SPI Serial Flash Programming Using ispJTAG on LatticeECP/EC FPGAs

October 2005 Technical Note TN1078

Introduction

Like all SRAM FPGAs the LatticeECP™ and LatticeEC™ devices need to be configured at power-up. This configuration can be done via:

1. Serial Peripheral Interface (SPI) boot memory
2. Traditional FPGA boot memory
3. JTAG
4. Microprocessor interface

If a boot memory is desired the SPI approach provides a number of advantages over traditional FPGA boot memory:

1. SPI devices are available from multiple vendors ensuring stable supply
2. The cost of SPI memory is up to 75% less than traditional FPGA boot memory
3. SPI memory is available in space saving 8-pin packages that are considerably smaller than packages used for traditional FPGA boot memory

Like all boot memories, SPI Serial Flash needs to be loaded with the FPGA configuration data. There are three options for programming an SPI memory used in conjunction with a LatticeECP/EC device. The SPI memory can be configured off-board using a stand-alone programmer, the memory can be programmed on-board using its SPI interface, or the memory can be programmed on-board via JTAG through the LatticeECP/EC device.

This technical note details the on-board configuration of SPI memory via the JTAG interface.

Related Documents

The following documents are available for download from the Lattice web site at www.latticesemi.com.

- *LatticeECP & EC – Low-Cost FPGA Configuration via Industry-Standard SPI Serial Flash*
- *LatticeECP/EC Family Handbook*
- Lattice technical note TN1053, *LatticeECP/EC sysCONFIG™ Usage Guide*
- *ispDOWNLOAD® Cable Data Sheet*

Hardware and Software Requirements

- An ispDOWNLOAD Cable, either USB or Parallel. Refer to the *ispDOWNLOAD Cable Data Sheet* for part numbers
- Properly installed ispLEVER® 4.2 or later
- Properly installed ispVM® System 14.3 or later

SPI/SPIX Differences

The majority of SPI Serial Flash on the market support a common read Operation Code (Op Code). LatticeECP/EC devices offer direct connection to these devices by hardwiring the read Op Code (03H) into the FPGA silicon. These devices are sometimes referred to as SPI3 devices because they support this common Read Op Code.

SPIX mode allows the LatticeECP/EC to easily interface to SPI Serial Flash devices that support a different read Op Code. This can be done with pull-up and pull-down resistors on the PCB wired to the SPID[7:0] pins, telling the

© 2004 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

FPGA which Read Op Code that particular Flash device supports. If the configuration pins CFG2, CFG1, and CFG0 are 0, 0, 1 (respectively) at power-up the FPGA will use the Read Op Code hardwired on the PCB to access the Flash.

Table 20-1. Device Configuration Codes

CFG2	CFG1	CFG0	Configuration Mode
0	0	0	SPI Serial Flash
0	0	1	SPIX Serial Flash
1	0	0	Master Serial
1	0	1	Slave Serial
1	1	0	Master Parallel
1	1	1	Slave Parallel
X	X	X	ispJTAG™ (always available)

Note: The configuration mode pins indicate the type of device the FPGA will configure from at power-up. For SPI Serial Flash the mode pins should be set to '000' or '001'.

Table 20-2. Manufacturers of "SPI3" Compatible Flash

Manufacturer	Device Family
ST Microelectronics	M25P
NexFLASH	NX25P
Silicon Storage Technology	SST25VF
Saifun	SA25F
Spansion	S25FL
PMC	Pm25LV
Atmel	AT25F

Note: This is not meant to be an exhaustive list of manufacturers or device families.

SPI Serial Flash Sizing

As depicted in Table 20-3, the density of the FPGA determines the size requirement for the SPI Serial Flash. Smaller Flash sizes can be realized by using the compression option in ispLEVER.

Table 20-3. Selecting Flash Density

Family	Device	Max Config Bits (Mb)	Required Boot Memory (Mb)	
			No Comp	Comp (typ)
LatticeECP/EC	EC1	0.6	1	25%
	EC3	1.1	2	25%
	ECP6/EC6	1.8	2	25%
	ECP10/EC10	3.1	4	25%
	ECP15/EC15	4.3	8	25%
	ECP20/EC20	5.3	8	25%
	ECP33/EC33	7.9	8	25%

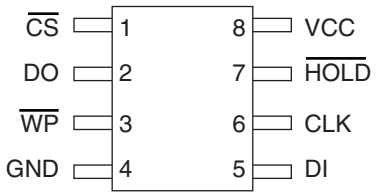
Hardware

This section describes how to physically wire the SPI Serial Flash into a design.

SPI Serial Flash Interface

The standard pin-out for 8-pin SPI Serial Flash memories is shown below (top view):

Figure 20-1. 8-Pin SPI Flash Memory, Standard Pinout



Note: V_{CCIO} for the bank that drives the signals to the SPI Serial Flash must match the SPI Serial Flash V_{CC} (today that voltage is 3.3V but this will change over time as Flash chip geometries decrease). For all LatticeECP/EC packages these signals are located in bank 3.

The SPI interface is a 4-wire serial interface comprised of the following signals:

1. **CS – Chip Select, Input:** Enables and disables device operation. When high the device is at standby power levels and the output is tri-stated. When low the device powers up and instructions can be written to and data read from the device.
2. **CLK – Serial Clock, Input**: Provides timing for the interface. The Serial Data Input (DI) is latched on the rising edge of CLK. Serial Data Output (DO) changes after the falling edge of CLK.
3. **DI – Serial Data In, Input**: When the device is enabled (CS is low) this pin allows instructions, addresses, and data to be serially written to the device. Data is latched on the rising edge of the CLK.
4. **DO – Serial Data Out, Output**: When the device is enabled (CS is low) this pin allows data and status to be serially read from the device. Data is shifted out on the falling edge of the CLK.

The SPI interface also supports the following two functions:

1. **HOLD – Input**: Allows device to be paused without de-selecting it. When HOLD is low DO will be tri-stated while DI and CLK are ignored. This function is useful when multiple devices are sharing the same SPI signals.
2. **WP – Write Protection, Input**: Used to prevent inadvertent writing of the Status Register Block Protect bits.

Note: The LatticeECP/EC SPI interface supports the basic 4-wire interface, but the user is free to implement these additional functions if desired.

ispJTAG Interface

The ispJTAG interface supports both IEEE 1149.1 Boundary Scan and IEEE 1532 In-System Configuration. Standard pinouts for 1x10, 1x8, and 2x5 download headers are shown in Table 20-4. The 1x10 header is preferred but ultimately the header chosen will depend on the available download cable. All new download cables have uncommitted “flywire” connections, so they can be attached to any of the header styles. Direction, in Table 20-4, refers to the cable, for example “output” indicates an output from the cable to the FPGA.

Table 20-4. Download Header Pinout

Pin Name	1x10	1x8	2x5	Direction	Description
V_{CCJ}	1	1	6	—	3.3V or 2.5V
TDO	2	2	7	Input	Test Data Out
TDI	3	3	5	Output	Test Data In
PROGRAMN	4	4	10	Output	Forces FPGA config, N/C
TRST	5	5	9	Output	Test Reset, N/C
TMS	6	6	3	Output	Test Mode Select
GND	7	7	2, 4, 8	—	Ground
TCK	8	8	1	Output	Test Clock
DONE	9			Input	FPGA configuration complete, optional
INITN	10			Input	FPGA ready for configuration, optional

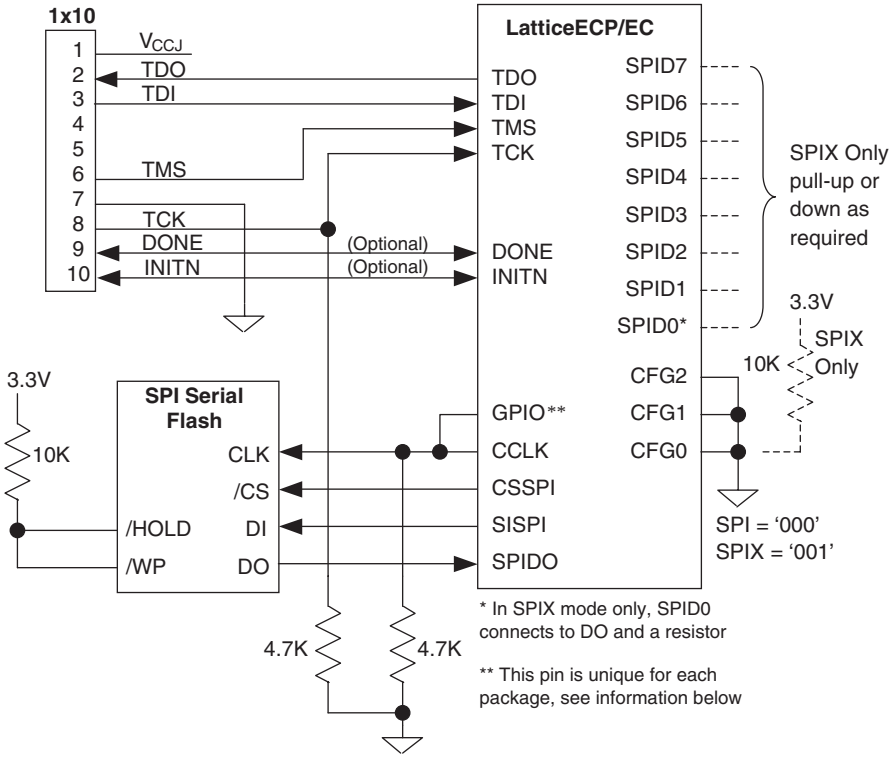
1. **VCCJ** – V_{CC} JTAG: Powers the ispDOWNLOAD Cable.
2. **TDO** – Test Data Out: Serial data read from the test device to the cable.
3. **TDI** – Test Data In: Serial data written from the cable to the device.
4. **PROGRAMN**: Initiates a configuration sequence when asserted low. Not used and should not be connected on the board.
5. **TRST** – Test Reset: Not used and should not be connected on the board.
6. **TMS** – Test Mode Select: Controls the IEEE 1149.1 state machine.
7. **TCK** – Test Clock: Clocks the IEEE 1149.1 state machine.
8. **GND**: Digital ground.
9. **DONE** – Optional, USB Only: Open drain, internal pull-up. A high indicates that the FPGA configuration sequence 9 was completed successfully.
10. **INITN** – Optional, USB Only: Open drain, internal pull-up. A high indicates that the FPGA is ready to be configured. A low indicates the FPGA is not ready to be configured, or an error occurred during configuration.

Note: Use of the DONE and INITN pins, while optional, does allow ispVM System to check that configuration completed successfully. If DONE or INITN are wired to the connector then the proper dialog box(es) must be checked in the Cable and I/O Port Setup section of ispVM System. See the Software section of this document for more details.

Schematic

The schematic in Figure 20-2 illustrates how to wire the ispJTAG connector, FPGA, and SPI Serial Flash.

Figure 20-2. Hardware Schematic



- The download header has standard 0.1 inch pin-to-pin spacing.
- The 4.7K pull-down resistors prevent spurious clock pulses during V_{CC} ramp-up. Place the resistors close to their clock line to keep the stub length as short as possible.
- The CCLK frequency can be as high as 50MHz, so keep this trace fairly short.
- V_{CCIO} for the bank that drives the signals to the SPI Serial Flash must match the SPI Serial Flash V_{CC} (today that voltage is 3.3V but this will change over time as Flash chip geometries decrease). For all packages these signals are located in bank 3.
- During configuration CCLK drives the SPI Serial Flash CLK pin, but once the FPGA completes configuration CCLK goes into tri-state. The CCLK pin is not accessible by user code so CCLK needs to be wired to a nearby General Purpose I/O pin (GPIO) to allow the FPGA fabric to supply a clock to the SPI Serial Flash. This pin is part of the Soft SPI Interface and is unique to each package. If the user embeds the Soft SPI Interface into their code then this pin, along with the other pins wired to the SPI Serial Flash, must be locked using the Pre-Map Preference Editor in ispLEVER. A complete list of these pins is found in Table 20-5.
- In addition to standard decoupling practices, place a decoupling capacitor close to the connector's V_{CCJ} pin. Any standard ceramic capacitor value may be used, for example 0.1 µF, 0.01 µF, etc.

Table 20-5. Pin Locations for Wiring ECP/EC to SPI Serial Flash[1]

LatticeECP/ EC Device	FPGA Pin Function	SPI Signal	672 fpBGA	484 fpBGA	256 fpBGA	208 PQFP	144 TQFP	100 TQFP
ECP33 EC33	GPIO	CLK	AB26	Y21				
	CCLK	CLK	V20	T20				
	CSSPIN	/CS	Y25	V21				
	SISPI	DI	W25	U21				
	SPIDO	DO	W26	V22				
ECP20 EC20	GPIO	CLK	AB26	Y21				
	CCLK	CLK	V20	T20				
	CSSPIN	/CS	Y25	V21				
	SISPI	DI	W25	U21				
	SPIDO	DO	W26	V22				
ECP15 EC15	GPIO	CLK		Y21	M13			
	CCLK	CLK		T20	L15			
	CSSPIN	/CS		V21	M16			
	SISPI	DI		U21	K16			
	SPIDO	DO		V22	J16			
ECP10 EC10	GPIO	CLK		Y21	M13	113		
	CCLK	CLK		T20	L15	130		
	CSSPIN	/CS		V21	M16	121		
	SISPI	DI		U21	K16	123		
	SPIDO	DO		V22	J16	124		
ECP6 EC6	GPIO	CLK		Y21	M13	113	77	
	CCLK	CLK		T20	L15	130	94	
	CSSPIN	/CS		V21	M16	121	85	
	SISPI	DI		U21	K16	123	87	
	SPIDO	DO		V22	J16	124	88	
EC3	GPIO	CLK			M13	113	77	52
	CCLK	CLK			L15	130	94	66
	CSSPIN	/CS			M16	121	85	57
	SISPI	DI			K16	123	87	59
	SPIDO	DO			J16	124	88	60
EC1	GPIO	CLK				113	77	52
	CCLK	CLK				130	94	66
	CSSPIN	/CS				121	85	57
	SISPI	DI				123	87	59
	SPIDO	DO				124	88	60

1. SPI is a four-wire interface; this table shows these wires plus the GPIO pin that needs to be wired to CCLK per Figure 20-2.

Software

The LatticeECP/EC FPGAs allow programming of the SPI Serial Flash via the ispJTAG port by using a small piece of code to redirect the ispJTAG 4-wire interface to the SPI Flash 4-wire interface.

With the Soft SPI Interface installed, ispJTAG is free to read or write the SPI Serial Flash while the FPGA is executing user code. This allows the user to perform functions such as background configuration updates.

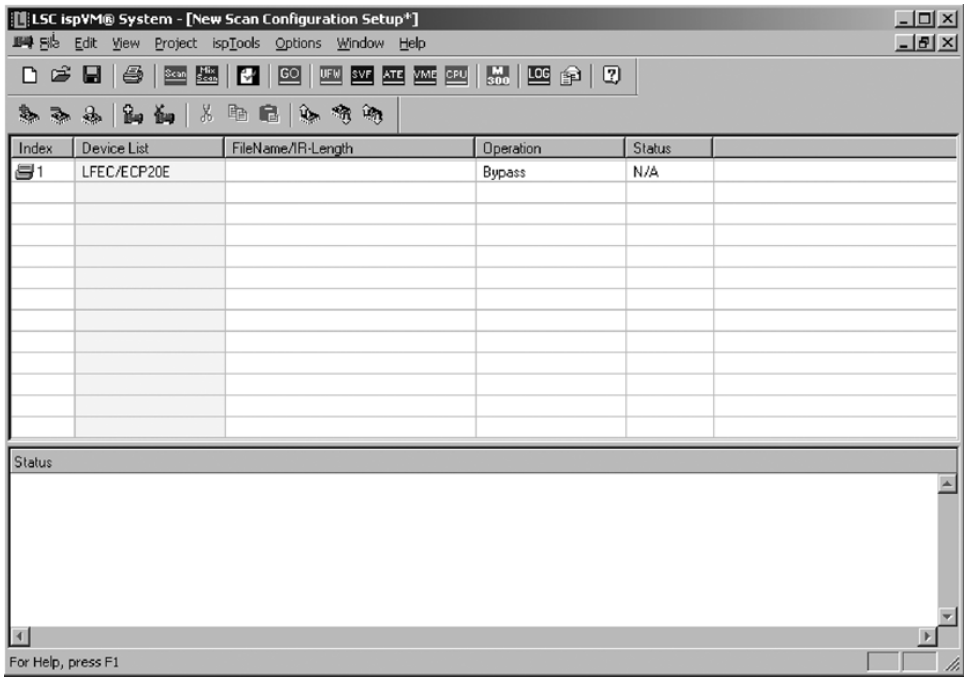
Programming Procedure

In order to program SPI Serial Flash via ispJTAG the FPGA must contain the Soft SPI Interface. Programming the SPI Serial Flash with ispVM System and an ispDOWNLOAD cable makes this transparent to the user. The software simply loads a default Soft SPI Interface bitstream into the FPGA and then loads the user bitstream into the Flash. Once programing of the SPI Serial Flash is complete the FPGA configures itself by reading the Flash. Again, software makes all of this transparent to the user so that it is no different than programming any other serial boot device.

The following instructions describe the process of selecting the FPGA, selecting the SPI Serial Flash, and programming the SPI Serial Flash. The following screen shots are from ispVM System 15.0.

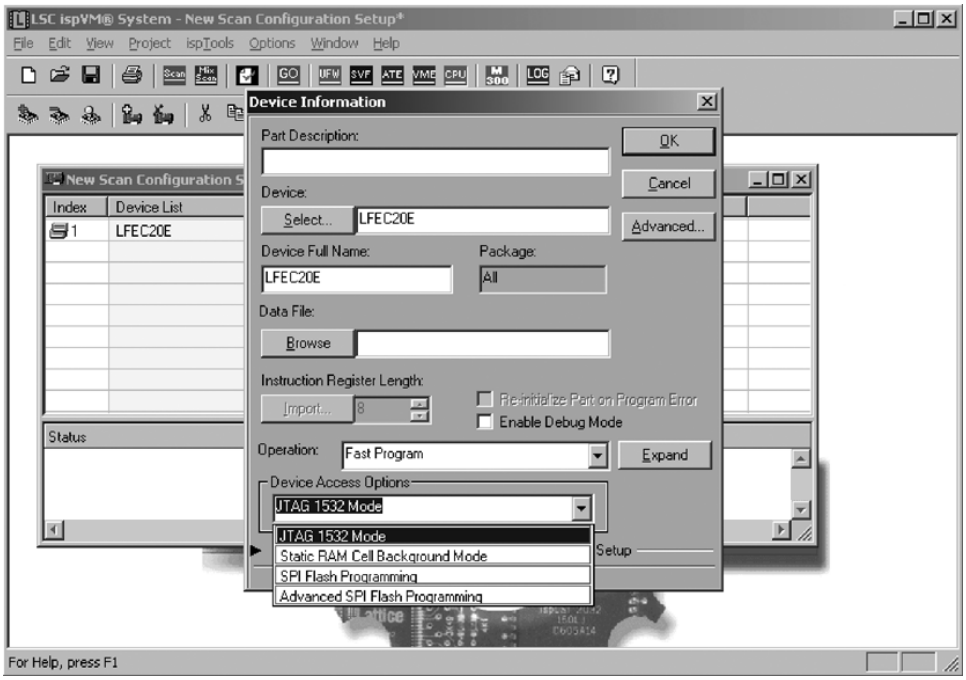
1. Connect the ispDOWNLOAD Cable to the appropriate header and apply power to the board.
2. Start the ispVM System software.
3. From the main window click on the **Scan** button located on the toolbar. The LatticeECP/EC device should be detected automatically (if it's not detected, check the ispJTAG connections and make sure the board is powered up). The resulting screen should be similar to Figure 20-3.

Figure 20-3. Main Window, Scan Complete



4. Double-click the number in the **Index** column and select the appropriate device to open the Device Information window, shown in Figure 20-4.

Figure 20-4. Device Information Window



5. Under **Device Access Options**, select either **SPI Flash Programming** or **Advanced SPI Flash Programming**. To use the default Soft IP and program the SPI Serial Flash with a single LatticeECP/EC bitstream, select **SPI Flash Programming**. If the Soft IP was instantiated into the your design, or if you need to merge multiple bitstreams into a single SPI Serial Flash device, select **Advanced SPI Flash Programming**.

    a. **SPI Flash Programming**

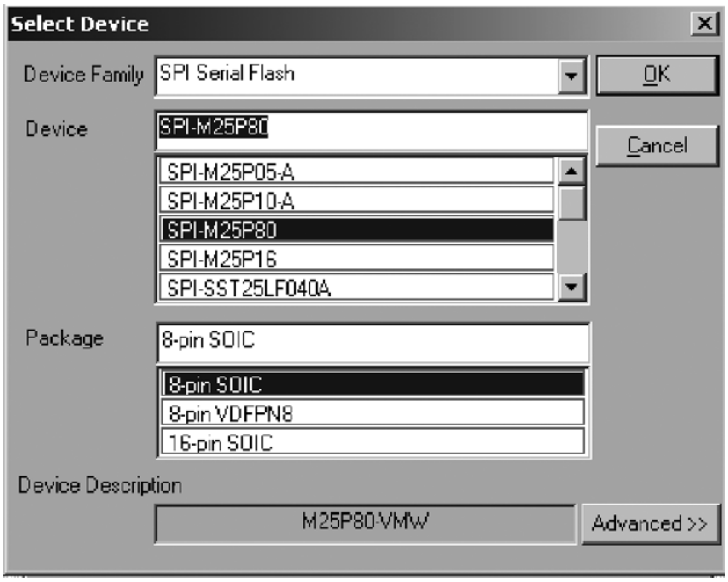
        i. Under **Device Access Options**, select **SPI Flash Programming**. The SPI Serial Flash Device dialog shown in Figure 20-5 will be displayed.

Figure 20-5. SPI Serial Flash Device Dialog



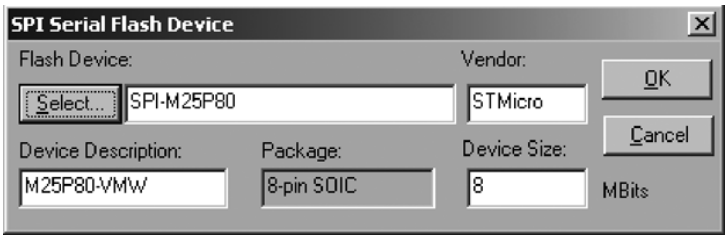
        ii. Click on the **Select** button to select the target SPI Serial Flash device. The SPI Serial Flash Select Device dialog shown in Figure 20-6 will be displayed.

Figure 20-6. Select SPI Serial Flash Device Dialog



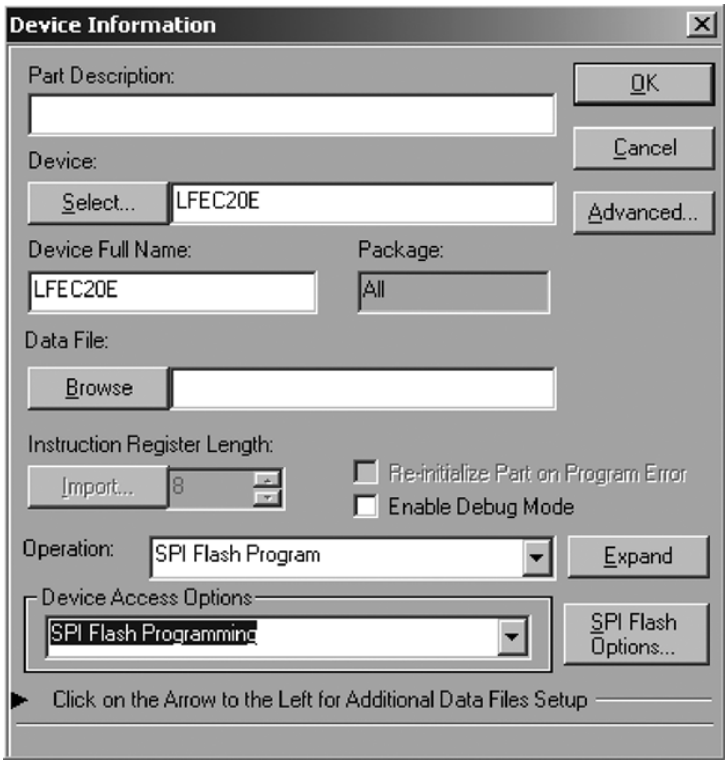
iii. Select the target SPI Serial Flash device and click the **OK** button. The SPI Serial Flash Device dialog shown in Figure 20-7 will be displayed.

Figure 20-7. SPI Serial Flash Device Dialog



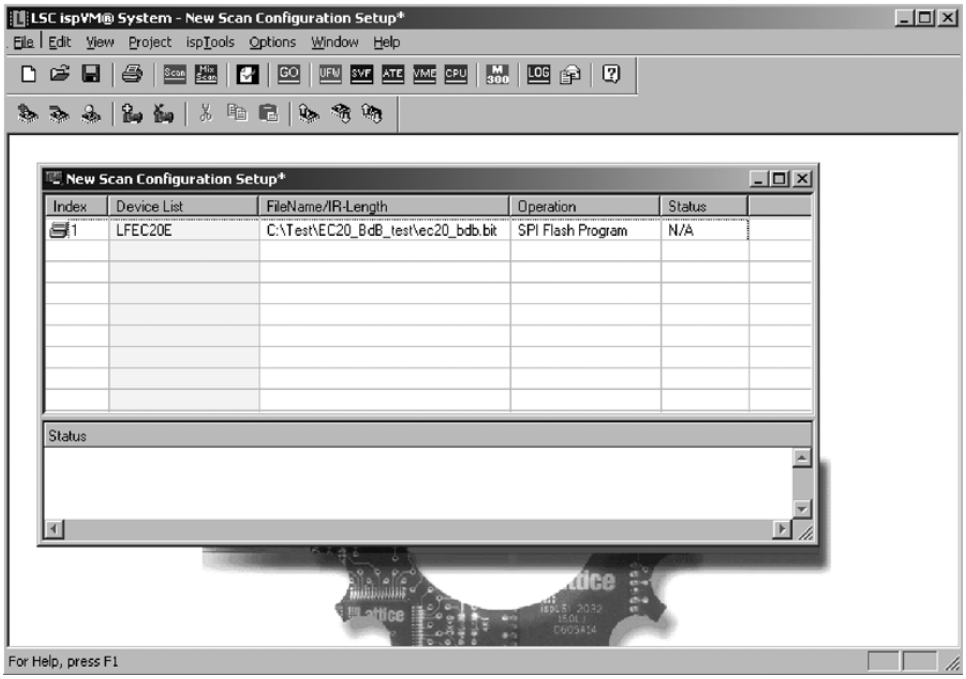
iv. Click the **OK** button. The Device Information dialog shown in Figure 20-8 will be displayed.

Figure 20-8. SPI Serial Flash Device Information Dialog



v. Under **Data File**, select the LatticeECP/EC bitstream to be programmed into the SPI Serial Flash device. Click the **OK** button. This will return you to the main ispVM window, shown in Figure 20-9. Proceed to step 6.

Figure 20-9. Main Window, SPI Programming



b. **Advanced SPI Flash Programming**

i. Under **Device Access Options**, select **Advanced SPI FLASH Programming**, as shown in Figure 20-4.

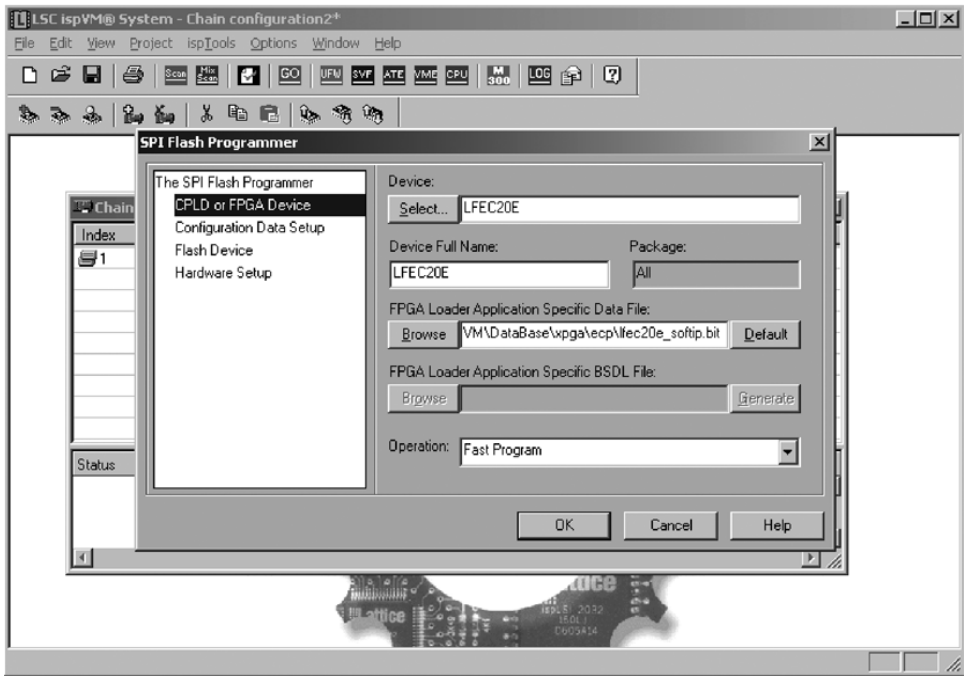
ii. The FPGA Loader Setup Dialog will be launched.

iii. Click on **CPLD or FPGA Device**. It should look similar to Figure 20-10.

iv. The default Soft IP will automatically be added as the **FPGA Loader Application Specific Data File**. If you instantiated the Soft IP into your own design, click on the **Browse** button to select your bitstream. Clicking on the **Default** button will reload the default Soft IP bitstream.

v. Under **Operation**, select **Fast Program**.

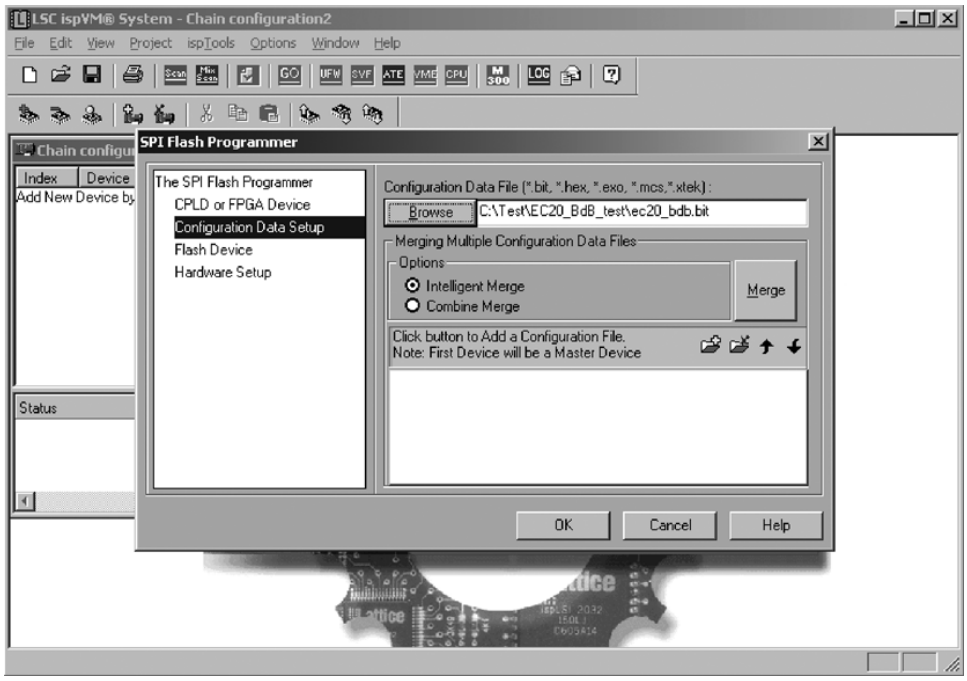
Figure 20-10. Select FPGA Device



vi. Click on **Configuration Data Setup** (see Figure 20-11).

vii. Under **Configuration Data File**, click on **Browse** to locate the user configuration file created using ispLEVER. Double-click on the file name. Use the **Merging Multiple Configuration Data Files** option if you want to merge multiple bitstreams into the SPI Serial Flash device. Multiple bitstreams would be used to configure multiple FPGAs using one SPI Serial Flash.

Figure 20-11. Configuration Data Setup

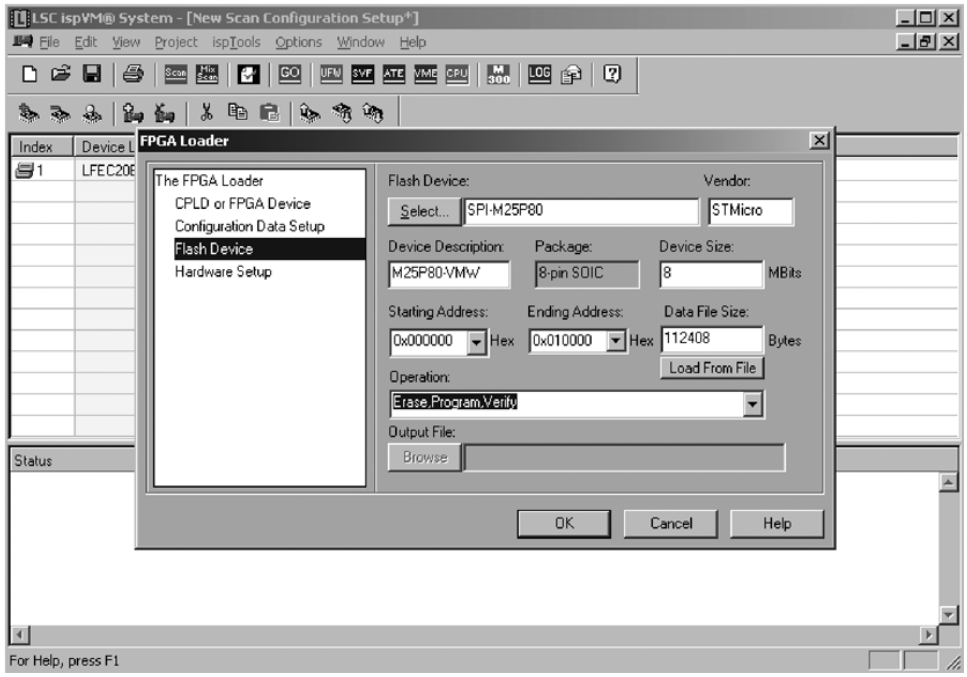


viii.Select **Flash Device** in the left window (see Figure 20-12).

ix. Under **Flash Device**, click **Select** to choose the desired device; in this case an ST Micro device was selected.

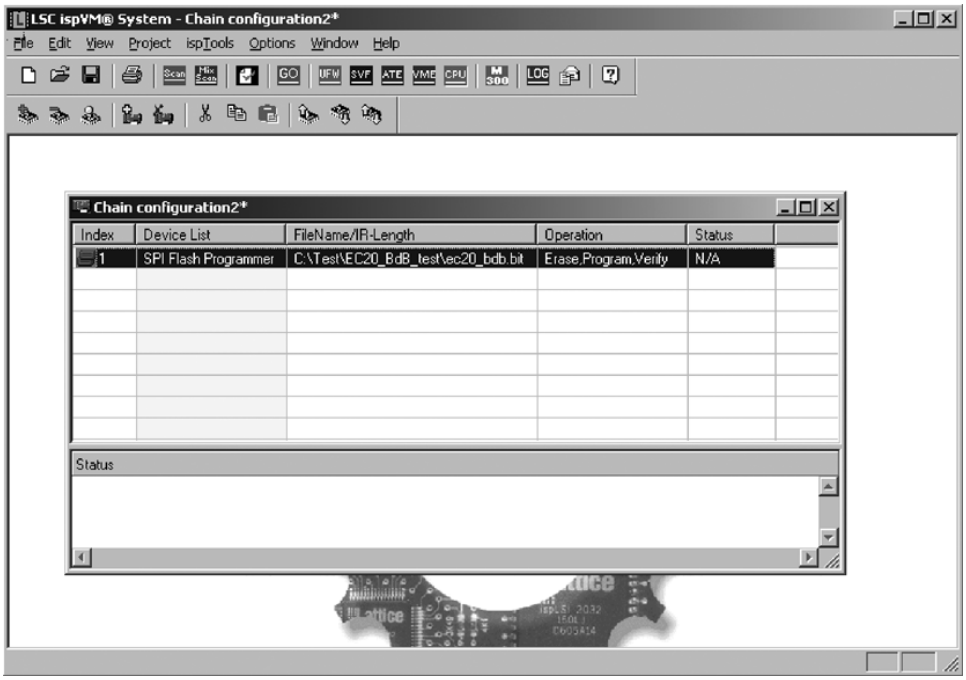
x. From the drop-down list under **Operation** select **Erase, Program, Verify**.

Figure 20-12. Select Flash Device



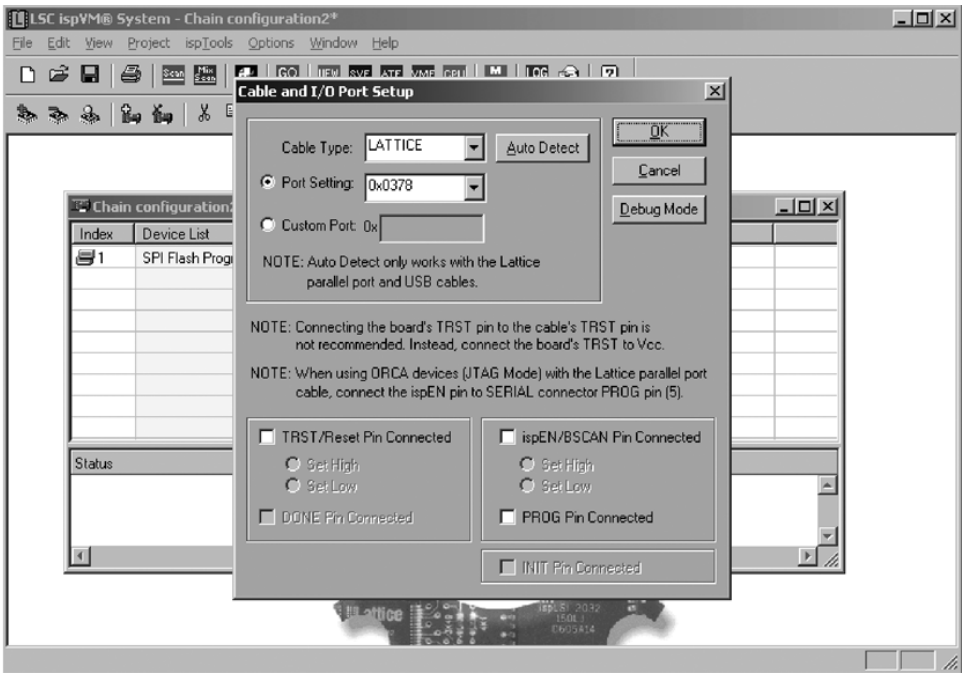
xi. Click **OK** to exit the FPGA Loader window. This will return you to the main ispVM window, shown in Figure 20-13.

Figure 20-13. Main Window, Advanced SPI Flash Programming



6. From the main project window, on the menu bar, click **Options > Cable** and **I/O Port Setup**.
7. Check that the proper cable type is selected (parallel or USB) and that the PROG, DONE, and INIT boxes (Figure 20-14) are properly selected/de-selected based on how the ispJTAG connector is wired up (see the Hardware Schematic section of this document).

Figure 20-14. Cable and I/O Port Setup



8. Click on **OK**.
9. From the main project window click the green **GO** button on the toolbar; this will begin the download process.
10. Upon successful download, in order to configure the FPGA with the new configuration, the user must cycle power to the FPGA or pulse the FPGA's Program pin low then high.

Including the SPI Interface in the FPGA Design

If the user wishes JTAG to have access to the SPI Serial Flash while the FPGA is operating, for instance to allow background configuration updates, then the Soft SPI Interface must be instantiated into the user code. Once a configuration bitstream containing the user code and the Soft SPI Interface has been created the programming sequences will be identical to those detailed above (see step 5).

Sample Code

The following code samples are simple VHDL and Verilog files that show how to instantiate the netlist file, i.e. the Soft SPI Interface. The netlist file should be placed in the same directory as the top design file. In the following examples the netlist file is called SPITOP.ngo. The netlist file, along with these sample source files, is freely available from the Lattice Semiconductor web site at www.latticesemi.com.

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity design_top is
    port ( rst     : in std_logic;
           sclk    : in std_logic;
           cnt_out : out std_logic_vector(7 downto 0);
  •
  •
  •
        -- SPI Serial Flash pins

          SPI_C   : out std_logic;  -- clock
          SPI_D   : out std_logic;  -- data input
          SPI_SN  : out std_logic;  -- chip select
          SPI_Q   : in  std_logic   -- data output
        );
end;

architecture behave of design_top is

--Instantiate the file

     component SPITOP
         port (
                 SPI_PIN_C  : out std_logic;
                 SPI_PIN_D  : out std_logic;
                 SPI_PIN_SN : out std_logic;
                 SPI_PIN_Q  : in  std_logic
                 );
     end component;

--User code

     signal cnt: std_logic_vector(7 downto 0);

begin
     process(sclk, rst)
     begin
          if rst = '1' then
           cnt <= (others => '0');
      elsif rising_edge(sclk) then
           cnt <= cnt + 1;
      end if;

 end process;

 cnt_out <= cnt;
  •
  •
  •

-- SPITOP port map

 spi_ip: SPITOP port map
      (
        SPI_PIN_C   =>  SPI_C,
        SPI_PIN_D   =>  SPI_D,
        SPI_PIN_SN =>  SPI_SN,
        SPI_PIN_Q   =>  SPI_Q
        );

end behave;
```

Verilog

Here is the same design in Verilog.

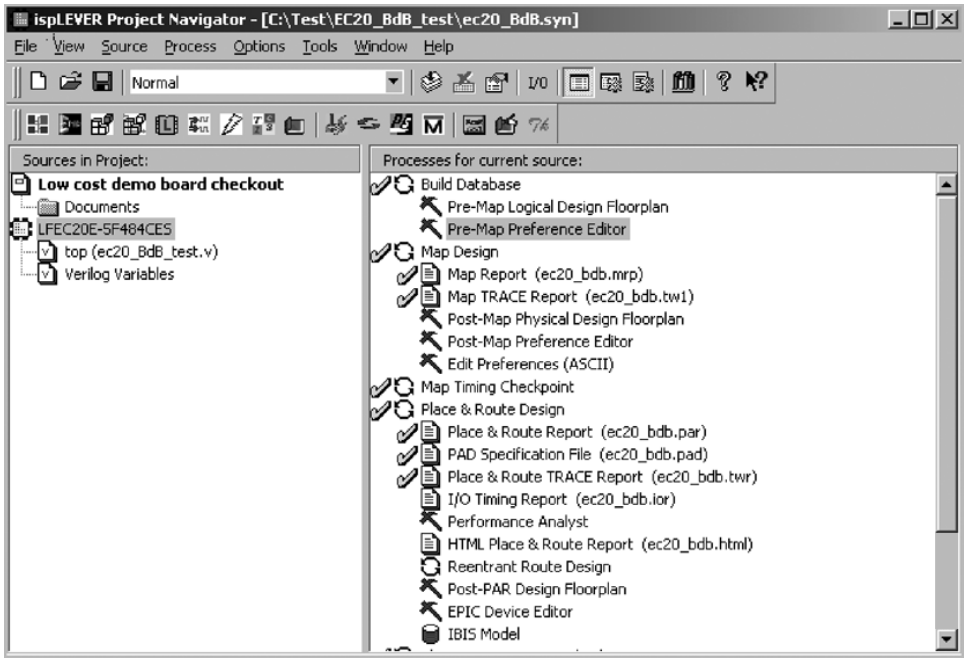
```
module design_top(rst, sclk, cnt_out, spi_c, spi_d, spi_sn, spi_q) ;

input         sclk, rst;
output [7:0]  cnt_out;

// SPI Serial Flash pins

input         spi_q;
output        spi_d, spi_sn, spi_c;

reg [7:0]     cnt_out;

// User code

always @ (posedge sclk or posedge rst)
     begin
          if (rst)
               cnt_out = 2'h00;
          else
               cnt_out = cnt_out + 1;
      end

// Instantiate Soft SPI Interface

SPITOP spi_ip
            (
             .SPI_PIN_C(spi_c),
             .SPI_PIN_D(spi_d),
             .SPI_PIN_SN(spi_sn),
             .SPI_PIN_Q(spi_q)
             );

endmodule

module SPITOP (SPI_PIN_C, SPI_PIN_D, SPI_PIN_SN, SPI_PIN_Q);

output     SPI_PIN_D, SPI_PIN_SN, SPI_PIN_C ;
input      SPI_PIN_Q;

endmodule
```

The Soft SPI Interface is only needed to connect JTAG to the SPI Serial Flash interface. If the SPI Serial Flash will be used by the user design, for instance as scratch memory, and background access via JTAG will not be required, then the Soft SPI Interface is not needed, and the user code can access the SPI pins directly. Remember that the configuration memory must start at address zero; any user defined memory space must be located above the configuration data. It is recommended that an address above the maximum possible configuration size be chosen. For instance if an ECP/EC20 device is being used, select a scratch pad starting address above 5.3Mb (see Table 20-3 in this document).

Locking the Pins

The last thing the user needs to do is tell ispLEVER which pins are connected to the SPI Serial Flash device (see Figure 20-2 and Table 20-5). From the ispLEVER Project Navigator window click on the package name in the left window, then double click on the Pre-Map Preference Editor in the right window (see Figure 20-15).

Figure 20-15. Select Pre-Map Preference Editor

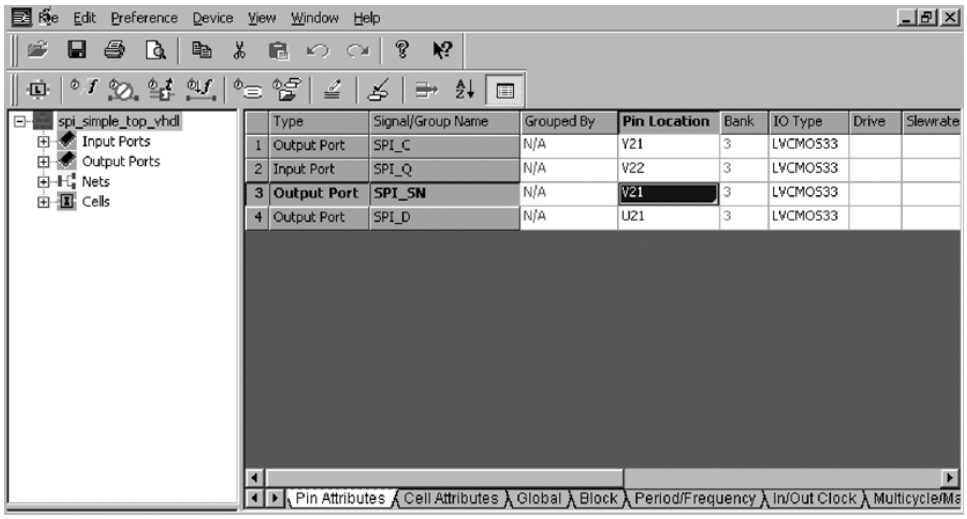


This will start the editor; it should look similar to Figure 20-16. The Pre-Map Preference Editor is where pin numbers and other attributes are assigned to the various I/O in a design. Figure 20-16 shows the proper pin selection for a 484 fpBGA, it also shows proper selection of the I/O type (in this case LVCMOS_3.3). Each package requires a different pin selection; refer to Table 20-5 and the schematic in Figure 20-2 (in the Hardware section of this document) for details.

While it is recommended that the pin listed in Table 20-5 be used as the GPIO to wire to CCLK, if the Soft IP is instantiated into the user design then any pin can be selected - with the following cautions:

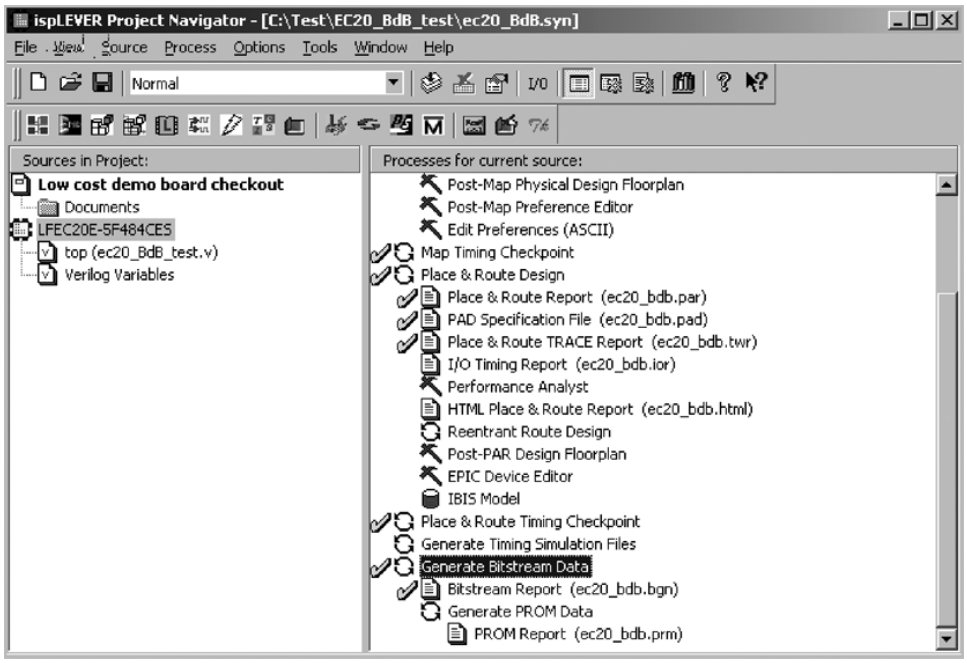
1. The Default Soft IP is built to use the default pin (see step 5 above under Programming Procedures), if a different pin is chosen the Default Soft IP will not work, i.e. you must instantiate the Soft IP in your design.
2. Do not select the DOUT pin as the GPIO to wire to CCLK. DOUT is an output during configuration, as is CCLK. Selecting DOUT will cause contention.
3. Select a pin from a bank that has its V_{CCIO} connected to 3.3V.

Figure 20-16. Preference Editor



After all of the pin attributes have been entered, close the Preference Editor, scroll down the right window, and double click on Generate Bitstream Data (see Figure 20-17). Once the bitstream has been generated, go to the Programming Procedure section of this document.

Figure 20-17. Generate Bitstream Data



Design Notes

The following tips will help insure first pass success when using LatticeECP/EC devices with SPI Serial Flash.

The PROGRAMN pin can be left open, wired to a button, microprocessor, etc. The PROGRAMN pin should be high at power-up, do not tie this pin to a pull-down resistor. There is a weak pull-up on this pin, but if needed, add an external 10K ohm pull-up resistor.

The INITN pin can be left open, connected to a microprocessor, status register, or other LatticeECP/EC devices. Holding this pin low during configuration will keep the device from configuring. Do not tie this pin to a pull-down resistor. There is a weak pull-up on this pin but if needed add an external 10K ohm pull-up resistor. This pin can drive 8 mA. If driving an LED that requires higher current, use an external driver/buffer.

The DONE pin can be left open, connected to a microprocessor, status register, or other Lattice devices with DONE pins. If you connect this pin to other DONE pins then all of the DONE pins in the chain will need to be set to open drain (using ispLEVER or an attribute in your code) and a pull-up resistor of about 10K will need to be added. Holding this pin low during configuration will keep the LatticeECP/EC from waking up. Do not tie this pin to a pull-down resistor. There is a weak pull-up on this pin but, if needed, add an external 10K ohm pull-up resistor. This pin can drive 8 mA. If driving an LED that requires higher current, use an external driver/buffer.

All of the SPI pins are part of I/O bank 3; therefore V_{CCIO} for bank 3 must be connected to the same voltage as the SPI Serial Flash.

When utilizing SPI Serial Flash, use of the SPI pins as user pins is generally not recommended. If you must use one or more of the SPI pins as user I/O, do not change the I/O type or direction. For example, if, during configuration, the SPI pin you wish to use is an input you may only use the pin as an input, not an output, and it must be of type LVCMOS33.

If you set Config_Mode in the ispLEVER Preference Editor to SPI3, and you are instantiating the Soft IP, you will get warnings when you compile your code. This is due to the Soft IP requiring use of the SPI3 port as normal I/O during SPI Serial Flash programming. You can avoid these warnings by selecting None or JTAG instead of SPI3 for the Config_Mode.

If you set Config_Mode to SPI3, or you instantiate the Soft IP into your code (assigning the SPI pins in the Preference Editor or your code), the SPI pins will be protected from use by the Place and Route tools. If you set Config_Mode to None or JTAG, and you are not instantiating the Soft IP, the SPI pins will not be protected from use by the Place and Route tools. In this case, consider using the Prohibit Site “<pin number>” command on the SPI pins to keep the Place and Route tools from using these pins. This is particularly useful if your design is I/O bound and you want to allow Place and Route as much flexibility as possible. You can use the Package View in the Preference Editor to place the Prohibits.

Try to select an SPI Serial Flash that is supported by your version of ispVM programming software. Check for the latest SPI Serial Flash vendor support in the latest version of ispVM software. The ispVM software is available at no charge from the Lattice web site at www.latticesemi.com. If your SPI Serial Flash is not supported on the latest software, please contact Lattice Technical Support.

If you have selected a GPIO pin to drive the CCLK that is other than the pin recommended in Table 20-5 of this document you will need to instantiate the Soft IP into your code. See the section above entitled “Including the SPI Interface in the FPGA Design”.

Conclusion

By combining the new low cost LatticeECP/EC family of devices with low cost, third party serial Flash, engineers can now take advantage of a very cost effective system solution. In addition to cost savings, the design also benefits from the space conscious 8-pin package.

This new capability, in addition to the traditional configuration methods, is fully supported by the latest Lattice tools.

For more information on Lattice's family of devices, visit our web site at www.latticesemi.com.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com