LatticeECP3 sysCONFIG Usage Guide

Introduction

Configuration is the process of loading or programming a design into volatile memory of an SRAM-based FPGA. This is accomplished via a bitstream file, representing the logical states, that is loaded into the FPGA internal configuration SRAM memory. The functional operation of the device after programming is determined by these internal configuration RAM settings. The SRAM cells must be loaded with configuration data each time the device powers up.

The configuration memory in LatticeECP3™ FPGAs is built using volatile SRAM; therefore, an external non-volatile configuration memory is required to maintain the configuration data when the power is removed. This non-volatile memory supplies the configuration data to the LatticeECP3 when it powers up or anytime the device needs to be updated.

To support multiple configuration options the LatticeECP3 supports the Lattice sysCONFIG™ interface as well as the dedicated JTAG port. The available configuration options, or modes, are listed in Table 15-1.

Table 15-1. Supported Configuration Modes

Interface	Port	Description
sysCONFIG	SPI	Serial Peripheral Interface to single or multiple FPGA devices.
	SPIm	Serial Peripheral Interface to single Flash memory devices with partitioned memory.
	SSPI	Configure and readback by standard SPI Master driver or devices.
	SCM	Slave Serial Mode for daisy chain configuration.
	SPCM	Slave 8-bit parallel CPU-like programming interface.
JTAG	JTAG (IEEE 1149.1 and IEEE 1532 compliant)	Standard 4-pin JTAG interface.

This technical note covers all of the configuration options available for LatticeECP3.

The LatticeECP3 configuration RAM can be loaded in a number of different modes. In these configuration modes, the FPGA can act as a master, a peripheral to a CPU, or a slave of other system devices. It also supports in-system configuration via the JTAG port.

The decision about which configuration mode to use is a system design concern. There are many methods for configuring the FPGA utilizing four basic schemes.

- **Master**: As a master, the FPGA is the source of the clock, and accesses an external PROM or EPROM storage device through a serial connection, with no additional timing or control signals used. This scheme includes Serial Programming Interface (SPI) that supports a seamless connection for programming using industry-standard external Flash-based memory devices.

- **Slave**: In slave mode the FPGA receives bit-serial or byte-wide data and a clock from an external data and timing source, either from a microprocessor, or from the lead device in an FPGA-daisy chain. As a slave device, the clock used to configure the FPGA is generated externally and provided to the CCLK input.

- **JTAG**: The device can be configured through the JTAG port. The JTAG port is always on and available regardless of the configuration mode selected.

  The system designer should determine the requirements for configuration very early in the design. Many factors must be considered when deciding which configuration mode is best suited for the design. The flexible features for configuration can provide a seamless design to the system.

General Configuration Flow

The LatticeECP3 enters Configuration mode when one of three things happens: power is applied to the device, the PROGRAMN pin is driven low, or a JTAG or SSPI Refresh instruction is issued. Upon entering Configuration mode the INITN pin and the DONE pin are driven low to indicate that the device is initializing (i.e. getting ready to receive configuration data).

Once initialization is complete, the INITN pin will be driven high. The low-to-high transition of the INITN pin causes the CFG pins to be sampled, telling the LatticeECP3 which port it will configure from. The LatticeECP3 then begins reading data from the selected port and starts looking for the preamble header (BDB3 hex for unencrypted bitstreams, and BFB3 for encrypted bitstreams). All data after the preamble is valid configuration data.

When the LatticeECP3 has finished reading all of the configuration data, and assuming there have been no errors, the DONE pin goes high and the LatticeECP3 enters user mode. In other words, the device begins to function according to your design.

Note that the LatticeECP3 may also be programmed via JTAG. When programming via JTAG, the INITN and DONE signals have no meaning, because JTAG, per the IEEE standard, takes complete control of all device I/Os. It is recommended that the PROGRAMN input be held high when using the JTAG port to configure the FPGA. This prevents the FPGA SRAM memory from being cleared when the JTAG programming cycle is complete.

The following sections define each configuration pin, each configuration mode, and all of the configuration options for the LatticeECP3.

Configuration Pins

The LatticeECP3 supports two types of configuration pins, dedicated and dual-purpose. The dedicated pins are used exclusively for configuration; the dual-purpose pins, when configuration is complete, are available as extra I/O pins. If a dual-purpose pin is to be used both for configuration and as a general purpose I/O (GPIO) the following conditions must be met:

• The I/O type must remain the same. For example, if the pin is a 3.3V CMOS pin (LVCMOS33) during configuration, it must remain a 3.3V CMOS pin as a GPIO.

• You must select the correct CONFIG_MODE setting and set the PERSISTENT attribute to OFF. Doing so permits the dual-purpose sysCONFIG pins to be used as GPIO when configuration completes. These settings can be found in the ispLEVER® Design Planner or Spreadsheet View in Lattice Diamond™ design software. See Table 15-3 for more information.

• You are responsible for insuring that no internal or external logic will interfere with the control signals required by configuration mode you have selected.

The dual-purpose configuration pins are controlled using HDL source file attributes, or with the ispLEVER or Diamond Preference Editor. You can read about how to apply HDL preferences in TN1177, LatticeECP3 sysIO Usage Guide.

The LatticeECP3 also supports JTAG for configuration, transparent read back, and JTAG testing. The following sections describe the function of the various sysCONFIG and JTAG pins. Table 15-2 is provided for reference.

Table 15-2. LatticeECP3 Configuration Pins

Pin Name	I/O Type	Pin Type	Configuration Mode					
			SPI	SPIm	SSPI[1]	SCM[1]	SPCM[1]	JTAG
CFG[2:0]	Input, weak pull-up	Dedicated	=000	=010	=001	=101	=111	
PROGRAMN	Input, weak pull-up	Dedicated	ALL					
INITN	Bi-directional open drain[5]	Dedicated	ALL					
DONE	Bi-directional open drain[5]	Dedicated	ALL					
CCLK	Input	Dedicated	Slave mode, determined by the CFG0 setting =1					
MCLK	Bi-directional, weak pull-up	Dual-Purpose	Master mode, determined by the CFG0 setting =0					
D0/SPIFASTN[2]	Bi-directional[2]	Dual-Purpose	SPIFASTN	SPIFASTN			D0	
D1[2, 3]	Bi-directional[2]	Dual-Purpose					D1	
D2[2, 3]	Bi-Directional[2]	Dual-Purpose					D2	
D3/SI[2, 3]	Bi-directional[2]	Dual-Purpose			SI		D3	
D4/SO[2, 3]	Bi-directional[2]	Dual-Purpose			SO		D4	
D5[2]	Bi-directional[2]	Dual-Purpose					D5	
D6[2, 3]	Bi-directional[2]	Dual-Purpose					D6	
D7/SPID0[2, 3]	Bi-directional[2]	Dual-Purpose	SPID0	SPID0	Note 4		D7	
CSN/SN	Bi-directional, weak pull-up	Dual-Purpose			SN		CSN	
CS1N/HOLDN	Bi-directional, weak pull-up	Dual-Purpose			HOLDN[3]		CS1N	
WRITEN	Active low control input pin	Dual-Purpose					WRITEN	
BUSY/SISPI	Bi-directional, weak pull-up	Dual-Purpose	SISPI	SSIPI	Note 4		BUSY	
DI/CSSPI0N	Bi-directional, weak pull-up[6]	Dual-Purpose	CSSPI0N	CSSPI0N	Note 4	DI		
DOUT/CSON	Bi-directional, weak pull-up	Dual-Purpose	DOUT		DOUT	DOUT	DOUT/CSON	

1. SSPI = Slave SPI, SCM = Serial Configuration Mode, SPCM = Slave Parallel Configuration Mode.
2. D[0:7] pins have no pull-up during power-up and configuration in all programming modes. This allows you to use large pull-up or pull-down resistors to pre-set those pins to a certain state while powering up your systems.
3. Weak pull-ups consist of a current source of 30μA to 150μA. The pull-ups for sysCONFIG dedicated and dual-purpose pins track VCCIO8. The pull-ups for TDI, TDO, and TMS track VCCJ.
4. This pin is used for programming the SPI Flash boot PROM.
5. Optional weak pull-up resistor available.
6. Requires external pull-up to VCCIO8.

Configuration Process and Flow

Prior to becoming operational, the FPGA goes through a sequence of states, including initialization, configuration and wake-up.

Figure 15-1. Configuration Flow
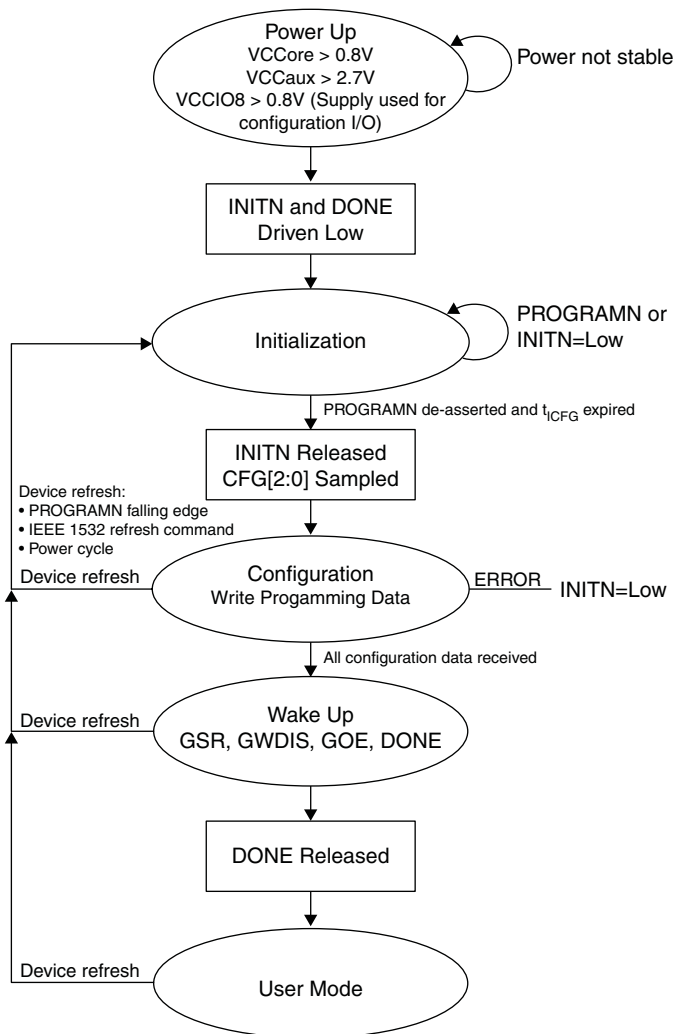
PROGRAMN must not be asserted low until after all power rails have reached stable operation.

PROGRAMN must not make a falling edge transition during the time the FPGA is in the Initialization state. PROGRAMN must be asserted for a minimum low period of t_{PRGM} in order for it to be recognized by the FPGA. Failure to meet this requirement can cause the device to become non-operational, requiring power to be cycled.

Power-up Sequence

In order for the LatticeECP3 to operate, power must be applied to the device. During a short period of time, as the voltages applied to the system rise, the FPGA will have an indeterminate state. Other devices in the system will also be in an indeterminate state.

As power continues to ramp, a Power On Reset (POR) circuit inside the FPGA becomes active. The POR circuit, once active, makes sure the external I/O pins are in a high-impedance state. It also monitors the V_{CCcore}, V_{CCaux}, and the V_{CCIO8} input rails. The POR circuit waits for the following conditions:

- $V_{CCcore} > 0.8V$
- $V_{CCaux} > 2.7V$
- $V_{CCIO8} > 0.8V$ (Supply used for configuration I/O)

When these conditions are met the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The FPGA samples the CFG[2:0] input pins to determine if a master or a slave mode configuration is selected. The FPGA uses this information to determine the t_{ICFG} initialization period. The next step is to assert INITN active low, and to drive DONE low. When INITN and DONE are asserted low the device moves to the initialization state, as shown in Figure 15-1.

The PROGRAMN input must not be asserted low as power is applied to the FPGA. Nor should it be allowed to transition from high to low at any time that INITN is in the initialization state.

Figure 15-2. Configuration from Power-On-Reset Timing
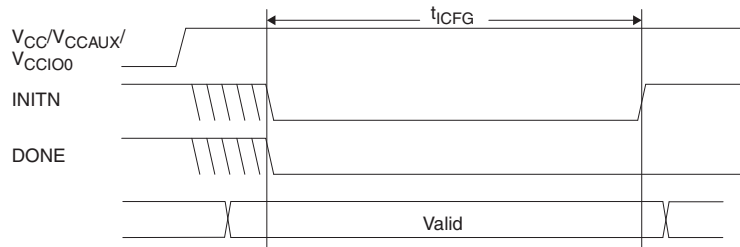
Initialization

The LatticeECP3 enters the memory initialization phase immediately after the Power On Reset circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all of the SRAM memory inside the FPGA.

The FPGA remains in the initialization state until all of the following conditions are met:

- The t_{ICFG} time period has elapsed
- The PROGRAMN pin is deasserted
- The INITN pin is no longer asserted low by an external master

The dedicated INITN pin provides two functions during the initialization phase. The first is to indicate the FPGA is currently clearing its configuration SRAM. The second is to act as an input preventing the transition from the initialization state to the configuration state.

During the t_{ICFG} time period the FPGA is clearing the configuration SRAM. When the LatticeECP3 is part of a chain of devices each device will have different t_{ICFG} initialization times. The FPGA with the slowest t_{ICFG} parameter can prevent other devices in the chain from starting to configure. Premature release of the INITN in a multi-device chain may cause configuration of one or more chained devices to fail to configure intermittently.

The active-low, open-drain initialization signal INITN must be pulled high by an external resistor when initialization is complete. To synchronize the configuration of multiple FPGAs, one or more INITN pins should be wire-ANDed. If one or more FPGAs or an external device holds INITN low, the FPGA remains in the initialization state.

Loading the Configuration Memory

The rising edge of the INITN pin causes the FPGA to enter the configuration state. The FPGA is able to accept the configuration bitstream created by the ispLEVER and Diamond development tools.

If the FPGA CFG[2:0] input pins have placed it in a master configuration mode it will begin fetching data from an external non-volatile memory.

If the FPGA CFG[2:0] input pins have placed it in a slave configuration mode, the FPGA waits for configuration data to be presented to it on each CCLK rising edge.

During the time the FPGA receives its configuration data the INITN control pin takes on its final function. INITN is used to indicate an error exists in the configuration data. When INITN is active high configuration is proceeding without issue. If INITN is asserted low, an error has occurred and the FPGA will not operate.

Wake-up

Wake-up is the transition from configuration mode to functional mode. Wake-up starts when the device has correctly received all of its configuration data. When all configuration data is received, the FPGA asserts an internal DONE strobe. The assertion of the internal DONE causes a Wake Up state machine to run that sequences four controls. The four control strobes are:

- External DONE
- Global Write Disable (GWDISn)
- Global Output Enable (GOE)
- Global Set/Reset (GSR)

External DONE is a bi-directional, open-drain I/O. The FPGA releases the open-drain DONE pin at the programmed wake-up phase. If an external agent is holding the external DONE pin low, the wake-up process of the LatticeECP3 does not proceed. Only after the external DONE is active high do the final wake-up phases complete. Once the final wake-up phases are complete, the FPGA enters user mode.

The Global Output Enable, when it is asserted, permits the FPGA's I/O to exit a high-impedance state and take on their programmed output function. The FPGA inputs are always active. However, they are typically prevented from performing any action on the FPGA logic by the assertion of the Global Set/Reset (GSR).

The Global Set/Reset is an internal strobe that, when asserted, causes all I/O/LUT flip-flops, distributed RAM output flip-flops, and Embedded Block RAM output flip-flops that have the GSR enabled attribute to be set/cleared per their HDL definition.

The Global Write Disable is a control that overrides the write enable strobe for all RAM logic inside the FPGA. The inputs on the FPGA are always active, as mentioned in the Global Output Enable section. Keeping GWDIS asserted prevents accidental corruption of the instantiated RAM resources inside the FPGA.

Clearing the Configuration Memory and Re-initialization

Several methods are available to clear the internal configuration memory of the LatticeECP3 device. The first is mentioned earlier when the device powers up (see the "Power-up Sequence" section of this document). A second method is to toggle the PROGRAMN pin. Also, JTAG can reinitialize the memory through an ISC Refresh command. SSPI can also initiate a reconfiguration with the Refresh command.

The other methods available are:

- Assertion of the PROGRAMn dedicated input
- Sending the ISC Refresh command using a configuration port (JTAG, or Slave SPI)

Invoking one of these methods causes the LatticeECP3 to drive INITN and DONE low. The FPGA enters the initialization state described above.

Reconfiguration Priority

There are many sources that can initiate a reconfiguration while a configuration is already in process. There is a priority as to which of these sources can interrupt the configuration process depending on which of the sources initiated the original configuration. Note that if an interruption occurs, the reconfiguration will occur without informing the original configuration source that the configuration did not complete. JTAG always has the highest priority and any JTAG initiated configuration event will cause a reconfiguration to occur. Toggling the PROGRAMN pin has the next highest priority. It will interrupt any current configuration other than a JTAG configuration.

FPGA Configuration Control Pin Definitions

The LatticeECP3 FPGA provides a set of I/O pins that can be used to load a configuration bitstream into the device. Some of these I/O are single purpose and are always available to perform configuration operations. Those configuration pins that are not dedicated can be configured for your use after the FPGA has entered user mode. This section describes what each I/O is, how it functions, and how to reclaim some for your own use.

Configuration Pin Management

The dual-purpose sysCONFIG pins described in the Table 15-2 are dedicated configuration pins during the device configuration process. The PERSISTENT attribute is used to determine whether the dual-purpose sysCONFIG pins remain sysCONFIG pins during normal operation. The LatticeECP3 provides three settings for the PERSISTENT feature. The available options are shown in Table 15-3.

Table 15-3. PERSISTENT Setting and Affected Pins

Persistent Setting	Pins
OFF	All dual-purpose configuration pins are available as GPIO
SLAVE_PARALLEL	D [0:7], CSN, CS1N, WRITEN, BUSY, CSON, MCLK[1]
SSPI	SI, SO, SN, HOLDN, SISPI, SPID0, SPID1, CSSPIN and CSSPI1N

1. These pins are not used by the Slave Parallel logic, but they are reserved by the Slave Parallel logic. They are not available for use as GPIO.

You can use the SLAVE_PARALLEL or the Slave SPI configuration port to access some of the resources connected to the FPGA. Accessing the FPGA resources requires special command sequences, which are described in other documents.

Dedicated Control Pins

The LatticeECP3 makes a set of dedicated control pins available to allow you to select the way you want to configure the FPGA. The following sub-sections describe the LatticeECP3 dedicated sysCONFIG pins. These pins are powered by V_{CCIO8}.

While the device is under IEEE 1149.1 or 1532 JTAG control the dedicated programming pins have no meaning. This is because a boundary scan cell will control each pin, per JTAG 1149.1, rather than normal internal logic.

JTAG Pins

The JTAG pins are standard IEEE 1149.1 TAP (Test Access Port) pins. The JTAG pins are dedicated pins and are always accessible when the LatticeECP3 device is powered up. While the device is under 1149.1 or 1532 JTAG control the dedicated programming pins INITN, DONE, and CCLK have no meaning. This is because a boundary scan cell will control each pin, per the IEEE standard, rather than normal internal logic. If the device is being accessed by JTAG then PROGRAMN will still be an active input even in JTAG mode.

These pins are powered by V_{CCJ}.

TDO
The Test Data Output pin is used to shift out serial test instructions and data. When TDO is not being driven by the internal circuitry, the pin will be in a high impedance state. This pin should be wired to TDO of the JTAG connector, or to TDI of a downstream device in a JTAG chain. An internal pull-up resistor on the TDO pin is provided. The internal resistor is pulled up to V_{CCJ}.

TDI
The Test Data Input pin is used to shift in serial test instructions and data. This pin should be wired to TDI of the JTAG connector, or to TDO of an upstream device in a JTAG chain. An internal pull-up resistor on the TDI pin is provided. The internal resistor is pulled up to V_{CCJ}.

TMS
The Test Mode Select pin controls test operations on the TAP controller. On the falling edge of TCK, depending on the state of TMS, a transition will be made in the TAP controller state machine. An internal pull-up resistor on the TMS pin is provided. The internal resistor is pulled up to V_{CCJ}.

TCK
The test clock pin, TCK, provides the clock to run the TAP controller state machine, which loads and unloads the JTAG data and instruction registers. TCK can be stopped in either the high or low state and can be clocked at frequencies up to that indicated in the LatticeECP3 Family Data Sheet. The TCK pin supports hysterisis; the typical hysterisis is approximately 100mV when V_{CCJ} = 3.3V. The TCK pin does not have a pull-up. A pull-down resistor between TCK and ground on the PCB of 4.7 K is recommended to avoid inadvertent clocking of the TAP controller as V_{CC} ramps up.

Optional TRST
Test Reset, TRST, is not supported on the LatticeECP3.

V_{CCJ}
Having a separate JTAG V_{CC} (V_{CCJ}) pin lets you apply a voltage level to the JTAG port that is independent from the rest of the device. Valid voltage levels are 3.3V, 2.5V, 1.8V, 1.5V, and 1.2V, but the voltage used must match the other voltages in the JTAG chain. V_{CCJ} must be connected even if JTAG is not used.

Please see In-System Programming Design Guidelines for ispJTAG Devices for further JTAG chain information.

Configuration and JTAG Pin Physical Description
All of the sysCONFIG dedicated and dual-purpose pins are part of Bank 8. Bank 8 V_{CCIO} determines the output voltage level of these pins, input thresholds are determined by the I/O Type selected in the ispLEVER Design Planner (default is 3.3V LVCMOS) and Diamond Spreadsheet View.

JTAG voltage levels and thresholds are determined by the V_{CCJ} pin, allowing the LatticeECP3 to accommodate JTAG chain voltages from 1.2V to 3.3V.

CFG**[2:0]**
The Configuration Mode pins, CFG[2:0], are used to inform the FPGA how you want to configure the device. The actions performed by the remaining configuration pins depend on the state of the CFG[2:0] inputs. The CFG[2:0] input pins have weak internal pull-up resistors, that guarantee a valid configuration mode is selected should they be left unconnected. Lattice recommends the CFG pins be connected with independent pull-up/pull-down resistors. It is also recommended that these pins not be directly connected to the power or ground planes.

The CFG[2:0] pins are sampled at two different points in the configuration process. The first sample point is when the Power-On Reset state machine starts up. The POR sample point determines if the FPGA to be configured in master or slave mode. The t_{ICFG} time period changes based on this information.

The second time the CFG pins are sampled is at the rising edge of the INITN pin. This sample is used to make the final configuration selection. Table 15-4 describes the configuration mode that is active based on the CFG input pins. The state on the CFG pins at any other time is not important. The state pins can be changed freely, which may be useful for selecting a new configuration mode.

Table 15-4. LatticeECP3 Configuration Pins[1]

Configuration Mode	Clock	CFG [2]	CFG [1]	CFG [0]
SPI Master (Single)	MCLK	0	0	0
SPI Master (Multiple)		0	1	0
Slave SPI	CCLK	0	0	1
		X	X	X
SCM	CCLK	1	0	1
SPCM	CCLK	1	1	1

1. JTAG is always available for IEEE 1149.1 and 1532 support.

PROGRAMN

The PROGRAMn is a dedicated input that is used to configure the FPGA. The PROGRAMn pin is a falling edge sensitive, and has an internal weak pull-up. When a falling edge occurs, the FPGA exits user mode and starts a device configuration sequence at the Initialization phase, as described earlier.

Proper operation of the LatticeECP3 FPGA depends on the PROGRAMn pin. The following conditions must be met:

- The PROGRAMn pin must not be asserted until after all of the supply rails are stable. This can be achieved by either placing an external pullup resistor and tying it to the VCCIO8 voltage, or permitting the FPGA's internal pull-up resistor to pull the input high.

- The PROGRAMn pin must make a high to low transition in order to cause the FPGA to enter configuration mode. This is not necessary when first powering the FPGA, as the FPGA will enter configuration mode after the internal Power On Reset circuit releases the internal reset.

- The PROGRAMn pin must not be allowed to transition from high to low at any time INITn is active (i.e. low) as a result of being in the Initialization state.

- PROGRAMn must meet the minimum active pulse width t_{PRGM}.

- PROGRAMn remains an active input even when the JTAG bus is being used to program the FPGA. The PROGRAMn pin should not be asserted during JTAG programming sequences.

Failing to follow these guidelines may prevent the FPGA from operating.

PROGRAMn must be de-asseted in order for the FPGA to exit the Initialization state.

Figure 15-3. Configuration from PROGRAMN Timing
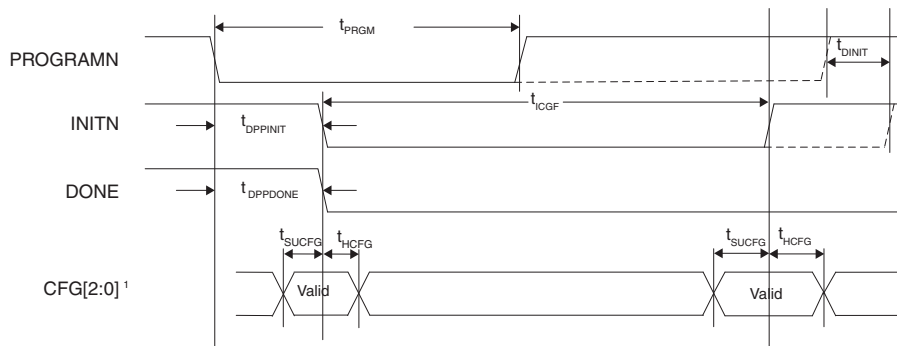
1. The CFG pins are normally static (hard wired).

INITN

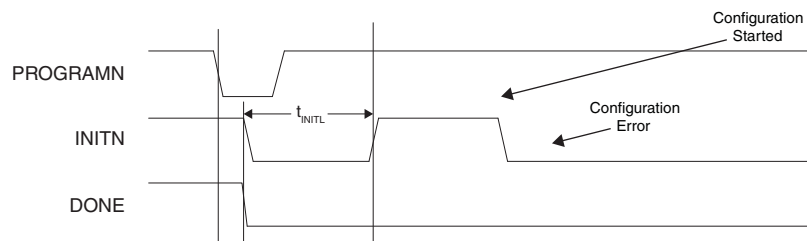The INITn pin is a bidirectional open-drain control pin. It has the following functions:

• After power is applied, or after a PROGRAMn assertion it goes low to indicate the FPGA configuration cells are being erased. The low time assertion is specified with the t_{ICFG} parameter.

• After the t_{ICFG} time period has elapsed the INITn pin is deasserted (i.e. is active high) to indicate the FPGA is ready for its configuration bits. In master mode the FPGA starts loading bits from an attached non-volatile memory. In slave mode the FPGA waits for the bits to arrive over the interface selected by the CFG[2:0] input pins. The rising edge of the INITn samples the CFG[2:0] inputs, allowing the FPGA to determine how it is to be configured.

• INITn can be asserted low by an external agent before the t_{ICFG} time period has elapsed in order to prevent the FPGA from reading configuration bits. This is useful when there are multiple FPGA's chained together. The FPGA with the longest t_{ICFG} time can hold all other FPGA's in the chain from starting to get data until it is ready itself.

• The last function provided by INITn is to signal an error during the time configuration data is being read. Once t_{ICFG} has elapsed, and the INITn pin has gone high, any INITn assertion signals the FPGA has detected an error during configuration.

The following conditions will cause INITN to become active, indicating the Initialization state is active:

• Power has just been applied

• PROGRAMN falling edge occurred

• The IEEE 1532 Refresh command has been sent using a slave configuration port (JTAG, SSPI, etc.).

If the INITN pin is asserted due to an error condition, the error can be cleared by correcting the configuration bitstream and forcing the FPGA into the Initialization state.

Figure 15-4. Configuration Error Notification

DONE

The DONE pin is a dedicated bi-directional open drain with a weak pull-up that signals the FPGA is in User mode. DONE is first able to indicate entry into User mode only after the internal DONE pin is asserted. The intenal DONE signal defines the beginning of the FPGA Wake-Up state.

The DONE output pin is controlled by the DONE_EX configuration parameter. The default state of this pin is OFF. The default mode causes the LatticeECP3 to start immediately after the internal DONE bit is asserted. The FPGA does not stall waiting for the DONE pin to be asserted high.

The FPGA can be held from entering User mode indefinitely by having an external agent keep the DONE pin asserted low. In order to use DONE to stall entering user mode the DONE_EX configuration preference must be set ON. A common reason for keeping DONE driven low is to allow multiple FPGAs to be completely configured. As each FPGA reaches the DONE state, it is ready to begin operation. The last FPGA to configure can cause all FPGAs to start in unison.

It is critical that DONE be asserted low when the LatticeECP3 is in a chain of FPGAs. The LatticeECP3 continues to pass configuration clock pulses to FPGAs attached downstream as long as DONE is de-assserted. Any FPGA

permitted to assert DONE and enter User mode will only pass a few hundred more configuration clock cycles. Downstream FPGAs will never complete their configuration process if this occurs.

The DONE pin drives low in tandem with the INITN pin when the FPGA enters Initialization mode. As described earlier, this condition happens when power is applied, PROGRAMN is asserted, or an IEEE 1532 Refresh command is received via a slave configuration port.

Sampling the DONE pin is a good way for an external device to tell if the FPGA has finished configuration. However, when using IEEE 1532 JTAG to configure SRAM the DONE pin is driven by a boundary scan cell, so the state of the DONE pin has no meaning during IEEE 1532 JTAG configuration (once configuration is complete, DONE takes on the behavior defined by the DONE_EX configuration parameter).

Configuration Clock (CCLK)
The CCLK is a dedicated input-only whose purpose is to provide a reference clock for the FPGA when one of the slave configuration modes is selected by the CFG[2:0] inputs.

Please refer to the LatticeECP3 AC Timing information in the LatticeECP3 Family Data Sheet for information about maximum clock rates, and data setup and hold parameters.

When the LatticeECP3 is in a chain of FPGAs it is necessary to continue to drive CCLK until all of the FPGAs have received their configuration data. It is recommended the CCLK continue to be toggled until the DONE signal is active.

Dual-Purpose sysCONFIG Pins

The Dual-Purpose sysCONFIG pins, depending on the configuration mode selected by the CFG[2:0] input pins, provide special configuration functions during the Configuration phase of the device wake-up process. The dual-purpose pins can be recovered for your use once the FPGA enters User mode. Successful recovery of the dual-purpose pins relies on following the guidelines described in the "Configuration Pins" section of this document.

The dual-purpose configuration pins are located in the same I/O bank as the dedicated configuration pins. The configuration pins in the LatticeECP3 are powered by the VCCIO8 voltage rail.

Master Clock (MCLK)
The MCLK provides a reference clock for synchronous non-volatile memories attached to the FPGA. MCLK is only active when the FPGA CFG[2:0] inputs select a master configuration mode. See Table 15-4 for a full description of the available configuration modes selectable by the CFG[2:0] input pins. MCLK acts as a general purpose I/O if the FPGA is in a slave configuration mode.

The LatticeECP3 generates MCLK from an internal oscillator. The initial output frequency of the MCLK is approximately 2.5MHz. The MCLK frequency can be altered using the MCCLK_FREQ parameter. You can select the MCCLK_FREQ using the Diamond Spreadsheet View. For a complete list of the supported MCLK frequencies, see Table 15-8.

During the initial stages of device configuration the frequency value specified using MCCLK_FREQ is loaded into the FPGA. Once the FPGA accepts the new MCLK_FREQ value the MCLK output begins driving the selected frequency. Make certain that when selecting the MCLK_FREQ that you do not exceed the frequency specification of your configuration memory, or of your PCB. Lattice recommends reviewing the LatticeECP3 AC specifications in the LatticeECP3 Family Data Sheet when making MCLK_FREQ decisions.

The LatticeECP3 provides the ability to be configured from a bitstream that is encrypted. There are additional requirements on the MCCLK_FREQ selection that you must adhere to when configuring the LatticeECP3 with an encrypted bitstream. These conditions are discussed in the "Bitstream Encryption/Decryption Flow" section of this document.

DI/CSSPI0N

The DI/CSSPI0N configuration pin provides one of two functions depending on the FPGA's configuration mode. When the FPGA is in Serial Configuration Mode the pin is set to DI (Data Input) mode. When the FPGA is in SPI Master or SPI Master Multiboot mode, the pin is set to CSSPI0N (Chip Select SPI 0).

When the FPGA is in Serial Configuration Mode the DI pin receives incoming configuration data. See the Serial Configuration Mode section of this document for more information.

When the FPGA is in SPI Master or SPI Master Multiboot mode the CSSPI0N is the chip select strobe to the attached SPI memory that contains the FPGA's configuration bits. The FPGA asserts this pin active low during the Configuration phase of the wake-up process.

An external pull-up resistor is required on CSSPI0N in SPI and SPIm modes of operation. Some SPI memory devices require the CSn input to rise in tandem with their input voltage. The internal pull-up on CSSPI0N does not become active until the FPGA determines all input voltage rails are stable. This does not meet the requirements of some SPI memory vendors.

DOUT/CSON

The DOUT/CSON configuration pin is used only when placing the LatticeECP3 into a chain of FPGAs requiring configuration.

The DOUT/CSON pin is an output from the LatticeECP3 and is only active when the FPGA is connected to another FPGA in a daisy chain.

When in a daisy chain, the pin may act as either a data output (DOUT) or a chip select (CSON). The behavior is described in detail in the Configuring Multiple FPGA Devices section of this document.

For serial and parallel configuration modes, when Bypass mode is selected, this pin becomes DOUT (see Figure 15-10). When the device is fully configured a Bypass instruction in the bitstream is executed and the data on DI, or D[0:7] in the case of a parallel configuration mode, will then be routed to the DOUT pin. This allows data to be passed, serially, to the next device. In a parallel configuration mode D0 will be shifted out first followed by D1, D2, and so on.

Daisy chaining the Parallel devices can be implemented with the Flowthrough attribute. This attribute allows the CSON pin to be driven when the done bit is set and configuration of the first part is complete. The CSON of the first part will drive the CSN of the second part.

You will find this attribute in the ispLEVER Generate Bitstream Data property under Chain Mode or in the Diamond Bitstream options window. See Appendix B for more information on setting these options in Diamond.

The DOUT/CSON drives out a high on power-up and will continue to do so until the execution of the Bypass instruction within the bitstream, or until the I/O Type is changed by your code.

CSN/SN

The CSN/SN is a bidirectional pin that is active in Slave Parallel Configuration mode, or in Slave SPI mode. The pin is a chip select that gates the incoming configuration data.

Detailed information about using the chip select pin can be found in the "Slave Parallel Mode (SPCM)" and "Slave SPI (SSPI)" configuration sections of this document.

CS1N/HOLDN

The CS1N/HOLDN configuration pin is active only in Slave Parallel Configuration mode or in Slave SPI mode.

When the LatticeECP3 is in a Slave Parallel Configuration mode the pin acts as a chip select that works in conjunction with CSN. Information detailing the interaction of these two pins is described in the Parallel Configuration mode section of this document.

The configuration pin takes on the HOLDN function when the LatticeECP3 is in Slave SPI Configuration mode. Assertion of the HOLDN input causes the FPGA to ignore the SPI SCLK. Details for using HOLDN are provided in the Slave SPI Configuration section.

When CSN or CS1N is high, the D[0:7], and BUSY pins are tri-stated. CSN and CS1N are interchangeable when controlling the D[0:7], and BUSY pins.

WRITEN

The WRITEN configuration pin is an input pin that is active in Slave Parallel Configuration mode. It is a write enable strobe that controls the direction data flows on the D[0:7] data bus pins. When WRITEN is asserted active low the FPGA D[0:7] pins are tri-stated to allow an external bus master to write data into the FPGA.

BUSY/SISPI

The BUSY/SISPI configuration pin is active in Slave Parallel Configuration mode and in SPI Master modes.

When the LatticeECP3 is in a Slave Parallel Configuration mode, the pin is a tri-state output pin. When the FPGA parallel bus is active due to the assertion of CSN/CS1N the BUSY pin indicates the FPGA's ability to accept a byte of configuration data. The FPGA is able to accept another configuration byte when this output is driven low.

When the LatticeECP3 is in SPI Master mode the pin is connected to the data input of the SPI PROM that contains the configuration data. SISPI is an output used by the LatticeECP3 to transmit commands to the SPI PROM.

D[0]/SPIFASTN

The D[0]/SPIFASTN configuration pin is available in Slave Parallel Configuration and SPI Master configuration modes.

In Slave Parallel Configuration mode the D[0] pin is the most-significant data bit in the combined D[0:7] parallel data bus.

In SPI Master configuration modes it becomes the SPIFASTN input. The input is sampled at the rising edge of the INITN output.

The SPIFASTN selects the Read Opcode transmitted to the SPI PROM. When SPIFASTN is deasserted (i.e. driven to Vih) the FPGA requests a Read Operation using the 03 hex Read Opcode. When SPIFASTN is asserted (i.e. driven to Vil) the FPGA requests a Read Operation using the 0B hex Read Opcode. A SPI PROM that accepts the 0B Read Opcode is able to operate at higher serial clock rates. Consult the SPI memory vendor's data sheet for the exact capabilities of the SPI memory device.

Do not leave this input floating when a SPI Master mode is selected.

In parallel mode this pin is D[0] and operates in the same way as D[1:6] below. Taken together D[0:7] form the parallel data bus, D[0] is the most significant bit in the byte. The D[0:7] data bus are open-drain I/O without a pullup resistor during the time that power is applied to the FPGA. They also remain in this state until the FPGA is fully configured. When the FPGA is configured the D[0:7] I/O take on the attributes defined in your HDL source code, or using the Spreadsheet View preference editor.

As with D[1:6], if SRAM (configuration memory) needs to be accessed using the parallel pins while the part is in user mode (the DONE pin is high) then the PERSISTENT control cell must be set to preserve this pin as D[0]. Note that SRAM may only be read using JTAG or Slave Parallel mode.

D[1], D[2], D[5] and D[6]

These configuration pins are only available in Slave Parallel Configuration mode and are intermediary data bits for the parallel data bus made up of bits D[0:7].

Remember that D[0] is the most-significant data bit and D[7] is the least-significant.

D[3]/SI

The D[3]/SI configuration pin is only available in Slave Parallel Configuration or in Slave SPI Configuration mode.

When the LatticeECP3 is in Slave Parallel Configuration mode the pin acts as D[3].

In Slave SPI Configuration mode the pin acts as the Serial Input pin for data supplied by a SPI Master Controller.

D[4]/SO

The D[4]/SO configuration pin is only available in Slave Parallel Configuration or in Slave SPI Configuraiton mode.

When the LatticeECP3 is in Slave Parallel Configuration mode the pin acts as D[4].

In Slave SPI Configuration mode the pin acts as the Serial Output pin for data transmitted from the FPGA back to the SPI Master Controller.

D[7]/SPID0

The D[7]/SPID0 configuration pin is only available in Slave Parallel Configuration or in Master SPI Configuration mode.

When the LatticeECP3 is in Slave Parallel Configuration mode the pin acts as D[7]. This is the least-significant-bit of the D[0:7] data bus.

In Master SPI Configuration mode the pin acts as the SPI Data Input pin receiving data from an attached SPI PROM.

Configuration Modes

LatticeECP3 devices support many different configuration modes, utilizing either serial or parallel data paths. The configuration method used by the LatticeECP3 is selected by driving the CFG[2:0] input pins. The CFG[2:0] input pins are sampled at the falling edge of INITN to determine if the part is in a master or a slave configuration mode. The pins are sampled a second time at the rising edge of INITN to determine the specific configuration mode.

The LatticeECP3 starts the configuration process in one of three ways:

- Initial application of power
- Assertion of the PROGRAMN input pin
- Reception of a REFRESH command form a configuration port (JTAG, Slave SPI, Slave Parallel)

SPI Interface

The Serial Peripheral Interface (SPI) is a four-wire de facto bus standard used to transmit and receive serial data. The LatticeECP3 can use a SPI data bus to retrieve its configuration data from most SPI ROMs.

The amount of ROM storage required depends on the number of Look Up Tables (LUTs) in the LatticeECP3 device you have selected. Figure 15-5 shows how many bits of configuration data are required for each member of the LatticeECP3 family.

Table 15-5. Maximum Configuration Bits – SPI Flash Mode Bitstream File[1]

Device	Bitstream Size[1]	SPI Flash	Dual Boot SPI Flash	Units
ECP3-17	4.5	8	16	Mb
ECP3-35	8.2	16	32	Mb
ECP3-70	22.5	32	64	Mb
ECP3-95	22.5	32	64	Mb
ECP3-150	35.7	64	128	Mb

1. The Bitstream Size column is the maximum number of bits the FPGA may require. This number takes into account the pre-initialization of all Embedded Block RAMs.

The estimated configuration time can be calculated by dividing the bitstream size (in bits) from Table 15-5 by the configuration clock (MCLK or CCLK) frequency. The MCLK frequency is modified using the Global Preferences tab within the Diamond Spreadsheet View or in the ispLEVER Design Planner.

The LatticeECP3 provides the following three SPI configuration modes:

• SPI Master (SPI)

• SPI Master Multiboot (SPIm)

• Slave SPI (SSPI)

SPI Master Mode
The simplest SPI configuration consists of one SPI Serial Flash connected to one LatticeECP3, as shown in Figure 15-5. In this configuration the LatticeECP3 is the master of the SPI bus. The FPGA controls the chip select and the MCLK, and receives the configuration data on the SPID0 input.

During FPGA configuration the SISPI output sends a command sequence to reset the SPI PROM's internal address pointer. The SPIFASTN informs the FPGA which SPI Read Command to send to the SPI PROM. When the SPIFASTN input is driven high, the FPGA sends a 03 Hex Read Opcode. When it is asserted active low, the FPGA sends a 0B Hex Read Opcode to the SPI PROM.

As mentioned in the section describing the MCLK behavior, the configuration clock frequency can be altered. The MCLK frequency must not exceed the clock input frequency of the SPI PROM.

Figure 15-5. One FPGA, One SPI Serial Flash
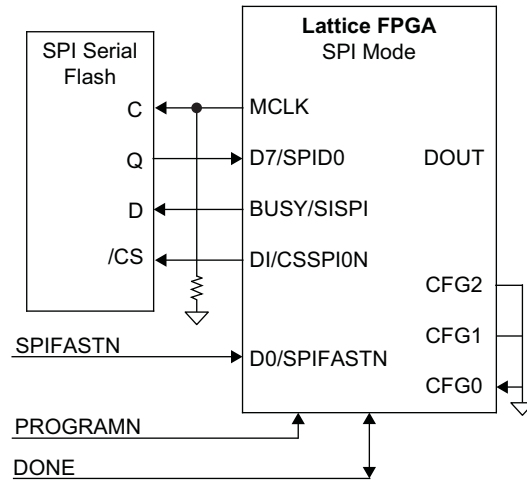
Note: The board-level pull-down on MCLK should have a 1-3 Kohms resistance. This counteracts the weak internal pull-up on MCLK and prevents an unintentional rising edge at power-up.

In order to configure properly, the LatticeECP3 must transmit at least 128 clock pulses before it receives the preamble code (BDB3 hex for unencrypted bitstreams, and BFB3 for encrypted bitstreams). It is required that the data in the SPI PROM be padded to account for these extra clocks. The bitstream generation tool automatically adds the necessary padding information.

There are some general rules that user has to follow in order to configure properly:

• The POR of the SPI flash device should be lower than the POR of LatticeECP3 or the SPI flash must be powered first.

• SPI flash maximum frequency must be higher than LatticeECP3 MCLK frequency.

• Board routing requirements need to ensure proper timing. Please refer to ECP3 data sheet for detailed timing requirement.

SPI Master Multiboot (SPIm) Mode

SPI Master Multiboot mode is an enhancement to the SPI Master boot mode. The SPI memory is attached to the FPGA in exactly the same way as SPI Master mode. SPIm enables you to partition the SPI PROM to store two configuration bitstreams. The FPGA will attempt to configure from the Primary image, and if the FPGA fails to configure from the primary image it tries to load a fail-safe Golden bitstream. Figure 15-6 shows the concept from a high level.

Figure 15-6. SPIm Mode for Dual-Boot Capability
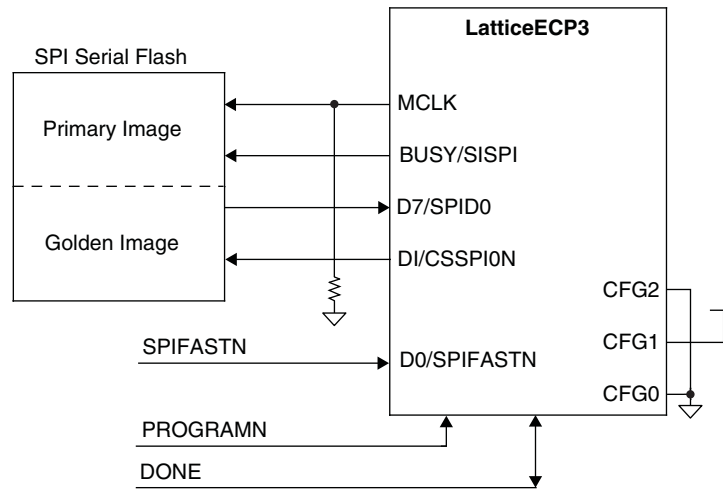
Internally, SPIm mode adds logic to detect a configuration failure and the ability to reattempt configuration from a different address within the SPI Flash device. While SPI mode treats the SPI Flash device as a single block of storage starting at address zero, SPIm mode allows segmentation of the Flash device for the golden bitstream.

In SPIm mode, the primary bitstream is stored at address offset 0x010000. When configuring, the LatticeECP3 device automatically reads the data beginning at this address first. If after 2*14 clocks the device still does not receive the pre-amble code or a bitstream error is encountered after receiving the pre-amble code, the configuration logic resets and loads the data located at address offset 0xFFFF00.

The LatticeECP3 uses a 24-bit addressing scheme to access the SPI memory array. The amount of storage remaining in the SPI starting at address 0xFFFF00 is only 256 bytes. This is not enough to store a complete LatticeECP3 configuration bitstream. The LatticeECP3 configuration bitstream is stored elsewhere in the SPI PROM. The data retrieved by the FPGA at address 0xFFFF00 is an instruction pointing to the start of the failsafe configuration data.

An example of the SPI Flash memory organization for SPIm mode is shown in Table 15-6.

Table 15-6. SPIm Mode Data Map[1, 2, 3]

Block (512Kb)	SPI Flash Address	Contents
0	0x000000	Unused
1 2 3 . . . 18	0x010000 0x020000 0x030000 . . . 0x120000	Primary Bitstream
32 33 34 . . . 49	0x200000 0x210000 0x220000 . . . 0x310000	Golden Bitstream
N	0xFF0000	
	0xFFFF00	Jump instruction to 0x200000

1. The bitstream sizes shown are examples. Actual sizes and address boundaries vary with device density.
2. After the golden bitstream is written into the SPI Flash device, the top half of the SPI Flash can be write-protected to secure the golden bitstream from alterations.
3. When the SPI Flash device reaches the address 0xFFFFFF, it rolls over to address 0x000000. If the last page is un-programmed, the device can read the jump instruction programmed on address 0x000000 effectively implementing a multiple patterns support for board development or debugging need.

Slave SPI (SSPI)

The LatticeECP3 can be configured by a SPI Master controller. Using the CFG[2:0] inputs to select SSPI configuration mode the FPGA becomes a SPI Slave device, receiving data from a SPI Master controller. The FPGA can be accessed using Mode 0 and Mode 3 bus transactions.

The slave SPI interface allows for a the following functions to be performed:

- Configuration of the FPGA

- Readback of the FPGA configuration bitstream

- Forcing the device to REFRESH as if PROGRAMN were asserted

- Read/Write access to a SPI PROM attached to the SPI Master configuration pins

- Clearing the FPGA configuration

- Reading the FPGA ID code

- Reading the FPGA User ID code

Table 15-7. Slave SPI Pins

Signal Name	Type	Description	Function
SN	Active Low Input	Chip Select	A falling edge on SN causes the FPGA to enter Command State. A rising edge on SN causes the FPGA to enter Reset State.
SI	Input	Serial Input Data	Serial input for command and data.
SO	Tri-state Output	Serial Output Data	Serial output data to the SPI Master.
SCK	Clock Input	Serial Data Clock	Serial input clock for command and data.
HOLDN	Asynchronous Active Low Input	Put the Device on Hold	Tri-state SO and set the device into the suspension state by ignoring the CCLK. Do not assert when loading encrypted bitstreams.

The chip select pin (SN) is a chip select input to the FPGA. The LatticeECP3 responds on the falling and rising edges of the SN input. SN is not a level sensitive input. When the SN falling edge occurs the FPGA is ready to accept commands from a SPI Master Controller. A rising edge on the SN input resets the internal state machine and tri-states the SO output pin. The only exception to this is when the FPGA has received a SPI_PROGRAM command. This command can only be interrupted by the assertion of the PROGRAMN input.

The HOLDN pin is provided to allow a SPI Master Controller to pause transactions on the Slave SPI port. When HOLDN is asserted low the FPGA tri-states the SO output and ignores the SCLK input. This allows the SPI Master Controller to interact with another SPI device and then resume transactions to the LatticeECP3. Encrypted bitstreams must be sent without interruption. You are not permitted to assert HOLDN or deassert SN once an encrypted bitstream transmission has begun.

Figure 15-7. Slave SPI Example
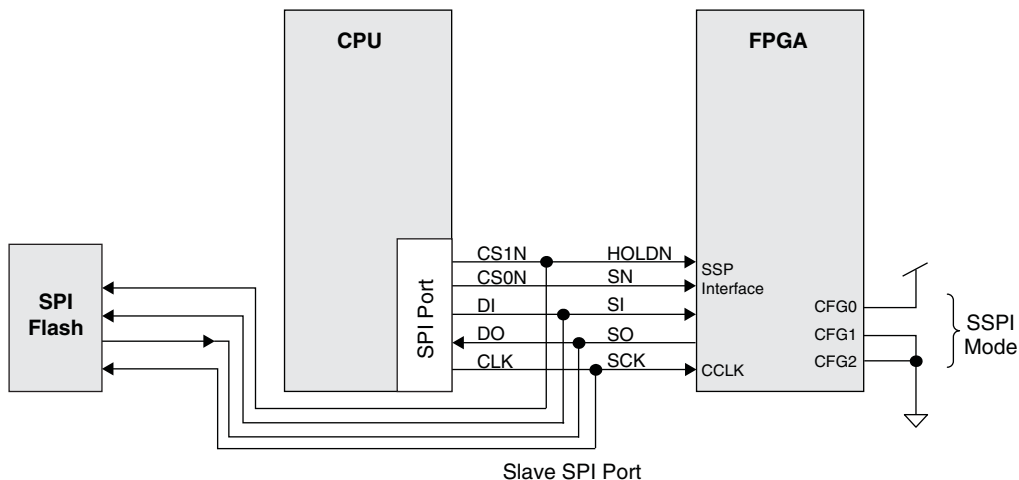
Figure 15-7 illustrates how an on-board CPU can be connected to the LatticeECP3 using the Slave SPI programming interface. The CPU can fetch configuration data from the attached SPI PROM. The CPU is not required to de-assert the SN input to the FPGA. When the CPU asserts the CS1N to access the SPI PROM the FPGA HOLDN is asserted causing it to ignore SCLK transitions. The HOLDN input can not be asserted when transferring an encrypted bitstream.

Full details on using Slave SPI mode on the LatticeECP3 are provided in TN1222, LatticeECP3 Slave SPI Port User's Guide.

Slave Serial Configuration Mode (SCM)

Slave Serial Configuration mode provides a simple, low pin count method for configuring one or more FPGAs. Data is presented to the FPGA on the Data Input pin DI at every CCLK rising edge.

Figure 15-8. Slave Serial Block Diagram
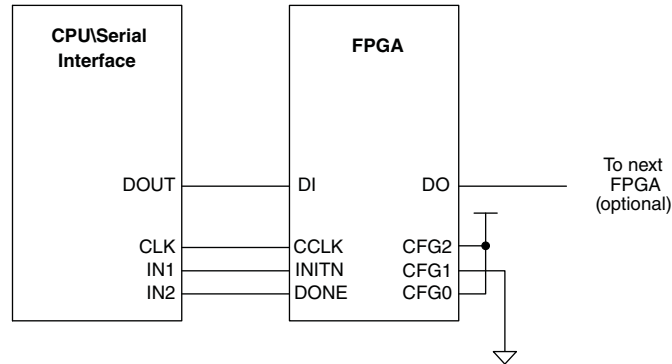
The bitstream data generated by Lattice Diamond is formatted so that it is ready to shift into the FPGA. Left shift the data out of the file in order for it to be correctly received by the FPGA.

The FPGA synchronizes itself on either a 0xBDB3 or 0xBFB3 code word. It is critical that any data presented on DIN not be recognized as one of these two synchronization words early. To guarantee proper recognition of the synchronization word it is recommended that the synchronization word always be preceded by a minimum of 128 '1' bits. Presenting any other bitstream data, Programmer generated header information for example, risks the being misinterpreted due to bit slippage.

Slave Serial Configuration Mode can be used to configure a chain of FPGAs. Details about configuring a chain of devices is discussed in of this document.

Slave Parallel Mode (SPCM)

The LatticeECP3 can be configured using Slave Parallel Configuration Mode. Slave Parallel permits an external master to configure the FPGA using an 8-bit synchronous SRAM bus interface. Slave Parallel Configuration Mode is a flexible method for configuring one or more FPGAs. It is also the fastest mode available for configuring the LatticeECP3.

The slave parallel interface allows for a multitude of different functions to be performed:

- Configuration of the FPGA
- Readback of the FPGA configuration bitstream
- Reading the device CRC
- Reading the programming status register
- Reading the FPGA ID code
- Reading the FPGA User ID code
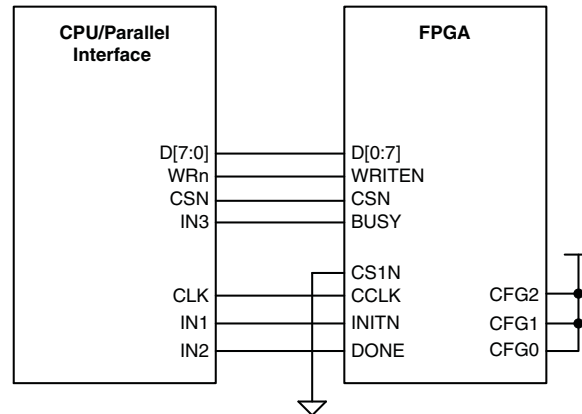
Figure 15-9. Slave Parallel Block Diagram

Figure 15-9 shows the typical Slave Parallel Configuration Mode usage. Configuration data can be written to the LatticeECP3 immediately following the INITN rising edge. The LatticeECP3 data bus bit ordering is denoted using a big-endian nomenclature. This means that D0 receives the MSBit, and D7 receives the LSBit. One byte of data can be sent to the FPGA on each rising CCLK edge as long as CSN, CS1N, and WRITEN are asserted. When the LatticeECP3 is the only device being configured the FPGA can receive configuration data at the full CCLK rate. The master device is not required to monitor the BUSY pin in this situation, because the configuration bitstreams are padded to avoid BUSY assertion.

Sending an encrypted bitstream must be done atomically, i.e. without interruption. The bus master is not permitted to pause the transfer of an encrypted bitstream by deasserting the CSN or CS1N inputs. The CCLK pin can be stretched or stopped if desired, but the CSN and CS1N pins must remain asserted.

Slave Parallel mode can also be to read status registers and the configuration bitstream. In order for the Slave Parallel port to be used to perform read operations the FPGA must have the PERSISTENT preference set to SLAVE_PARALLEL mode. See the Configuration Pin Management section of this document for more information.

Figure 15-10. Parallel Port Write Timing Diagram
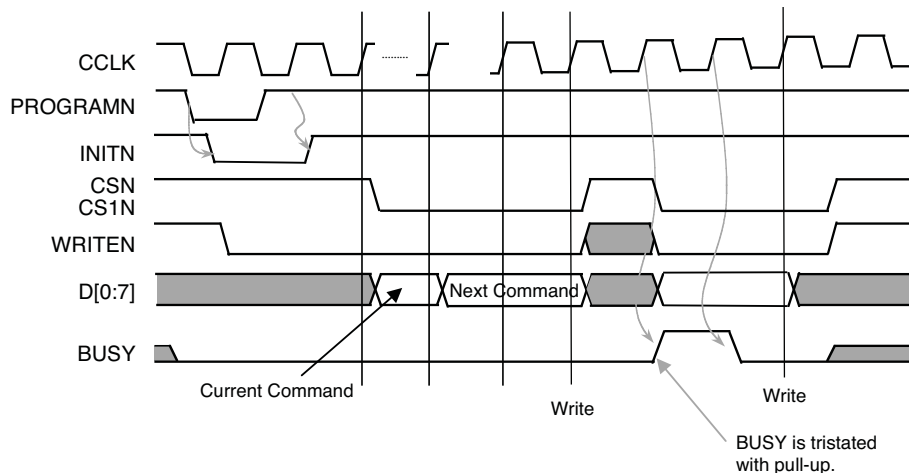
Note: The BUSY pin cannot go high while both CS1N and CSN are low. The second BUSY high shown is OK since CS1N or CSN was low previously.

JTAG Mode (IEEE 1149.1 and IEEE 1532)

The LatticeECP3 provides an IEEE 1532 compliant interface. The IEEE 1532 specification, a superset of the IEEE 1149.1 JTAG specification, describes a standard methodology for configuring programmable logic devices. The LatticeECP3 only requires the four IEEE 1149.1 control signals (TCK, TMS, TDI, and TDO) in order to initiate and complete programming operations. The LatticeECP3 JTAG port is always available for use, regardless of the configuration mode selected.

Programming the LatticeECP3 using the JTAG port is typically accomplished in one of several ways:

- You can use Lattice Diamond Programmer software in combination with a Lattice download cable

- You can use Automatic Test Equipment that can read Serial Vector Format (SVF), In-System Configuration (ISC), STAPL/JAM, or ATE vector files

- You can use an embedded microprocessor to run Lattice's ispVM Embedded configuration software

Lattice Diamond Programmer's Fast Program
The Lattice Diamond development tools translate your design into a bitstream containing an optional header, mandatory preamble, and the device configuration data. The configuration data includes its own preamble, fuse data, and finally a trailing CRC. This basic structure is used for all of the configuration modes supported by the LatticeECP3. The IEEE 1532 mode adds some additional operations to the device configuration process.

Prior to sending the configuration data the FPGA's Boundary Scan I/O Cells are placed in a high-impedance state, and the FPGA's configuration memory is cleared. Because the I/O are tri-stated the DONE and INITn output signals do not provide status information while the configuration data is being written to the FPGA. The JTAG configuration mode uses an internal status register to confirm the FPGA DONE and INITn status signals indicate the device configured correctly. After the internal DONE and INITn controls are confirmed, the Boundary Scan I/O Cells are re-enabled, and all I/O take on the function assigned to them.

The JTAG interface, because it can control the Boundary Scan I/O Cells, can also be used to configure the LatticeECP3 without putting the I/O into a high-impedance state. During device configuration the I/O cells can be locked in their last known active state. This mode of operation is called TransFR Programming. A full description of how to use TransFR is provided in TN1087, [Minimizing System Interruption During Configuration Using TransFR Technology](#).

JTAG Configuration Data Read and Save
The JTAG interface can be used to read the configuration data stored in the FPGA's SRAM array. There are two modes available to retrieve the data, foreground mode or background mode.

Foreground readback is accessed using IEEE 1532 mode. When using this method the JTAG Boundary Scan Cells are placed in a high-impedance state, and the configuration data read. Once the configuration data is retrieved the Boundary Scan Cells are restored, and the FPGA returns to normal operation.

The Background Read and Save operation allows you to read the content of the device while the device remains in operation. All I/O, as well at the non-JTAG configuration pins, continue normal operation during the Background Read and Save operation. You must not violate the following conditions when using the Background Read and Save function:

- The Soft Error Detection system must not be running. De-assert the SEDENABLE pin to prevent the SED circuit from interfering with the Background Read and Save operation. It is recommended that you wait at least one full SEDSTART to SEDDONE time period after the deassertion of the SEDENABLE to make sure the SED circuit has discontinued operation. Alternately monitor the SEDINPROG output, and wait for it to de-assert.

- Write operations to distributed RAM blocks must be suspended. Write operations that occur at the same time the SRAM cell is being read are non-deterministic. It is possible for the SRAM to receive, or retain, incorrect RAM data.

Regardless of which read and save mode is used the configuration data will not include the EBR or the distributed RAM contents. Distributed RAM contents will always be return zeroes.

Boundary Scan and Boundary Scan Description Language (BSDL) Files
The LatticeECP3, as mentioned previously, provides an IEEE 1149.1 compliant JTAG interface. The JTAG interface can be controlled by Automatic Test Equipment (ATE) that uses Boundary Scan Description Language (BSDL) files. Lattice makes BSDL files available for the LatticeECP3 on the Lattice Semiconductor website.

The boundary scan ring covers all of the I/O pins, as well as the dedicated and dual-purpose sysCONFIG pins. Note that PROGRAMN, CCLK, and the CFG pins are observe only (BC4, JTAG read-only) boundary scan cells.

When performing JTAG 1149.1 EXTEST instructions, the SERDES CML termination for both Tx and Rx is set to 50 ohm pull-ups. This allows the high-speed channels to operate properly if DC data is sent or received. During JTAG EXTEST, the termination will be set to 50 ohm. This overrides the termination resistance programmed into the SERDES logic.

Bitstream Generation Software Usage

This section describes the settings for bitstream generation performed by the Diamond software program that generates a bitstream. These options are controlled through the Global Preferences of the Diamond Spreadsheet View and the property settings of the Bit Generation Software tool. To set the Global Preferences and properties in Diamond, see Appendix B. By setting the proper parameter in the Lattice design software the selected configuration options are set in the generated bitstream. As the bitstream is loaded into the device the selected configuration options take effect. These options are described in the following sections.

Bit Generation takes a fully routed Physical Design (.ncd file) as input and produces a configuration bitstream (bit images). The bitstream file contains all of the configuration information from the Physical Design defining the internal logic and interconnections of the FPGA, as well as device-specific information from other files associated with the target device. The data in the bitstream can then be downloaded directly into the FPGA memory cells or used to generate files for PROM programming (using a separate program, ispVM). Please refer to the ispVM documentation for details on creating PROM image files.

Configuration Options

Several configuration options are available for each configuration mode. These options are controlled from the Spreadsheet View for each Strategy. They include the following items.

- When daisy chaining multiple FPGA devices an overflow option is provided for serial and parallel configuration modes
- When using SPI or SPIm mode, the master clock frequency can be set
- A security bit can be set to prevent SRAM readback
- The Persistent option can be set
- Configuration pins can be protected
- DONE pin options can be selected

By setting the proper parameter in the Lattice design software the selected configuration options are set in the generated bitstream. As the bitstream is loaded into the device the selected configuration options take effect. These options are described in the following sections.

Master Clock
If the CFG pins indicate an SPI or SPIm mode, the MCLK pin will become an output with a default frequency, or one selected when you added preferences to your design. The default Master Clock Frequency is 2.5 MHz. For a complete list of the supported Master Clock frequencies, please see the [LatticeECP3 Family Data Sheet](#). When using the LatticeECP3 devices, the available frequencies are restricted, as shown in the data sheet.

You can change the Master Clock frequency by setting the MCCLK_FREQ global preference in the Diamond Spreadsheet view tool. During configuration one of the first pieces of information loaded is the MCCLK_FREQ parameter. When this parameter is loaded the master clock frequency changes to the selected value without glitching. Care should be exercised not to exceed the frequency specification of the slave devices or the signal integrity capabilities of the PCB layout.

Configuration time is computed by dividing the maximum number of configuration bits, as given in Table 15-5 above, by the Master Clock frequency.

Table 15-8. Selectable Master Clock (MCCLK) Frequencies During Configuration (Nominal)

MCCLK (MHz)	MCCLK (MHz)
2.5[1]	10
	13
4.3	15[2]
5.4	20
6.9	26
8.1	
9.2	33[3]

1. Software default MCLK frequency. Hardware default is 3.1MHz.
2. Maximum MCCLK with encryption enabled.
3. Maximum MCCLK without encryption

Security Bit

Setting the CONFIG_SECURE option to ON prevents readback of the SRAM from JTAG or the sysCONFIG pins. When CONFIG_SECURE is set to ON the only operations available are erase and write. The security control bit is updated as the last operation of SRAM configuration. If a secured device is read it will output all ones.

For LatticeECP3 devices the CONFIG_SECURE option is accessed via the Design Planner in ispLEVER. To set this option in Diamond, see Appendix B. The default is OFF.

Persistent Option

The PERSISTENT option is used to direct the place and route tools about how it can use the sysCONFIG pins. By default the PERSISTENT option is turned OFF, which allows the place and route tools to reclaim most of the configuration pins as general purpose input/output. Changing the PERSISTENT configuration option from its default state prevents the place and route tools from either the Slave SPI or the Slave Parallel configuration ports from becoming general purpose I/O.

Enabling the dedicated sysCONFIG ports is useful for performing additional capabilities while the FPGA is in user mode.

You use the SLAVE_PARALLEL setting when:

- You want to read back the FPGAs SRAM contents. The LatticeECP3 provides a command set and access protocol that allows the configuration SRAM to be read from the FPGA. The Slave Parallel port can read all of the configuration data, except the EBR and the distributed RAM contents.

- You have a LatticeECP3, configured as a SPI Master, in series of FPGAs in a device chain. The SPI Master FPGA must keep the MCLK pin active in order to provide a configuration clock for all of the chained FPGAs. Table 15-3 describes the configuration pins that are preserved. The MCLK output is only preserved Slave Parallel configuration mode. If PERSISTENT is set to OFF, or SSPI the MCLK output tri-states after the lead FPGA is configured, which prevents chained FPGAs from configuring.

Use the SSPI PERSISTENT setting when:

- You want to access a SPI PROM attached to the SPI Master configuration pins. You can attach a SPI memory

controller and using a custom command you can perform erase, program, and verify sequences on the SPI PROM while the FPGA is in operation. Table 15-3 describes the configuration pins that are preserved. Information about the Slave SPI transactions are published in TN1222 LatticeECP3 Slave SPI Port User's Guide. You can also use the SSPI Embedded device programming software provided by Lattice.

Configuration Mode

The CONFIG_MODE option tells the software which configuration port the hardware is using to program the FPGA. Setting this parameter permits the design software to check to make sure configuration port pins are not oversubscribed. The oversubscription is only flagged as a warning. In some cases it is acceptable to oversubscribe the configuration port. For example it is acceptable to have the FPGA in SPI Master configuration mode and use the SISPI, SPID0,a nd SPICS pins as general purpose I/O.

The CONFIG_MODE is also used to make sure encrypted bitstreams are generated correctly. To guarantee correct operation of encrypted bitstreams you need to set the CONFIG_MODE parameter.

DONE EX

During configuration the DONE output pin is low. Once configuration is complete, indicated by assertion of an internal DONE bit, the device wake-up sequence takes place. The external DONE pin is able to operate in one of two modes during the wake up sequence. The default behavior, set when DONE_EX = OFF, is for it to actively drive to VIL. When DONE_EX is set ON, the external DONE pin becomes an open-drain output. The LatticeECP3 wake up sequence will stall until the external DONE pin is pulled high. Set DONE_EX to ON when you want to synchronize the when a chain of FPGAs wakes up. Make sure you place a pullup resistor that is able to drive all of the DONE pins.

Device Wake-Up

When configuration is complete the device will wake up in a predictable fashion. Wake-Up occurs after successful configuration, without errors, and provides the transition from Configuration Mode to User Mode. The Wake-Up process begins when the internal DONE bit is set.

Table 15-9 provides a list of the Wake-Up sequences supported by the LatticeECP3; Figure 15-11 shows the Wake-Up timing. The default Wake-Up sequence works fine for most single device applications.

Table 15-9. Wake-Up Options

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
1	DONE	GOE, GWDIS, GSR		
2	DONE		GOE, GWDIS, GSR	
3	DONE			GOE, GWDIS, GSR
4[1]	DONE	GOE	GWDIS, GSR	
5	DONE	GOE		GWDIS, GSR
6	DONE	GOE	GWDIS	GSR
7	DONE	GOE	GSR	GWDIS
8		DONE	GOE, GWDIS, GSR	
9		DONE		GOE, GWDIS, GSR
10		DONE	GWDIS, GSR	GOE
11		DONE	GOE	GWDIS, GSR
12			DONE	GOE, GWDIS, GSR
13		GOE, GWDIS, GSR	DONE	
14		GOE	DONE	GWDIS, GSR
15		GOE, GWDIS	DONE	GSR
16		GWDIS	DONE	GOE, GSR
17		GWDIS, GSR	DONE	GOE

Table 15-9. Wake-Up Options (Continued)

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
18		GOE, GSR	DONE	GWDIS
19			GOE, GWDIS, GSR	DONE
20		GOE, GWDIS, GSR		DONE
21[2]		GOE	GWDIS, GSR	DONE
22		GOE, GWDIS	GSR	DONE
23		GWDIS	GOE, GSR	DONE
24		GWDIS, GSR	GOE	DONE
25		GOE, GSR	GWDIS	DONE

1. Default when DONE_EX=ON.
2. Default when DONE_EX=OFF.

Figure 15-11. Wake-Up Timing Diagram
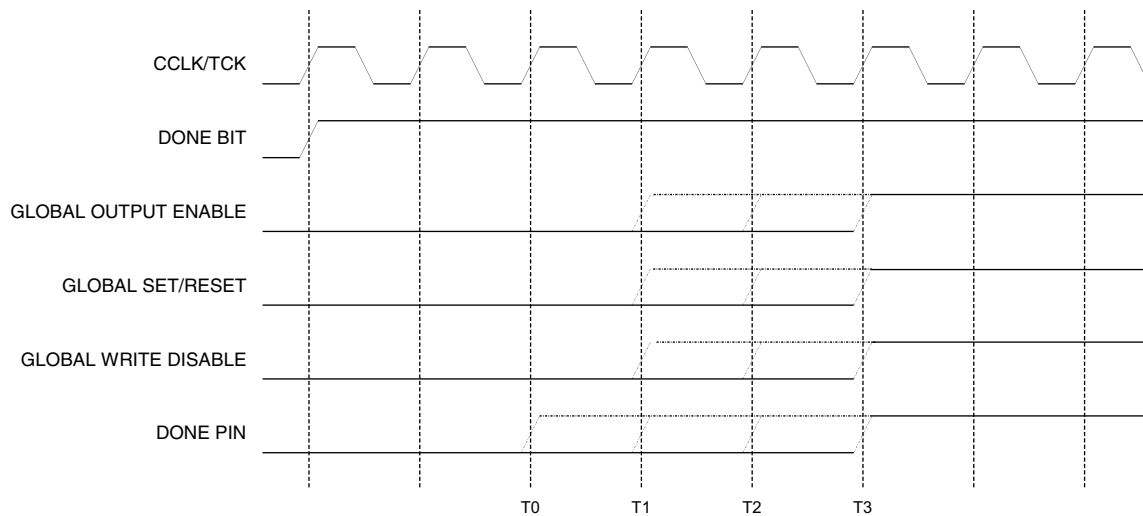
Synchronizing Wake-Up

The LatticeECP3 is, in most cases, configured using one of the master configuration modes. The FPGA, when in master configuration mode, is driving the configuration clock. The configuration clock is used for stepping through the final four Wake-Up states described in the previous section.

The LatticeECP3 has the ability to use an external clock source to control the final state transitions in the Wake_Up process. There are three possible sources for the clock. The JTAG TCK, the Slave Configuration CCLK, and a general-purpose input.

Start_Up Clock Selection
Once the FPGA is configured, it enters the start-up state, which is the transition between the configuration and operational states. This sequence is synchronized to a clock source, which defaults to CCLK when a slave configuration mode is used, or TCK when JTAG is used.

If desired, a user-defined clock source can be used instead of CCLK/TCK. You need to specify this clock signal, and instantiate the STRTUP library element in your design. The example shown below shows the proper syntax of instantiating the STRTUP library element.

Verilog
```
  STRTUP u1 (.UCLK(<clock_name>)) /* synthesis syn_noprune=1 */;
```

VHDL
```
  component STRTUP
     port(STRTUP: in STD_ULOGIC );
  end component;

  attribute syn_noprune: boolean ;
  attribute syn_noprune of STRTUP: component is true;
  begin
     u1: STRTUP port map (UCLK =><clock name>);
```

Synchronous to Internal DONE Bit
If the LatticeECP3 is the only device in the configuration chain, or the last device in the chain, DONE_EX should be set to the default value (OFF). The Wake-Up process will be initiated by setting of the internal DONE bit on successful completion of configuration.

Synchronous to External DONE Pin
The DONE pin can be used to synchronize Wake-Up to other devices in a configuration chain. If DONE_EX (see the DONE EX section above) is ON then the DONE pin is an open-drain bi-directional pin. If an external device drives the DONE pin low then the Wake-Up sequence stalls until DONE is active high. Once the DONE pin goes high the device will follow the selected WAKE_UP sequence.

In a configuration chain, a chain of devices configuring from one source (such as Figure 15-17), it is usually desirable, or even necessary, to delay wake-up of all of the devices until the last device finishes configuration. This is accomplished by setting DONE_EX to OFF on the last device while setting DONE_EX to ON for the other devices.

Wake-up Sequence Options
The wake-up sequence options determine the order of application for three internal signals, GSR, GWDIS, and GOE, and one external signal, DONE.

• GSR is used to set and reset the core of the device. GSR is asserted (low) during configuration and de-asserted (high) in the Wake-Up sequence.

• When the GWDIS signal is low it safeguards the integrity of the RAM Blocks and LUTs in the device. This signal is low before the device wakes up. The GWDIS signal is internal to the FPGA, and does not appear on any FPGA I/O. During the time it is driven low all EBR and LUT RAM elements are safe from being modified.

• During initialization and configuration the FPGA I/O are placed in a high impedance state. The GOE control controls when the FPGA I/O leave the high impedance state. The I/O are Hi-Z when GOE is asserted low.

• The DONE pin, when high, indicates that FPGA has completed configuration and is in user mode. DONE will only be high if DONE_EX=ON, the output driver is released, and the external pin is pulled up.

If DONE_EX (see DONE EX above) is OFF then sequence 21 is the default, but you can select any sequence from 8 to 25; if DONE_EX is ON the default sequence is 4, but you can select any sequence from 1 to 7.

WAKE_ON_LOCK
The wake-up sequence can be delayed until the selected PLLs have a chance to lock. The WAKE_ON_LOCK attribute selects which PLLs will delay the wake-up sequence until the PLL locks. If you choose an external signal for PLL feedback rather than an internal clock signal, wake-up must occur without waiting for PLL lock because all I/Os are tri-stated until the device wakes up, preventing the PLL from locking.

Using the default mode of operation, the device PLLs do not have to be locked for wake-up to commence. You can choose to make the wake-up sequence dependent on any of the PLLs. If multiple PLLs are included in the design, all PLLs in the design have to be locked to satisfy the wake-up sequence.

Bitstream Generation Property Options

Run DRC (T/F)

When the Run DRC option is set to TRUE, a physical design rule check will be run prior to generating a bitstream. The output will be saved to the Bit Generation report (.bgn file). Running DRC before a bitstream is produced will detect any errors that could cause the FPGA to function improperly. If no fatal errors are detected, it will produce a bitstream file. The default is True and will run DRC. When this option is set to False, a design rule check (DRC) will not be run prior to generating a bitstream.

Create Bitfile (T/F)

This option allows you to decide whether or not to generate an output bitstream. The default setting is to create a bitstream.

Bitstream File Formats

- Bit file (binary)

- Raw bit file

- Mask & Readback file (ASCII)

- Mask & Readback file (binary)

These options allow you to chose the format of a bitstream file. The Raw Bit option causes the Bitstream Generator to create a Raw Bit (.rbt) file instead of a binary file (.bit). A binary .bit file can be viewed with a binary editor.

The Raw Bit File is a text file containing ASCII ones and zeros representing the bits in the bitstream file. If you are using a microprocessor to configure a single FPGA, you can include the Raw Bit file in the source code as a text file to represent the configuration data. The sequence of characters in the Raw Bit file is the same as the bit sequence that will be written into the FPGA. This file is a large file.

A Mask file (.msk) can be generated in either an ASCII formatted file or binary file. The Mask file is used to compare relevant bit locations for executing a readback of configuration data contained in an operating FPGA. You can compare readback data from the device to the mask file after downloading the bitstream. The ASCII mask file will contain 1's and 0's, and X's. The file contains all FPGA data frames. It contains no header, ID frames, address frames and no preloaded frames.

No Header (T/F)

The generated bitstream contains no header. The default will be false and will always produce a bitstream file including all the header information.

Bitstream Encryption/Decryption Flow

The LatticeECP3 supports both encrypted and non-encrypted bitstreams. The encrypted flow adds only two steps to the normal FPGA design flow, encryption of the configuration bitstream and programming the encryption key into the LatticeECP3 devices.

Encrypting the Bitstream

As with any other Lattice FPGA design flow, the engineer must first create the design using a device and version of ispLEVER or Diamond which supports the encryption feature. You must obtain the Encryption Installer from Lattice prior to using Encryption capabilities. The design is synthesized, mapped, placed and routed, and verified. Once the engineer is satisfied with the design a bitstream is created and loaded into the FPGA for final debug. After the design has been debugged it is time to secure the design.

The bitstream can be encrypted using an appropriate version of ispLEVER by going to the Tools pull-down menu and selecting Security features or by using the Universal File Writer (UFW), which is part of the Lattice ispVM™

System tool suite. The file is encrypted using ispVM as follows. To encrypt the bitstream in Diamond, see Appendix B.

Figure 15-12. ispVM Main Window
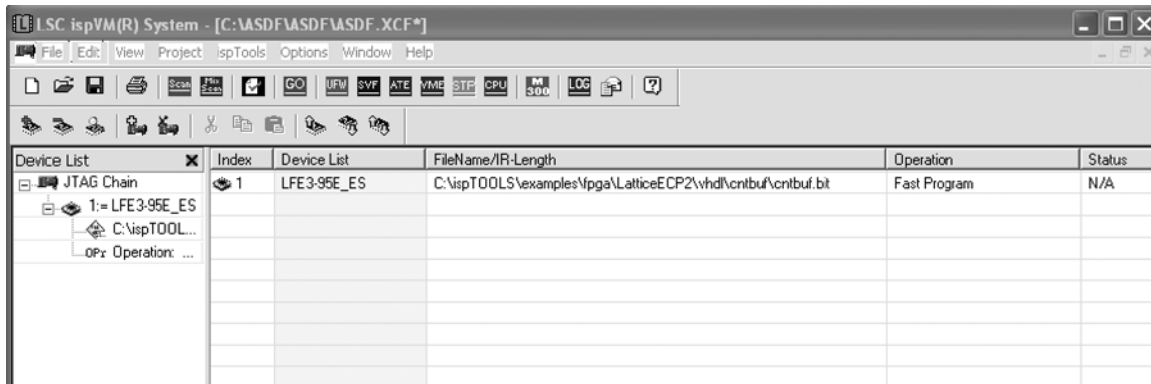
1. Start ispVM. You can start ispVM from the Tools menu in ispLEVER or from the **Start -> Programs** menu in Windows. ispVM is not accessible from the Tools menu in Diamond. You should see a window that looks similar to Figure 15-12. Click on the **UFW** button on the toolbar. You will see a window similar to Figure 15-13.

Figure 15-13. Universal File Writer (Encryption Option)
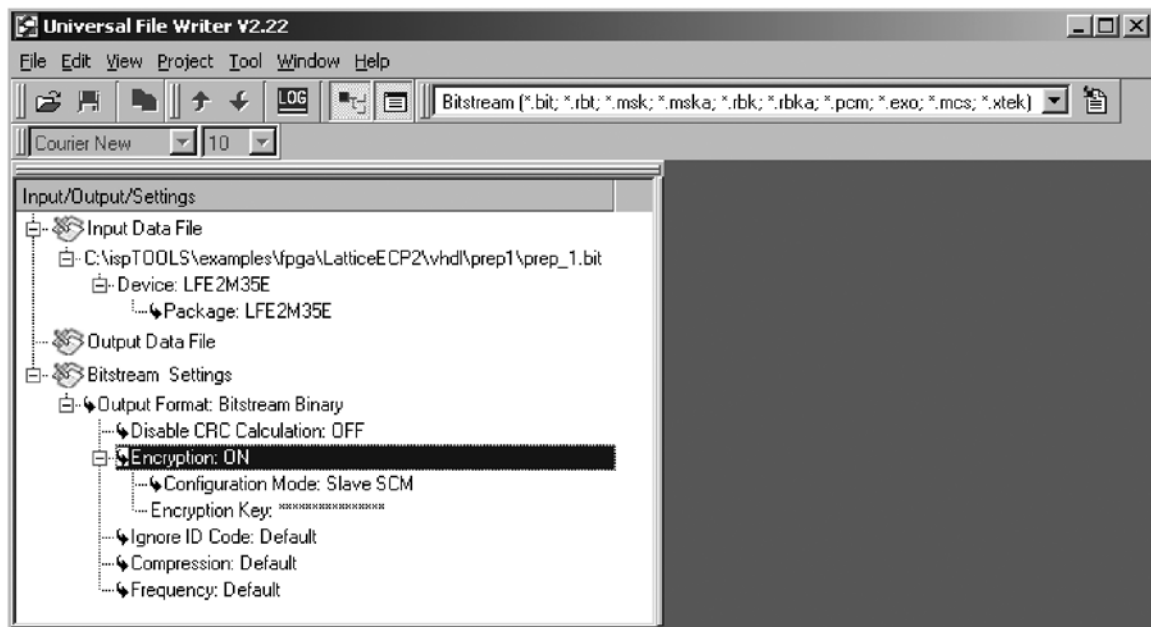
2. Double click on **Input Data File** and browse to the non-encrypted bitstream created using ispLEVER or Diamond. Double-click on **Output Data File** and select an output file name. Right-click on **Encryption** and select **ON**. Right-click on **Configuration Mode** and select the type of device the FPGA will be configuring from, such as SPI Serial Flash. Right-click on **Encryption Key** and select **Edit Encryption Key**. You will see a window that looks similar to Figure 15-14.

Figure 15-14. Encryption Key Dialog Window
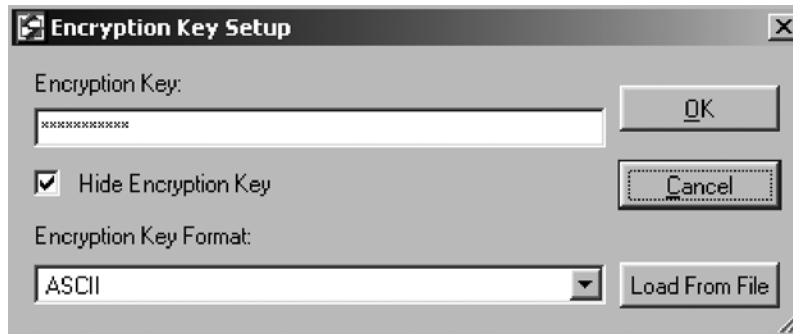
3. Enter the desired 128-bit key. The key can be entered in Hexadecimal or ASCII. Hex supports 0 through f and is not case sensitive. ASCII supports all alphanumeric characters, as well as spaces, and is case sensitive. Note: be sure to remember this key. Lattice cannot recover an encrypted file if the key is lost. Click on **OK** to go back to the main UFW window.

4. From the menu bar, click on **Project -> Generate** to create the encrypted bitstream file.

5. The bitstream can now be loaded directly into non-volatile configuration storage (such as SPI Serial Flash) using a Lattice ispDOWNLOAD® Cable, a third-party programmer, or any other method normally used to program a non-encrypted bitstream. However, before the LatticeECP3 can configure from the encrypted file the 128-bit key used to encrypt the file must be programmed into the one-time programmable cells on the FPGA.

Programming the 128-bit Key

The next step is to program the 128-bit encryption key into the one-time programmable cells on the LatticeECP3. This is done through the device JTAG interface. Note that this step is separated from file encryption to allow flexibility in the manufacturing flow. For instance, the board manufacturer might program the encrypted file into the SPI Serial Flash, but the key might be programmed at your facility. This flow adds to design security and it allows you to control over-building of a design. Over-building occurs when a third party builds more boards than are authorized and sells them to grey market customers. If the key is programmed at the factory, then the factory controls the number of working boards that enter the market. The LatticeECP3 will only configure from a file that has been encrypted with the same 128-bit key that is programmed into the FPGA.

To program the key into the LatticeECP3, proceed as follows.

1. Attach a Lattice ispDOWNLOAD cable from a PC to the JTAG connector wired to the LatticeECP3 (note that the 128-bit key can only be programmed into the LatticeECP3 using the JTAG port). Apply power to the board.

2. Start the ispVM System software. ispVM can be started from within the ispLEVER Tools menu or from the **Start -> Programs** menu in Windows. ispVM cannot be invoked from the Tools menu in Diamond. You should see a window that looks similar to Figure 15-12. If the window does not show the board's JTAG chain then proceed as follows. Otherwise, proceed to step 3.

   a. Click the **SCAN** button in the toolbar to find all Lattice devices in the JTAG chain. The chain shown in Figure 15-12 has only one device, the LatticeECP3.

Figure 15-15. Device Information Window (Encryption Option)
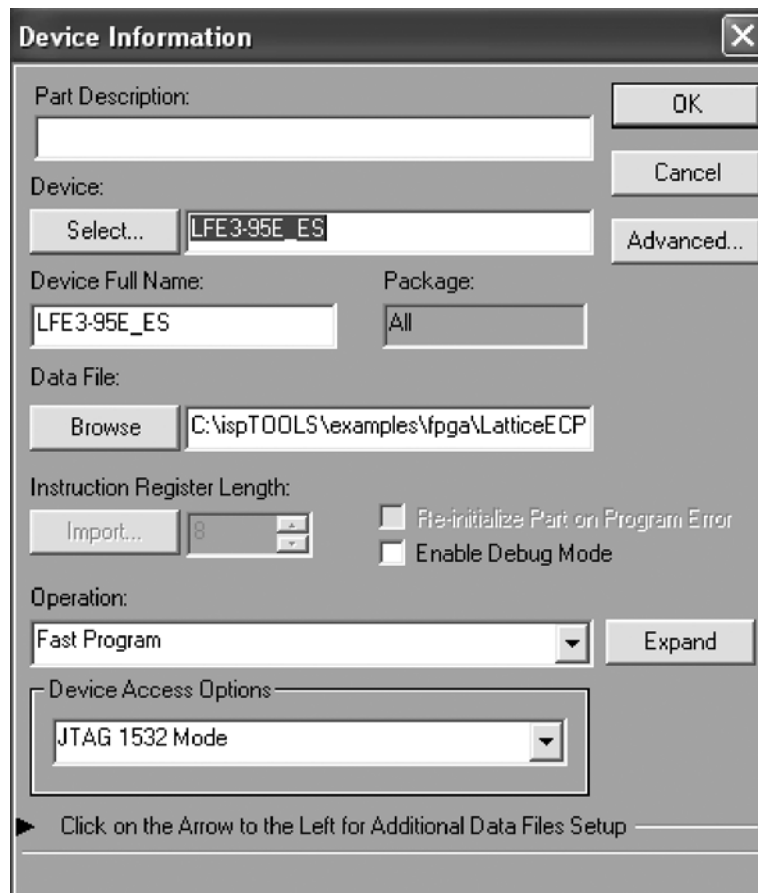
3. Double-click on the line in the chain containing the LatticeECP3. This will open the Device Information window (see Figure 15-17). From the Device Access Options drop-down box select **Security Mode**, then click on the **Security Key** button to the right. The window will look similar to Figure 15-16.

Figure 15-16. Enter the Encryption Key
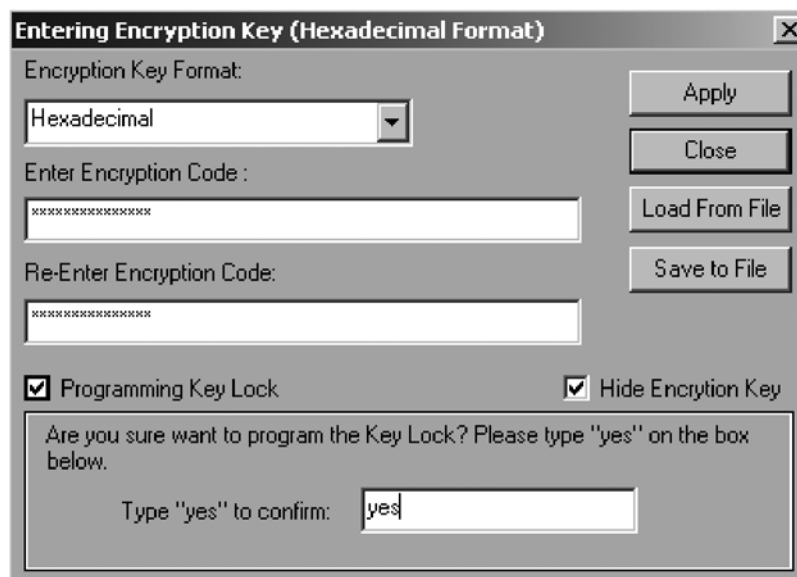
4. Enter the desired 128-bit key. The key can be entered in Hexadecimal or ASCII. Hex supports 0 through f and is not case sensitive. ASCII supports all alphanumeric characters, as well as spaces, and is case sensitive. This key must be the same as the key used to encrypt the bitstream. The LatticeECP3 will only configure from an encrypted file whose encryption key matches the one loaded into the FPGA's one-time programmable cells. *Note: be sure to remember this key. Once the Key Lock is programmed, Lattice Semiconductor cannot read back the one-time programmable key.*

   a. The key can be saved to a file using the **Save to File** button. The key will be encrypted using an 8-character password that you select. The name of the file will be <project_name>.bek. In the future, instead of entering the 128-bit key, simply click on **Load from File** and provide the password.

5. Programming the Key Lock secures the 128-bit encryption key. Once the Key Lock is programmed and the device is power cycled, the 128-bit encryption key cannot be read out of the device. When satisfied, type **Yes** to confirm, then click **Apply**.

6. From the main ispVM window (Figure 15-12) click on the green **GO** button on the toolbar to program the key into the LatticeECP3 one-time programmable cells. When complete, the LatticeECP3 will only configure from a bitstream encrypted with a key that exactly matches the one just programmed.

Verifying a Configuration

As an additional security step when an encrypted bitstream is used, the readback path from the SRAM fabric is automatically blocked. In this case, for all ports, a read operation will produce all 1's. However, even when the configuration bitstream has been encrypted and readback disabled, there are still ways to verify that the bitstream was successfully downloaded into the FPGA.

If the SRAM fabric is programmed directly, the data is first decrypted and then the FPGA performs a cyclic redundancy code (CRC) on the data. (CRC) circuitry is used to validate each configuration data frame (sequence of data bits) as it is loaded into the target device. If all CRCs pass, configuration was successful. If a CRC does not pass, the DONE pin will stay low and INITN will go from high to low.

If the encrypted data is stored in non-volatile configuration memory, such as SPI Serial Flash, the data is stored encrypted. A bit-for-bit verify can be performed between the encrypted configuration file and the stored data.

File Formats

The base binary file format is the same for all non-encrypted, non-1532 configuration modes. Different file types (hex, binary, ASCII, etc.) may ultimately be used to configure the device, but the data in the file is the same. Table 15-10 shows the format of a non-encrypted bitstream. The bitstream consists of a comment field, a header, the preamble, and the configuration setup and data.

Table 15-10. Non-Encrypted Configuration Data

Frame	Contents	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator
Header	1111...1111	16 Dummy bits
	1011110110110011	16-bit Standard Bitstream Preamble (0xBDB3)
Verify ID		64 bits of command and data
Control Register 0		64 bits of command and data
Reset Address		32 bits of command and data
Write Increment		32 bits of command and data
Data 0		Data, 16-bit CRC, and Stop bits
Data 1		Data, 16-bit CRC, and Stop bits
. . .	. . .	. . .
Data n-1		Data, 16-bit CRC, and Stop bits
End	1111...1111	Terminator bits and 16-bit CRC
Usercode		64 bits of command and data
SED CRC		64 bits of command and data
Program Security		32 bits of command and data
Program Done		32 bits of command and data, 16-bit CRC
NOOP	1111...1111	64 bits of NOOP data
End	1111...1111	32-bit Terminator (all ones)

Note: The data in this table is intended for reference only.

Table 15-11 shows a bitstream that is built for encryption but has not yet been encrypted. The highlighted areas will be encrypted. The changes between Table 15-10 and Table 15-11 include the following:

- The Program Security frame (readback disable) has been moved to the beginning of the file so that readback is turned off at the very beginning of configuration. This is an important security feature that prevents someone from interrupting the configuration before completion and reading back unsecured data.

- A copy of the usercode is placed in the non-encrypted comment string. This has been done to allow you a method to identify an encrypted file. For example, the usercode could be used as a file index. Note that the usercode itself, while encrypted in the configuration data file, is not encrypted on the device. At configuration the usercode is decrypted and placed in the JTAG Usercode register. This allows you a method to identify the data in the device. The JTAG Usercode register can be read back at any time, even when all SRAM readback paths have been turned off. The usercode can be set to any 32-bit value. For information on how to set usercode, see the ispLEVER or Diamond help facility.

- A copy of CONFIG_MODE, one of the global preferences, is placed in the non-encrypted comment string. CONFIG_MODE can be SPI/SPIm, SSPI, Slave SCM, Slave PCM, Master PCM, or JTAG.

Table 15-11. Configuration File Just Before Encryption

Frame	Contents	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator
Header	1111...1111	16 Dummy Bits
		16-bit Standard Bitstream Preamble
Verify ID		64 bits of Command and Data
Control Register 0		64 bits of Command and Data
Program Security		32 bits of Command and Data
Reset Address		32 bits of Command and Data
Write Increment		32 bits of Command and Data
Data 0		Data, 16-bit CRC, and Stop Bits
Data 1		Data, 16-bit CRC, and Stop Bits
. . .	. . .	. . .
Data n-1		Data, 16-bit CRC and Stop Bits
End	1111...1111	Terminator Bits and 16-bit CRC
Usercode		64 Bits of Command and Data
SED CRC		64 Bits of Command and Data
Program Done		32 Bits of Command and Data, 16-bit CRC
NOOP	1111...1111	64 bits of NOOP data
End	1111...1111	32-bit Terminator (All Ones).

Note: The data in this table is intended for reference only. The shaded areas will be encrypted.

Once encrypted, besides the obvious encryption of the data itself, the file will have additional differences from a non-encrypted file (refer to Tables 15-12, 15-13, and 15-14).

• There are three preambles, the encryption preamble, alignment preamble, and the bitstream preamble. The alignment preamble marks the beginning of the encrypted data. The entire original bitstream, including the bitstream preamble are all encrypted, per Table 15-11. The comment string, the encryption preamble, dummy data, and alignment preamble are not encrypted.

• The decryption engine within the FPGA takes some time to perform its task; extra time is provided in one of two ways. For master configuration modes (SPI and SPIm) the FPGA drives the configuration clock, so when extra time is needed the FPGA stops sending configuration clocks. For slave configuration modes (Bitstream-Burst, Slave Serial, and Slave Parallel) the data must be padded to create the extra time. Because of this there are several different file formats for encrypted data (see Tables 15-12, 15-13, and 15-14). Note that because of the time needed to decrypt the bitstream it takes longer to configure from an encrypted data file than it does from a non-encrypted file. The bitstream sizes may vary depending on the configuration mode.

Table 15-12. LatticeECP3 Master Serial Encrypted Bitstream File Format

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator[1].
Header	(LSB) 1111…1111	2 Dummy Bytes.
	1011111110110011	16-bit Encryption Preamble (0xBFB3).
30,000 Filler Bits	11…1111	30,000 bits of filler data (all 1's). Ignored by device.
Key Expansion Preamble	1011101010110011	16-bit Key Expansion Preamble (0xBAB3).
240 Filler Bits	11…1111	240 bits of data (all 1's). Ignored by device.
Alignment Preamble	1011110010110011	16-bit Alignment Preamble (0xBCB3).
	1	1-bit Dummy Data.
Encrypted Configuration Data	Encrypted Data[2]	There are no dummy filler bits when the bitstream is generated for master programming mode. The CCLK of the master device stops the clock when it needs time to decrypt the data. It resumes the clock when ready for new data.[2]
Encrypted Frame Program Done	32 Bits Encrypted	32-bit Program Done Command – Encrypted.
Frame (End)	32-bit Encrypted Terminator	32-bit Terminator (all ones) – Encrypted.
Filler Bits	11…1111	Insert filler to meet the 128-bit bound requirement (all 1's).
Dummy Data	111…1111	207-bit Dummy Data (all ones), to provide delay to turn off the decryption engine.

1. Two new fields, Usercode and Config Mode, have been added to the comment string for the ECP2 family encrypted bitstreams.
2. If bitstream compression is selected, the bitstream must be compressed prior to encryption.

Table 15-13. LatticeECP3 Slave Serial, JTAG Burst, and Slave SPI Encrypted Bitstream File Format

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator[1].
Header	(LSB) 1111…1111	2 Dummy Bytes.
	1011111110110011	16-bit Encryption Preamble (0xBFB3).
30,000 Filler Bits	11…1111	30,000 bits of filler data (all 1's). Ignored by device.
Key Expansion Preamble	1011101010110011	16-bit Key Expansion Preamble (0xBAB3).
240 Filler Bits	11…1111	240 bits of data (all 1's). Ignored by device.
Alignment Preamble	1011110010110011	16-bit Alignment Preamble (0xBCB3).
	1	1-bit Dummy Data.
Configuration Data Frames	128 Bits of Encrypted Data	128-bit Configuration Data.[2]
	64 Bits of Filler Data	64 bits of filler data (all 1's). Provide delay clocks for the decryption engine to decrypt the 128 bits of data just received. If the peripheral device can provide 64 extra clocks, then the 64 bits of dummy data is not required.
	………………	
	128 Bits of Encrypted Data	Last 128 bits of the last Frame of the Configuration data.
	64 Bits of Filler Data	64 bits of filler data (all 1's). Delay to decrypt the last 128 bits of Configuration data.
Encrypted Frame Program Done	32 Bits Encrypted	32-bit Program Done Command – Encrypted.
Frame (End)	32-bit Encrypted Terminator	32-bit Terminator (all ones) – Encrypted.
Filler Bits	11…1111	Insert filler to meet the 128-bit bound requirement (all 1's).
	64 Bits of Filler Data	64 bits of filler data (all 1's). Delay to decrypt the Program Done command and the filler.
Dummy Data	111…1111	207-bit Dummy Data (all ones), to provide delay to turn off the decryption engine.

1. Two new fields, Usercode and Config Mode, have been added to the comment string for the ECP2 family encrypted bitstreams.
2. If bitstream compression is selected, the bitstream must be compressed prior to encryption.

Table 15-14. LatticeECP3 Master Parallel Encrypted Bitstream File Format

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator[1].
Header	(LSB) 1111…1111	2 Dummy Bytes.
	1011111110110011	16-bit Encryption Preamble (0xBFB3).
240,000 Filler Bits	11…1111	240,000 bits of filler data (all 1's). Ignored by device.
Key Expansion Preamble	1011101010110011	16-bit Key Expansion Preamble (0xBAB3).
2032 Filler Bits	11…1111	240 bits of data (all 1's). Ignored by device.
Alignment Preamble	1011101010110011	16-bit Alignment Preamble (0xBCB3).
	11111111	8-bit Dummy Data.
Encrypted Configuration Data	Encrypted Data[2]	There are no dummy filler bits when the bitstream is generated for master programming mode. The CCLK of the master device stops the clock when it needs time to decrypt the data. It resumes the clock when ready for new data.[2]
Encrypted Frame Program Done	32 Bits Encrypted	32-bit Program Done Command – Encrypted.
Frame (End)	32-bit Encrypted Terminator	32-bit Terminator (all ones) – Encrypted.
Filler Bits	11…1111	Insert filler to meet the 128-bit bound requirement (all 1's).
Dummy Data	111…1111	1600-bit Dummy Data (all ones), to provide delay to turn off the decryption engine.

1. Two new fields, Usercode and Config Mode, have been added to the comment string for the ECP2 family encrypted bitstreams.
2. If bitstream compression is selected, the bitstream must be compressed prior to encryption.

Decryption Flow

Compared to the encryption flow just discussed, the decryption flow is much simpler.

When data comes into the FPGA the decoder starts looking for the preamble and all information before the preamble is ignored. The preamble determines the path of the configuration data.

If the decoder detects a standard bitstream preamble in the bitstream it knows that this is a non-encrypted data file. The decoder then selects the Raw data path.

If the decoder detects an encryption preamble in the bitstream it knows that this is an encrypted data file. If an encryption key has not been programmed, the encrypted data is blocked and configuration fails (the DONE pin stays low), if the proper key has been programmed then configuration can continue. The next block read contains 30,000 clocks of filler data. This delay allows time for the FPGA to read the key cells and prepare the decryption engine. The decoder keeps reading the filler data looking for the alignment preamble. Once found, it knows that the following data needs to go through the decryption engine. It first looks for the standard preamble. Once found, then the SRAM cells' programming begins.

But what happens if the key in the FPGA does not match the key used to encrypt the file? Once the data is decrypted, the FPGA expects to find a valid standard bitstream preamble (BDB3), along with proper commands and data that pass CRC checks. If the keys do not match then the decryption engine will not produce a proper configuration bitstream; either configuration will not start because the preamble was not found (the INITN pin stays high and the DONE pin stays low) or CRC errors will occur, causing the INITN pin to go low to indicate the error.

Combining Configuration Modes

Multiple FPGAs, One SPI Flash

With a sufficiently large SPI Flash, multiple FPGAs can be configured as shown in Figure 15-17. The first FPGA is configured in SPI mode; the following FPGAs are configured in Slave Serial mode. Full details on using SPI mode combined with Slave Serial mode are provided in TN1222 LatticeECP3 Slave SPI Port User's Guide.

Figure 15-17. Multiple FPGAs, One SPI Serial Flash
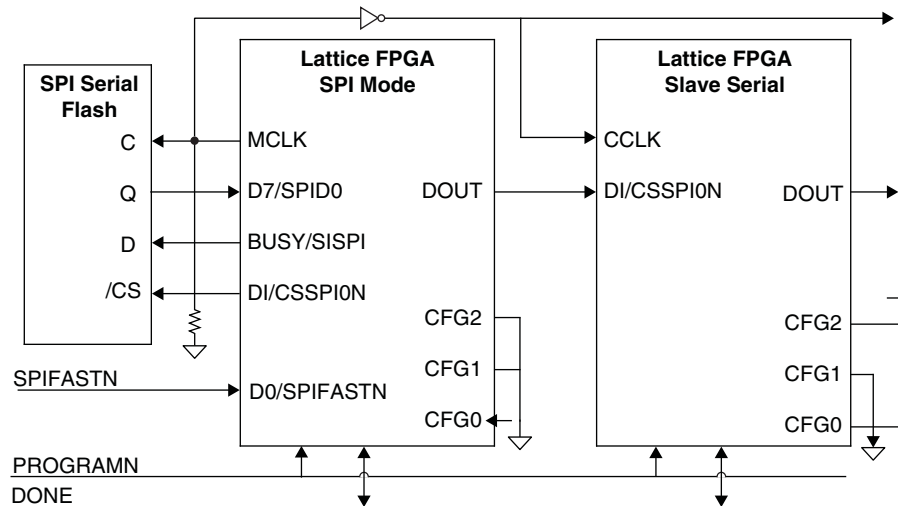
Note: The inverter for MCLK guarantees the hold time for data input to the daisy chained FPGAs.

Figure 15-18. Slave SPI Example 1
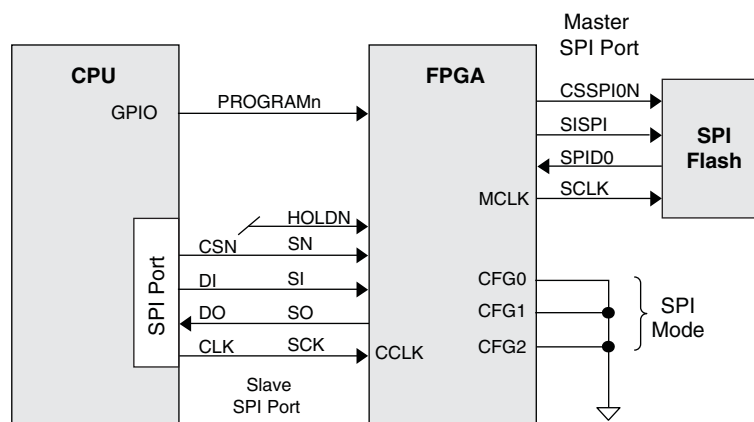
The system diagram shown in Figure illustrates one application of the Slave SPI interface, where the FPGA selects the SPI Flash as the primary boot source. The modern CPU has the capability to program the SPI Flash boot PROM as well as to command the FPGA to re-boot from the SPI Flash by toggling the PROGRAMN pin. This requirement can only be met if the CPU drives the CCLK, and the MCLK is driven is by the FPGA for the SPI Flash boot PROM as shown in Figure .

CS1N/HOLDN: When Slave SPI mode is used, this pin is an asynchronous active Low Input that tri-states the serial read out data of the SPI port and sets the device to the suspend state by ignoring the clock. Set the SSPI PERSISTENT to on to retain the pin as HOLDN pin to access the Slave SPI port in user mode.

Figure 15-19. Slave Parallel with Flowthrough

Chain Mode Options

The LatticeECP3 can be one of many FPGAs in a chain that each need to get configuration data. The Bypass and Flowthrough options control how each FPGA in the chain of devices pass configuration bits to the other devices in the chain. Successful configuration of a chain of FPGAs depends on a thorough understanding of the Bypass and Flowthrough features.

Bypass Option

This option is used when you are configuring a chain of FPGAs in either parallel or serial daisy chain configurations. The Bypass option, when enabled, adds an additional command to the end of the configuration bitstream being sent to the LatticeECP3. The LatticeECP3 receives all of the configuration bits, and upon reception of the BYPASS command it enables a serial bypass register. This bypass register passes all incoming configuration bits to the DOUT pin for use by the next FPGA in the chain. Prior to the LatticeECP3 receiving the BYPASS command the internal bypass register is initialized to '1'. Any FPGA receiving data from the DOUT pin will see a long string of ones until the BYPASS command is accepted by the LatticeECP3.

The following conditions must be met when Bypass is enabled:

- The PERSISTENT option must be set to Slave Parallel mode

- The bitstream can not be encrypted

- The LatticeECP3 can not be in SPIm mode

The Bypass Option is DISABLED by default.

Flowthrough Option

The Flowthrough option is used when you are configuring a chain of FPGAs in Slave Parallel Configuration mode. It is not applicable to any other configuration mode. The Flowthrough option, when enabled, adds a FLOWTHROUGH command to the end of the bitstream for the LatticeECP3. The LatticeECP3 receives all of the configuration data over the Slave Parallel data bus. When it receives the FLOWTHROUGH command it asserts the CSON output pin driving the parallel bus chip select input of the next FPGA in the chain. Until reception of the

FLOWTHROUGH command the CSON pin is deasserted high, which prevents any downstream FPGA from loading the incoming data bytes.

The following conditions must be met to use the Flowthrough option:

- The PERSISTENT option must be set to Slave Parallel

- The bitstream can not be encrypted

- The CONFIG_MODE must be Slave Parallel Configuration mode

The Flowthrough option is DISABLED by default.

Reset Configuration RAM in Reconfiguration (T/F)
When this switch is set to true, it directs the bitstream to reinitialize the device when configuration starts. When the switch is set to false, it directs the bitstream to program the device to retain the current configuration and allows for additional bitstream configuration. The default is true. Use of the feature requires you to be aware of potential contention issues with the prior configuration loaded into the FPGA.

References

- Federal Information Processing Standard Publication 197, Nov. 26, 2001. Advanced Encryption Standard (AES)

Technical Support Assistance

e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
June 2009	01.1	Added SSPI command table. Major changes to SPI support.
October 2009	01.2	Added clarification for dual boot function.
November 2009	01.3	Compression support removed.
January 2010	01.4	Updated SPI Flash mode.
March 2010	01.5	Updated Parallel Port Write Timing diagram.
June 2010	01.6	Updated for Lattice Diamond design software support.
December 2010	01.7	Removed EBR_READ command from the Slave SPI Commands table.
March 2011	01.8	Updated Selectable Master Clock (MCCLK) Frequencies During Configuration (Nominal) table.
August 2011	01.9	Added footnote to "One FPGA, One SPI Serial Flash" figure.
September 2011	02.0	Added new table to Appendix A: Maximum Configuration Bits - Serial and Parallel Mode Bitstream Files.
February 2012	02.1	Updated document with new corporate logo.
August 2012	02.2	Added recommendation to pull up CSSPI0N.
April 2013	02.3	Added pins and footnote to the PERSISTENT Setting and Affected Pins table.
May 2013	02.4	Updated the PERSISTENT Setting and Affected Pins table.
	02.5	Added reference for using SPI mode combined with Slave Serial mode in the Combining Configuration Modes section.
	02.6	Updated the Multiple FPGAs, One SPI Serial Flash diagram.
June 2013	02.7	Changed BAB3 to BFB3 in the General Configuration Flow and Configuration Modes sections,
		Changed 0xBAB3 to 0xBFB3 in the General Configuration Flow and Configuration Modes sections,
		Updated the Encrypted File Format for a Master Mode table contents and changed title to LatticeECP3 Master Serial Encrypted Bitstream File Format table,
		Updated the Encrypted File Format for a Slave Serial Mode table contents and changed title to LatticeECP3 Slave Serial, JTAG Burst, and Slave SPI Encrypted Bitstream File Format.
		Updated the Encrypted File Format for a Slave Parallel Mode contents and changed title to LatticeECP3 Master Parallel Encrypted Bitstream File Format table.
July 2013	02.8	Added Reconfiguration Priority section.
		Added general rules when configuring in the Configuration Modes section.
		Updated MCLK FREQ values in the Global Preferences table.
		Changed t_{PRGMRJ} to t_{PRGM}.
		Updated Technical Support Assistance information.

Appendix A. Configuration Memory Requirements

Table 15-15. Bitstream Memory

Description		Number of Bits
Header		16
Preamble		16
Verify ID		64
Reserved		136
CReg0		64
NOOP		8
Reset address		32
Write inc		32
Data frames (by device)	-35	2067
	-70/95	2819
	-150	3607
Bits per frame (by device)	-35	3416
	-70/95	6728
	-150	8384
Total data frame bits (by device)		Data frames multiplied by bits per frame
CRC bits per frame		16
Stop bits per frame		32
End frame		160
CRC		16
Usercode		64
SED CRC		64
Program security		32
EBR frames		Device and user design specified
Write command		32
EBR data		18432
CRC		16
Stop bits		32
CRC		16
EBR bits per frame		18528
Total EBR frame bits		Equal to EBR Frames multiplied by EBR bits per frame
Program done		48
End		32

Table 15-16. Maximum Configuration Bits – Serial and Parallel Mode Bitstream Files

Device	All Modes Unencrypted Bitstream Size	Slave Serial Mode Encrypted Bitstream Size	Slave Parallel Mode Encrypted Bitstream Size	Units
LatticeECP3-17 No EBR	3.88	5.84	19.60	Mb
LatticeECP3-17 Max EBR	4.41	6.64	22.26	Mb
LatticeECP3-35 No EBR	6.83	10.28	34.38	Mb
LatticeECP3-35 Max EBR	8.10	12.18	40.74	Mb
LatticeECP3-95 No EBR	18.22	27.36	91.32	Mb
LatticeECP3-95 Max EBR	22.46	33.72	112.53	Mb
LatticeECP3-150 No EBR	29.01	43.54	145.27	Mb
LatticeECP3-150 Max EBR	35.58	53.40	178.13	Mb

Appendix B. Lattice Diamond Usage Overview

This appendix discusses the use of Lattice Diamond design software for projects that include the LatticeECP2M SERDES/PCS module .

For general information about the use of Lattice Diamond, refer to the Lattice Diamond Tutorial.

If you have been using ispLEVER software for your FPGA design projects, Lattice Diamond may look like a big change. But if you look closer, you will find many similarities because Lattice Diamond is based on the same toolset and work flow as ispLEVER. The changes are intended to provide a simpler, more integrated, and more enhanced user interface.

Converting an ispLEVER Project to Lattice Diamond

Design projects created in ispLEVER can easily be imported into Lattice Diamond. The process is automatic except for the ispLEVER process properties, which are similar to the Diamond strategy settings, and PCS modules. After importing a project, you need to set up a strategy for it and regenerate any PCS modules.

Importing an ispLEVER Design Project

Make a backup copy of the ispLEVER project or make a new copy that will become the Diamond project.

1. In Diamond, choose **File > Open > Import ispLEVER Project**.

2. In the ispLEVER Project dialog box, browse to the project's .syn file and open it.

3. If desired, change the base file name or location for the Diamond project. If you change the location, the new Diamond files will go into the new location, but the original source files will not move or be copied. The Diamond project will reference the source files in the original location.

The project files are converted to Diamond format with the default strategy settings.

Adjusting PCS Modules

PCS modules created with IPexpress have an unusual file structure and need additional adjustment when importing a project from ispLEVER. There are two ways to do this adjustment. The preferred method is to regenerate the module in Diamond. However this may upgrade the module to a more recent version. An upgrade is usually desirable but if, for some reason, you do not want to upgrade the PCS module, you can manually adjust the module by copying its .txt file into the implementation folder. If you use this method, you must remember to copy the .txt file into any future implementation folders.

Regenerate PCS Modules

1. Find the PCS module in the Input Files folder of File List view. The module may be represented by an .lpc, .v, or .vhd file.

2. If the File List view shows the Verilog or VHDL file for the module, and you want to regenerate the module, import the module's .lpc file:

    a. In the File List view, right-click the implementation folder ( ) and choose **Add > Existing File**.

    b. Browse for the module's .lpc file, *<module_name>*.**lpc**, and select it.

    c. Click **Add**. The .lpc file is added to the File List view.

    d. Right-click the module's Verilog or VHDL file and choose **Remove**.

3. In File List, double-click the module's .lpc file. The module's IPexpress dialog box opens.

4. In the bottom of the dialog box, click **Generate**. The Generate Log tab is displayed. Check for errors and close.

In File List, the .lpc file is replaced with an .ipx file. The IPexpress manifest (.ipx) file is new with Diamond. The .ipx file keeps track of the files needed for complex modules.

Using IPexpress with Lattice Diamond

Using IPexpress with Lattice Diamond is essentially same as with ispLEVER.

The configuration GUI tabs are all the same except for the Generation Options tab. Figure 15-20 shows the Generation Options tab window.

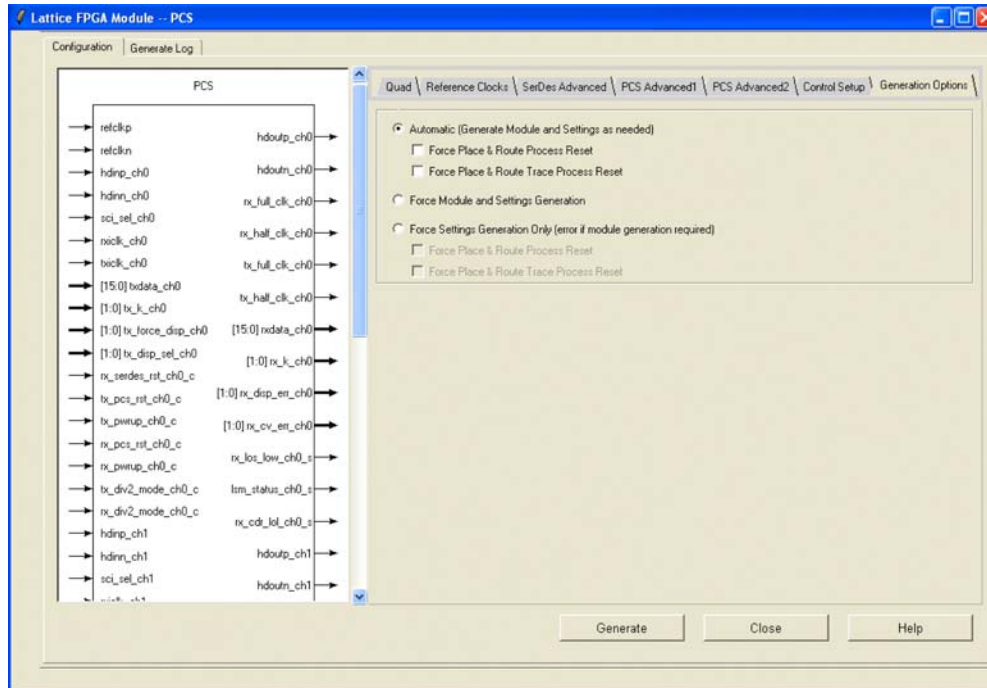*Figure 15-20. Generation Options Tab*

Table 15-17. SERDES_PCS GUI Attributes – Generation Options Tab

GUI Text	Description
Automatic	Automatically generates the HDL and configuration(.txt) files as needed. Some changes do not require regenerating both files.
Force Module and Settings Generation	Generates both the HDL and configuration files.
Force Settings Generation Only	Generates only the attributes file. You get an error message if the HDL file also needs to be generated.
Force Place & Route Process Reset	Resets the Place & Route Design process, forcing it to be run again with the newly generated PCS module.
Force Place & Route Trace Process Reset	Resets the Place & Route Trace process, forcing it to be run again with the newly generated PCS module.

Note:
Automatic is set as the default option. If either Automatic or Force Settings Generation Only and no sub-options (Process Reset Options) are checked and the HDL module is not generated, the reset pointer is set to Bitstream generation automatically.

After the Generation is finished, the reset marks in the process window will be reset accordingly.

Creating a New Simulation Project Using Simulation Wizard

This section describes how to use the Simulation Wizard to create a simulation project (.spf) file so you can import it into a standalone simulator.

1. In Project Navigator, click **Tools > Simulation Wizard**. The Simulation Wizard opens.

2. In the Preparing the Simulator Interface page click **Next**.

3. In the Simulator Project Name page, enter the name of your project in the Project Name text box and browse to the file path location where you want to put your simulation project using the Project Location text box and Browse button.

   When you designate a project name in this wizard page, a corresponding folder will be created in the file path you choose. Click **Yes** in the popup dialog that asks you if you wish to create a new folder.

4. Click either the Active-HDL® or ModelSim® simulator check box and click **Next**.

5. In the Process Stage page choose which type of Process Stage of simulation project you wish to create Valid types are RTL, Post-Synthesis Gate-Level, Post-Map Gate-Level, and Post-Route Gate-level+Timing. Only those process stages that are available are activated.

   Note that you can make a new selection for the current strategy if you have more than one defined in your project.

   The software supports multiple strategies per project implementation which allow you to experiment with alternative optimization options across a common set of source files. Since each strategy may have been processed to different stages, this dialog allows you to specify which stage you wish to load.

6. In the Add Source page, select from the source files listed in the Source Files list box or use the browse button on the right to choose another desired source file. Note that if you wish to keep the source files in the local simulation project directory you just created, check the **Copy Source to Simulation Directory** option.

7. Click **Next** and a Summary page appears and provides information on the project selections including the simulation libraries. By default, the Run Simulator check box is enabled and will launch the simulation tool you chose earlier in the wizard in the Simulator Project Name page.

8. Click **Finish**.

The Simulation Wizard Project (.spf) file and a simulation script DO file are generated after running the wizard. You can import the DO file into your current project if desired. If you are using Active-HDL, the wizard will generate an .ado file and if you are using ModelSim, it creates and .mdo file.

Note: PCS configuration file, (.txt) must be added in step 6.

Setting Global Preferences in Diamond

To set any of the Global preferences in Table 15-18, do the following in Diamond:

• Invoke the Spreadsheet View by selecting **Tools > Spreadsheet View**.

• Select the **Global Preferences Tab** beneath the Spreadsheet View pane as shown in Figure 15-21.

• Right-click on the **Preference Value** to be set. In the drop-down menu, select the desired value.

Table 15-18. Global Preferences

Preference Name	Values
PERSISTENT	OFF SLAVE_PARALLEL SSPI
CONFIG_MODE	SPI SLAVE_SERIAL JTAG SLAVE_PARALLEL SPIm MASTER_PARALLEL SSPI
DONE_EX	OFF ON
MCCLK_FREQ	2.5 4.3 5.4 6.9 8.1 9.2 10 13 15 20 26 33
CONFIG_SECURE	OFF ON
WAKE_UP	1 4 6 7 10 14 17 21 22 23 24 25
WAKE_ON_LOCK	OFF ON
ENABLE_NDR	OFF ON
CONFIG_IOVOLTAGE	2.5 1.2 1.5 1.8 3.3
STRTUP	EXTERNAL TCLK CCLK MCLK

Figure 15-21. Global Preferences Tab
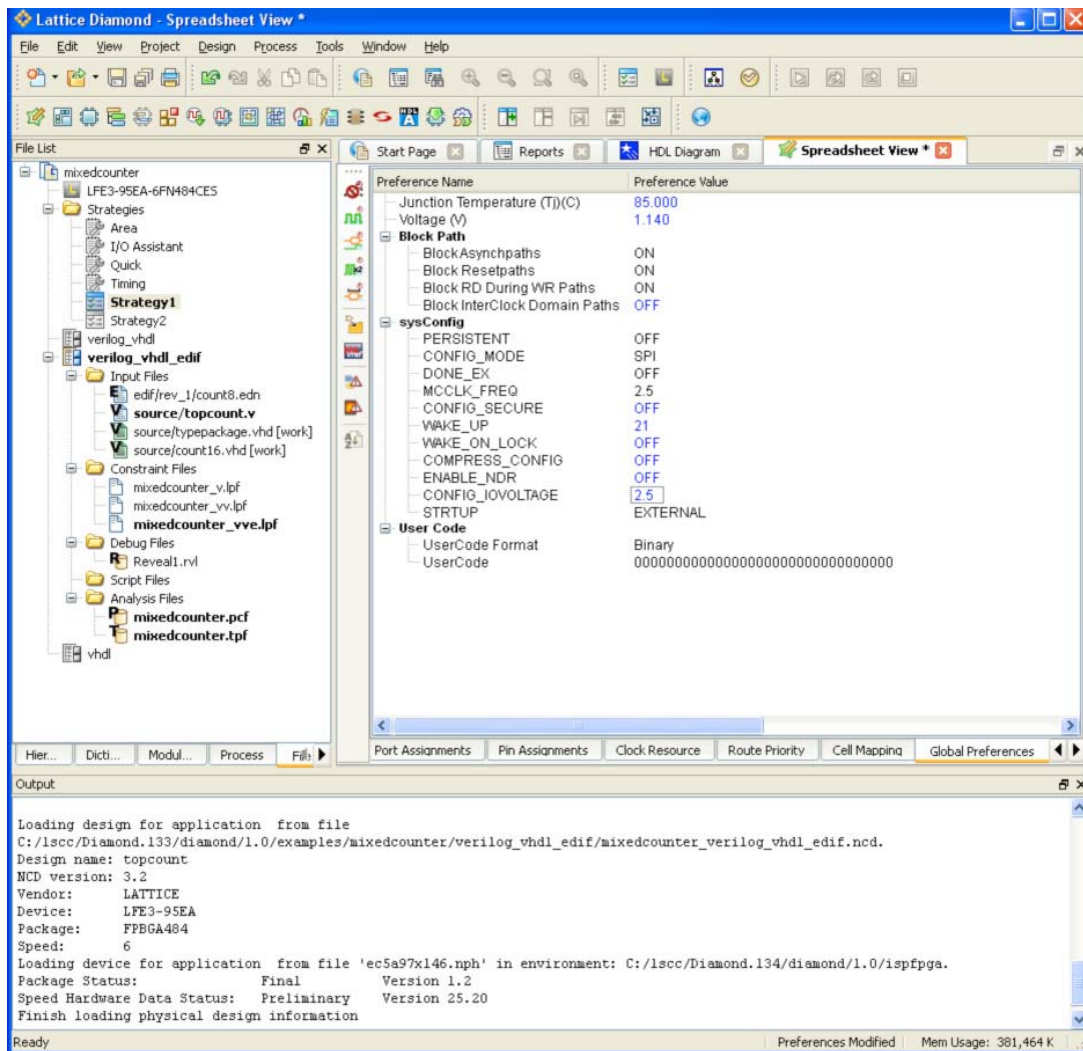
## Setting Bitstream Generation Options in Diamond

To set any of the Bitstream Generation options listed in Table 15-19, do the following:

• In the **File List** pane, double-click the left mouse button on a **Strategy** to invoke the Strategy settings window.

• In the **Process** pane, left-click on **Bitstream**. All options related to generating a bitstream can be set in this window.

Table 15-19. Bitstream Generation Options

Preference Name	Values
Chain Mode	Bypass Disable Flowthrough
Create bit file	True False
No Header	False True
Output Format	Bit File (Binary) Mask and Readback File (ASCII) Mask and Radback File (Binary) Raw Bit file (ASCII)
PROM Data Output Format	Intel Hex 32-bit Motorola Hex 32-bit
Reset Config RAM in re-configuration	True False
Run DRC	True False
Search Path	(Enter a value or browse to specify the search path)

Figure 15-22. Bitstream Options
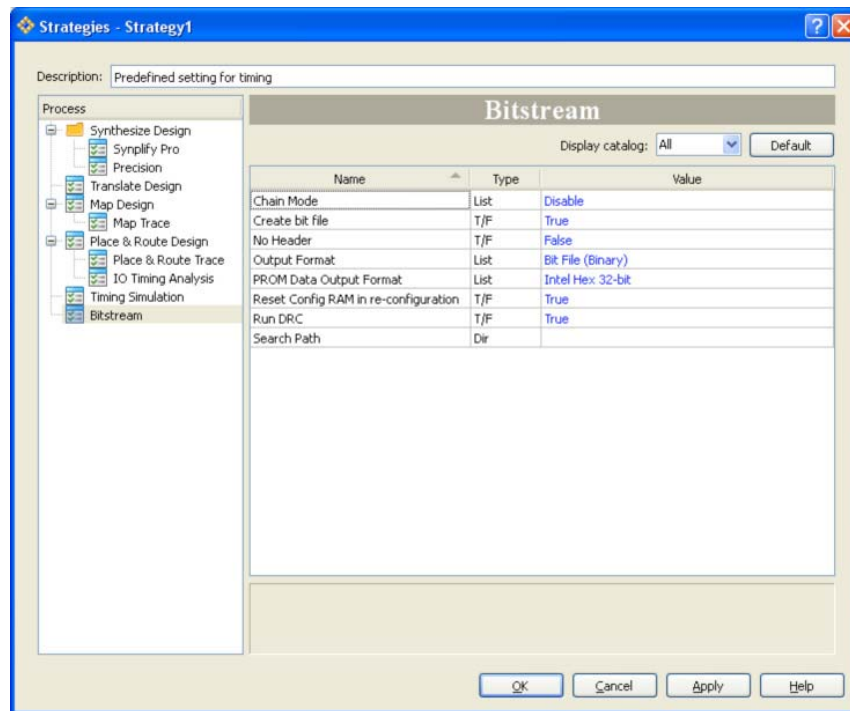
- Double-click the left mouse button on the **Value** you want to set. Select the desired value from the drop-down menu.

  *Note: An explanation of the option is displayed at the bottom of the window. The **Help** button also invokes online help for the option*

- Select **OK**. You can then run the Bitstream File process.

Setting Security Options in Diamond

Prior to setting security options in Diamond, you must have installed the Encryption Control Pack. You must also have selected an encrypted device in your project.

To Set Security Settings, do the following:

• Select the **Tools > Security Setting** option. The following dialog box appears:

• If desired, select **Change** and enter a password.

• Select **OK**. A dialog window appears to enter an encryption key.

• If you do not want to enable an encryption key, select **OK**.

• If you do want to enable an encryption key, select the **Advanced Security Settings** checkbox, enter the **Key Format**, and then enter the **Encryption Key**.
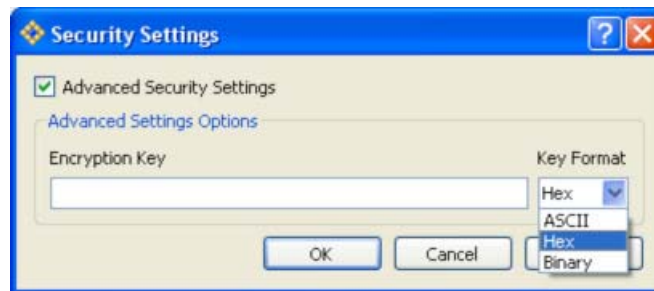
• Select **OK** to create the encryption files.