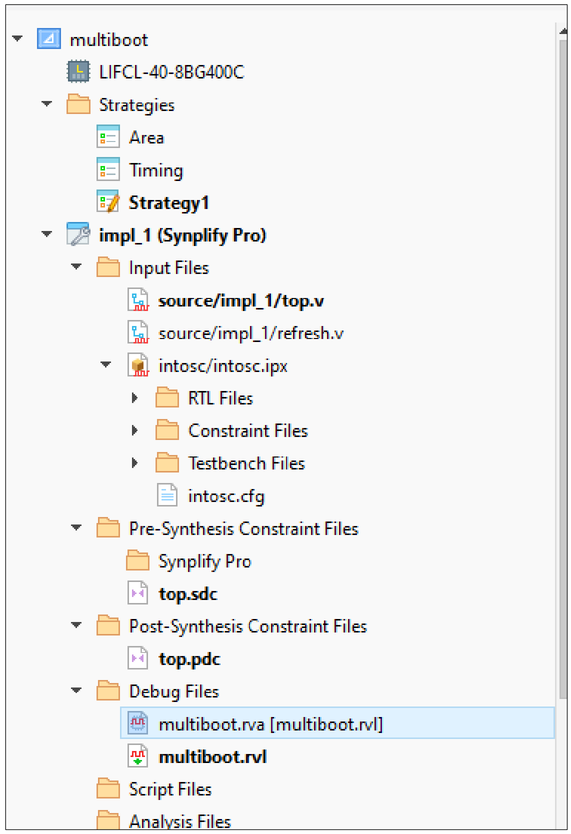

Figure 7.2. multiboot.rva File

c. Open the waveform tab and apps_image displays CAFE1, indicating that Primary Image is running on the device (Figure 7.3).

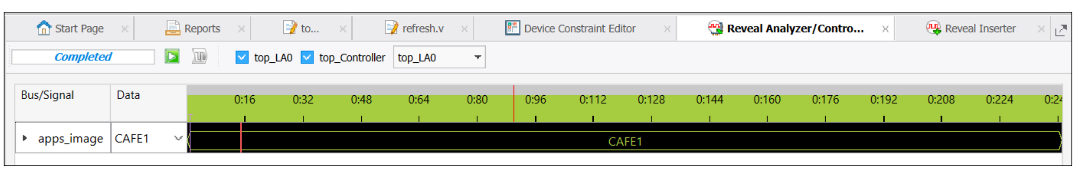

Figure 7.3. apps_image Showing CAFE1

2. Reboot the device with Alternate Pattern 1: Launch the Reveal Controller by selecting the top_Controller from the drop-down menu (Figure 7.4).

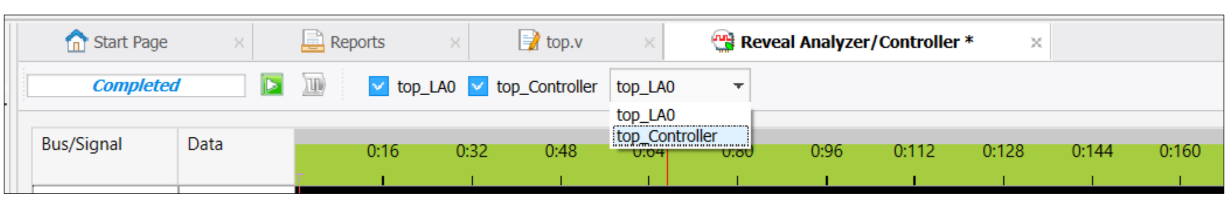

Figure 7.4. Select top_Controller

3. Toggle the switches in Reveal Controller to reboot/reconfigure the device with the desired image. Check the Direct Mode box before toggling the switches (Figure 7.5).

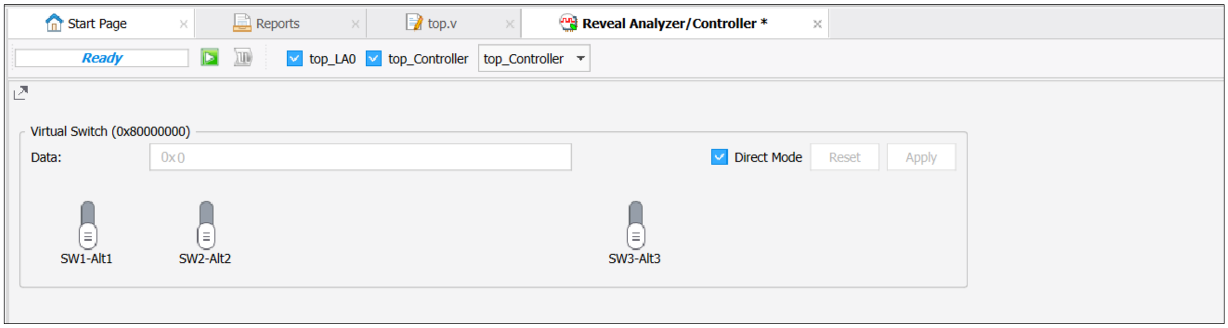

Figure 7.5. Toggle Switches in Reveal Controller

4. Toggle SW1-Alt1 (Figure 7.6).

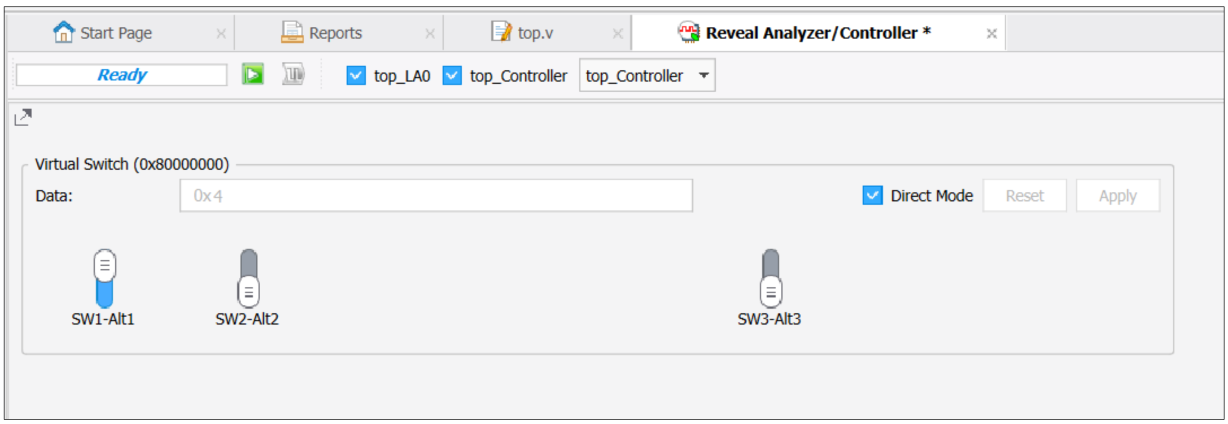

Figure 7.6. Toggle SW1-Alt1

5. Run Reveal Analyzer by clicking the button and return to the waveform tab. The apps_image still displays CAFE1 and LED D3 is blinking, indicating that Alternate Pattern 1 is running on the device (Figure 7.3).
   Note: You are seeing the same output as before because the Alternate Pattern 1 is set as the primary pattern.
6. Reboot the device with Alternate Pattern 2:
   a. Toggle SW2-Alt2 in top_Controller (Figure 7.7).

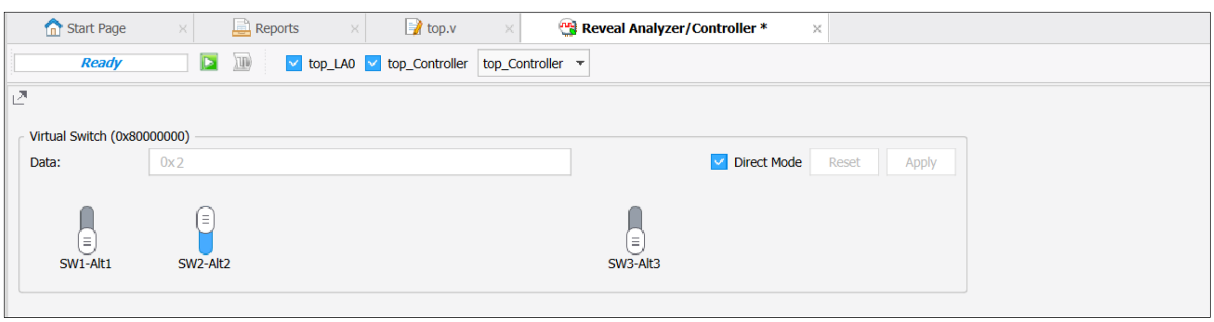

Figure 7.7. Toggle SW2-Alt2

   b. Run Reveal Analyzer and the apps_image displays CAFE2. LED D3 and D4 are blinking, indicating that Alternate Pattern 2 is currently running on the board (Figure 7.8).

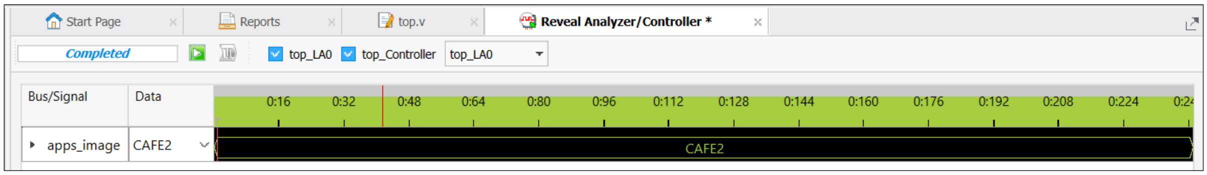

Figure 7.8. apps_image Showing CAFE2

7. Reboot the device with Alternate Pattern 3:
   a. Toggle SW3-Alt3 in top_Controller (Figure 7.9).

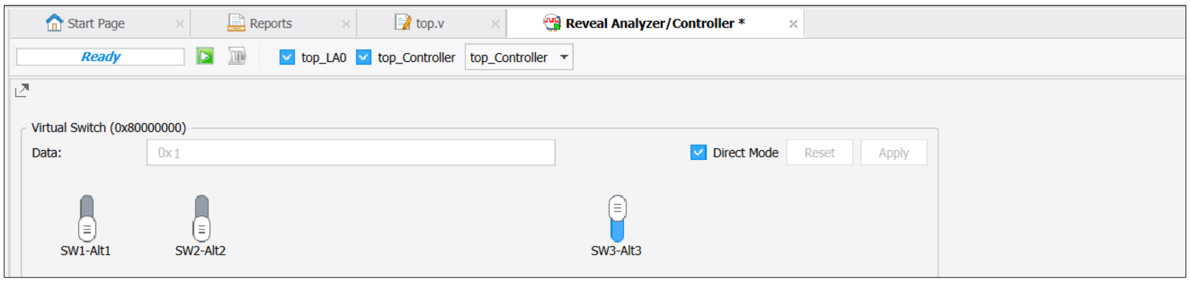

Figure 7.9. Toggle SW3-Alt3

   b. Run Reveal Analyzer and the apps_image displays CAFE3. LED D3, D4, and D5 are blinking, indicating that Alternate Pattern 3 is currently running on the board (Figure 7.10).

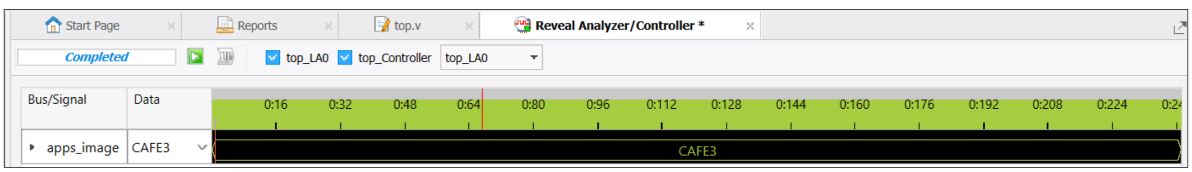

Figure 7.10. apps_image Showing CAFE3

7.3.2. Performing Multi-Boot – with Corrupted Pattern

To test fallback to the golden pattern function, reprogram Alternate Pattern 3 with a corrupted pattern and then boot the board with the corrupted pattern.

1. Close Reveal Analyzer/Controller and open Radiant Programmer.
2. Reprogram a corrupted image (multiboot/impl_3/multiboot_impl_3_corrupted.bit) at Start Address: 0x005A0000 using Background Programming mode (Figure 7.11).

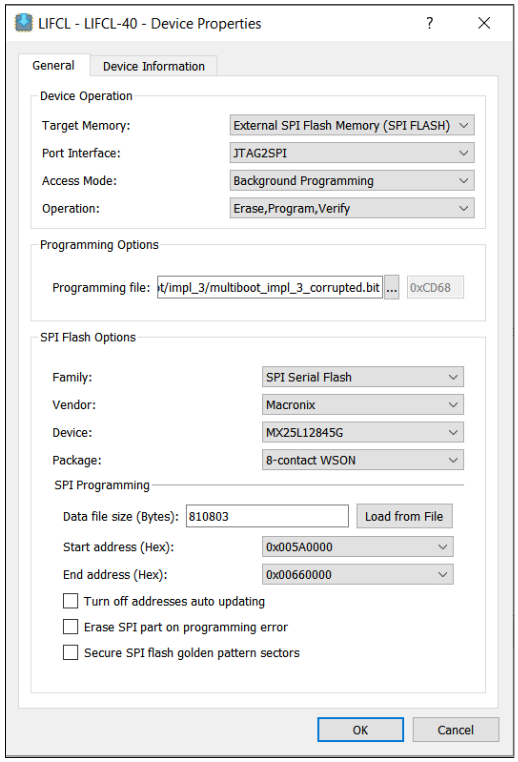

Figure 7.11. Reprogram Corrupted Image

a. After SPI flash programming is completed, run the Reveal Analyzer/Controller.
b. Toggle SW3-Alt3 in top_Controller to boot using Alternate Pattern 3 (Figure 7.9).
c. Run Reveal Analyzer and now the apps_image displays CAFE4 (Figure 7.12). LED D3, D4, D5, and D6 are blinking, indicating that the device fails to boot from the corrupted Alternate Pattern 3 and falls back to the golden pattern.

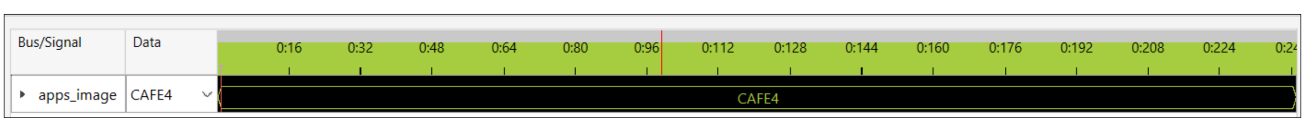

Figure 7.12. apps_image Showing CAFE4

References

- Lattice Nexus Platform web page
- Multi-Boot User Guide for Nexus Platform (FPGA-TN-02145)
- sysCONFIG User Guide for Nexus Platform (FPGA-TN-02099)
- Nexus OSC Module - Lattice Radiant Software (FPGA-IPUG-02065)
- Lattice Radiant Software Help
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 1.0, July 2026

Section	Change Summary
All	Updated the document title to its current.
Functional Description	In the Design Components section, updated the following statement from *Both sys_clk and lmmi_clk clocks are running at 12.1622 MHz* to *Both sys_clk and lmmi_clk clocks are running at 50 MHz*.

Revision 0.80, August 2024

Section	Change Summary
All	Initial preliminary release.

LATTICE SEMICONDUCTOR®

www.latticesemi.com