LatticeECP3, LatticeECP2/M, ECP5 and ECP5-5G Dual Boot and Multiple Boot Feature

Technical Note

FPGA-TN-02203-1.7

September 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

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
EBR	Embedded Block RAM
FPGA	Field-Programmable Gate Array
JTAG	Joint Test Action Group
PCS	Physical Coding Sublayer
PROM	Programmable Read-Only Memory
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
USB	Universal Serial Bus

1. Introduction

One of the biggest risks in field upgrade applications is disruption during the field upgrade process. Disruption can occur as:

- Power disruption
- Communications disruption
- Data file corruption

To eliminate the risk completely, the device has a Dual Boot Feature that switches to load from the second known good (Golden) pattern when the first pattern is corrupted. This feature enhances the reliability of a field upgradeable system.

Even if the system does not require a field upgrade, the pattern corruption can still occur due to the following problems caused by the SPI Flash devices:

- Read fatigue
- Charge loss

The Golden pattern is less affected by these problems since most of the time it is in a dormant state.

The LatticeECP™ and LatticeSC™ families were the industry's first FPGA devices to support industry standard SPI Flash devices as the single image boot PROM. The LatticeECP2/M is the first FPGA family to support the Dual Boot feature using only one industry standard SPI Flash device. This approach has many advantages:

- Lower cost
  - One-chip solution
  - Industry standard SPI Flash devices
  - Density can be as high as 128 Mbit
- Much smaller board space
  - 8-pin SOIC packages for 16 Mbit or less
  - 16-pin SOIC packages for 32 Mbit or larger
- Simple field upgrade
  - Uses industry standard SPI interface
  - Uses JTAG port
- Reliable field upgrade
  - Uses the SPI Flash device's Sector Lock feature to protect the Golden pattern
  - Locates the Golden pattern strategically into the locked sectors

The abundant supply of SPI Flash devices offers system designers many low cost system solutions. For example:

- Density ranges from 1 Mbit to 128 Mbit
  - One SPI Flash device can store up to two bitstreams with 64-Mbit bits each
- Attractive features that are applicable to the Dual Boot feature:
  - Soft Flash Lock by sectors protecting the patterns from erroneous re-programming
  - High-speed read at >50 MHz for fast boot-up time
  - Power-down option to minimize power consumption

2. Glossary

Alternative Boot

After the FPGA device has been configured, this pattern is loaded when the PROGRAMN pin is toggled or the Refresh instruction is issued. Up to four Alternative Boot patterns are possible.

Binary Hex Data File (.bin File)

The data image of the Hex data file in binary format. All Hex data files are converted into this format prior to consumption. This type of file is not printable.

Bitstream Data File (.bit File)

The configuration data file, for a single FPGA device, in the format that can be loaded directly into the FPGA device to configure the SRAM cells. The file is expressed in binary Hex format. The file is not printable.

Configure

Write the pattern into the SRAM fuses of the FPGA device and wake up.

Dual Boot

The device has two patterns, a Primary pattern and a Golden pattern, to choose to load.

Flash Lock

The feature provides protection to the Flash fuses against accidental erase or corruption. Most of the SPI Flash devices support Soft Lock. Lock choices include:

- Whole device
- Bottom half
- Bottom quarter
- Last sector

Details can be found in the SPI Flash device data sheet.

Golden Boot

The guaranteed good pattern loaded into the FPGA device when booting failure occurs. It is also known as the root boot. Only one Golden boot pattern is allowed.

Hex Data File (.exo, .mcs, .xtek Files)

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 its small size and it is printable, and thus good for record keeping. This type of file is not directly consumable by the utilities supporting it.

Multiple Boot (Only Available for ECP5 and ECP5-5G)

The device has more than two patterns, a Primary pattern, a Golden pattern and some Alternative patterns, to choose to load.

Primary Boot

Upon power cycling, the FPGA device loads this pattern in first. Only one Primary pattern is allowed.

Program

Writes into the selected Flash cells state a logical zero (0) (close fuse).

Refresh

The action loads the pattern from a non-volatile source to configure the FPGA device.

SPI

Stands for the Serial Peripheral Interface defined originally by Motorola.

Sector (Block)

The smallest number of bytes of Flash fuses can be erased at the same time by the erase command.

3. Resources

The minimum SPI Flash density required to support the Dual Boot feature is listed in Table 3.1.

Note: LatticeECP2/M™, ECP5™, and ECP5-5G™ devices support bitstream compression. Table 3.1 does not consider compression. The LatticeECP3™ bitstream size shown is the maximum bitstream size including the optional EBR initialization data stream.

Table 3.1. Required SPI Flash Device Size for Dual Boot

Device Name	Bitstream Size (Uncompressed)	Minimum SPI Flash Density for Dual Boot	Units
LatticeECP2™ Family			
ECP2-6	1.48	4	Mbits
ECP2-12	2.84	8	Mbits
ECP2-20	4.41	16	Mbits
ECP2-35	6.25	16	Mbits
ECP2-50	8.90	32	Mbits
ECP2-70	13.27	32	Mbits
LatticeECP2M Family			
ECP2M-20	5.91	16	Mbits
ECP2M-35	9.81	32	Mbits
ECP2M-50	15.78	64	Mbits
ECP2M-70	19.79	64	Mbits
ECP2M-100	25.60	64	Mbits
LatticeECP3 Family			
ECP3-17	4.41	16	Mbits
ECP3-35	8.10	32	Mbits
ECP3-70	22.46	64	Mbits
ECP3-95	22.46	64	Mbits
ECP3-150	35.58	128	Mbits
ECP5 Family			
LFE5/UM-25	5.42	16	Mbits
LFE5/UM-45	9.74	32	Mbits
LFE5/UM-85	18.36	64	Mbits
ECP5-5G Family			
LFE5UM5G-25	5.42	16	Mbits
LFE5UM5G-45	9.74	32	Mbits
LFE5UM5G-85	18.36	64	Mbits

4. LatticeECP2/M and LatticeECP3 Dual Boot Mode

The LatticeECP2/M and LatticeECP3 families support the SPIm configuration mode. This feature is also known as Dual Boot mode.

This document provides easy steps for implementing the Dual Boot feature using Lattice Diamond® design software, Diamond Programmer, and Deployment Tool. The implementation of the dual boot feature using older tools such as ispLEVER, ispVM, and ispUFW are described in an appendix of this document.

Information about the differences of the Dual Boot feature in the LatticeECP2/M and LatticeECP3 devices is provided for those readers interested in the technical detail.

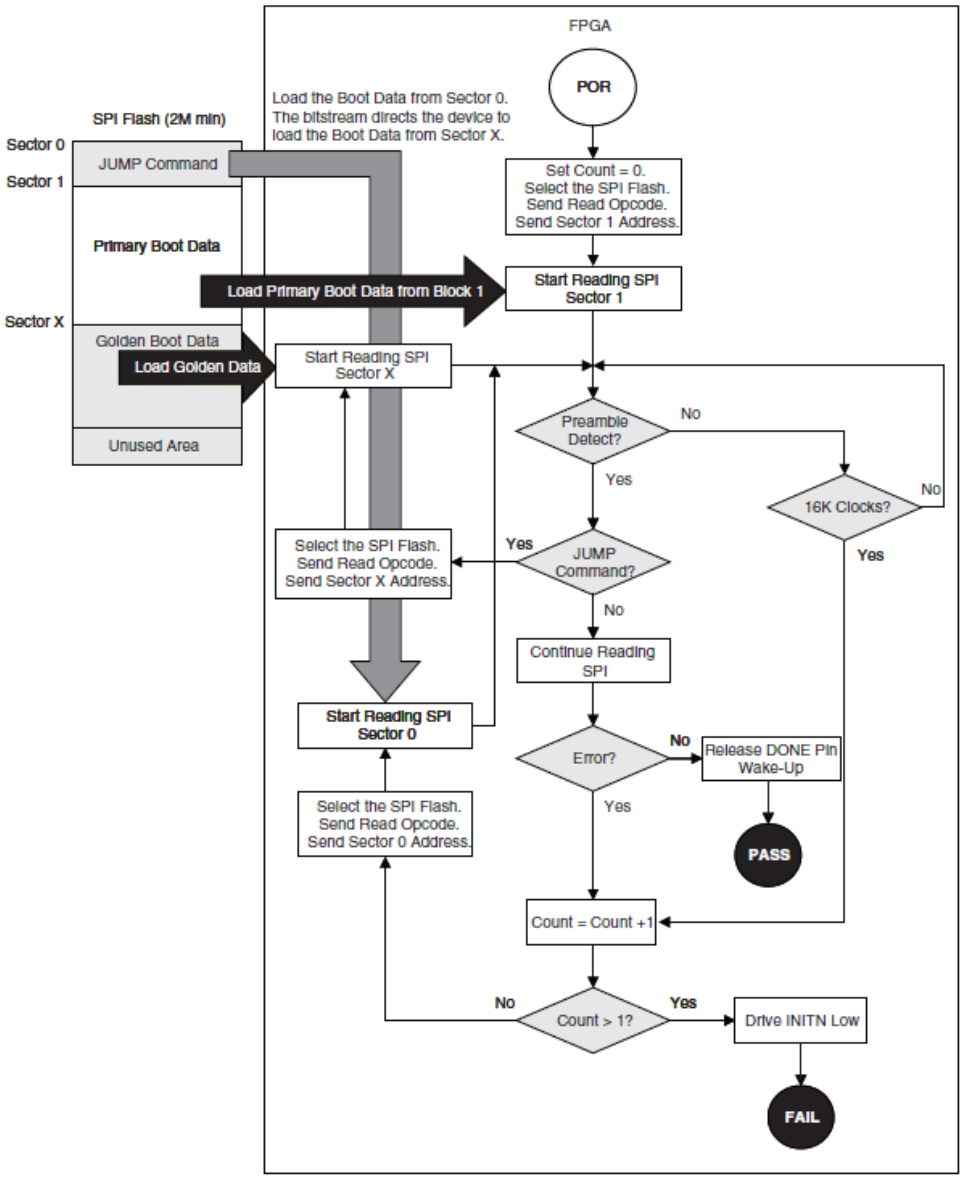


Figure 4.1. LatticeECP2/M Dual Boot Block Diagram

4.1. Description of the LatticeECP2/M Dual Boot Flow Diagram

Note: This flow is triggered either by a power cycle, the PROGRAMN pin being toggled, or the Refresh instruction being issued. The configuration pin setting must be CFG[0:2] = [010].

1. When Dual Boot mode is selected, in addition to the standard CRC checking, a time-out check is performed while reading the Primary pattern, the Golden pattern, and the Jump command.
   - Time Out Check
     As for all configuration protocols, the LatticeECP2/M or LatticeECP3 device searches for the preamble code 0xBDB3 from the Primary pattern. If after 16K clocks the preamble code is not received, the devices issues a time-out error and load the Jump command, which directs it to the location of the Golden pattern. If after another 16K clocks the preamble code is not received, the device issues a time-out error, drive the INTIN pin low to indicate an error, and stop driving the clock. The purpose of this test is to detect blank SPI Flash devices. A blank SPI Flash can be due to a SPI Flash erase disruption during a field upgrade as well as a newly manufactured board.
   - Data Corruption Check
     After the detection of the preamble code within 16K clocks, the CRC engine is turned on to detect whether the bitstream is corrupted. This determines whether the SPI Flash device has a corrupted Primary or Golden pattern due to SPI Flash program disruption or data loss.
2. If the Primary pattern fails one of the two checks above, the LatticeECP2/M or LatticeECP3 device knows that the Primary pattern is not valid. It drives the INITN pin low briefly to indicate an error and resets the configuration engine. After clearing all the SRAM fuses, it drives the INITN pin high, and read the Jump command which directs it to the location of the Golden pattern in the SPI Flash.
3. If the Jump command is corrupted, the device can read from Sector 1 where the Primary pattern is located. If the corrupted Primary pattern triggered Dual Boot initially, then loading the corrupted Primary pattern causes configuration to fail and the clock stops. A corrupted Jump command also causes a configuration failure. It is important to note that a corrupted Golden pattern is not the only possible cause for Dual Boot configuration failure.
4. If the failure is due to time-out checking failure, the LatticeECP2/M or LatticeECP3 device stops the clock and drive the INITN to low to indicate a configuration failure.
5. If the Jump command is valid, the LatticeECP2/M or LatticeECP3 device stops the clock, drive the INITN pin low, reset the configuration engine, and perform a Clear All operation. The device then drives the INITN pin high after the completion of the Clear All action, restart the clock, and read the Golden pattern from the SPI Flash Sector X address contained in the Jump command.
6. When reading in the Golden pattern, the LatticeECP2/M or LatticeECP3 device performs the time-out checking and the CRC checking if the preamble is detected. If the Golden pattern is also corrupted, configuration fails, the clock stops, and the INITN pin is driven low.
7. There are three special cases:
   - The SPI Flash is blank. This is normally the case when the board is powered up for the first time, unless the SPI Flash device is programmed before it is mounted on the board.
     The LatticeECP2/M or LatticeECP3 device settles down in a time-out checking failure while trying to read in the Jump command as described in item number 3 above.
   - The SPI Flash device has a single boot configuration and the LatticeECP2/M or LatticeECP3 CFG pin setting is for Dual Boot (SPIm mode). This is the case if a bitstream is programmed in Sector 0 (starting address 0x000000) of the SPI Flash device.
     The LatticeECP2/M or LatticeECP3 device is expected to configure successfully. This is due to the fact that when reading from Sector 1, the device is missing the preamble code, and thus fails time out checking. There is also the possibility that the device recognizes the array data as the preamble, but this results in a CRC error or an invalid command failure. Either case causes the device to read eventually from Sector 0 where the valid Single Boot bitstream is located.
   - The SPI Flash has a Dual Boot configuration but the LatticeECP2/M or LatticeECP3 CFG pin setting is for Single Boot (SPI mode). This is the case when the CFG[0:2] switch from 010 (Dual Boot) to 000 (Single Boot). The SPI mode (also known as Single Boot) setting causes the FGPA device to boot from Sector 0 (address 0x000000).

- LatticeECP2/M Devices – The Jump command is found at Sector 0. The device still supports the Jump command in SPI mode. The result is that the device jumps to and configure from the Golden pattern.
- LatticeECP3 Devices – Sector 0 is unused, and contains all one (1) data in the entire Sector 0. Since time-out checking is not available in SPI mode, the device shifts through the entire Sector 0 while searching for the preamble code until it reaches Sector 1 where the Primary pattern is located. LatticeECP3 devices is configured successfully by the Primary pattern.

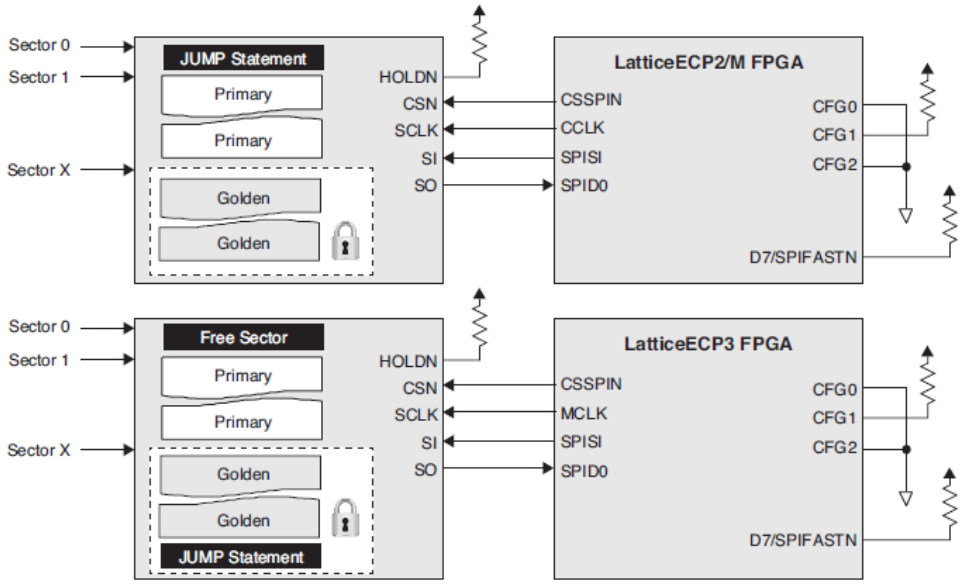


Figure 4.2. Location of the Primary and Golden Bitstreams

5. ECP5 and ECP5-5G Dual Boot Mode

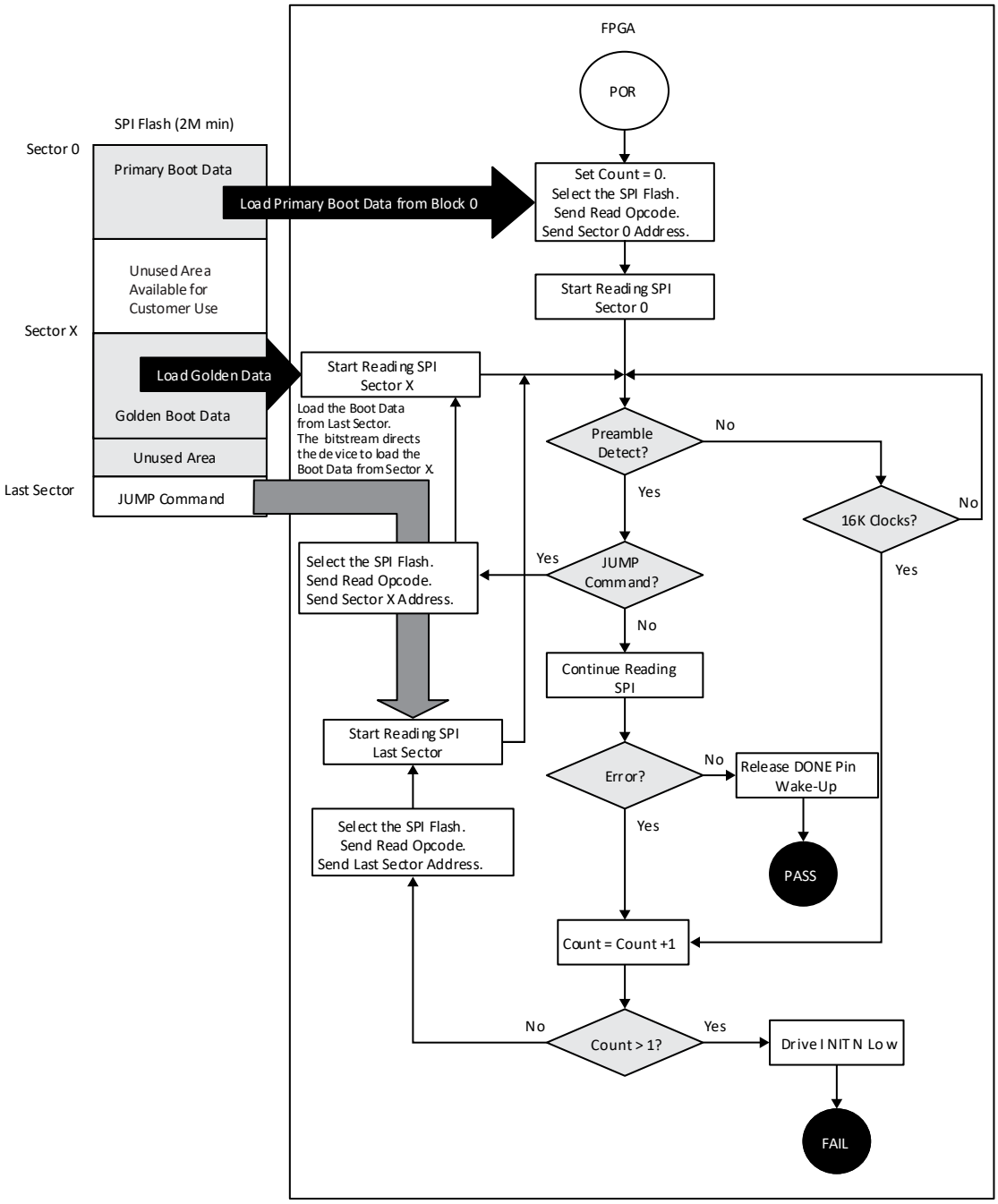


Figure 5.1. ECP5 and ECP5-5G Dual Boot Block Diagram

5.1. Description of the ECP5 and ECP5-5G Dual Boot Flow Diagram

Note: This flow is triggered either by a power cycle, the PROGRAMN pin being toggled, or the Refresh instruction being issued. The configuration pin setting must be CFG[0:2] = [010].

1. When Dual Boot mode is selected, in addition to the standard CRC checking, a time-out check is performed while reading the Primary pattern, the Golden pattern, and the Jump command.
   - Time Out Check
     As for all configuration protocols, the ECP5 and ECP5-5G devices search for the preamble code 0xBDB3 from the Primary pattern. If after 16K clocks the preamble code is not received, the devices issues a time-out error and load the Jump command, which directs it to the location of the Golden pattern. If after another 16K clocks the preamble code is not received, the devices issues a time-out error, drive the INTIN pin low to indicate an error, and stop driving the clock. The purpose of this test is to detect blank SPI Flash devices. A blank SPI Flash can be due to a SPI Flash erase disruption during a field upgrade as well as a newly manufactured board.
   - Data Corruption Check
     After the detection of the preamble code within 16K clocks, the CRC engine is turned on to detect whether the bitstream is corrupted. This determines whether the SPI Flash device has a corrupted Primary or Golden pattern due to SPI Flash program disruption or data loss.
2. If the Primary pattern fails one of the two checks above, the ECP5 and ECP5-5G devices know that the Primary pattern is not valid. It drives the INITN pin low briefly to indicate an error and resets the configuration engine. After clearing all the SRAM fuses, they drives the INITN pin high, and read the Jump command which directs it to the location of the Golden pattern in the SPI Flash.
3. If the Jump command is corrupted, It also causes a configuration failure. It is important to note that a corrupted Golden pattern is not the only possible cause for Dual Boot configuration failure.
4. If the Jump command is valid, the ECP5 and ECP5-5G devices stop the clock, drive the INITN pin low, reset the configuration engine, and perform a Clear All operation. The devices then drives the INITN pin high after the completion of the Clear All action, restart the clock, and read the Golden pattern from the SPI Flash Sector X address contained in the Jump command.
5. When reading in the Golden pattern, the ECP5 and ECP5-5G devices performs the time-out checking and the CRC checking if the preamble is detected. If the Golden pattern is also corrupted, configuration fails, the clock stops, and the INITN pin is driven low.
6. There is a special case wherein the SPI Flash is blank.

   This is normally the case when the board is powered up for the first time, unless the SPI Flash device is programmed before it is mounted on the board.

   The ECP5 and ECP5-5G devices settle down in a time-out checking failure while trying to read in the Jump command as described in item number 3 above.

6. Critical Points

- When the bitstream is to be programmed into a SPI Flash device, the Strategy Bitstream option No Header must be set to False.
  Reason: Some SPI Flash devices do not guarantee the data integrity of the first few bytes during a Fast Read. To address the issue, the LatticeECP2/M, LatticeECP3, ECP5 and ECP5-5G devices are designed to ignore the first 128 bits of data from the SPI Flash devices. Therefore, the first 128 bits (or 16 bytes) in the bitstream file must be dummy (0xFF). The presence of the header (~300 bytes) serves as the dummy.
- LatticeECP2/M devices do not support Dual Boot if the Primary pattern is encrypted.
  Reason: If the Primary pattern is corrupted, the LatticeECP2/M device fails to read the Golden pattern located at Sector 0. Instead, it reads continuously from the corrupted Primary pattern located at Sector 1.
  Solution: Add circuitry to drive CFG1 pin low (0) to set the device into SPI mode to force the device to boot from Golden pattern located at Sector 0.
- The Dual Boot feature supports SPI Flash devices with sector size of 64K (65535) bytes. The sector size of some high density SPI Flash devices is 256K bytes.
  Reason: When updating the Primary pattern, the original Primary pattern must be erased first. The Erase by Sector command is used to keep the Jump command and the Golden Pattern intact. The erase command for 256K byte sector size SPI Flash devices not only erases Sector 1, but also Sectors 0-3. As a consequence, the Jump command located at Sector 0 is erased.
  Solution: When updating the Primary pattern, restore the Jump command at Sector 0 as well. LatticeECP3, ECP5 and ECP5-5G devices do not have this issue since the Jump command is located at the last page.
- The Dual Boot feature requires a SPI Flash device with at least four sectors.
  Reason: The Jump command, Primary pattern, and Golden pattern each occupies at least one sector, or three sectors total. The smallest SPI Flash device that meets this requirement is a 2M bit (four-sector) SPI Flash device.
- Check the maximum read frequency supported by the SPI Flash device. Do not set the MCLK_FREQ frequency in Diamond Spreadsheet View beyond the maximum specified in the data sheet of the SPI Flash device. Do not set the MCLK_FREQ frequency over 10Mhz either due to Lattice device speed limitation on frame by frame CRC check feature.
  Reason: All SPI Flash devices support Read (0x03) and Fast Read (0x0B) commands. For LatticeECP2/M and LatticeECP3 devices, the SPIFASTN pin is set to high or low to select the Read or Fast Read, respectively. For the ECP5 and ECP5-5G devices, there is no SPIFASTN pin. Read(0x03) or Fast Read (0x0B) command is integrated in the bitstream.

Note: The LatticeECP2/M family only supports the Dual Boot feature using the Read (0x03) command. The SPIFASTN pin must be pulled high.

When the device is powered up, the clock frequency runs at the silicon default frequency, which is approximately 3.1 MHz. Once the Primary pattern is read, the clock frequency changes to the value contained in the bitstream. This is the MCLK_FREQ frequency value selected in Diamond Spreadsheet View. This is the default frequency used to read the Jump command and Golden pattern until a new value is set by a different bitstream. For LatticeECP2/M devices, this is also the default frequency any time the PROGRAMN pin is toggled. For LatticeECP3, ECP5 and ECP5-5G devices, any time the PROGRAMN pin is toggled or power is cycled, the clock frequency resets to the silicon default frequency (3.1 MHz for LatticeECP3 and 2.4 MHz for ECP5 and ECP5-5G devices).

7. Creating a Dual Boot or Multiple Boot PROM Hex File

The Dual Boot feature on LatticeECP2/M, LatticeECP3, ECP5, and ECP5-5G devices is simple and flexible. It is simple because only one SPI Flash device is required. It is flexible because the intelligent use of the Jump command. The Diamond software provides a turn-key solution to implement the feature.

- Merge the Golden pattern, Primary pattern, and the Jump command into one PROM hex file using the Lattice Deployment Tool. The Dual Boot PROM hex file can later be programmed into the SPI Flash device using Diamond Programmer or a third-party programmer.
- Reprogram the Primary pattern in the SPI Flash devices through the JTAG port of the FPGA devices using Diamond Programmer.
- Generate JTAG embedded programming files to program the SPI Flash devices through the JTAG port of the FPGA devices. The Multiple Boot feature on ECP5 and ECP5-5G devices is very similar to the Dual Boot feature. The only difference is the interface to generate the Boot PROM Hex File.

7.1. Using the Lattice Deployment Tool to Create a Dual Boot PROM Hex File

The steps below provide the procedure for generating a Dual Boot PROM hex file using the Deployment Tool.

1. Generate the Golden and Primary bitstream files in Diamond.
2. Invoke Lattice Deployment Tool from the **Start > Programs > Lattice Diamond > Accessories** menu or from **Start > Lattice Diamond Programmer** menu if Programmer and the Deployment Tool were installed standalone.
3. In the Diamond Deployment Tool window, select **External Memory** as the Function Type and select **Dual Boot** as the Output File Type. Select **OK**.

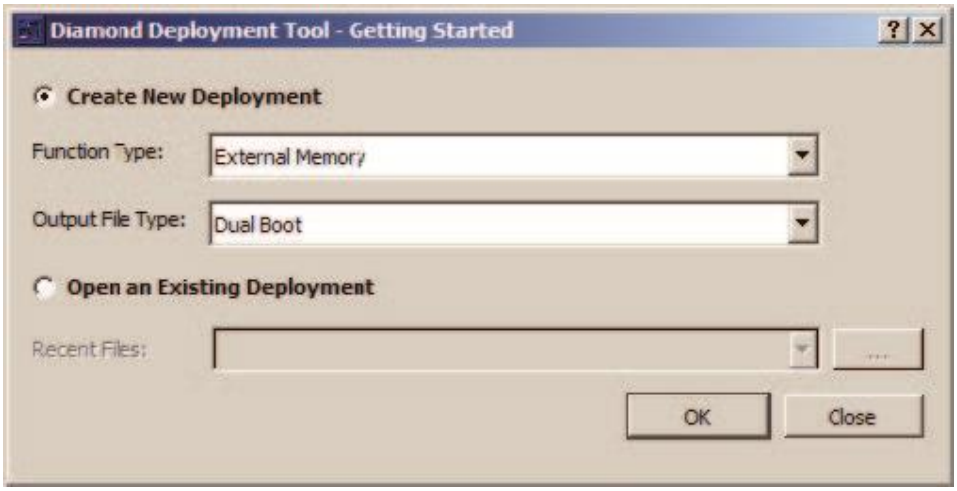


Figure 7.1. Creating New Deployment for Dual Boot

4. In the Step 1 window, click the File Name fields to browse and select the two bitstream files to be used to create the PROM hex file. One file is the Primary bitstream and the other file is the Golden bitstream. Select **Next**.

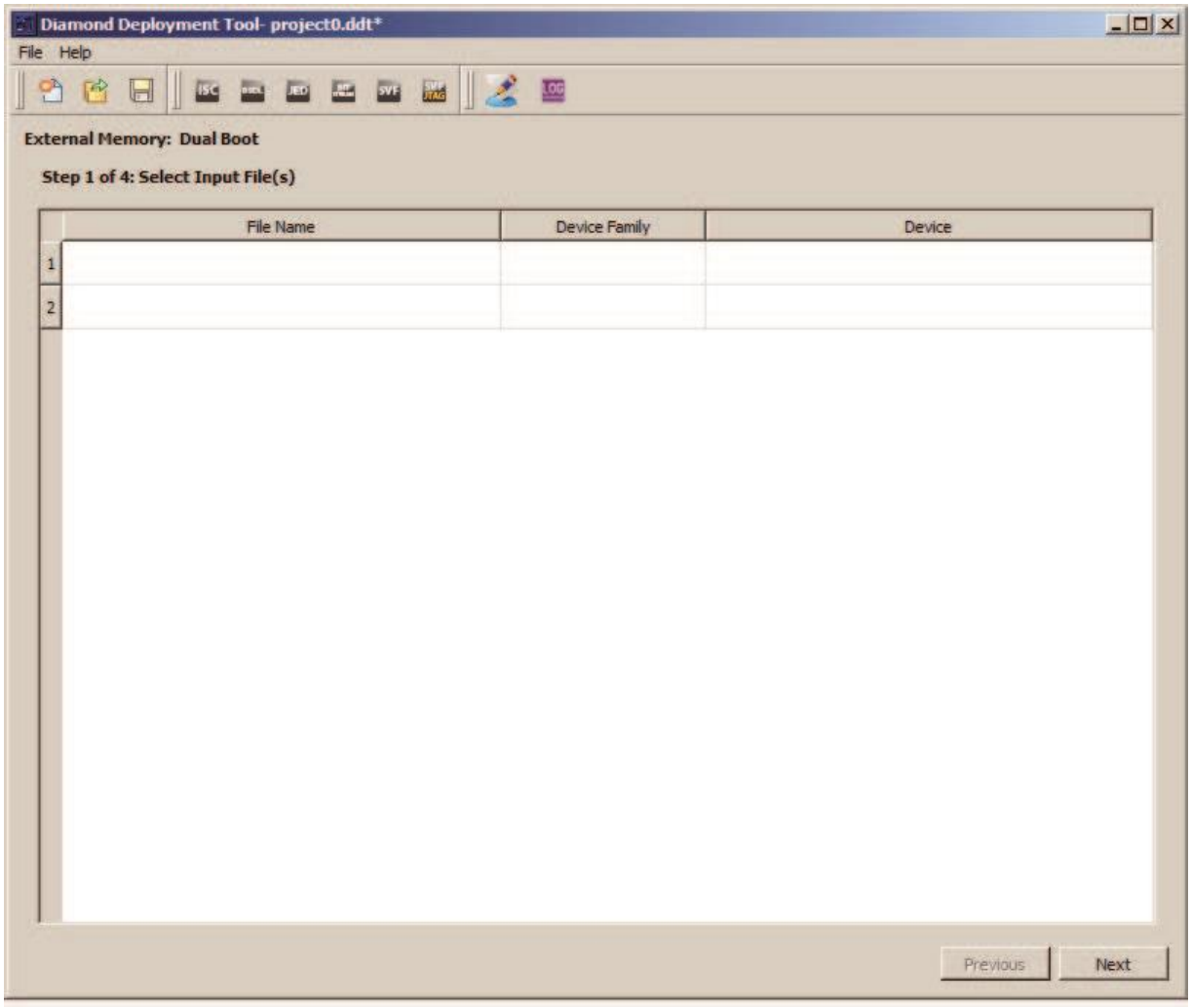


Figure 7.2. Selecting Input Files for Dual Boot

5. In the Step 2 window, select the Output Format. Choices are Intel Hex, Motorola Hex, and Extended Tektronix Hex.
6. In the Step 2 window, select the SPI Flash Size. Choices are, in Mb, 4, 8, 16, 32, 64, or 128.
7. In the Step 2 window, select the bitstream files to be used for the Golden and Primary Pattern.
8. In the Step 2 window, if needed, select the options to Protect Gold Sector (typically selected for Dual Boot), Byte Wide Bit Mirror, Retain Bitstream Header or Optimize Memory Space.

   Protect Golden Sector – By default, the golden sector is located immediately after the primary sector, saving SPI Flash space (less wasted space). When selected, locates the golden pattern at the first sector in the upper half of the SPI Flash. This allows you to protect the golden pattern from accidental erase/re-programming by protecting the upper half of the SPI Flash when it is programmed for the first time.

   Byte Wide Bit Mirror – Flips each byte in Intel, Extended Tektronix, and Motorola hexadecimal data files.

   Retain Bitstream Header – By default, Deployment Tool uses the Deployment Tool name, version number, and date the file was generated as the header. This option retains the header of the original input file as the header.

   Optimize Memory Space – By default, the Deployment Tool uses the worst case file size for SPI Flash memory space allocation. The worst case size is an uncompressed bitstream with maximum EBR and PCS. This allows maximum flexibility for field upgrades. If a new Primary pattern file size grows significantly due to less compression or adding EBR blocks, it is guaranteed to fit in the sectors already allocated for the Primary pattern. When this option is selected, the Deployment Tool uses the actual file size for the address allocation. This reduces the amount of wasted SPI Flash space. It may also allow you to use a small SPI Flash if you are not using EBR memory.

However, if the new Primary pattern has less compression ratio or more EBR/PCS, the new Primary could extend into the Golden bitstream memory space. If that happens, the entire SPI Flash, including Golden pattern, has to be erased and re-programmed with a new Hex file.

Select **Next**.

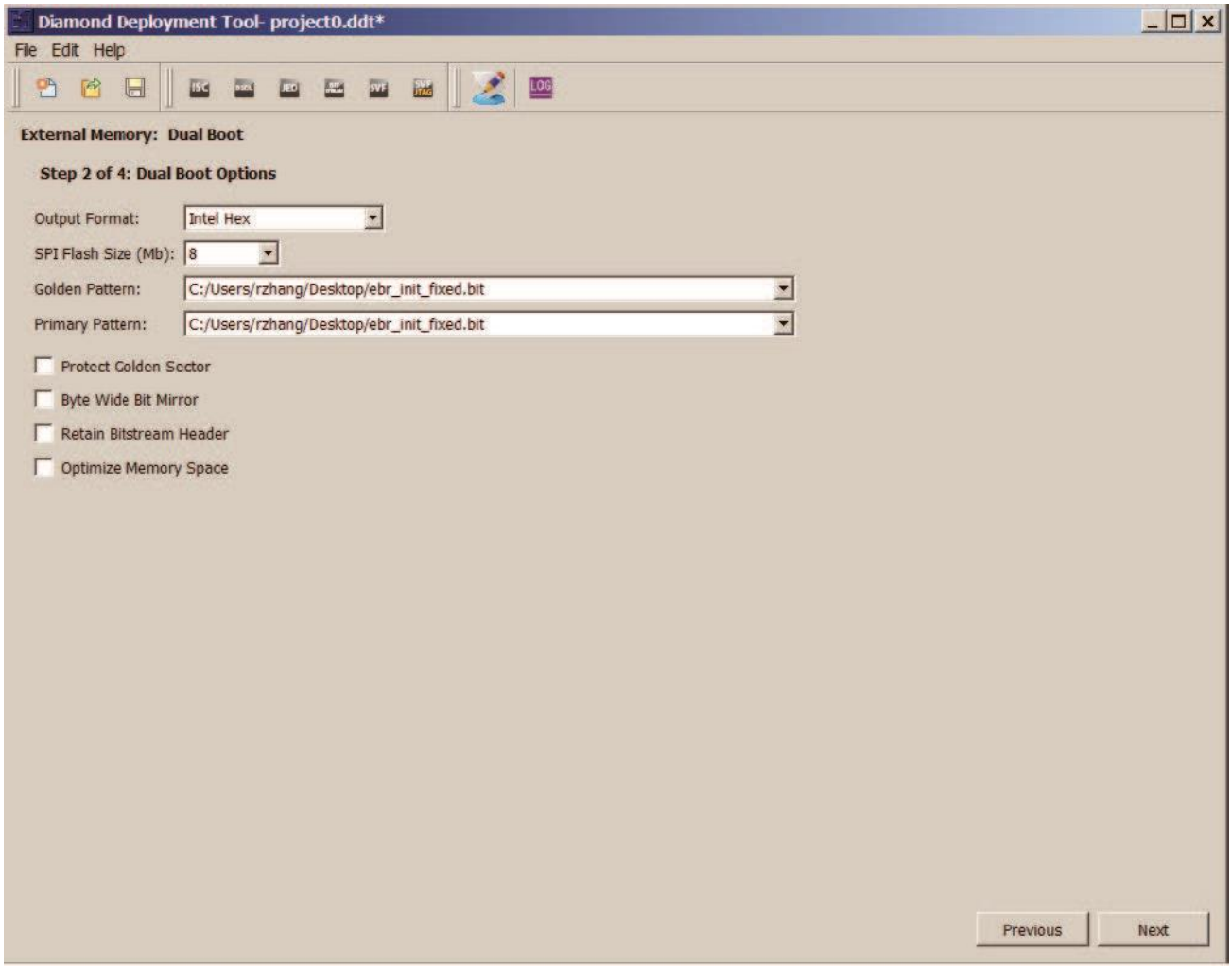


Figure 7.3. Selecting Dual Boot Options

9. In the Step 3 window, specify the Output File name for the PROM hex file. Select **Next**.
10. In the Step 4 window, review the information and select the **Generate** button. The Deployment Generation Status pane should indicate the PROM file was generated successfully.
11. Select **File > Save** and select **File > Exit**. When the deployment is saved, it can be re-opened later and all settings are saved.

7.2. Using the Lattice Deployment Tool to Create a Multiple Boot PROM Hex File (for ECP5 or ECP5-5G)

The following steps provide the procedure for generating a Multiple Boot PROM hex file using the Deployment Tool.

1. Generate the Golden and Primary bitstream files in Diamond.
2. Invoke Lattice Deployment Tool from the **Start > Programs > Lattice Diamond > Accessories** menu or from **Start > Lattice Diamond Programmer** menu if Programmer and the Deployment Tool were installed standalone.
3. In the Diamond Deployment Tool window, select **External Memory** as the Function Type and select **Advanced SPI Flash** as the Output File Type. Select **OK**.

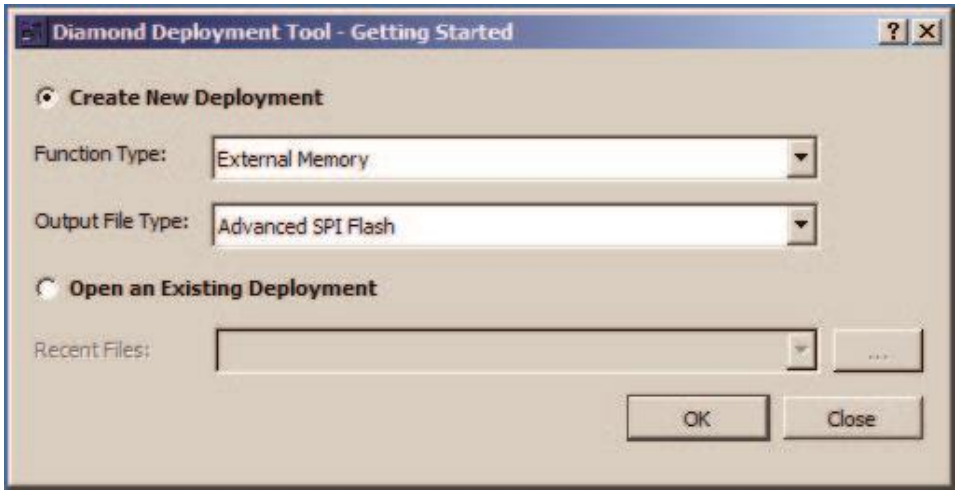


Figure 7.4. Creating New Deployment for Multiple Boot

4. In the Step 1 window, point and click the left mouse button to select the Primary bitstream (this step is mainly for the Tool to read the device information). Select **Next**.
5. In the Step 2 window Options Tab, select the Output Format. Choices are Intel Hex, Motorola Hex, and Extended Tektronix Hex.
6. In the Step 2 window Options Tab, select the SPI Flash Size. Choices are, in Mb, 4, 8, 16, 32, 64, or 128.

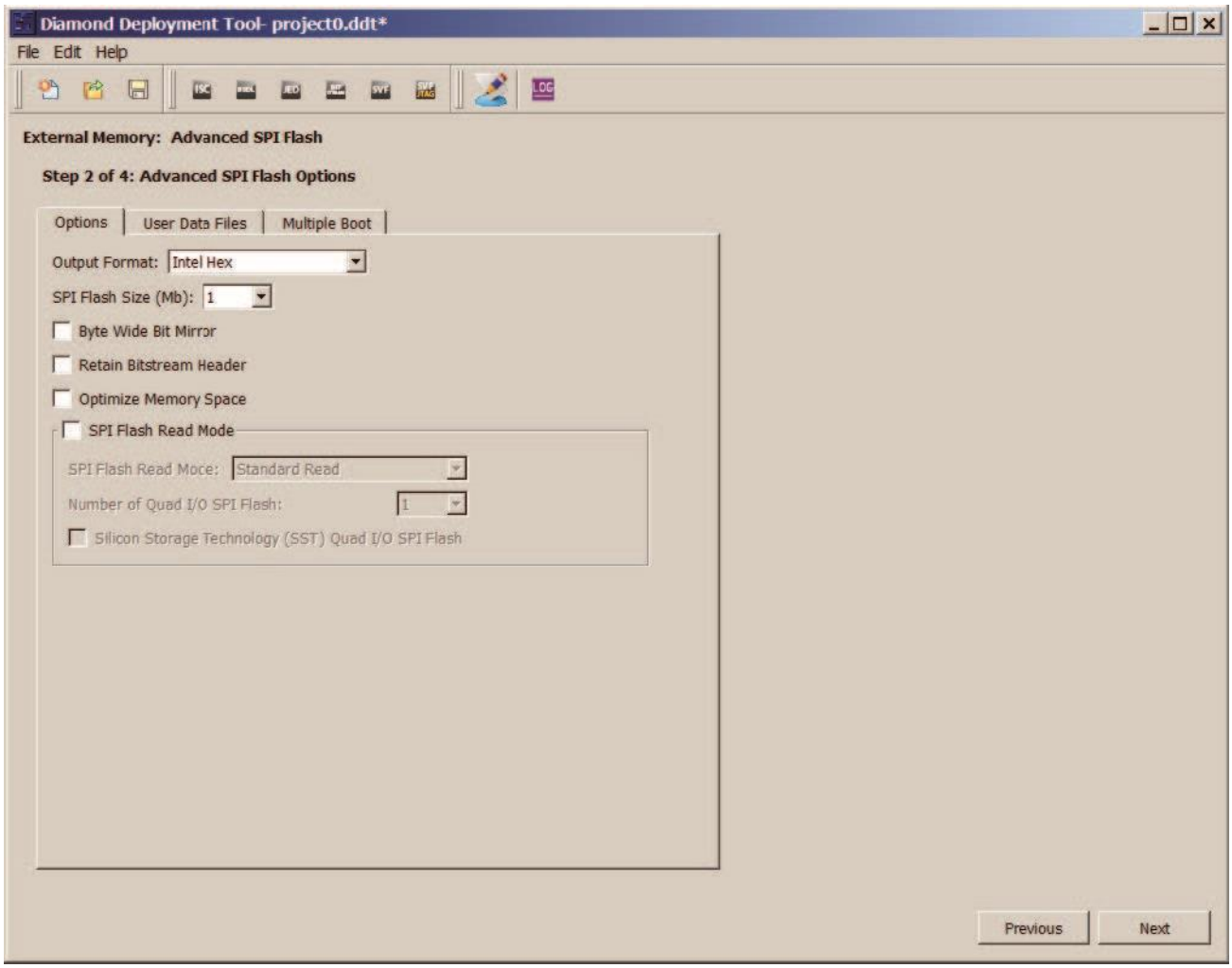


Figure 7.5. Selecting Advanced SPI Flash Options

7. In the Step 2 window Multiple Boot Tab, select Multiple Boot. select the bitstream files to be used for the Golden, Primary and Alternate Patterns.

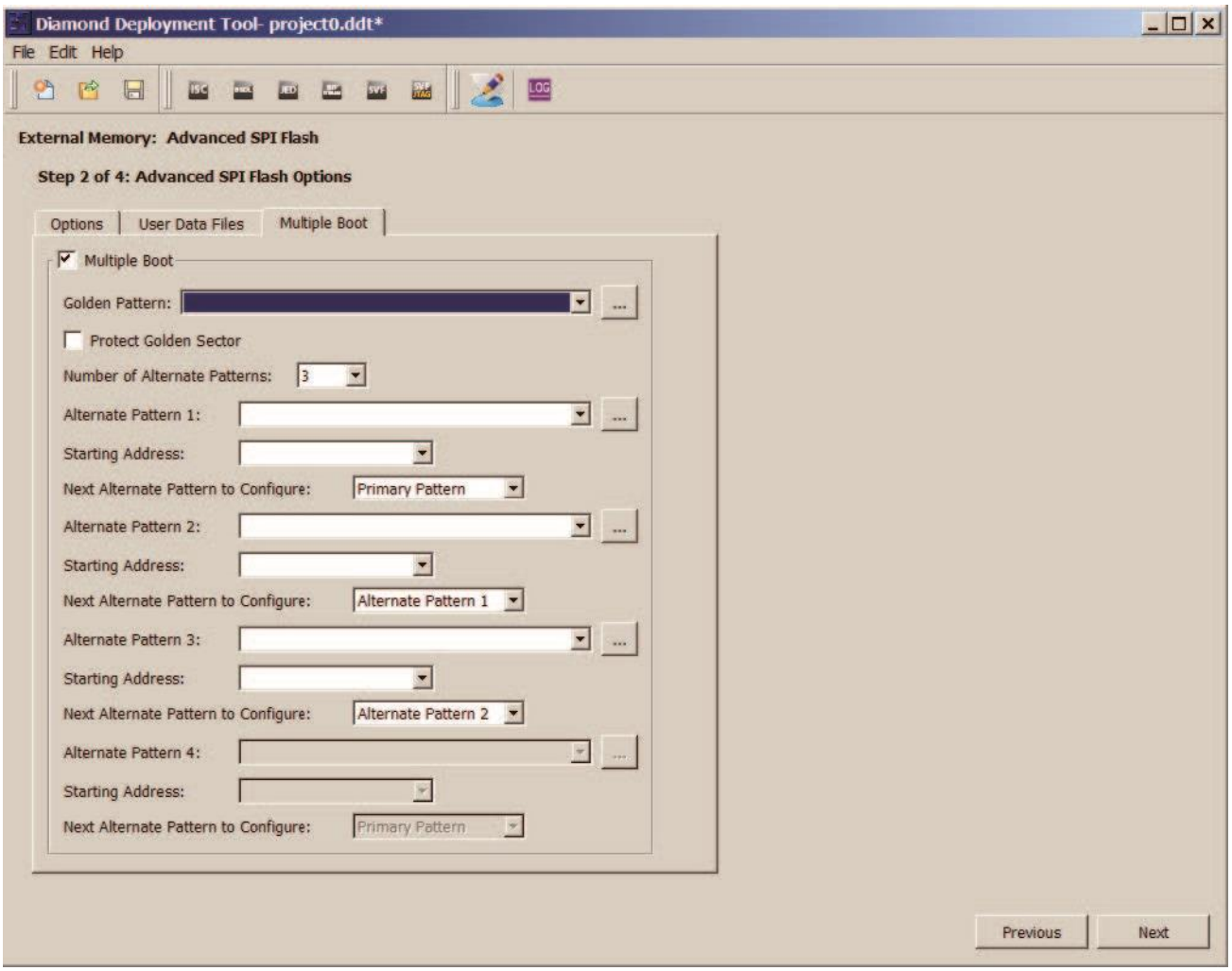


Figure 7.6. Selecting Input Files for Multiple Boot

8. In the Step 2 window, if needed, select the options to Protect Gold Sector (typically selected for Dual Boot), Byte Wide Bit Mirror, Retain Bitstream Header, Optimize Memory Space or SPI Flash Read Mode.
   - Protect Golden Sector – By default, the golden sector is located immediately after the primary sector, saving SPI Flash space (less wasted space). When selected, locates the golden pattern at the first sector in the upper half of the SPI Flash. This allows you to protect the golden pattern from accidental erase/reprogramming by protecting the upper half of the SPI Flash when it is programmed for the first time.
   - Byte Wide Bit Mirror – Flips each byte in Intel, Extended Tektronix, and Motorola hexadecimal data files.
   - Retain Bitstream Header – By default, Deployment Tool uses the Deployment Tool name, version number, and date the file was generated as the header. This option retains the header of the original input file as the header.
   - Optimize Memory Space – By default, the Deployment Tool uses the worst case file size for SPI Flash memory space allocation. The worst case size is an uncompressed bitstream with maximum EBR and PCS. This allows maximum flexibility for field upgrades. If a new Primary pattern file size grows significantly due to less compression or adding EBR blocks, it is guaranteed to fit in the sectors already allocated for the Primary pattern. When this option is selected, the Deployment Tool uses the actual file size for the address allocation. This reduces the amount of wasted SPI Flash space. It may also allow you to use a small SPI Flash if you are not using EBR memory. However, if the new Primary pattern has less compression ratio or more EBR/PCS, the new Primary could extend into the Golden bitstream memory space. If that happens, the entire SPI Flash, including Golden pattern, has to be erased and re-programmed with a new Hex file.

- SPI Flash Read Mode – Standard Read and Fast Read are providing 1 bit of data per clock. Dual I/O SPI Flash Read provides two bits of data per clock. Quad I/O SPI Flash Read provides four bits of data per clock. Please see ECP5 sysCONFIG Usage Guide (FPGA-TN-02039) for more details.

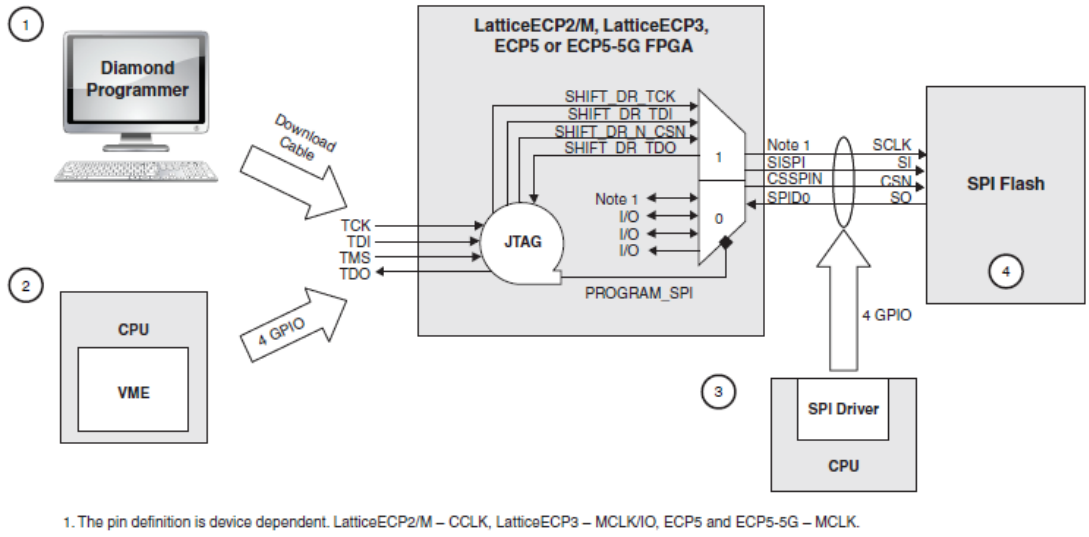
Select **Next**.

9. In the Step 3 window, specify the Output File name for the PROM hex file. Select **Next**.
10. In the Step 4 window, review the information and select the Generate button. The Deployment Generation Status pane should indicate the PROM file was generated successfully.
11. Select **File > Save** and select **File > Exit**. When the deployment is saved, it can be re-opened later and all settings.

8. Programming the Dual Boot or Multiple Boot Pattern into the SPI Flash Device

There are three methods for programming SPI Flash devices.

1. Run Diamond Programmer on a PC with a download cable connected to a board.
2. Port VME or SSPI into the on-board CPU. This method is used primarily if a field upgrade is required.
3. Pre-program the SPI Flash devices on a third-party programmer.



1. The pin definition is device dependent. LatticeECP2/M – CCLK, LatticeECP3 – MCLK/IO, ECP5 and ECP5-5G – MCLK.

Figure 8.1. Methods for Programming the SPI Flash Device

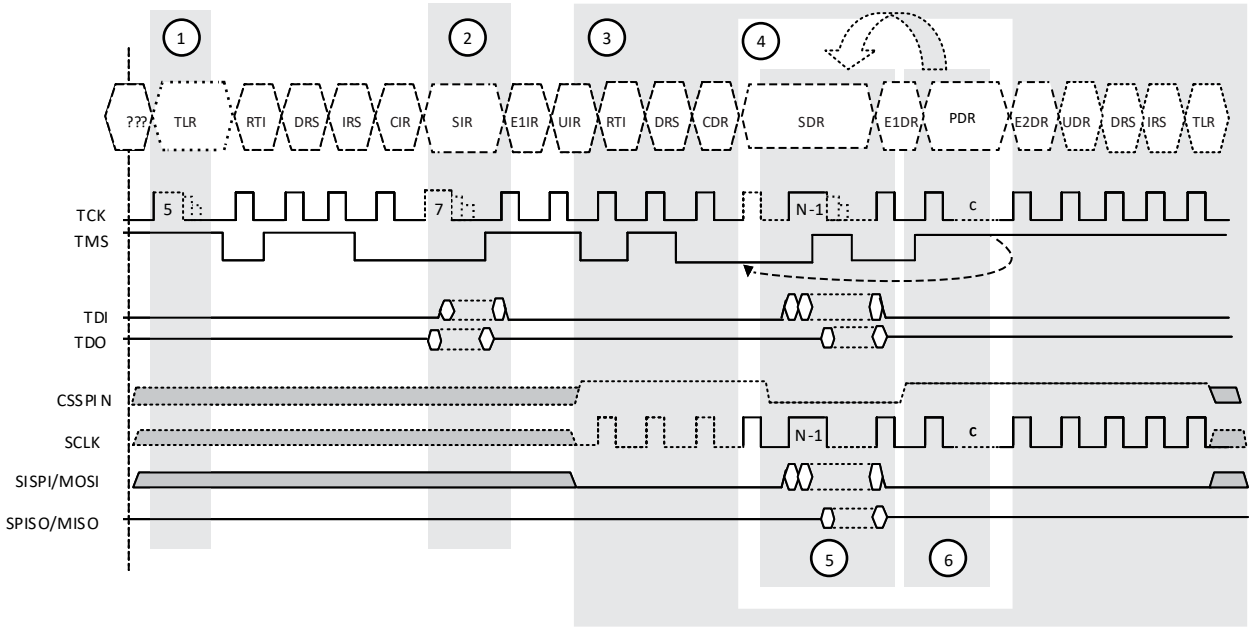
Procedure for Programming the Dual Boot Patterns into the SPI Flash Device using Diamond Programmer:

1. Connect power to the board and connect download cable from the board to the PC
2. Invoke Diamond Programmer from the Diamond window, **Tools > Programmer** or select the **Programmer** icon in the Diamond toolbar. Programmer can also be selected outside of Diamond from the **Start > Lattice Diamond > Accessories** menu or from the **Start > Lattice Diamond Programmer** menu.
3. Select **Create a New Project from a Scan**. You could also create a new blank project or open an existing Programmer project.
4. Select **Detect Cable** to scan the PC to determine what cable is connected or you can manually select the type of Cable and Port. Cable options are USB, USB2, or LATTICE. The USB or USB2, the Port is automatically set. If LATTICE is selected as the Cable, you also need to specify the Port Address. Select **OK**.
5. Select the **Operation** field using the left mouse button.
6. For Access Mode, select **SPI Flash Background Programming**.
7. For Operation, select **SPI Flash Erase, Program, Verify**.
8. For Programming File, browse to select the boot PROM file (.mcs, .exo, or .xtek).
9. For SPI Flash Options, specify the Family, Vendor, Device, and Package. Select **OK**.
10. In the Programmer window, select **Design > Program** or select the **Program** icon in the toolbar.

9. SPI Flash Programming Implementation with Diamond Programmer

The LatticeECP2/M, LatticeECP3 and ECP5 devices are designed to support a JTAG instruction, PROGRAM_SPI, that when executed, effectively connects the four SPI interface pins of SPI Flash devices to the four JTAG port pins. Diamond Programmer automatically takes care of the details to program the SPI Flash devices via the JTAG port. A detailed waveform diagram is shown in Figure 9.1.

Note: For LatticeECP2/M devices, the PERSISTENT must be set to ON in the Diamond Spreadsheet View in order to program the SPI Flash device when the LatticeECP2/M is in user mode. For LatticeECP3, ECP5 and ECP5-5G devices, it is not required, however it is recommended to ensure the pins are allocated for this purpose.



Note: The Persistent fuse must be ON for this waveform to work.

Figure 9.1. JTAG SPI Flash Programming Waveforms

Table 9.1. JTAG SPI Flash Programming Descriptions

Block Number	Title	Description
1	Reset JTAG Port	The standard method to set the JTAG state machine to a known state.
2	Send Instruction	Shift in the PROGRAM_SPI instruction (OPCODE = 0x0X→). The arrow indicates the shift- ing direction (bit 0 first).
3	Connect	The 4-pin JTAG port is connected to the 4-pin SPI interface. SCLK following TCK indicates a connection is made.
4	Repeat	Loop around to erase by sectors and programming by pages.
5	Shift Data	Send in the command to erase a sector or shift in one page of programming data. The FPGA responds by driving the CSSPIN pin to low to gates on SCLK, SPID0, and SISPI.
6	Burn Time Delay	Drive the CSSPIN to high to command the SPI Flash device to start the erase or program- ming action. Wait for the required erase or programming delay time then poll the complete status. Consult the SPI Flash data sheet for the polling method required.

10. Implementing Incremental Design Changes Using Diamond Programmer

The procedures described so far are for static design applications: creating the Dual Boot PROM hex file and programming the SPI Flash with the Dual Boot PROM hex file. This section describes how to update the Golden or Primary bitstream in the SPI Flash device.

The Intel, Motorola, and Extended Tektronix PROM record file format does not allow file manipulations (edit, cut, or insert). Instead, incremental design changes for memory devices are carried out directly on the memory devices. The procedure is as follows:

1. Start with a pre-programmed SPI Flash device with the current Dual Boot PROM hex file (created in Deployment Tool).
2. Program the new Golden or Primary Boot bitstream into the SPI Flash device.

Note: The Jump command does not change, thus there is no need to re-program.

3. Optional: Read and save the newly programmed SPI Flash into a PROM hex file.

Procedure for Updating the Primary or Golden Pattern in a SPI Flash Device using Diamond Programmer:

1. Launch Programmer within Diamond, **Tools > Programmer**, or from the **Start > Lattice Diamond > Accessories** menu or from the **Start > Lattice Diamond Programmer** menu if you have a standalone version of Programmer installed. Select **Diamond Programmer** and select **Create a new Project from a Scan**, then specify the scan details, or select **Open an existing Programmer Project**, then browse to select the .xcf file and select **OK**.
2. The Diamond Programmer GUI appears indicating Device Family, Device, Operation, File Name, etc.
3. Select **SPI Flash Background Programming** as the Access Mode.
4. Select **SPI Flash Erase, Program, Verify** as the Operation.
5. Specify the bitstream file to be re-programmed.
6. Specify the SPI Flash Options.
7. If the bitstream is the Primary pattern, then select **0x010000 (sector 1)** as the Start Address for LatticeECP2/M and LatticeECP3 devices or select **0x000000 (sector 0)** as the Start Address for ECP5 and ECP5-5G devices. This is also known as the offset address. If the bitstream is the Golden pattern, then find sector X address in Table A.3.
8. Select **OK** to close the Device Properties window.
9. Programmer generates a chain description file as a .xcf file and re-programs the SPI Flash. Programmer only erases and reload the program from the starting address to the ending address. The other sectors remain untouched.

Appendix A. Detailed Device-Specific Information

Table A.1. Configuration Mode Selection for LatticeECP2/M and LatticeECP3 Families

Configuration Mode	LatticeECP2/M	LatticeECP3	ECP5 and ECP5-5G
CFG[2:0]			
SPI (Single Boot)	000	000	010
SPIm/SPI (Dual Boot)	010	010	010
SPI (Multi Boot)	-	-	010

Table A.2. Configuration Mode Selection for LatticeECP2/M and LatticeECP3 Families

Device Name	Bitstream Size[1]	SPI Flash Density for Dual Boot	Sectors Required	Unused (Free) Sectors
	Mbits	Mbits	(65536 Bytes/Sector)	(65536 Bytes/Sector)
LatticeECP2 Family				
ECP2-6	1.48	4	7	1
ECP2-12	2.84	8	13	3
ECP2-20	4.41	16	19	13
ECP2-35	6.25	16	27	5
ECP2-50	8.90	32	37	27
ECP2-70	13.27	32	55	9
LatticeECP2M Family				
ECP2M-20	5.91	16	25	7
ECP2M-35	9.81	32	41	23
ECP2M-50	15.78	64	65	63
ECP2M-70	19.79	64	81	47
ECP2M-100	25.60	64	105	17
LatticeECP3 Family[2]				
ECP3-17	4.41	16	19	13
ECP3-35	8.10	32	35	29
ECP3-70	22.46	64	91	37
ECP3-95	22.46	64	91	37
ECP3-150	35.58	128	145	111
ECP5 Family[2]				
LFE5/UM-25	5.42	16	23	9
LFE5/UM-45	9.74	32	41	23
LFE5/UM-85	18.36	64	75	53
ECP5-5G Family[2]				
LFE5UM5G-25	5.42	16	23	9
LFE5UM5G-45	9.74	32	41	23
LFE5UM5G-85	18.36	64	75	53

Notes:

1. The bitstream size shown is the maximum uncompressed bitstream size.
2. The bitstream size shown is the maximum with all EBR initialization data included.

A.1. Deployment Tool Memory Space Allocation

The Deployment Tool allocates memory as shown in Table A.1. even if the bitstream is compressed or the SPI Flash density selected is larger than the SPI Flash density shown. The advantage of using the maximum bitstream size approach is that once the device is known, the location of the Golden Pattern starting address (Sector X address) is deterministically known. Otherwise it could vary from design to design and would be impossible to keep track of when there are design changes or field updates for the design.

Table A.3. Deployment Tool Memory Space Allocation

Device Name	SPI Flash Density	Physical Location of Bitstreams (Sector Address Range)							
		Jump Command		Primary Pattern		Golden Pattern Sector X Address		Unused (Free)	
		Start	End	Start	End	Start[1]	End	Start	End
LatticeECP2 Family									
ECP2-6	4M	0x000000	0x0000FF	0x010000	0x03FFFF	0x040000	0x06FFFF	0x070000	0x07FFFF
ECP2-12	8M			0x010000	0x06FFFF	0x070000	0x0CFFFF	0x0D0000	0x0FFFFF
ECP2-20	16M			0x010000	0x09FFFF	0x0A0000	0x12FFFF	0x130000	0x1FFFFF
ECP2-35	16M			0x010000	0x0DFFFF	0x0E0000	0x1AFFFF	0x1B0000	0x1FFFFF
ECP2-50	32M			0x010000	0x12FFFF	0x130000	0x24FFFF	0x250000	0x3FFFFF
ECP2-70	32M			0x010000	0x1BFFFF	0x1C0000	0x36FFFF	0x370000	0x3FFFFF
LatticeECP2M Family									
ECP2M-20	16M	0x000000	0x0000FF	0x010000	0x0CFFFF	0x0D0000	0x18FFFF	0x190000	0x1FFFFF
ECP2M-35	32M			0x010000	0x14FFFF	0x150000	0x28FFFF	0x290000	0x3FFFFF
ECP2M-50	64M			0x010000	0x20FFFF	0x210000	0x40FFFF	0x410000	0x7FFFFF
ECP2M-70	64M			0x010000	0x28FFFF	0x290000	0x50FFFF	0x510000	0x7FFFFF
ECP2M-100	64M			0x010000	0x34FFFF	0x350000	0x68FFFF	0x690000	0x7FFFFF
LatticeECP3 Family									
ECP3-17	16M	0x1FFF00	0x1FFFFF	0x010000	0x09FFFF	0x0A0000	0x12FFFF	0x000000	0x00FFFF
								0x130000	0x1EFFFF
ECP3-35	32M	0x3FFF00	0x3FFFFF	0x010000	0x11FFFF	0x120000	0x22FFFF	0x000000	0x00FFFF
								0x230000	0x3EFFFF
ECP3-70	64M	0x7FFF00	0x7FFFFF	0x010000	0x2DFFFF	0x2E0000	0x5AFFFF	0x000000	0x00FFFF
								0x5B0000	0x7EFFFF
ECP3-95	64M	0x7FFF00	0x7FFFFF	0x010000	0x2DFFFF	0x2E0000	0x5AFFFF	0x000000	0x00FFFF
								0x5B0000	0x7EFFFF
ECP3-150	128M	0xFFFF00	0xFFFFFF	0x010000	0x48FFFF	0x490000	0x90FFFF	0x000000	0x00FFFF
								0x910000	0xFEFFFF
ECP5 Family									
LFE5-25	16M	0x1FFF00	0x1FFFFF	0x000000	0x0AFFFF	0x0B0000	0x15FFFF	0x160000	0x1FFEFF
LFE5-45	32M	0x3FFF00	0x3FFFFF	0x000000	0x13FFFF	0x140000	0x27FFFF	0x280000	0x3FFEFF
LFE5-85	64M	0x7FFF00	0x7FFFFF	0x000000	0x24FFFF	0x250000	0x49FFFF	0x4A0000	0x7FFEFF
ECP5-5G Family									
LFE5UM5G-25	16M	0x1FFF00	0x1FFFFF	0x000000	0x0AFFFF	0x0B0000	0x15FFFF	0x160000	0x1FFEFF
LFE5UM5G-45	32M	0x3FFF00	0x3FFFFF	0x000000	0x13FFFF	0x140000	0x27FFFF	0x280000	0x3FFEFF
LFE5UM5G-85	64M	0x7FFF00	0x7FFFFF	0x000000	0x24FFFF	0x250000	0x49FFFF	0x4A0000	0x7FFEFF

Note: The Golden pattern starting address, also known as the Sector X address, is shaded.

Table A.4. Jump Command Syntax

Frame	Contents (D0..D7)	Description
Header	(LSB)1111…1111	16 Dummy Bytes
	1011110110110011	16-bit Preamble (0xBDB3)
Frame (Control Register 0)	11000100	Bits 0 to 7 of Write Control Register 0 command
	0000…0000	24-bit Command Information (24 zeros)
	0000…0000	Control Register 0 data (32 zeros)
Frame (Jump Command)	11111110	Bits 0..7 of Jump Command
	0000…0000	24-bit Command Information (24 zeros)
	(SPI Flash Read Opcode)	8-bit SPI Flash Read opcode. 0x03 (regular read) 0x0B (fast read) 0xBB (fast read dual I/O) 0xEB (fast read quad I/O)
	(Sector Address)	24-bit Sector Address of the SPI Flash to which the LatticeECP2/M or LatticeECP3, ECP5 and ECP5-5G jumps. For example, for SPI Flash Sector X address 0x050000 = b000001010000000000000000 Note: The Sector X address for all the devices can be found on the Start Address of the Golden Pattern column in Table A.3.
Frame (End)	1111…1111	16 Dummy Bytes

A.2. Memory Space Allocation Proposal When Using the Sector Protect Feature

The Golden pattern is the pattern that never changes. The Jump command is also considered part of the Golden pattern. The sector protection feature on SPI Flash devices can protect the Golden pattern and the Jump command. There is always the potential that the Golden pattern could be corrupted due to an accidental erase or programming activity, for example, while updating the Primary pattern or reading and writing the free area of the SPI Flash devices. The Jump command for LatticeECP2/M devices is located at Sector 0. Only Winbond's W25X family of SPI Flash devices supports the sector-by-sector protection including Sector 0. All the other SPI Flash devices support protection by quadrants. This means the Primary pattern and the Jump command is in the same quadrant. If the protection feature is turned on, both Jump command and the Primary pattern is locked and disables the ability to field upgrade the Primary pattern. The LatticeECP3, ECP5 and ECP5-5G devices changed the location of Jump command to the last page to meet the quadrant boundary requirement.

Also, due to the quadrant boundary restriction of the protect feature for most of the SPI Flash devices, the starting address of the Golden pattern must be located at the bottom (second) half, of the SPI Flash devices selected.

Note: The SPI Flash density shown is the minimum SPI Flash density required to support the Dual Boot feature. If the actual SPI Flash device density is larger, the starting address of the Golden Pattern and the last page address of the Jump command should be adjusted accordingly.

Requirements:

- LatticeECP2/M devices must use the Winbond W25X family of SPI Flash devices
- LatticeECP3, ECP5, and ECP5-5G devices can use any SPI Flash device

Table A.5. Memory Space Allocation Proposal When Using the Sector Protect Feature

Device Name	SPI Flash Density	Physical Location of Bitstreams (Sector Address Range)							
		Jump Command		Primary Pattern		Golden Pattern Sector X Address		Unused (Free)	
		Start	End	Start	End	Start[1]	End	Start	End
LatticeECP2 Family									
ECP2-6	4M	0x000000	0x0000FF	0x010000	0x3FFFFF	0x040000	0x06FFFF	0x070000	0x07FFFF
ECP2-12	8M			0x010000	0x06FFFF	0x080000	0x0DFFFF	0x0E0000	0x0FFFFF
ECP2-20	16M			0x010000	0x09FFFF	0x100000	0x18FFFF	0x190000	0x1FFFFF
ECP2-35	16M			0x010000	0x0CFFFF	0x100000	0x1BFFFF	0x1C0000	0x1FFFFF
ECP2-50	32M			0x010000	0x12FFFF	0x200000	0x31FFFF	0x320000	0x3FFFFF
ECP2-70	32M			0x010000	0x1BFFFF	0x200000	0x3AFFFF	0x3B0000	0x3FFFFF
LatticeECP2M Family									
ECP2M-20	16M	0x000000	0x0000FF	0x010000	0x0CFFFF	0x100000	0x1BFFFF	0x1C0000	0x1FFFFF
ECP2M-35	32M			0x010000	0x14FFFF	0x200000	0x33FFFF	0x340000	0x3FFFFF
ECP2M-50	64M			0x010000	0x20FFFF	0x400000	0x5FFFFF	0x600000	0x7FFFFF
ECP2M-70	64M			0x010000	0x28FFFF	0x400000	0x67FFFF	0x680000	0x7FFFFF
ECP2M-100	64M			0x010000	0x34FFFF	0x400000	0x73FFFF	0x740000	0x7FFFFF
LatticeECP3 Family									
ECP3-17	16M	0x1FFF00	0x1FFFFF	0x010000	0x09FFFF	0x100000	0x18FFFF	0x000000	0x00FFFF
								0x190000	0x1EFFFF
ECP3-35	32M	0x3FFF00	0x3FFFFF	0x010000	0x11FFFF	0x200000	0x30FFFF	0x000000	0x00FFFF
								0x310000	0x3EFFFF
ECP3-70	64M	0x7FFF00	0x7FFFFF	0x010000	0x2DFFFF	0x400000	0x6CFFFF	0x000000	0x00FFFF
								0x6D0000	0x7EFFFF
ECP3-95	64M	0x7FFF00	0x7FFFFF	0x010000	0x2DFFFF	0x400000	0x6CFFFF	0x000000	0x00FFFF
								0x6D0000	0x7EFFFF
ECP3-150	128M	0xFFFF00	0xFFFFFF	0x010000	0x48FFFF	0x800000	0xC7FFFF	0x000000	0x00FFFF
								0xC80000	0xFEFFFF
ECP5 Family									
LFE5-25	16M	0x1FFF00	0x1FFFFF	0x000000	0x0AFFFF	0x100000	0x1AFFFF	0x1B0000	0x1FFEFF
LFE5-45	32M	0x3FFF00	0x3FFFFF	0x000000	0x13FFFF	0x200000	0x33FFFF	0x340000	0x3FFEFF
LFE5-85	64M	0x7FFF00	0x7FFFFF	0x000000	0x24FFFF	0x400000	0x64FFFF	0x650000	0x7FFEFF
ECP5-5G Family									
LFE5UM5G-25	16M	0x1FFF00	0x1FFFFF	0x000000	0x0AFFFF	0x100000	0x1AFFFF	0x1B0000	0x1FFEFF
LFE5UM5G-45	32M	0x3FFF00	0x3FFFFF	0x000000	0x13FFFF	0x200000	0x33FFFF	0x340000	0x3FFEFF
LFE5UM5G-85	64M	0x7FFF00	0x7FFFFF	0x000000	0x24FFFF	0x400000	0x64FFFF	0x650000	0x7FFEFF

Note: The Golden pattern starting address, also known as the Sector X address, is shaded.

Appendix B. Dual Boot Feature using ispVM and ispUFW

B.1. Using the ispUFW to Create a Dual Boot PROM File

Table B.1. and Figure B.1 provide the procedure for generating a Dual Boot PROM file using the ispUFW. In ispUFW, right click the mouse to show the options, left click the mouse selects the option.

Table B.1. Procedure for Creating a Dual Boot PROM File using the ispUFW

Step	Description
1	Launch ispUFW from the ispVM GUI by clicking on the UFW icon.
2	Select PROM file as the output file format. Only a PROM file can store a memory record of multiple memory images.
3	Select the two bitstream files, one for the Golden and one for the Primary.
4	Browse to the first bitstream file. The device name is extracted from the header of the bitstream file.
5	Browse to the second bitstream file. The device name is extracted from the header of the bitstream file.
6	Select the PROM density large enough to store the Dual Boot patterns.
7	Select the Dual Boot feature by selecting Yes.
8	Select the Golden file from one of the two bitstream files selected in Steps 4 and 5. Two information dialogs pops up giving the address locations of the Golden file and Jump command.
9	Select the Primary file from one of the two bitstream files selected in Steps 4 and 5. An information dialog pops up giving the address location of the Primary file.
9.1	This step is optional. The byte-wide flipping is off by default for serial configuration modes such as SPI and SPIm.
10	Select the output data file name.
11	Generate the Dual Boot PROM file record.
Notes	
D1	The device name is extracted from the bitstream header. Otherwise, you must select the device name.
D2	When launching ispUFW, the SPI Flash PROM Size defaults to 2 Mbit. If the SPI Flash density is too small to store the Dual Boot patterns, the error message is issued here.
D3	Make sure the density SPI Flash density is large enough to store the dual boot patterns. The SPI Flash density for the LatticeECP2/M and LatticeECP3 devices can be found at the Addendum section ofthis document.
D4	The Jump command for the LatticeECP2/M family is at Sector 0. For LatticeECP3, the location is the last page of the SPI Flash selected at Step 6.
D5	The destination of the JUMP command for the LatticeECP2M-35E devices is sector 0x15 (21 in decimal). This is known as the Sector X address. ispUFW sets the Sector X address based on Table A.3. If the sector protection fea- ture is selected, then the destination is adjusted to the middle sector of the SPI Flash density selected.
D6	The Primary pattern is located in Sector 1 for all devices.
D7	The Jump command offers flexibility in locating the Golden pattern. However, you are recommended to follow the addresses shown in Table A.3. The benefit of knowing where they are located becomes obvious when re-programming the Primary pattern or the Golden pattern.
D8	For the LatticeECP2/M family, the first free sector after the Golden pattern to the end of the SPI Flash device is free memory space that can be used for other applications, such as scratch pad memory, etc. For LatticeECP3, from the first free sector after the Golden pattern to the end of the second last sector and the entire sector 0 are free memory space. The difference is due to the physical location of the Jump command.

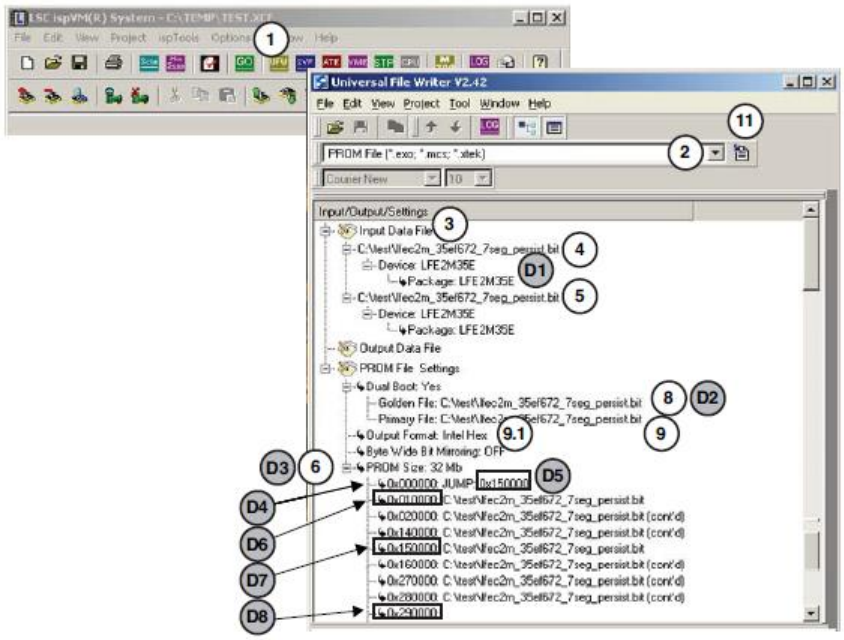


Figure B.1. Using the ispUFW to Create a Dual Boot PROM File

B.2. Programming the Dual Boot Pattern into the SPI Flash Device

There are four methods for programming SPI Flash devices.

1. Run ispVM System on a PC with the ispDOWNLOAD™ cable. This method is useful especially during the board design phase.
2. Port ispVME into the on board CPU. This method is used primarily if a field upgrade ise required.
3. Port the SPI driver into the on-board CPU. This method is possible but it is not recommended due to the complexity in resolving the bus contention issues on the SCLK pin and CSSPIN pin.
4. Pre-program the SPI Flash devices on a third-party programmer.

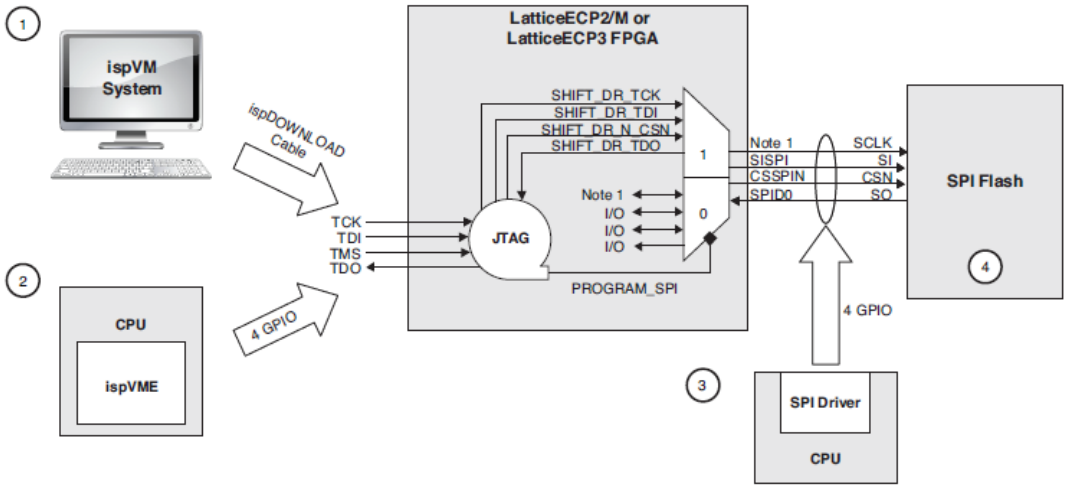


Figure B.2. Methods for Programming the SPI Flash Device

Table B.2. Procedure for Programming the Dual Boot Patterns into the SPI Flash Device using ispVM

Step	Description
1	Launch the device selection menu.
2	Select the FPGA device.
3	Select Dual Boot SPI Flash Programming under Device Access Options.
4	Launch the SPI Serial Flash Device dialog by clicking on the SPI Flash Options button.
5	Select the SPI Flash device. Make sure the density is large enough to store both Dual Boot bitstreams as well as the Jump command.
	If you have not already generated a Dual Boot PROM file, jump to Step 7.
	If the Dual Boot PROM file was generated using the ispUFW as described in Table , make sure if is the same size as selected previously in Step 6 of Table . If you wish to use an existing Dual Boot PROM file, continue to Step 6.
6	Select the existing Dual Boot PROM file, and jump to Step 11.
	Generate a Dual Boot PROM file.
7	Select the Golden Boot file. The bitstream must have the persistent bit set to on.
8	Select the Primary Boot file. The bitstream must have the persistent bit set to on.
8.1	This step is optional. The default PROM file name is derived from the Golden Boot file name and default file type is .mcs. You can change the name or location of the resulting Dual Boot PROM file.
9	Generate the Dual Boot PROM file from the bitstream patterns and the Jump command. See Note D2.
10	Load the PROM file generated at Step 9 above to the PC memory buffer by clicking the Load From File button.
11	Click OK to close the SPI Serial Flash Device dialog. The PROM file memory size is checked against the SPI Flash density at this moment. An error flag is issued if the SPI Flash density is smaller.
12	Click OK to close the Device Information dialog. ispVM generates a chain description file, which can be saved as a .xcf file.
13.1	Select the GO button to program the Dual Boot PROM file into the SPI Flash device.
13.2	Select the VME menu to generate the embedded programming files.
Notes	
D1	The data file size shown is the absolute memory size of the PROM file.
D2	ispVM issues the error message if the SPI Flash density is smaller than required, or if the device type selected and the bitstream sized does not match.
D3	The PROM file is attached to the data file directory on the menu. The PROM file can be viewed by right-clicking the mouse to launch the special menu, and then scrolling to the bottom to select the viewer.

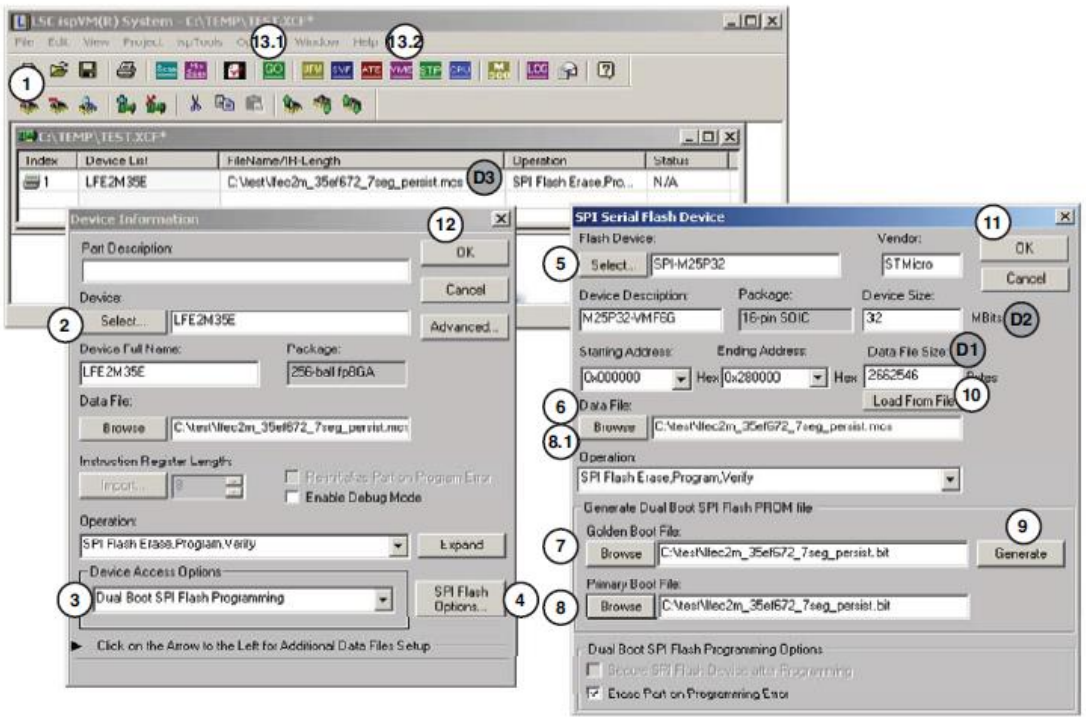


Figure B.3. Using the ispVM GUI to Program the Dual Boot Bitstream onto the SPI Flash Device

B.3. SPI Flash Programming Implementation on ispVM System

The LatticeECP2/M and LatticeECP3 devices are designed to support a JTAG instruction, PROGRAM_SPI, that when executed, effectively connects the four SPI interface pins of SPI Flash devices to the four JTAG port pins. The ispVM System software and ispVME automatically takes care of the details to program the SPI Flash devices via the JTAG port. A detailed waveform diagram is shown in Figure B.4.

Note: For LatticeECP2/M devices, the PERSISTENT must be set to ON in the ispLEVER Design Planner or Diamond Spreadsheet View in order to program the SPI Flash device when the LatticeECP2/M is in user mode. For LatticeECP3 it is not required, however it is recommended to ensure the pins are allocated for this purpose.

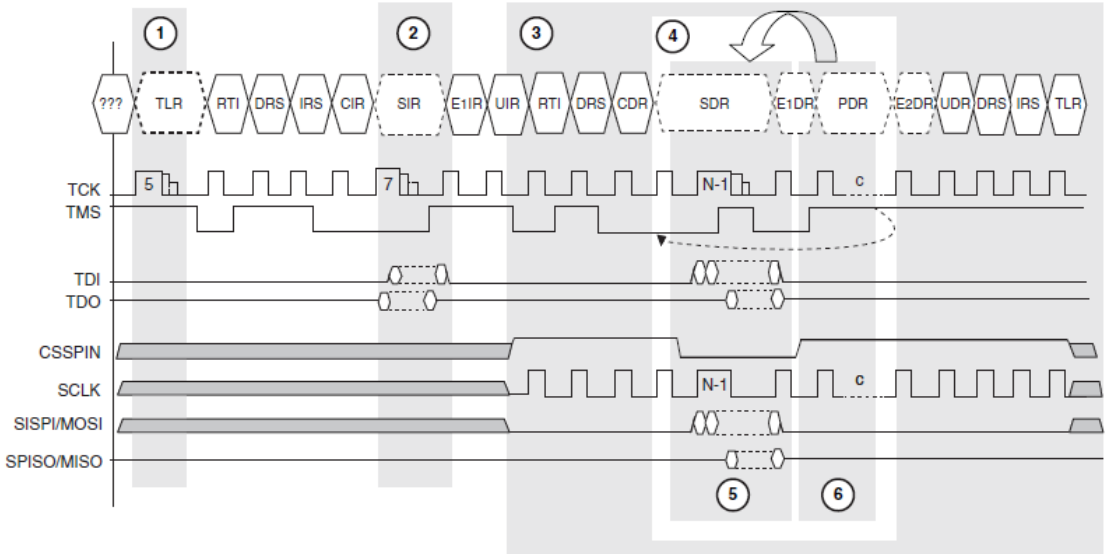


Figure B.4. JTAG SPI Flash Programming Waveforms

Table B.3. JTAG SPI Flash Programming Descriptions

Block Number	Title	Description
1	Reset JTAG Port	The standard method to set the JTAG state machine to a known state.
2	Send Instruction	Shift in the PROGRAM_SPI instruction (OPCODE = 0x0X→). The arrow indicates the shift- ing direction (bit 0 first).
3	Connect	The 4-pin JTAG port is connected to the 4-pin SPI interface. SCLK following TCK indicates a connection is made.
4	Repeat	Loop around to erase by sectors and programming by pages.
5	Shift Data	Send in the command to erase a sector or shift in one page of programming data. The FPGA responds by driving the CSSPIN pin to low to gates on SCLK, SPID0, and SISPI.
6	Burn Time Delay	Drive the CSSPIN to high to command the SPI Flash device to start the erase or program- ming action. Wait for the required erase or programming delay time then poll the complete status. Consult the SPI Flash data sheet for the polling method required.

B.3.1. Implementing Incremental Design Changes Using the ispVM System

The procedures described so far are for static design applications: creating the Dual Boot PROM file and programming the SPI Flash with the Dual Boot PROM file. This section describes how to update the Golden or Primary bitstream in the SPI Flash device.

The Intel and Motorola PROM record file format does not allow file manipulations (edit, cut, or insert). Instead, incremental design changes for memory devices are carried out directly on the memory devices. The procedure is as follows:

1. Start with a pre-programmed SPI Flash device with the current Dual Boot PROM file.
2. Program the new Golden or Primary Boot bitstream into the SPI Flash device.
   Note: The Jump command does not change, thus there is no need to re-program.
3. Optional: Read and save the newly programmed SPI Flash into a PROM file.

Table B.4. Procedure for Updating the Primary or Golden Pattern in a SPI Flash Device using ispVM

Steps	Description
1	Launch ispVM System and select the device.
2	Select SPI Flash Programming mode under Device Access Options.
3	Select the SPI Flash Erase, Program, Verify operation. When reprogramming a Flash device, an erase action is mandatory.
4	Launch the SPI Serial Flash Device dialog by clicking on the SPI Flash Options button.
5	Select the SPI Flash device that needs to be re-programmed.
6	Browse for the bitstream file to be re-programmed.
7.1	If the bitstream is the Primary pattern, then select 0x010000 (Sector 1) as the starting address. This also known as the offset address.
7.2	If the bitstream is the Golden pattern, then find the Sector X address in Table A.3.
8	Reload the bitstream file again to adjust the ending address to account for the offset address selected at Step 7.1 or 7.2 above.
9	Click OK to close the SPI Serial Flash Device dialog.
10	Click OK to close the Device Information dialog. ispVM generates a chain description file, which can be saved as a .xcf file.
11	Click the GO button to instruct ispVM to begin the re-programming action. ispVM only erases and re-program from the starting address to the ending address. The other sectors are left untouched. Only the Primary or Golden pattern is changed.
Notes	
D1	The starting address specifies where the sector-by-sector erase and programming action begins.
D2	The ending address specifies where the sector-by-sector erase and programming action ends. In conjunction with the starting address, the boundary of the erase and programming action is thus defined. The Jump com- mand is outside the boundary and thus is not changed.
D3	When loading the bitstream, the file size in bytes is shown. The number of sectors is required to hold the pattern is reflected on the starting address and ending address.

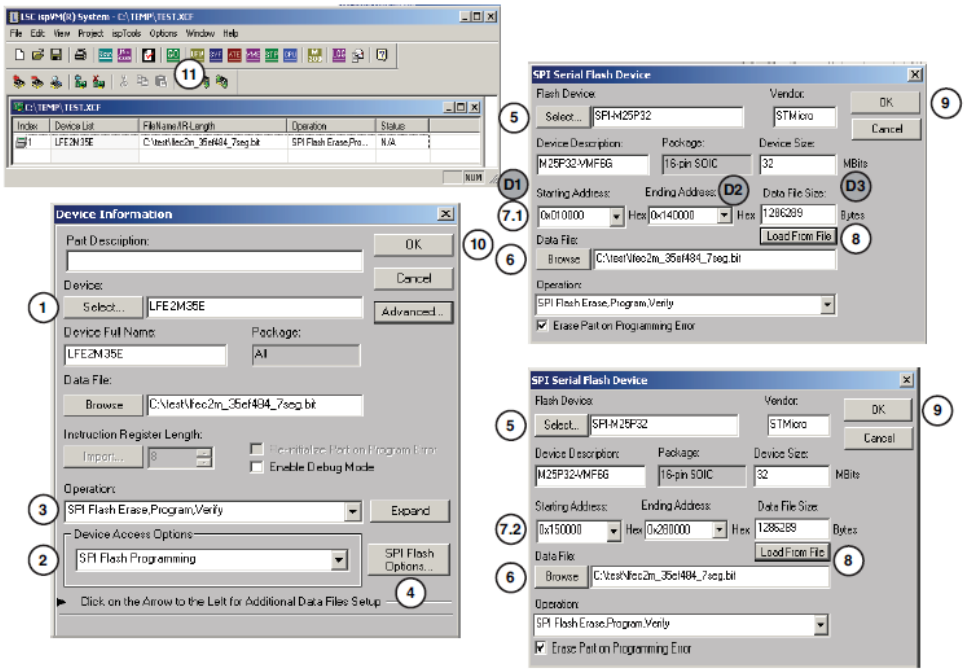


Figure B.5. Incremental SPI Flash Programming on ispVM System

Appendix C. Differences Between Hex and Binary Files

The Hex file format and binary file formats differ in their bit orientation. The binary file must be written from LSB to MSB (D0 to D7), while the Hex files must be written MSB to LSB (D7 to D0). Thus, 0xBD becomes 0xBD, and 0xB3 becomes 0xCD. The 16-bit preamble code 0xBDB3 in the binary file becomes 0xBDCD in the Hex file.

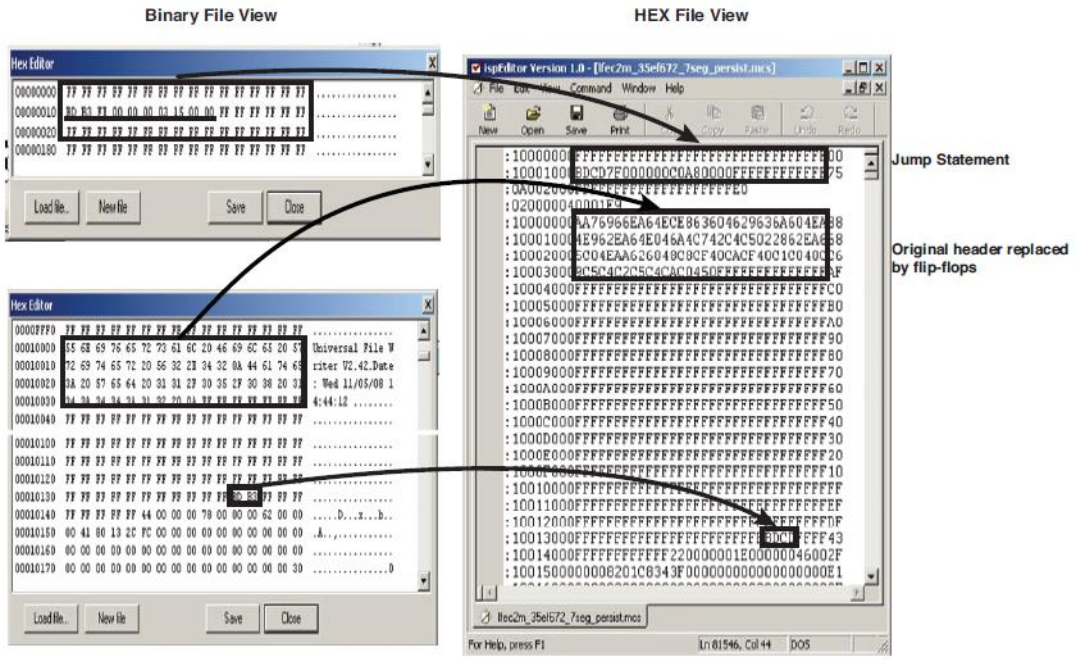


Figure C.1. Binary File and Hex File Comparison

Appendix D: Reference Material - Flash Configuration Primary

Background

Lattice was the first FPGA vendor to use industry standard SPI Flash devices as the boot PROM. Lattice was also the first vendor providing in-system SPI Flash programming support through the JTAG port of the host FPGA devices. The business and economic benefits to users is well documented. The technical merits of using SPI Flash devices as the boot PROM is a worthwhile topic to explore.

Generic Behavior of SPI Flash Devices

Nearly all SPI Flash devices support two read opcodes:

- Slow Read (0x03): The first byte of data from Slow Read is guaranteed to be valid.
- Fast Read (0x0B): The first byte of data from Fast Read is not guaranteed to be valid.

All SPI Flash devices support exactly the same random read command format:

Read Command: = <8-bit opcode><24-bit Sector X address>

Note: Data out available: All SPI Flash devices present the first bit of the first byte to the data out immediately after the 24-bit Sector X address. However, the validity of the first byte varies among vendors and the read opcode.

Lattice FPGA Devices' SPI Flash Configuration Protocol From a Single SPI Flash Device

Note: This process is triggered by power cycling or toggling the PROGRAMN pin. Lattice FPGA devices can read data out from up to four SPI Flash devices.

1. Set the clock frequency per the Control Register 0 setting. Silicon default is ~3.1 MHz.
2. Issue eight clocks to send out the Slow or Fast Read opcode, per the SPIFASTN pin setting for LatticeECP2/M and LatticeECP3. The ECP5 and ECP5-5G devices does not have the SPIFASTN pin. Slow or Fast Read opcode is integrate in the bitstream.
3. For LatticeECP2/M and LatticeECP3, issue 24 clocks to send the Sector X address 0x000000 or 0x010000, per the CFG1 pin setting. For the ECP5 and ECP5-5G devices, Sector X address is always 0x000000.
4. After clocking out the address, issue another 96 clocks.
5. Start reading the data on the SPID0/MISO pin.

Note: From the FPGA's perspective, it ignores the data on the SPID0/MISO pin for the first 128 clocks (8+24+96 from Steps 2, 3, and 4, above). From the SPI Flash device's perspective, the FPGA ignores the first 96 bits of data shifted out of the SPI Flash device. This is due to the fact that the SPI Flash device does not shift data out until after the 24 address clocks. This is shown in Figure D.1.

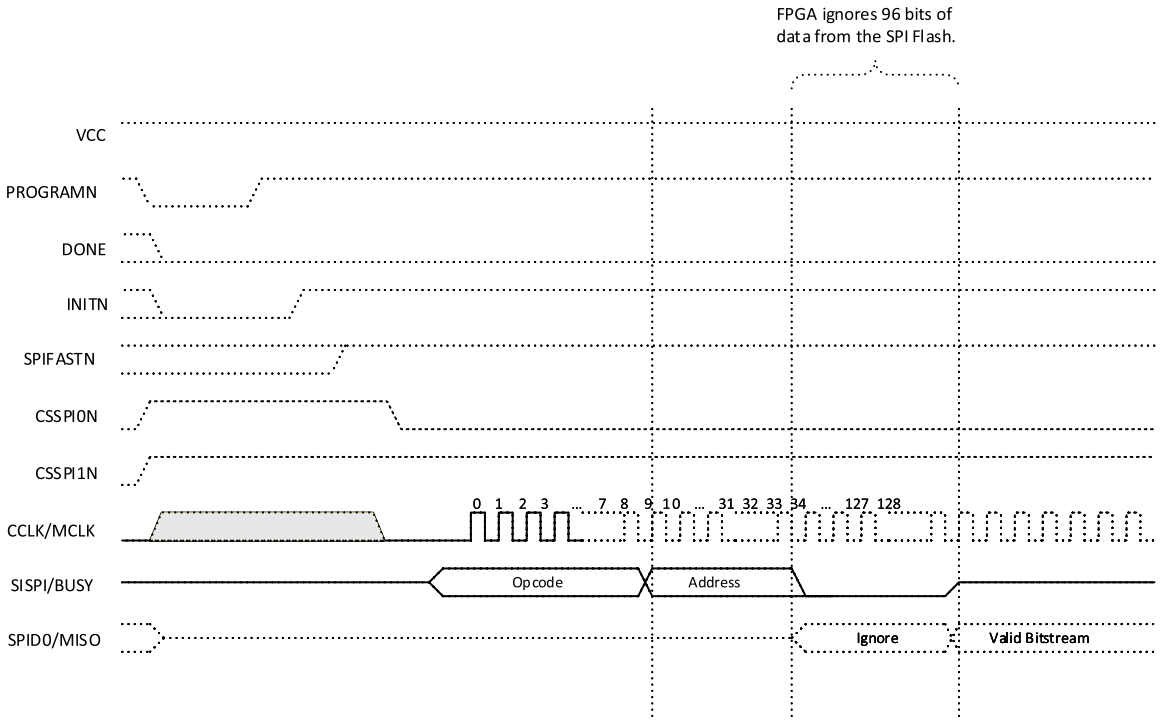


Figure D.1. SPI Mode Configuration Waveform Diagram

6. From this point on, the LatticeECP2/M, LatticeECP3, ECP5 or ECP5-5G device behaves exactly the same as industry standard FPGA devices. It searches the incoming bitstream for the preamble code, also known as the alignment code.
7. Once the preamble code is received, the LatticeECP2/M, LatticeECP3, ECP5 or ECP5-5G device immediately expects command information. For example, before the preamble code, the byte 0xFF is referred to as dummy data. It has no meaning. After the preamble code, 0xFF becomes a NOOP command.
8. One of the first commands in the bitstream is to write onto the Control Register 0, which sets the read clock to the frequency you selected when the bitstream was generated. If you do not specify a frequency preference, the software defaults to 2.5 MHz for LatticeECP2/M and LatticeECP3 devices and 2.4 MHz for ECP5 and ECP5-5G devices. This is done to make all LatticeECP2/M and LatticeECP3 devices appear to be the same.
   Note: It is important to note that when toggling the PROGRAMN pin to re-configure, LatticeECP3 devices reset the frequency setting to the silicon default of 3.1 MHz. LatticeECP2/M devices do not do this and keeps the current frequency setting despite clearing of the Control Register 0. This is because the frequency setting is controlled by latching in from Control Register 0. The only way to reset the frequency setting latches in LatticeECP2/M devices is by power cycling the device.
9. Control Register 0 also sets the following functions:
   a. Bitstream compression on or off
   b. Wake up sequence settings
   c. Auto Clear SRAM (Yes or No) when PROGRAMN pin toggle
   d. TransFR™ feature settings
   e. SPI Flash interface single or double
   f. Multiple Boot selections
10. The rest of the process involves reading the bitstream from the SPI Flash device to configure the FPGA. If the frame-by-frame CRC check is OK, the configuration is successful. If the command is to overflow the incoming data to the output, do so until the external DONE pin is high. Pulse 128 clocks, then end.

Technical Support Assistance

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

Revision History

Revision 1.7, September 2020

Section	Change Summary
All	• Changed document number from TN1216 to FPGA-TN-02203. • Updated document template.
Disclaimers	Added this section.
Acronyms in This Document	Added this section.
Glossary	Updated content to add ECP5 and ECP5-5G support for Multiple Boot function.
Creating a Dual Boot or Multiple Boot PROM Hex File	Updated content to add ECP5 and ECP5-5G support on Using the Lattice Deployment Tool to Create a Multiple Boot PROM Hex File (for ECP5 or ECP5-5G).

Revision 1.6, October 2015

Section	Change Summary
All	• Changed document title to LatticeECP3, LatticeECP2/M, ECP5 and ECP5-5G Dual Boot and Multiple Boot Feature. • Added support for ECP5-5G.
Technical Support Assistance	Updated Technical Support Assistance section.

Revision 1.5, March 2014

Section	Change Summary
All	• Added support for ECP5 device family. • Changed document title to LatticeECP3, LatticeECP2/M, and Lattice ECP5 Dual Boot and Multiple Boot Feature. • Changed instances of LatticeECP4UM to ECP5.

Revision 1.4, January 2014

Section	Change Summary
All	Changed document title to LatticeECP3, LatticeECP2/M, and Lattice ECP4UM Dual Boot and Multiple Boot Feature.
ECP5 and ECP5-5G Dual Boot Mode	Added LatticeECP4UM Dual Boot Mode section
Creating a Dual Boot or Multiple Boot PROM Hex File	Added Creating a Dual Boot or Multiple Boot PROM Hex File section.
Appendix C. Differences Between Hex and Binary Files	Moved the Difference Between Hex and Binary Files section to the end of the document. This section is now Appendix C. Differences Between Hex and Binary Files.
Appendix D: Reference Material - Flash Configuration Primary	Moved the Reference Material section to the end of the document. This section is now Appendix D: Reference Material - Flash Configuration Primary.
Technical Support Assistance	Updated Technical Support Assistance information.

Revision 1.3, April 2013

Section	Change Summary
Critical Points	Added information on setting MCLK_FREQ frequency in Critical Points section.

Revision 1.2, August 2012

Section	Change Summary
All	Removed references to LatticeECP3 compression.

Revision 1.1, April 2012

Section	Change Summary
All	• Updated document with new corporate logo. • Updated to use Diamond Programmer and Lattice Deployment Tool. • Removed references to LatticeXP2.

Revision 1.0, October 2010

Section	Change Summary
All	Initial release.

www.latticesemi.com