

Using SPI Flash with 7 Series FPGAs

Author: Rohan Balar

Summary

This application note describes the advantages of selecting a serial peripheral interface (SPI) flash as the configuration memory storage for the Xilinx 7 series FPGAs and the details for implementing the solution. This document includes the required connections between the FPGA and the SPI flash memory and the details necessary to select the proper SPI flash.

[Programming the SPI Flash In-System](#) provides details about using XILINX tools for in-system programming of the SPI flash via the FPGA. This allows for configuration flexibility during the debugging stages of development. More information on the Vivado tools can be found at www.xilinx.com. The designer should be familiar with [UG470, 7 Series FPGAs Configuration User Guide](#) that contains additional information on FPGA configuration and details on other configuration methods.

Introduction

This application note addresses the two flows shown in [Figure 1](#):

- Indirect SPI flash programming using the Vivado Design Suite and ISE Design Suite iMPACT tools.
- SPI flash configuration that delivers the FPGA configuration bitstream stored in a SPI flash memory to the 7 series FPGAs.

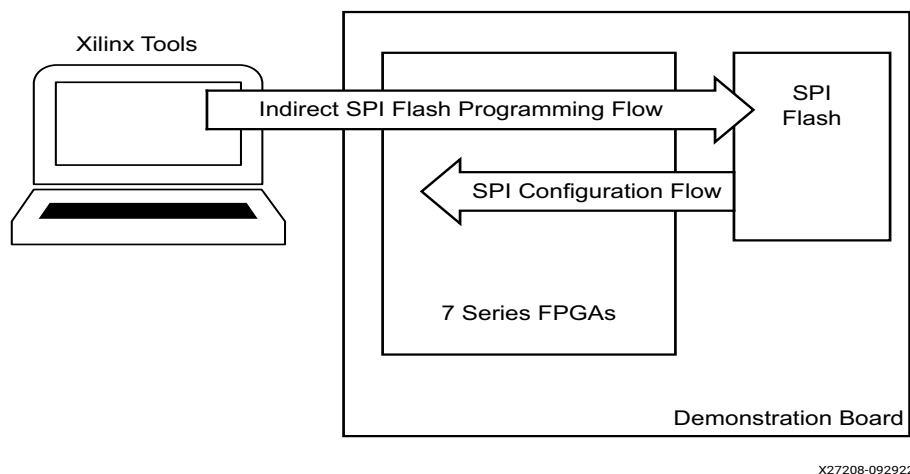


Figure 1: SPI Flash Configuration and Indirect Programming Flows

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing noninclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this [link](#) for more information.

Xilinx FPGAs require that a configuration bitstream is delivered at power-up. The SPI flash memories use a 4-wire synchronous serial data bus. The SPI flash configuration requires only four pins, which allows 1- or 2-bit data width for delivery of the configuration bitstream. Newer SPI flash devices offer the option to use six pins to enable 4-bit data width, thereby decreasing configuration time appropriately. FPGA configuration via the SPI interface is a very low pin count configuration solution and many vendors have devices in a large range of density options.

Other options for FPGA configuration, such as a byte peripheral interface (BPI) parallel NOR flash, supports a wider configuration data bus that allows for faster configuration at power-up, however this mode requires a minimum of 25 pins.

Because parallel NOR flash devices have higher density options than SPI flash, BPI flash should be considered if the application requires large amounts of nonvolatile data storage or if several FPGA bitstreams need to be stored.

Xilinx also provides the ability to program the SPI flash in-system using the existing configuration connections between the SPI flash and the FPGA. The Xilinx programming tool uses JTAG to configure the FPGA to enable a path between the configuration cable and the SPI flash. This allows design flexibility in a lab environment to easily program new configuration bitstreams into the SPI flash without removing the flash from the board and using an external desktop programmer.

The sections in this document are:

- [SPI Flash Basics](#): Review of the SPI flash pin functions and device features.
- [SPI Flash Configuration Interface](#): Details on the FPGA configuration interface with the SPI flash.
- [SPI Flash Configuration Time](#): Details the steps for determining the maximum clock frequency.
- [SPI Flash Using Xilinx Vivado Design Suite](#)
 - [SPI Flash Using ISE Design Suite](#): Describes the option for generating the bitstream.
 - [Preparing the SPI Flash Programming File: Vivado Design Suite IDE Example](#): Provides the instructions to generate a SPI flash data file.
 - [Programming the SPI Flash In-System: Vivado Design Suite IDE Example](#): Provides instructions to program the SPI Flash.
- [SPI Flash Using ISE Design Suite](#)
 - [SPI Flash Configuration Options](#): Describes the options for generating the bitstream.
 - [Preparing the SPI Flash Programming File](#): Provides instructions to generate a SPI flash data file.
 - [Programming the SPI Flash In-System](#): Provides instructions to program the SPI flash.
- [Appendix](#)
 - [Programming Time](#)

- [Debugging and Troubleshooting Guidelines](#)

SPI Flash Basics

This section reviews the SPI flash pins and their connections to 7 series FPGAs. Details about the SPI flash configuration options, such as density selection, data width, and FPGA configuration time, are also included.

Figure 2 shows the basic connectivity between 7 series FPGAs and the SPI flash with a x1 data width. The read and address instructions are sent from the FPGA to the SPI flash via the master-out-slave-in (MOSI) pin. The data is returned from the SPI flash via the master-in-slave-out (MISO) pin. SCK is the clock pin and SS is the active-Low slave select pin. A x2 data width has the same connections, however the MOSI becomes bidirectional and is used as an additional data pin.

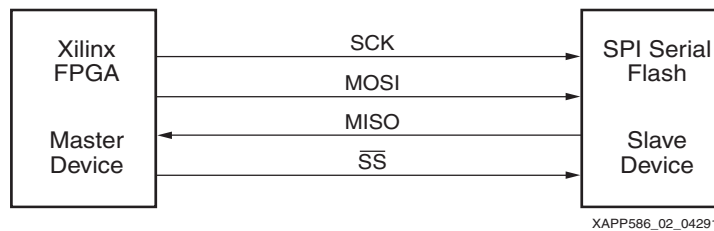


Figure 2: Basic SPI Flash to FPGA Connections—x1 Data Width

In addition to the pins described above, the SPI flash can have additional pins that can be used to control other special functions. These additional pins can vary with the SPI flash vendor, however two common special function pins are *hold* and *write protect*. Newer SPI flash devices enable these hold and write protect pins with a dual function of additional data output pins to increase the data bus up to 4 bits.

Table 1: SPI Flash Pin Names

Pin Names Used in This Document	Alternate Pin Names Used by Other Vendors	Pin Function
SCK	C, CLK	Clock for SPI flash instructions and data
MOSI	DQ0, DI, SI, IO0	Master out; slave in. Can be an additional data pin in x2 or x4 output modes
MISO	DQ1, IO1, SO, DO	Master in; slave out
\overline{SS}	S/, CS/	Slave select
\overline{HOLD}	DQ3, IO3	Hold or pause without deselecting the device. Can be an additional data pin in x4 output mode.
\overline{W}	DQ2, WP/, IO2	Write protect portions of the SPI flash memory. Can be an additional data pin in x4 output mode.

Selecting an SPI Flash

The first criteria in selecting a SPI flash is density. For many designs this means selecting a flash device that is large enough to store the configuration bitstream of the target FPGA. For some designs, other considerations narrow the options of which flash to use, such as the need to store multiple bitstreams, or have a daisy chain of FPGAs to be configured, or configuration speed.

The minimum density required is always the size of the FPGA configuration bitstream. See [UG470, 7 Series FPGAs Configuration User Guide](#) for details. If the design requires multiple bitstreams, multiply the size of the bitstream accordingly. The Xilinx tools allow bitstream compression, however it is not recommended to rely on compression when determining SPI flash size because compression varies greatly with the user's design and is not predictable.

Some designs require the FPGA to configure in a specified amount of time. In this case, the designer should consider using a SPI flash that allows for the fastest read-clock rate and ensuring the support of x4 data width read operations (sometimes called quad output fast read in SPI flash data sheets).

The designer also must consider the I/O voltage compatibility. The Artix™-7 and Kintex™-7 families support configuration I/O voltages up to 3.3V and the Virtex®-7 family supports up to 1.8V. The SPI flash vendors generally use the same voltage supply for the core voltage and the I/O voltage. However, some vendors can use a separate I/O voltage pin. The differences in the type of voltage supplies affect the ability to use different vendors as a second source. See [XAPP1328, Configuration or Boot with NOR Flash for Sourcing Flexibility and Cost Reduction Application Note](#).

The list of the flash devices that are tested and supported by the Xilinx Design Suite tools for the 7 Series families can be found at ISE Design Suite:

https://www.xilinx.com/html_docs/xilinx14_7/pim_r_supported_spi_bpi_proms.htm

See [UG908, Vivado Design Suite User Guide: Programming and Debugging](#) for a list of supported devices.

SPI Flash Configuration Interface

[Figure 3](#) shows the pins of the FPGA required for SPI flash configuration. Many of these pins are also required for other configuration methods and are not specific to SPI flash configuration.

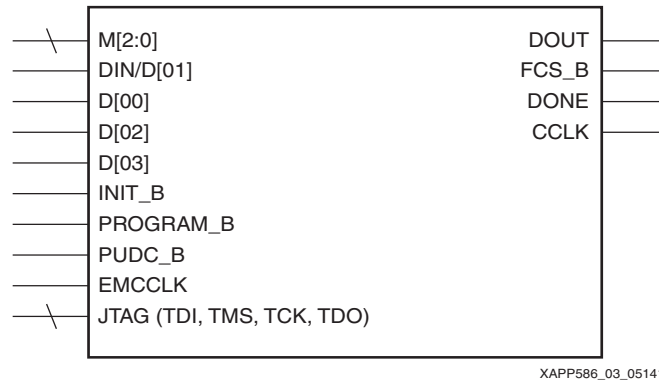


Figure 3: 7-Series FPGA SPI Flash Configuration Interface Block Diagram

Table 2 details the functions of the FPGA pins during SPI flash configuration. In addition to the pins mentioned in the [SPI Flash Basics](#) section, other configuration interface signals are shown which give status information and control of FPGA configuration.

Table 2: SPI Flash Configuration Pins

7-Series FPGA Pin Name	FPGA Direction	Dedicated or Dual Purpose	Description
M[2:0]	Input	Dedicated	Determines the FPGA configuration mode. M[2:0] = 001 for master SPI flash mode. Connect each mode pin either directly, or via a 1 kΩ (or stronger) resistor, to VCCO_0 or GND.
DIN/D[01]	Input	Dual-Purpose	Receives data from the SPI flash MISO pin. In x1 mode, this is the only data input pin to the FPGA.
D[00]	Input/Output	Dual-Purpose	At the start of FPGA configuration, this pin drives the SPI flash's MOSI pin and delivers a read instruction and the address. In x1 mode, this pin is output only. In x2 and x4 data width modes, this pin is bidirectional and receives data from the SPI flash.
D[02]	Input	Dual-Purpose	Receives data bit 2 from the SPI flash in x4 data width mode.
D[03]	Input	Dual-Purpose	Receives data bit 3 from the SPI flash in x4 data width mode.
INIT_B	Bidirectional, Input, Output, Open-drain	Dedicated	Driven Low during FPGA power-up, indicating the FPGA is performing self-initialization prior to initiating configuration. After self-initialization is complete, and before the mode pins are sampled, this pin can be externally driven Low to delay configuration. After mode pins are sampled, INIT_B becomes open drain. During configuration bitstream loading, this pin acts as an indicator for CRC error.
PROGRAM_B	Input	Dedicated	Active-Low asynchronous full-chip reset.
PUDC_B	Input	Dual-Purpose	Controls I/O (except bank 0 dedicated I/Os) pull-up resistors during configuration. This pin must be externally terminated. 0 = Pull-up resistors during configuration 1 = 3-state output during configuration
EMCCLK	Input	Dual-Purpose	An input for supplying an external configuration clock (optional). This clock is then internally routed to the CCLK FPGA pin.

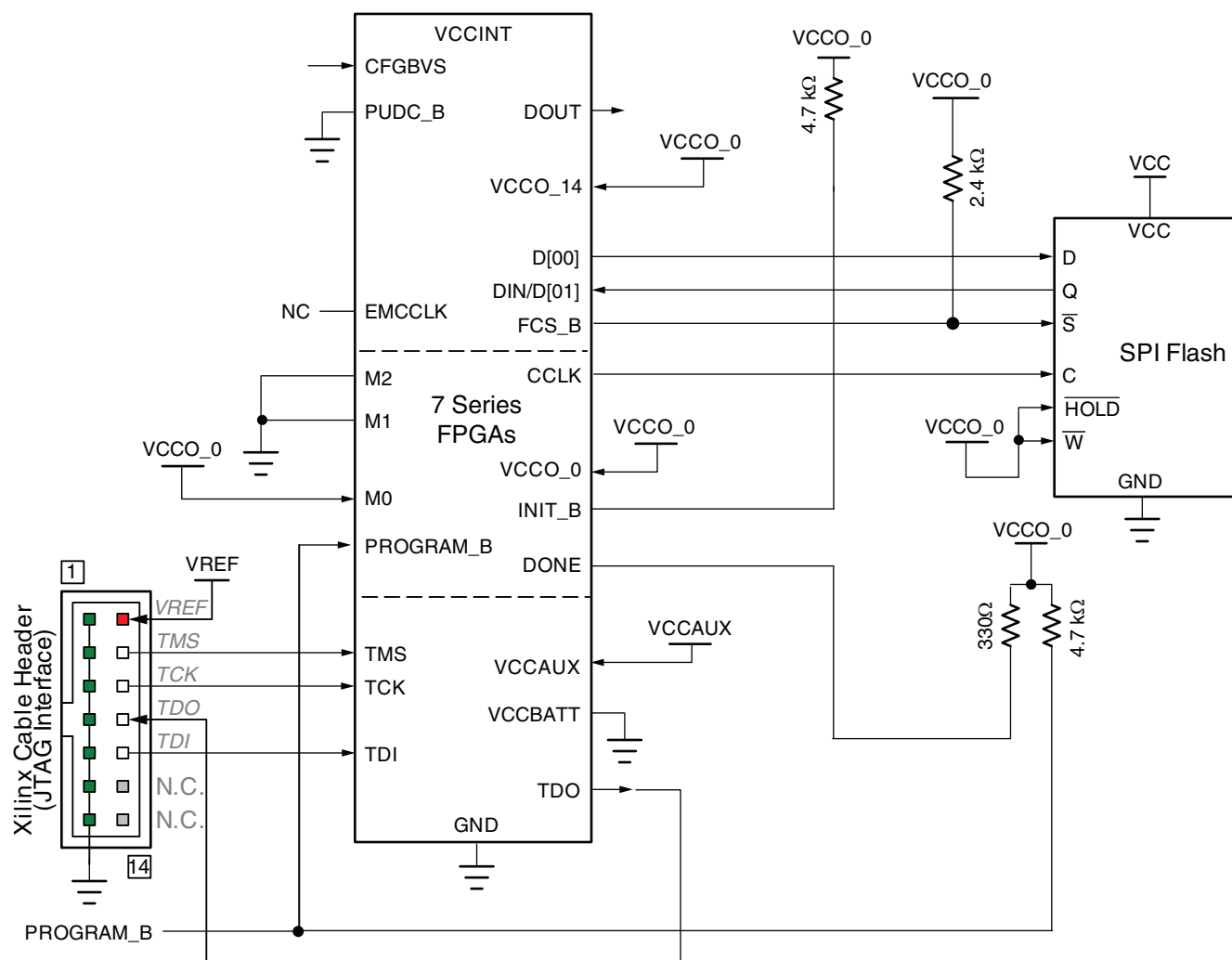
Table 2: SPI Flash Configuration Pins (Cont'd)

7-Series FPGA Pin Name	FPGA Direction	Dedicated or Dual Purpose	Description
CCLK	Input/Output	Dedicated	Initial configuration clock source for all configuration modes except JTAG. Drives the SCK pin of the SPI flash.
FCS_B	Output	Dual-Purpose	Drives the SPI flash SS/ pin Low during configuration to enable the SPI flash.
DOUT	Output	Dual-Purpose	Only used in x1 SPI flash configuration modes when daisy chaining multiple FPGAs. See UG470, 7 Series FPGAs Configuration User Guide for more information on daisy-chain configuration.
DONE	Output/ Open-Drain	Dedicated	Active-High signal indicating configuration is complete. 0 = FPGA not configured 1 = FPGA configured
CFGBVS	Input	Dedicated	For the Artix-7 and Kintex-7 families, this pin determines the voltage standard supported in the configuration I/O banks. For the Virtex-7 family, the only allowable configuration voltage is 1.8V or less. See the Configuration Bank Voltage Select section in UG470, 7 Series FPGAs Configuration User Guide for additional information. 1 = 2.5V or 3.3V 0 = 1.8V or less
TDI	Input	Dedicated	JTAG test data input port ⁽¹⁾
TMS	Input	Dedicated	JTAG test mode select input port ⁽¹⁾
TCK	Input	Dedicated	JTAG test clock input port ⁽¹⁾
TDO	Output/ open-drain	Dedicated	JTAG test data out port ⁽¹⁾

Notes:

1. JTAG pins are optional for the SPI flash configuration interface, but are required for indirect SPI flash programming.

[Figure 4](#) and [Figure 5](#) are nearly identical. [Figure 4](#) illustrates the connections for a SPI flash configuration solution in x1 or x2 data width mode. [Figure 5](#) illustrates the connections for a SPI flash configuration solution in x4 data width mode. The only difference between [Figure 4](#) and [Figure 5](#) is the handling of the $\overline{\text{HOLD}}$ and the $\overline{\text{W}}$ pins, which are terminated in x1 and x2 data width modes and are connected to the 7 series FPGAs in x4 data width mode.

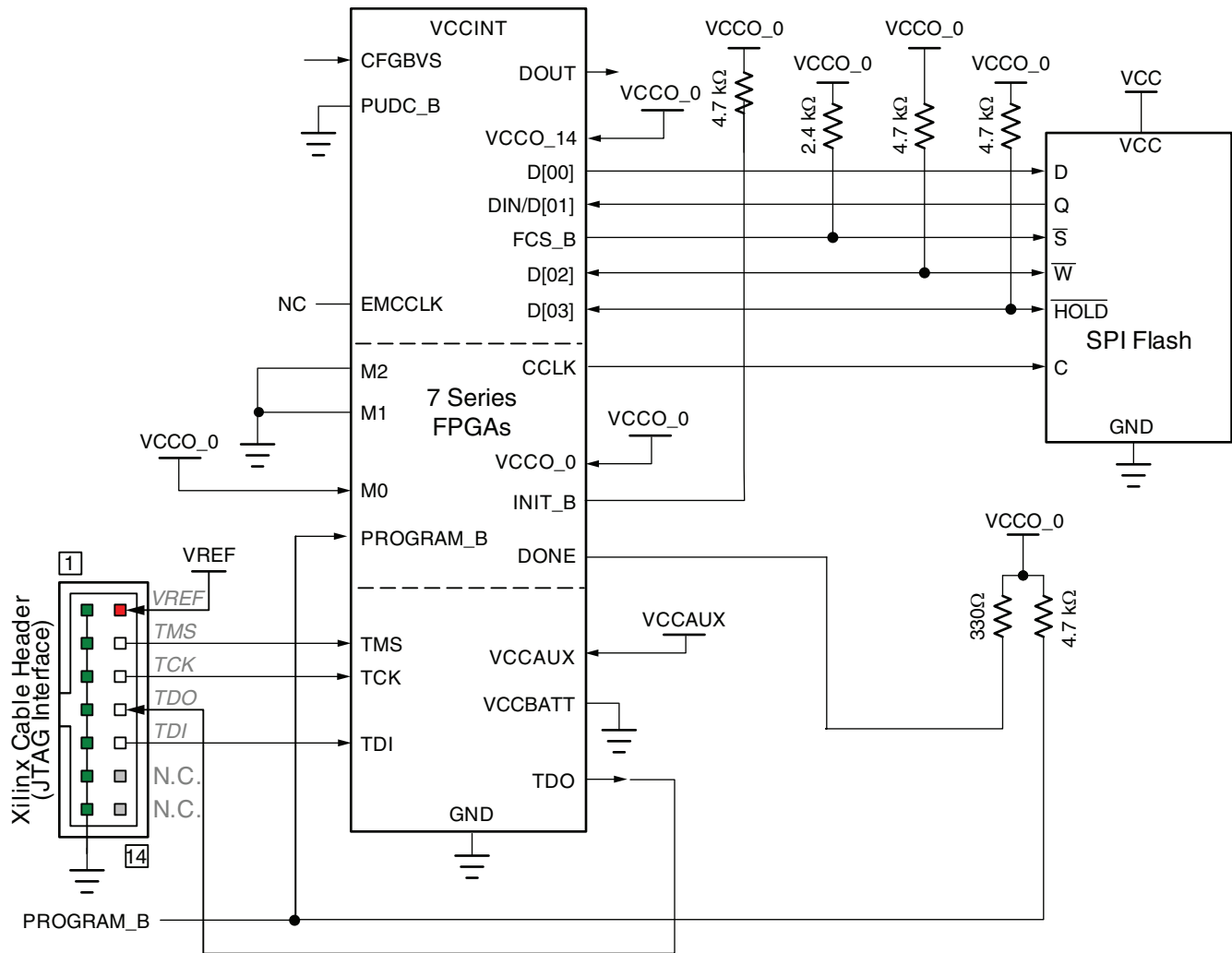


XAPP586_04_051112

Figure 4: SPI Flash x1/x2 Configuration Schematic

Notes relevant to Figure 4, SPI flash x1/x2 configuration schematic:

1. DONE is by default an open-drain output. An external pull-up resistor is recommended.
2. INIT_B is a bidirectional open-drain pin. An external pull-up resistor is required.
3. CCLK signal integrity is critical.
4. Series resistors should be considered for the datapath from the SPI flash to the FPGA to minimize overshoot. The proper resistor value can be determined from simulation.
5. The VCCO supply of the 7 series FPGAs must be compatible with the VCC of the SPI flash. CFGBVS=GND when VCCO_0=1.8V or CFGBVS=VCCO_0 when VCCO_0=3.3V. (Flash [I/O] voltage, VCCO_0, VCCO_14, and CFGBVS must all be aligned hto support same voltage: 1.8V or 3.3V. 2.5V is also possible but not used since there is not any 2.5V SPI flash.
6. VCCBATT is the power source for the AES key stored in SRAM. For details about AES encryption, see [UG470, 7 Series FPGAs Configuration User Guide](#).



XAPP586_05_012113

Figure 5: 7 Series FPGA SPI Flash x4 Configuration Schematic

Notes relevant to [Figure 5](#) SPI flash x4 configuration schematic:

1. DONE is by default an open-drain output. An external pull-up resistor is recommended.
2. INIT_B is a bidirectional open-drain pin. An external pull-up resistor is required.
3. CCLK signal integrity is critical.
4. Series resistors should be considered for the datapath from the SPI flash to the FPGA to minimize overshoot. The proper resistor value can be determined from simulation.
5. The VCCO supply of the 7 series FPGAs must be compatible with the VCC of the SPI flash. CFGBVS=GND when VCCO_0=1.8V or CFGBVS=VCCO_0 when VCCO_0=3.3V. (Flash [I/O] voltage, VCCO_0, VCCO_14, and CFGBVS must all be aligned to support same voltage: 1.8V or 3.3V. 2.5V is also possible but not used since there is not any 2.5V SPI flash).
6. VCCBATT is the power source for the AES key stored in SRAM. For details about AES encryption, see [UG470, 7 Series FPGAs Configuration User Guide](#).

Power-On Considerations for SPI Flash

At power-on, a race condition exists between the FPGA and the SPI flash. After the FPGA completes a self-initialization, it transmits a read command to the SPI flash to retrieve the configuration data, at which time the SPI flash must be ready to respond to this command. Refer to the specifications in the SPI flash data sheets for the time required for the flash to complete its own self-initialization (see the [References](#) section for the list of SPI flash data sheets). In general the self-initialization time (also called power-on reset) of the 7 series FPGAs is an order of magnitude (milliseconds) more than the SPI flash (hundreds of microseconds), however the designer should evaluate the time. This is certainly more important if the SPI flash and the FPGA are on different supply rails.

Configuring the FPGA from SPI Flash

After the FPGA finishes self-initialization, INIT is released and the FPGA samples the mode pins (M[2:0]) to determine which configuration mode to use. With the mode pins M[2:0] = 001, the FPGA then begins to output clocks on CCLK at a frequency of approximately 3 MHz. Shortly afterwards, FCS_B drives Low, followed by the OPCODE for a x1 fast-read instruction and address on the D[00] pin as shown in [Figure 6](#).

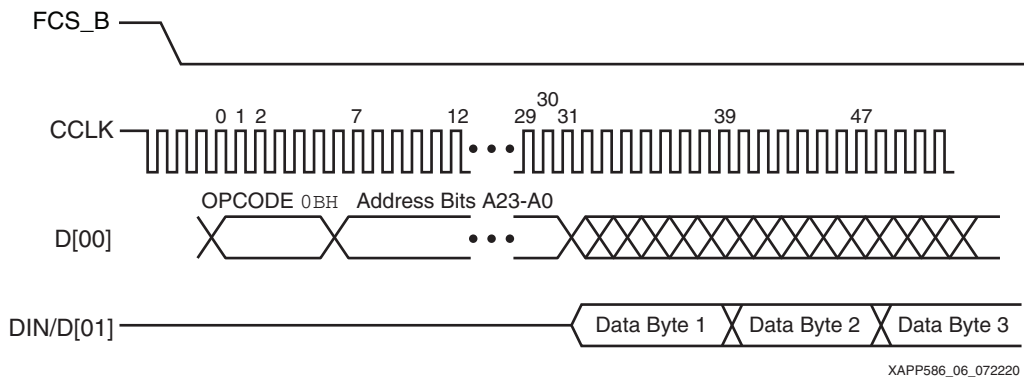


Figure 6: 7 Series FPGA SPI Flash Timing

Data is initially transmitted from the SPI flash to the FPGA in x1 mode. The commands to switch to an external clock, x2 or x4 bus width, or other options are all contained within the early portion of the bitstream. After reading these options, the FPGA makes mid-configuration adjustments.

The default behavior is for data to be output from the SPI flash on the falling edge of CCLK and captured by the FPGA on the rising edge of CCLK. The default behavior can be changed to capture on the falling edge by enabling the option set_property BITSTREAM.CONFIG.SPI_FALL_EDGE: YES" (Vivado Design Suite) and bitgen -g SPI_FALL_EDGE:yes (ISE Design Suite).

SPI Flash Configuration Time

[Equation 1](#) should be used to determine the maximum frequency that the SPI flash can safely operate and still deliver the bitstream reliably. The SPI flash delivers data on the falling edge of the clock. The default for 7 series FPGAs is to capture data on the rising edge of the clock. For the equation below to be true, it is assumed that the SPI flash configuration option that enables the FPGA to capture data on the falling edge is enabled for Vivado Design Suite (set_property BITSTREAM.CONFIG.SPI_FALL_EDGE: YES) and for ISE Design Suite (bitgen -g spi_fall_edge: yes). This allows the full clock cycle to be utilized, therefore higher frequencies can be reached.

[Equation 1](#) describes the SPI flash configuration frequency calculation.

Maximum configuration clock frequency =

Equation 1

$$\frac{1}{\text{Flash clock to out}(T_{SPITCO}) + \text{FPGA data setup}(T_{SPIDDC}) + \text{Trace Propagation Delay}(T_{TPD})}$$

In the 7 series FPGAs, the frequency tolerance of the internal oscillator (fMCCKTOL) is significant. If minimum configuration time is critical, it is recommended the designer use an external clock (EMCCLK).

After the optimum configuration rate is determined, the designer needs to divide the total bitstream size by the configuration rate to determine the total configuration time in x1 mode. If using x2 or x4 data widths, divide by the width.

Configuration Clock Calculation Example

This section demonstrates the steps to determine the maximum operating configuration frequency for the SPI flash solution.

The SPI flash selected is N25Q128A13 and the target is a Kintex-7 XC7K325T FPGA.

The SPI flash clock to out, per the SPI flash data sheet, has multiple values depending on VCC and the capacitance on the output pin.

These values are as fast as 5 ns and as slow as 7 ns according to the N25Q128A13 data sheet. In this example, the value of 7 ns represents the SPI flash operating with a load of 30 pF or less.

The FPGA setup time for a Kintex-7 XC7K325T FPGA is 3.0 ns (for the current value, refer to [DS182](#), *Kintex-7 FPGAs Data Sheet*).

The trace propagation delays from the CCLK to C pin and the longest propagation delay of any of the data pins provide T_{TPD} . For this example, a rule of thumb of 165 ps per inch and a trace length of 6 inches from the FPGA to the SPI flash are used. (For more accurate results, other techniques such as IBIS simulation are recommended.) A trace value of 12 inches at 165 ps/inch gives 2.0 ns.

$1 / (7 \text{ ns} + 3.0 \text{ ns} + 2.0 \text{ ns})$ yields a clock frequency of 83.3 MHz.

The designer should consider using the FPGA's internal oscillator and the closest value to 83.3 MHz is 66 MHz. However, the frequency tolerance (fMCCKTOL) for the XC7K325T is $\pm 50\%$ (for the current value, refer to [DS182](#), *Kintex-7 FPGAs Data Sheet*) so this clock frequency could potentially be $(66 \text{ MHz} \times 1.5) = 99 \text{ MHz}$, which would be too fast for the calculated maximum.

The next fastest configuration rate is 50 MHz, which has a maximum frequency of $(50 \text{ MHz} \times 1.5) = 75 \text{ MHz}$. This rate is well below the calculated maximum and nominally operates at 50 MHz, which is well below the desired 83.3 MHz.

The bitstream of a Kintex-7 XC7K325T FPGA is 91,548,896 bits.

$91,548,896 / 50,000,000 = 1.83$ seconds to configure XC7K325T in x1 data width @ 50 MHz

Assume that an 80 MHz oscillator was already on the board for another application or device, and so can serve a dual purpose as the clock for the FPGA configuration.

$91,548,896 / 80,000,000 = 1.144$ seconds to configure XC7K325T x1 data width @ 80 MHz

$1.144 / 4 = 286$ ms to configure the XC7K325T FPGA in SPI flash x4 data width

Design Considerations for the External Master Clock

When using the external master clock (EMCCLK) as a configuration clock source, EMCCLK must be included in the user's design. Not doing so results in the FPGA failing to complete the start-up sequence. There are no special design requirements needed when using the internal oscillator for FPGA configuration.

SPI Flash Using Xilinx Vivado Design Suite

SPI Flash configuration Options

This section briefly overviews the various device configuration properties for 7 Series FPGA. [Table 3](#) lists the necessary options to properly generate a configuration bitstream that is compatible with the SPI flash. If the option is unspecified, the default value is used.

Table 3: 7 Series Properties

Settings	Default Value	Possible Values	Description
BITSTREAM.CONFIG.EXTM ASTERCCLK_EN	Disable	Disable, Div-1, Div-2, Div-4, Div-8	When set to disable, an internal oscillator is the source of the configuration clock (CCLK). When set to Div-1, Div-2, Div-4, or Div-8, the source of the CCLK is switched to the external clock source from the EMCCLK pin at the beginning of the configuration procedure. The CCLK frequency becomes the frequency of the EMCCLK source divided by the Div setting number.
BITSTREAM.CONFIG.CONFIGRATE	3	3, 6, 9, 12, 16, 22, 26, 33, 40, 50, 66	When the default internal oscillator is the source of the CCLK, this option sets the CCLK frequency. See the data sheet for the range of variation of the CCLK frequency when based on the internal oscillator.
BITSTREAM.CONFIG.SPI_32 BIT_ADDR	No	No, Yes	Enables SPI 32-bit address style, which is required for SPI devices with storage of 256 Mb and larger
BITSTREAM.CONFIG.SPI_BU SWIDTH	None	None, 1, 2, 4	Sets the SPI bus to Dual (x2) or Quad (x4) mode for Master SPI configuration from third party SPI flash devices.
BITSTREAM.CONFIG.SPI_FAL L_EDGE	No	No, Yes	Sets the FPGA to use a falling edge clock for SPI data capture. This improves timing margins and may allow faster clock rates for configuration.
CONFIG_VOLTAGE	1.8	1.5, 1.8, 2.5, 3.3	Informs the tools of the voltage used for configuration. Set this to the voltage being used on the configuration banks.
CFGBVS	GND	GND, VCCO	Informs the tools of the voltage driving the CFGBVS pin. If the VCCO_0 supply for bank 0 is supplied with 2.5V or 3.3V, then the CFGBVS pin must be tied High (i.e. connected to VCCO_0) and CFGBVS property must be set to VCCO. If VCCO_0 is supplied with 1.5V or 1.8V, then the CFGBVS pin must be tied Low (i.e. connected to GND) and the CFGBVS property must be set to GND.
BITSTREAM.GENERAL.COM PRESS	False	True, False	Uses the multiple frame write feature in the bitstream to reduce the size of the bitstream, not just the Bitstream (.bit) file. Using compress does not guarantee that the size of the bitstream shrinks.

These options are available through Device Properties in the Vivado Design Suite shown in the following figure.

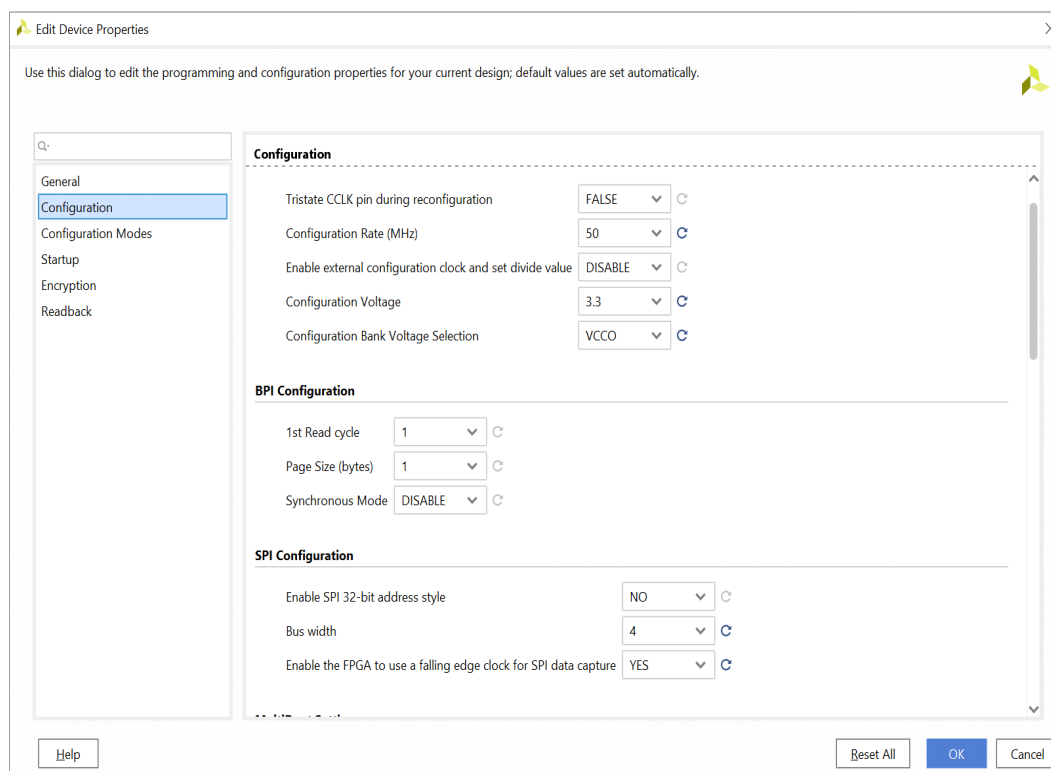


Figure 7: Bitstream Configuration Option

The Vivado IDE automatically sets the configuration properties in the design constraints (XDC) file, based on selections made in the Device Properties dialog, shown in Figure 8. Alternatively, the following Tcl commands can be manually added to the XDC file:

```
set_property BITSTREAM.GENERAL.COMPRESS False [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 50 [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
set_property BITSTREAM.CONFIG.SPI_32BIT_ADDR No [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property BITSTREAM.CONFIG.SPI_FALL_EDGE YES [current_design]
```

Note: The constraint example values above were used for the KC705 evaluation board.

Preparing the SPI Flash Programming File: Vivado Design Suite IDE Example

Use the following IDE flow in your Vivado tools design based in Project Mode to generate a bitstream for master SPI x4 configuration:

1. Add the recommended constraints from the SPI Flash Configuration Options to the designs constraint file. This step should ideally be done before synthesis and implementation. Adding constraints after synthesis or implementation causes these runs to be marked as out-of-date upon saving the constraints file. If no other constraints (or design files) are otherwise changed, the runs can safely be forced up-to-date using the Force-Up-to-Date option as described in **Generating SPI Configuration Constraints: IDE Flow**.
2. In the Flow Navigator, locate the Program and Debug tab and left click the Bitstream Settings button as shown in [Figure 8](#).



Figure 8: Flow Navigator

3. In the Project Settings dialog, select `-bin_file` which instructs the `write_bitstream` command to generate a headerless bitstream for programming the SPI flash. Click **OK** to accept this change.

Note: See [UG908](#), *Vivado Design Suite User Guide: Programming and Debugging* for details on creating an alternative `.mcs` programming file using `write_cfgmem`.

4. In the Flow Navigator, click **Generate Bitstream** under the Program and Debug flow or select **Generate Bitstream** from the Flow menu.



Figure 9: Flow Navigator

This starts the Generate Bitstream flow. Upon completion, if no errors are encountered, a bitstream that can be used for SPI programming is found at:

```
<Project_Dir>\<Project_Name>.runs\impl_1\
```

Because the `-bin_file` option was selected earlier, the bitstream was generated as a BIN file with a `.bin` extension in addition to the standard `.bit` file. The `.bin` file is used for programming the SPI flash in SPI Programming File Generation.

Generating Bitstream: Vivado Tools TCL Batch FILE Example

As an alternative to the IDE flow presented in the previous section, a TCL batch file is presented here that can be used with a non-project Vivado tools flow. There must have been a checkpoint created previously for this flow to succeed. This script assumes the checkpoint is saved in a subdirectory of the current directory from where the script is run:

```
## Non-project TCL flow to generate a BIN file used for SPI Flash programming

# Note: write_checkpoint must have been run prior to running this script open_checkpoint (post_route.dcp)

set_property BITSTREAM.GENERAL.COMPRESS FALSE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 50 [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
set_property BITSTREAM.CONFIG.SPI_32BIT_ADDR hNo [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property BITSTREAM.CONFIG.SPI_FALL_EDGE YES [current_design]
write_bitstream -verbose -force -bin_file spi_programming.bit
```

Note: The example above are constraints used on KC705 evaluation board.

Programming the SPI Flash In-System: Vivado Design Suite IDE Example

The Vivado Design Suite is capable of programming the SPI flash attached to the FPGA by downloading an indirect programming bitstream to the target FPGA that contains an SPI controller. The Vivado tools communicate with the SPI controller via a JTAG cable to transfer the programming file into the SPI flash. This in-system programming feature can enable the testing and debugging of multiple design iterations in the prototyping phase and for debugging. This feature is not intended for high-volume production programming. For production programming, consider solutions from BPM Microsystems or Data I/O.

Programming the SPI Flash: Vivado Design Suite IDE Example

The first step to programming the SPI flash in-system requires that the 7 series FPGAs is first loaded with an interface design that bridges the programming cable to the SPI flash. This step clears the contents of the FPGA. This SPI interface design leaves the unused FPGA pins floating. The user should be aware of this and ensure that this does not have any undesired effects on other devices attached to the FPGA. For example, certain I/O pins might need to remain Low or High during SPI flash programming. These pins would normally be pulled low or high by the final FPGA design. In these cases, the designer might need to add external pull-down or pull-up resistors on these pins to ensure correct behavior during SPI programming.

The following demonstration targets the Kintex-7 XC7K325T FPGA and the Micron MT25QL128 SPI flash present on the KC705 Kintex-7 FPGA evaluation board.

Ensure the board is powered and the USB cable is attached. In the case of the KC705 board, a standard USB-to-micro-USB cable is used instead of a platform cable USB II. In most cases, there is no specialized hardware on the board to handle the PC-to-JTAG interface so a Xilinx Platform Cable USB II or other supported cable is required.

1. Ensure the board is connected and a programming cable is connected. Turn on the power to the target board.
2. Open the Vivado tools Hardware Manager by selecting **Open Hardware Manager** under the Program and Debug flow as show in [Figure 10](#). Alternatively, select **Flow > Open Hardware Manger**.



Figure 10: **Open Hardware Manager**

- Open and connect to the hardware target by clicking on the **Open Target** link in the Hardware Manager pane and then selecting **Auto Connect** as shown in Figure 11.

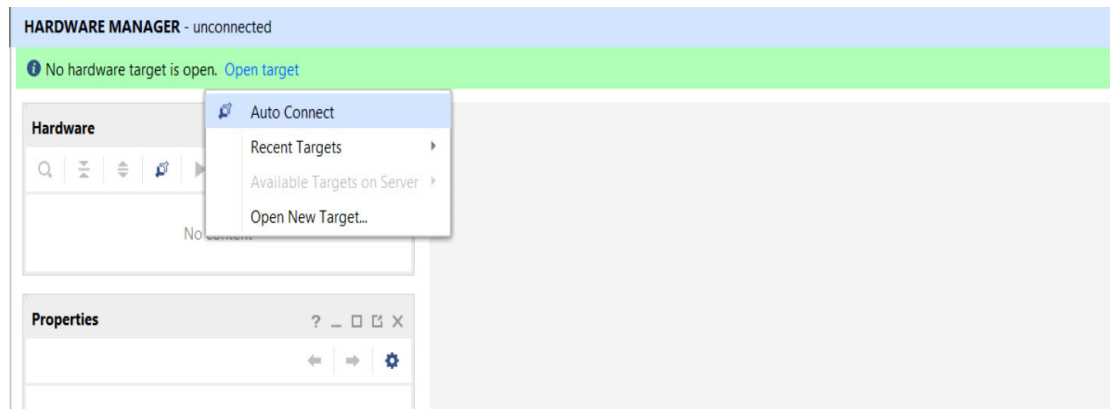


Figure 11: Hardware Manager

Note: This step should automatically connect to the 7 series FPGA target if it is connected to the workstation that is running the Vivado Design Suite. Ensure the cable is properly connected and the board is powered on. It is also possible to connect remotely to another workstation that is running hw_server and is physically connected to the target board via a programming cable. See the *Opening a New Hardware Target* section in [UG908, Vivado Design Suite User Guide: Programming and Debugging](#) for a list of supported devices for more details if this flow is needed or auto connection fails.

- Launch the Add Configuration Memory Device dialog by right clicking the target FPGA in the hardware manager as shown in Figure 12.

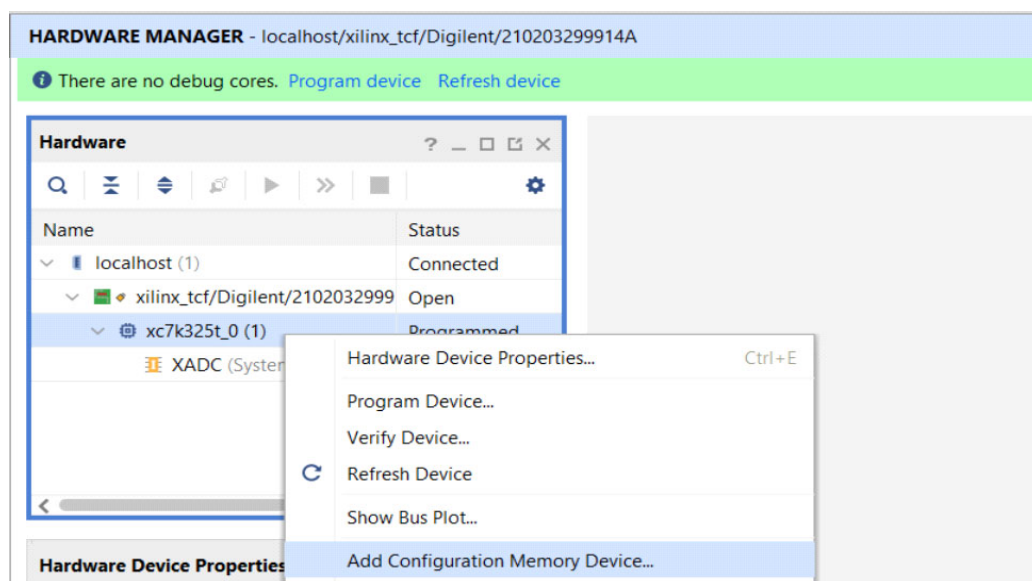


Figure 12: Add Configuration Memory Device

- In the Add Configuration Memory Device window, select the proper SPI flash from the list. The filter options and the search field can both be used to simplify the search and narrow down the options as shown in Figure 13. Click **OK**.

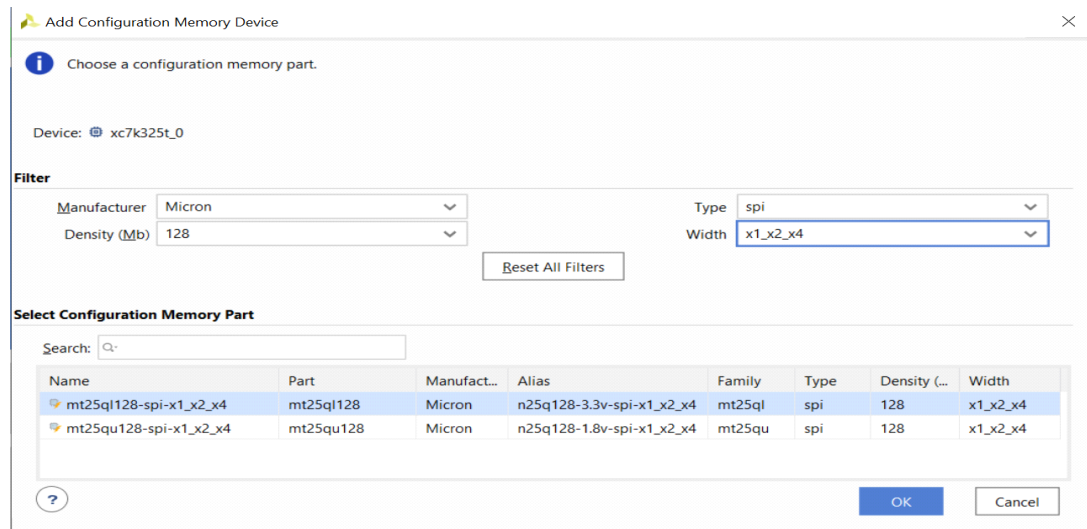


Figure 13: Add Configuration Memory Device

Note: For SPI widths x1, x2, or x4, use SPI flash devices that end with "-x1_x2_x4" (or possibly "-x1" or "-x1_x2"), but not "-x1_x2_x4_x8". The latter is only for use with master SPI x8 configuration mode.

Note: Flash selection present on KC705 board is used.

- Click **OK** in the Add Configuration Memory Device Completed dialog box that appears next to program the configuration memory devices at this time.
- In the Program Configuration Memory Device dialog, click on the browse button to the right of the Configuration file field and select the **.bin** created previously. By default, the **.bin** file is located in the **impl_1** project directory in the following subdirectory:

`<Project_Dir>\<Project_Name>.runs\impl_1\`

Note: The location of the Vivado tools project subdirectory is referred to as `<Project_Dir>` in this application note and the project name is referred to as `<Project_Name>`.

- Set the **Program Operations** as needed for your design and click **OK** when finished (Figure 14).
 - Address Range:** Specify to erase just enough sectors to cover the size of the programming file (**Configuration File Only**) or the entire SPI flash device (**Entire Configuration Memory Device**).
 - Erase:** Erase the sectors or the entire device as indicated by **Address Range**.
 - Blank Check:** Check that the sectors as indicated by **Address Range** are blank.
 - Program:** Program the SPI device with the **Configuration** file.
 - Verify:** Read back the programming contents and verify they match the **Configuration** file.

- **SVF Options:** An alternative way to program FPGAs and configuration memory devices is through the use of a serial vector format (SVF) file. The SVF file generated through Vivado® Design Suite and Vivado Lab Edition contains low level JTAG instructions and data required to program these devices. Refer to [UG908](#), *Vivado Design Suite User Guide: Programming and Debugging* for more information.

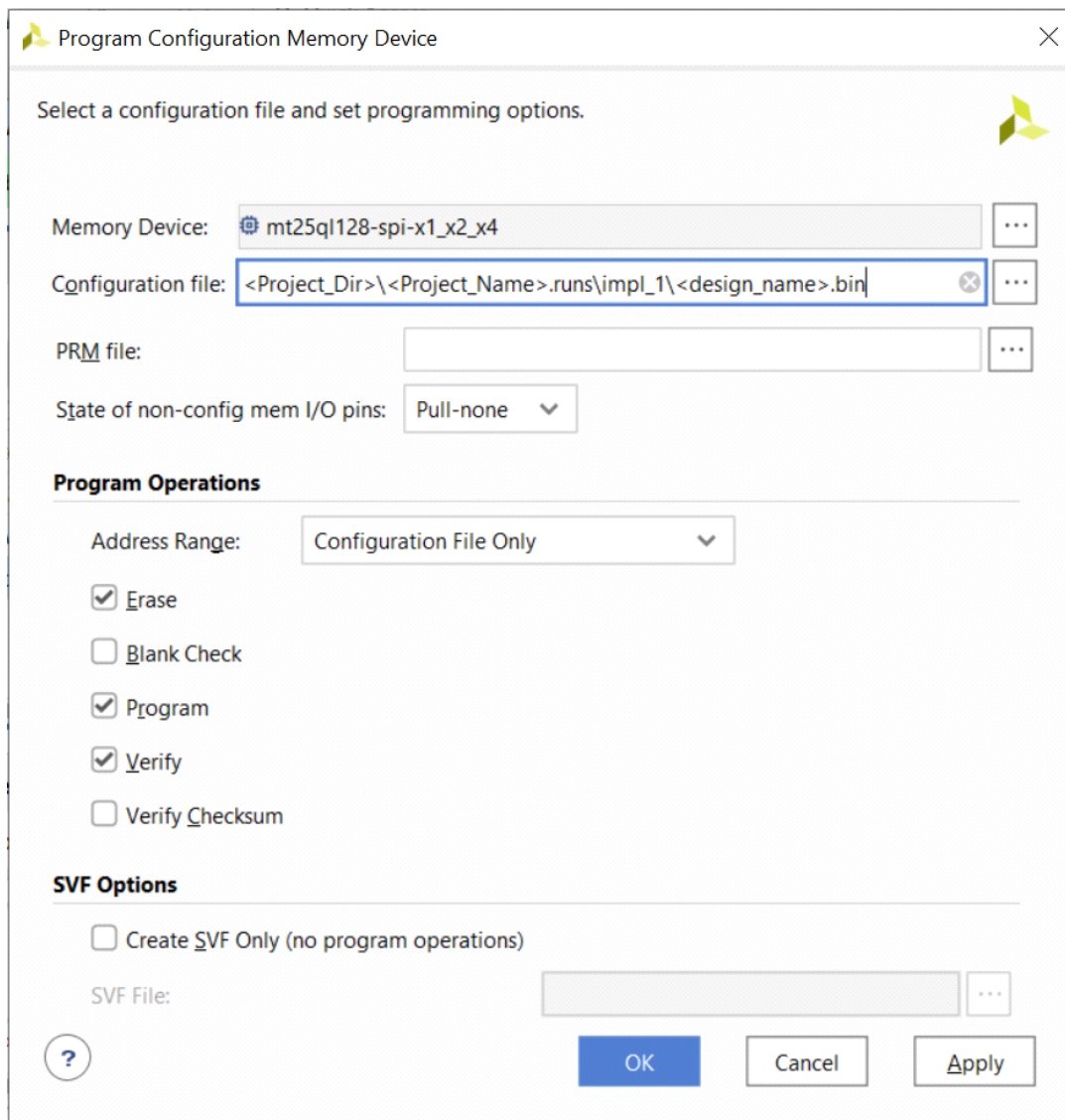


Figure 14: Program Configuration Memory Device

Programming should begin as soon as **OK** is clicked. A Programming Configuration Memory Device window is loaded showing the progress of the various stages selected in the Program Operations dialog.

Programming the SPI Flash: Vivado Tools TCL Batch File Example

As an alternative to the IDE flow presented in the previous section, a Tcl batch file is presented here that can be used with a non-project Vivado tools flow. For this flow to succeed, there must

have been a .bin file created previously. Save the following Tcl script as program_spi.tcl or adjust the instructions on sourcing with the Vivado tools as appropriate:

```
# Simple Vivado script to program a SPI flash
#
# The board should be connected via a programming cable and powered prior to
# running. The programming file is specified by the variable
# "programming_files" in this example
#
# Run this script from a Vivado command prompt:
# vivado -mode batch -source program_spi.tcl
open_hw_manager
connect_hw_server -url localhost:3121
current_hw_target [get_hw_targets]
open_hw_target
# Set the current Xilinx FPGA device. If more than one FPGA is in the JTAG
# chain, you may need to use the get_hw_devices command to help set the proper
# one with the current_hw_target command current_hw_device [lindex #[get_hw_devices] 0]
# Set my_mem_device variable for the SPI flash device get_cfgmem_parts can be
# used to find the supported flash. See "help get_cfgmem_parts" in the Vivado
# Tcl Console when in the Hardware Manager for options which can help narrow
# the search. UG908 also lists supported SPI flash devices. Be sure to use the
# parts ending with _x1_x2_x4 for width x1, x2, and x4 and parts ending with
# _x1_x2_x4_x8 for Dual Quad SPI (x8 width).
set my_mem_device [lindex [get_cfgmem_parts {mt25ql128-spi-x1_x2_x4}] 0]
# Set a variable to point the to BIN file to program
set programming_files {spi_programming.bin}
# Create a hardware configuration memory object and associate it with the
# hardware device. Also, set a variable with which to point to this object
set my_hw_cfgmem [create_hw_cfgmem -hw_device \
[lindex [get_hw_devices] 0] -mem_dev $my_mem_device]
# Set the address range used for erasing to the size of the programming file
set_property PROGRAM.ADDRESS_RANGE {use_file} $my_hw_cfgmem
# Set the programming file to program into the SPI flash
set_property PROGRAM.FILES $programming_files $my_hw_cfgmem
```

```
# Set the termination of unused pins when programming the SPI flash
set_property PROGRAM.UNUSED_PIN_TERMINATION {pull-none} $my_hw_cfgmem

# Configure the hardware device with the programming bitstream
program_hw_devices [lindex [get_hw_devices] 0]

# Set programming options

# Do not perform a blank check, but erase, program and verify
set_property PROGRAM.BLANK_CHECK 0 $my_hw_cfgmem

set_property PROGRAM.ERASE 1 $my_hw_cfgmem

set_property PROGRAM.CFG_PROGRAM 1 $my_hw_cfgmem

set_property PROGRAM.VERIFY 1 $my_hw_cfgmem

# Now program the part
program_hw_cfgmem -hw_cfgmem $my_hw_cfgmem
```

SPI Flash Using ISE Design Suite

SPI Flash Configuration Options

[Table 4](#) lists the BitGen options necessary to properly generate a configuration bitstream that is compatible with the SPI flash. These options are available through the Generate Programming File properties in the ISE tools shown in [Figure 15](#). If the option is unspecified, the default value listed first and displayed bold/ text is used.

Table 4: BitGen Configuration Options

BitGen Option	Description
-g spi_buswidth: 1 2 4	Selects the data width to be used when reading from the SPI flash.
-g spi_32bit_addr: No Yes	Set to Yes when targeting a SPI flash 256 Mb or larger. This instructs the FPGA to transmit a larger address space required for larger flash devices.
-g SPI_Fall_Edge: No Yes	Set to Yes when trying to achieve higher configuration speeds. Yes = capture data on the falling edge of the clock No = capture data on the rising edge of the clock
-g ConfigRate: 3 6 9 12 16 22 26 33 40 50 66	Sets the approximate configuration clock frequency driven by the 7 series FPGAs to the SPI flash when using the internal oscillator (in MHz). The actual values for this option can be seen in UG628, Command Line Tools User Guide and might vary depending on the selected FPGA.
-g ExtMasterCclk_en: Disable div-8 div-4 div-2 div-1	Instructs the FPGA to use the clock signal on EMCCLK as the configuration clock instead of the internal oscillator. Select div-1 to use the clock on EMCCLK at the same frequency and the other div options to divide down the EMCCLK clock by the appropriate value prior to output on CCLK.

To access these options from the ISE tools, right-click **Generate Programming File** then select **Process Properties > Configuration Options**. Alternatively, from the Process pull-down on the menu bar, select **Process Properties**, then select **Configuration Options**. From the Property display level pull-down at the bottom of the window, select **Advanced** to see all the options.

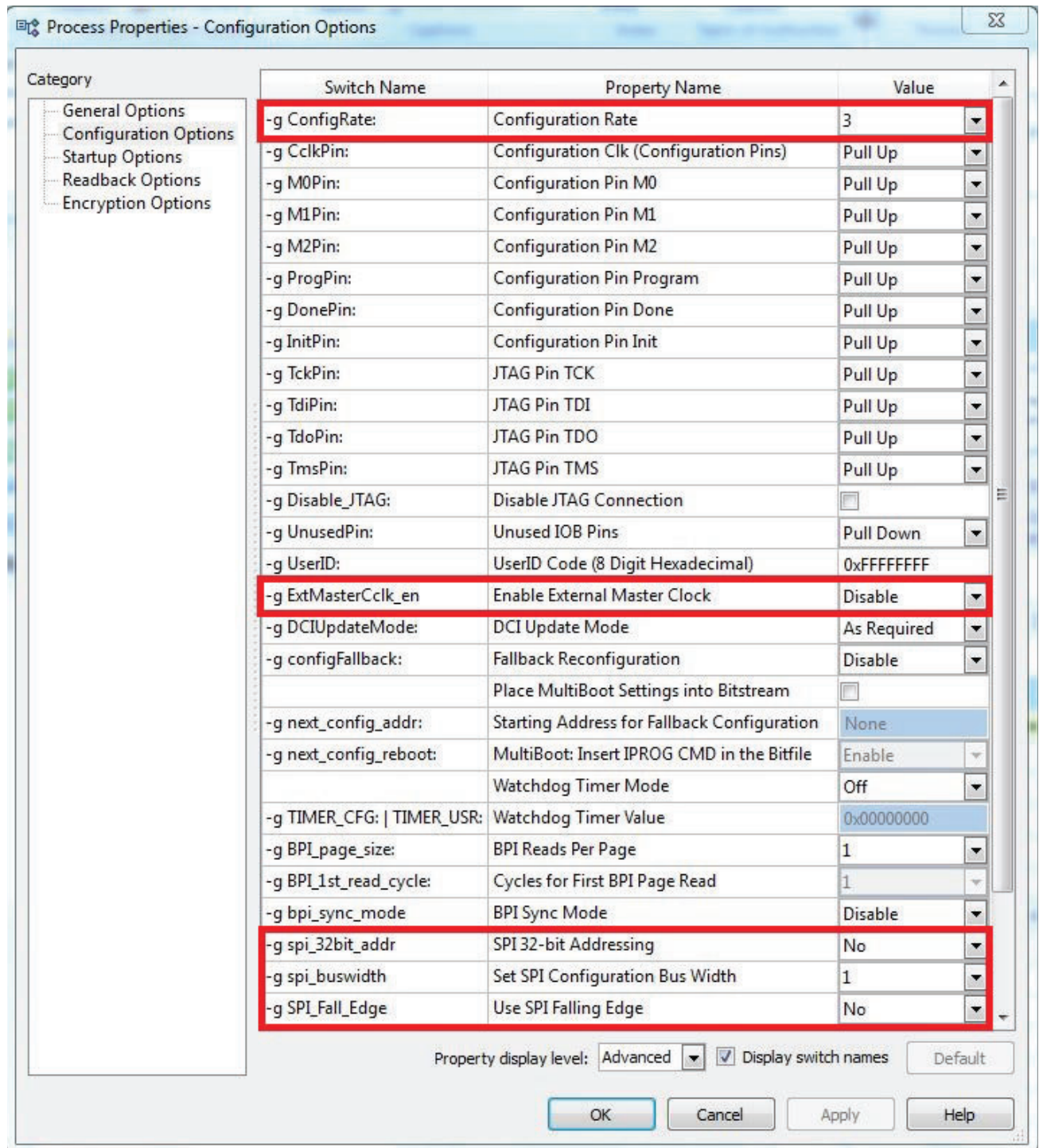


Figure 15: BitGen Configuration Options

XAPP586 07 042912

Preparing the SPI Flash Programming File

The Xilinx ISE tools take the FPGA bitstream (.bit) and generate a PROM file (.mcs) that is then used to program the SPI flash. PROMGen (the program that performs this task) is located within the iMPACT programming tool under the Create PROM File flow. Selecting this flow walks the user through the options for generating a file. The user can also generate the PROM file by using the command line and the underlying program PROMGen.

Table 5: PROMGen Options

Option	Description
-spi	Used to maintain the correct bit ordering required to configure the FPGA from a SPI flash device.
-p mcs exo	PROM output file format. Commonly accepted PROM file formats include Intel Hex (.mcs) and Motorola Hex (.exo). Only MCS is supported for Xilinx iMPACT indirect programming flows.
-s <size>	Specifies the SPI flash size in kilobytes. The SPI flash size must be a power of two for this option and the default setting is 64 KB.
-u <address>	Loads the bitstream starting at the specified address in an upward direction. If not entered, the default setting is at address 0. Most designs use address 0.
-o <filename>	Specifies the output file name.

The following command line example demonstrates how the options are used:

```
promgen -spi -p mcs -o spi_flash.mcs -s 16384 -u 0 design.bit
```

The example command line instructs PROMGen to:

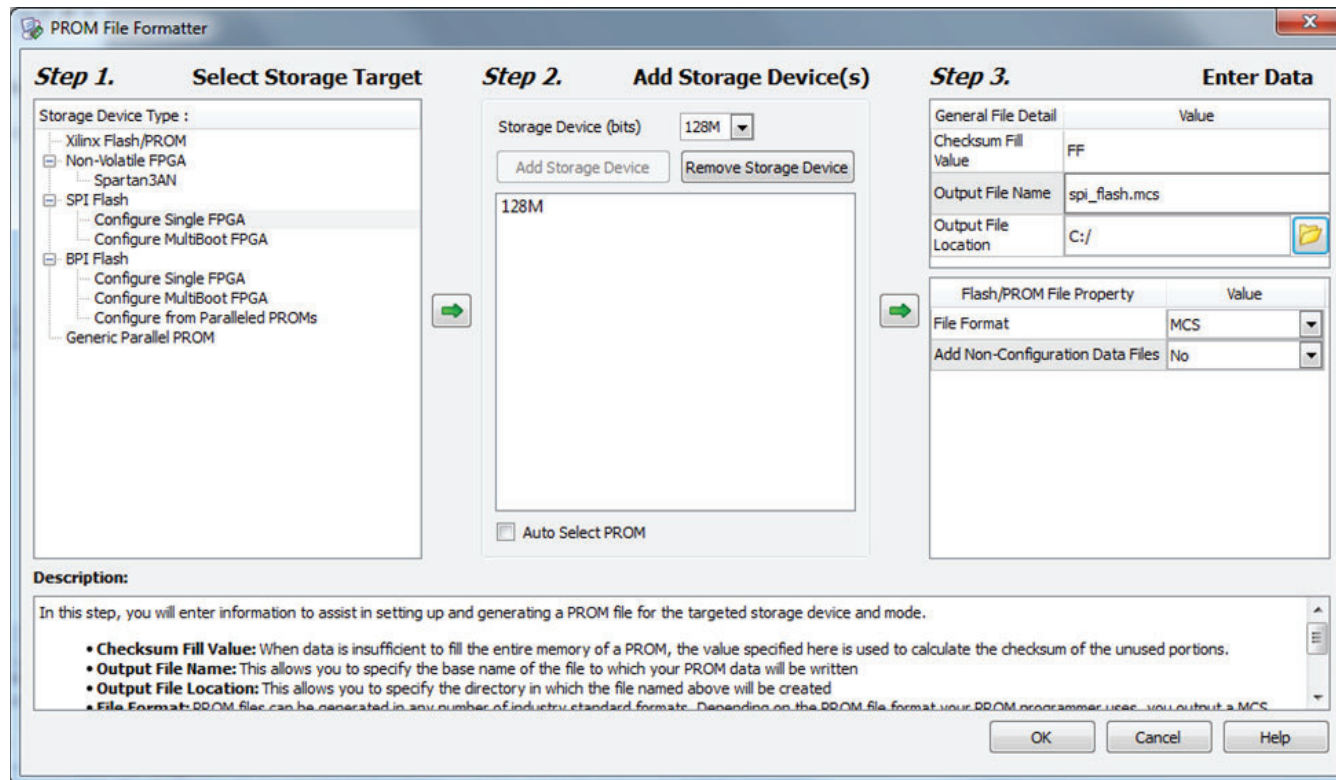
- Create a file with the bit ordering for a SPI flash, using the MCS file format, with the output file name `spi_flash.mcs`
- Select a 128 Mb flash target (16384 x 1024 bytes x 8 bits = 134,217,728, which is the actual size of a 128 Mb SPI flash).

Launch the ISE iMPACT tool by starting the Configure Target Device process. [Figure 16](#) shows the GUI interface for PROMGen.

The GUI steps to generate the flash file are:

1. Open iMPACT and select **Create PROM File (PROM File Formatter)** in the upper left box. [Figure 16](#) shows the GUI interface for PROMGen.
2. Use the wizard that appears to generate the PROM file:
 - a. Under Step 1 select **Storage Target** and under SPI Flash select **Configure Single FPGA**. Then select the green arrow that goes to Step 2.
 - b. Under Step 2 Add Storage Device, from the Storage Device (bits) pull-down menu, select the proper SPI flash density (128 Mb is the example shown in [Figure 16](#)), and click **Add Storage Device**. Then select the green arrow that goes to Step 3.

- c. Under Step 3 Enter Data, type the output file name, browse to the location to place the file, ensure the file format is MCS, then click **OK**.



XAPP586_08_042912

Figure 16: PROM File Formatter Flow

Next, a dialog appears that provides instructions to begin adding configuration bitstream files.

3. Select **OK** and add the target BIT file.
4. For a single design image, click **No** at the next prompt, which is used for multiple designs.
5. Click **OK** to confirm completion of the design file entry.
6. Double-click **Generate File** to create the MCS flash programming file. The console log displays the files generated.

Programming the SPI Flash In-System

The iMPACT programming tool offers the ability to program the SPI flash in-system utilizing the existing connections between the SPI flash and the FPGA. This is referred to as *indirect programming* because the SPI flash is programmed through the FPGA, not directly from iMPACT. Also needed to perform indirect programming is access to the JTAG pins of the FPGA, a JTAG programming cable such as the Xilinx Platform Cable USB II, and a computer that has the iMPACT programming tool installed. Using this setup, the SPI flash can be reprogrammed without removing it from the board, which is extremely useful for debugging in a lab environment.

Note: The indirect programming solution is not intended for high-volume production. For production programming, consider solutions from BP Microsystems or Data I/O.

The first step to programming the SPI flash in-system requires that the 7 series FPGAs be first loaded with an interface design that bridges the programming cable to the SPI flash. This step clears the contents of the FPGA. This SPI interface design leaves the unused FPGA pins floating. The user should be aware of this and ensure that this does not have any undesired effects on other devices attached to the FPGA. For example, certain I/O pins might need to remain Low or High during SPI flash programming, and these pins would normally be pulled Low or High by the final FPGA design. In such cases, the designer might need to add external pull-down or pull-up resistors on these pins to ensure correct behavior during SPI programming.

The following demonstration targets the Kintex-7 XC7K325T FPGA and the Micron N25Q128A13 SPI flash present on the KC705 Kintex-7 FPGA evaluation board.

Ensure the board is powered and the USB cable is attached. In the case of the KC705 board, a standard USB-to-micro-USB cable is used instead of a Platform Cable USB II. In most cases, there is no specialized hardware on the board to handle the PC-to-JTAG interface, therefore a Platform Cable USB II is required.

Figure 17 shows the GUI interface for the boundary-scan mode.

1. Start iMPACT by double clicking the **Configure Target Device** process and then select **Boundary Scan** in the upper left box (Figure 17).
2. Select the **Initialize Chain** icon to identify JTAG devices on the board.
3. The JTAG chain populates on the screen. The designer can choose to assign a bitstream to the FPGA because it does not matter for purposes of programming the SPI flash. In the example below the file `toplevel.bit` is assigned to the Kintex-7 FPGA.

Note: A box with SPI/BPI? appears above the FPGA. This indicates that no PROM file is currently assigned to any attached flash device to this FPGA.

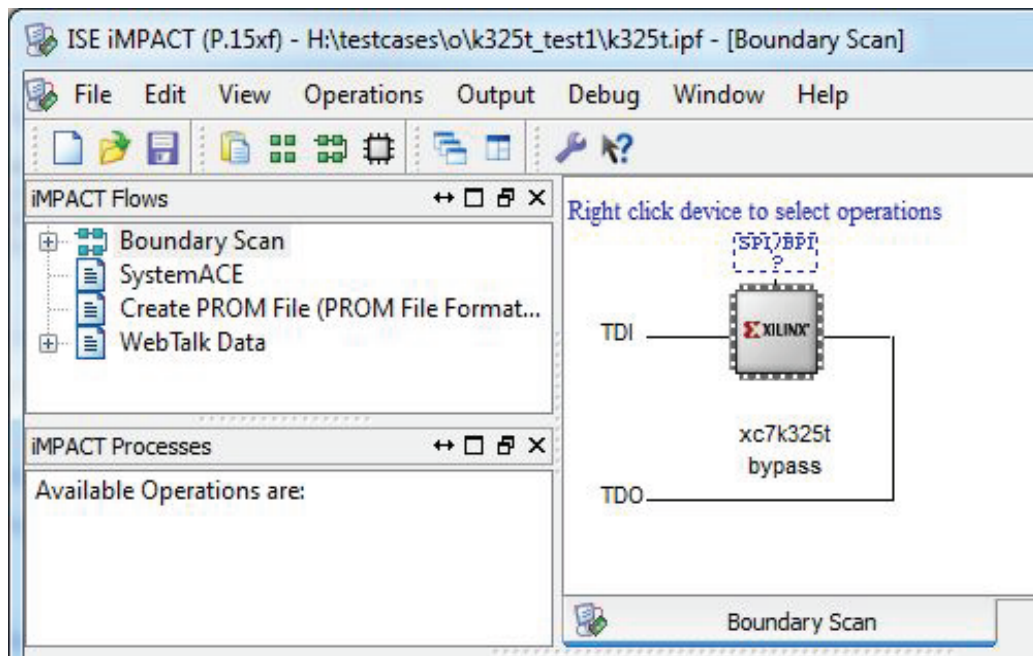


Figure 17: iMPACT JTAG Chain

4. Right-click **SPI/BPI** and select **Add SPI/BPI flash** (Figure 18). This step brings up an Explorer window. Navigate to the MCS file and select it.
5. iMPACT then asks which type of flash device is the target. Select the appropriate device in the pull-down menus.

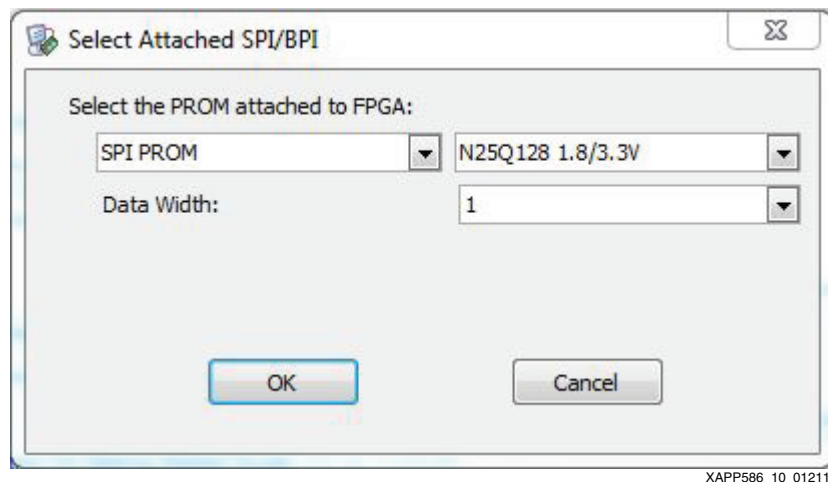
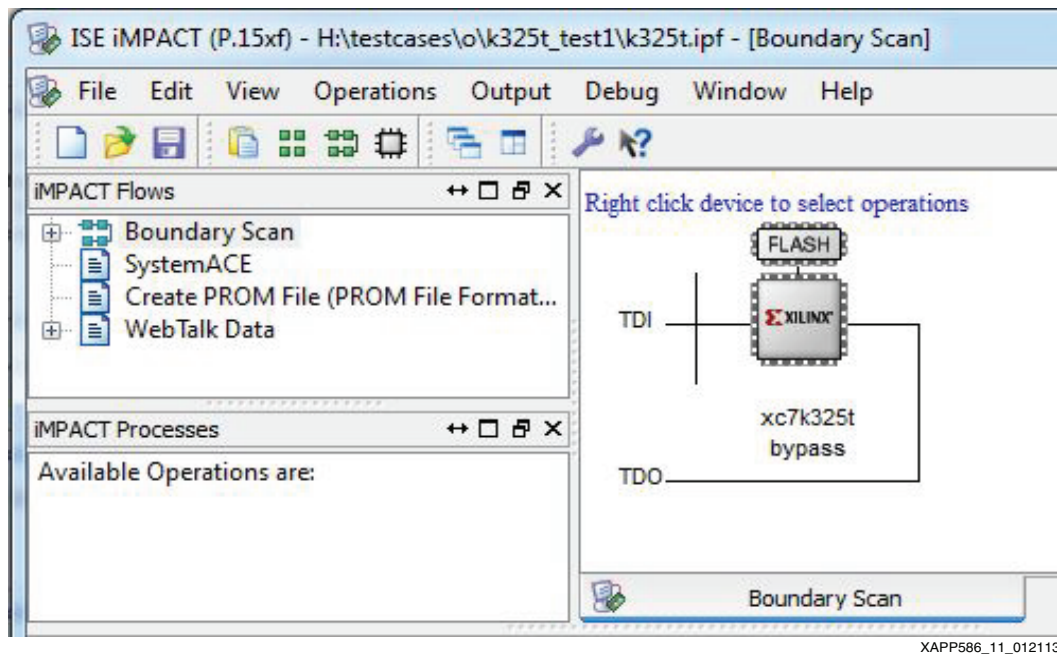


Figure 18: iMPACT SPI Flash Selection

The image on the right side of the window now changes from a dotted box labeled SPI/BPI? to a flash device image, indicating that a SPI flash device has been attached (Figure 19).

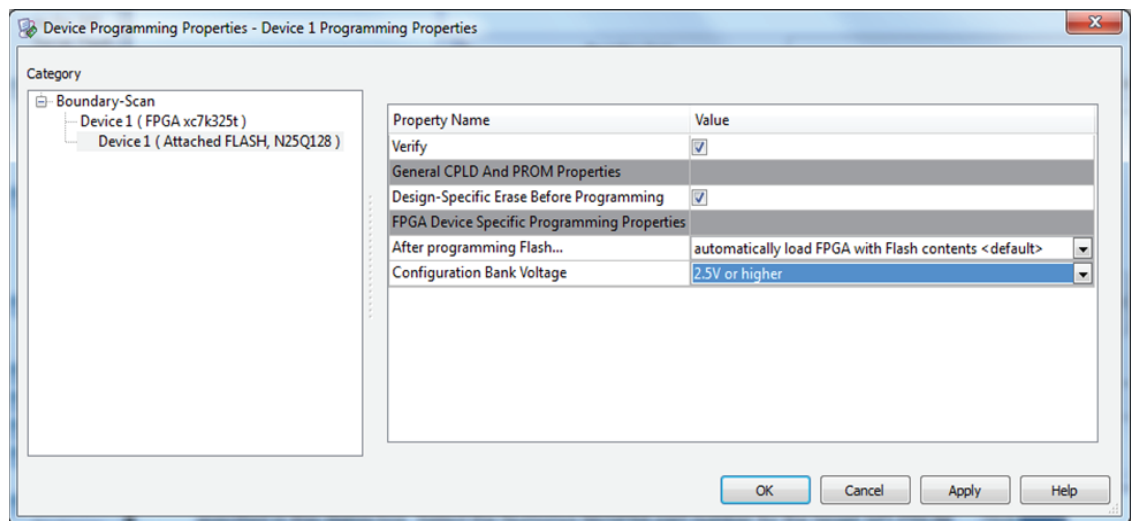


XAPP586_11_012113

Figure 19: iMPACT JTAG Chain with the SPI Flash Assigned

6. Right-click **Flash** and select **Set Programming Properties** (Figure 20).

From here, operations such as Erase and Verify are enabled by default. This step executes all operations **Erase > Program > Verify** when the designer programs the SPI flash. These instructions can be done individually, if desired.



XAPP586_12_042912

Figure 19: iMPACT SPI Flash Programming Properties

Conclusion

The SPI flash is a low-pin count and simple solution for configuring 7 series FPGAs. Support of indirect programming enhances ease of use by allowing in-system programming updates of the SPI flash by reusing connections already required for the configuration solution. Although some other configuration options permit faster configuration times or higher density, the SPI flash solution offers a good balance of speed and simplicity.

Appendix

Programming Time

Using iMPACT and Vivado Design Suite with the KC705 evaluation board, the operation times are provided. These times are not guaranteed values and are for reference only.

Kintex-7 XC7K325T FPGA's single bitstream (~91 Mb):

ISE iMPACT Design Tool:

- Design-specific erase time: 190 seconds
- Program time: 440 seconds
- Verify time: 220 seconds

Vivado Design Suite Tool:

- Design Specific Erase Time: 120.59 seconds
- Program Time: 214 seconds
- Verify: 34.14 seconds

Debugging and Troubleshooting Guidelines

Configuration issues can have many different causes that are not readily differentiated at the outset. For troubleshooting master SPI configuration failures, the following sections summarize some of the common debugging steps that can help narrow in on the root cause and possible solutions.

Verify Layout

- It is very important that the FPGA signal integrity of the JTAG TCK and FPGA CCLK signals are maintained. Avoid using lengthy connections where possible. Using long connections can result in unwanted noise or voltage waveform reflections that degrade the integrity of FPGA signals.
- Review the master SPI x4 schematic for the correct connectivity.

- Scope the CCLK, FCS_B, and D00_MOSI lines. After power-up or following a PROGRAM_B pulse, trigger on the falling edge of FCS_B and verify that the waveform matches the master SPI x1 or x4 read commands. Refer to the 7 Series FPGAs Configuration (UG470).
- For faster configuration and programming times, use the bitstream compression setting. For troubleshooting purposes, a simple bitstream can be used (that simply toggles a single LED or example) to get higher compression ratios and shorter programming times.
- Ensure the ConfigRate option does not exceed the maximum frequency supported by the target flash and FPGA. Refer to the FMCKTOL timing parameter in the appropriate 7 Series device data sheet for the FPGA CCLK tolerance. For debugging, consider dropping the configuration rate lower than the maximum calculated frequency or even leaving it at the default. See SPI Flash Configuration Time for instructions on calculating the maximum configuration rate.
- Ensure the JTAG Integrity is good:
 - Perform a basic FPGA IDCODE operation to verify the connections
 - Program a simple bitstream into the FPGA
- Review the FPGA status register for additional insight. Figure 9. Shows the CONFIG STATUS register after a successful configuration.

Register Name	Value
BOOT_STATUS	00000000000000000000000000000001
CONFIG_STATUS	01000000000010000010110011111100
BIT00_CRC_ERROR	0
BIT01_DECRYPTOR_ENABLE	0
BIT02_PIL_LOCK_STATUS	1
BIT03_DCL_MATCH_STATUS	1
BIT04_END_OF_STARTUP_EOS_STATUS	1
BIT05_GTS_CFG_B_STATUS	1
BIT06_GWE_STATUS	1
BIT07_GHIGH_STATUS	1
BIT08_MODE_PIN_M[0]	1
BIT09_MODE_PIN_M[1]	0
BIT10_MODE_PIN_M[2]	0
BIT11_INIT_B_INTERNAL_SIGNAL_STATUS	1
BIT12_INIT_B_PIN	1
BIT13_DONE_INTERNAL_SIGNAL_STATUS	0
BIT14_DONE_PIN	1
BIT15_IDCODE_ERROR	0
BIT16_SECURITY_ERROR	0
BIT17_SYSTEM_MONITOR_OVER_TEMP_ALARM_STATUS	0
BIT18_CFG_STARTUP_STATE_MACHINE_PHASE	100
BIT21_RESERVED	0000
BIT25_CFG_BUS_WIDTH_DETECTION	00

Figure 20: CONFIG STATUS Register

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the [Design Hubs](#) page.

Note: For more information on Documentation Navigator, see the [Documentation Navigator](#) page on the Xilinx website.

References

These documents provide supplemental material useful with this design:

1. [UG470](#), *7 Series FPGAs Configuration User Guide*
2. [DS180](#), *7 Series FPGAs Overview*
3. [DS181](#), *Artix-7 FPGAs Data Sheet*
4. [DS182](#), *Kintex-7 FPGAs Data Sheet*
5. [DS183](#), *Virtex-7 FPGAs Data Sheet*
6. [UG883](#), *Kintex-7 FPGA KC705 Evaluation Kit Getting Started Guide*
7. [XAPP1328](#), *Configuration or Boot with NOR Flash for Sourcing Flexibility and Cost Reduction Application Note*
8. [XAPP1233](#), *SPI Configuration and Flash Programming in UltraScale FPGAs Application Note*
9. Xilinx ISE Design Suite Documentation:
www.xilinx.com/support/documentation-navigation/development-tools/hardware-development/ISE-design-suite.html
10. [UG628](#), *Command Line Tools User Guide*
11. [UG908](#), *Vivado Design Suite User Guide: Programming and Debugging*
12. Micron Technology, Inc. for SPI flash data sheets: www.micron.com

13. Infineon Technologies AG. for SPI flash data sheets: www.infineon.com
14. Winbond Electronic Corp. site for SPI flash data sheets: www.winbond.com

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
05/30/2012	1.0	Initial Xilinx release.
02/01/2013	1.1	Updated pin functions in Table 1 . Updated SPI flash and 7 series FPGAs blocks in Figure 5 . Updated GUI steps to generate flash file in Preparing the SPI Flash Programming File . Updated Figure 17 , Figure 18 , and Figure 19 . Updated Programming the SPI Flash In-System .
01/31/2014	1.2	Added Vivado Design Suite support to second paragraph of Summary . Updated Kintex-7 FPGA bitstream number and calculations in Configuration Clock Calculation Example . Updated description of M[2:0] in Table 2 . In iMPACT Indirect Flash Operation Time Estimates , updated iMPACT software version and operation times. Updated URLs for ISE Design Suite documentation and <i>Command Line Tools User Guide</i> in References .
10/28/2016	1.3	Added paragraph about Artix-7 and Spartan-7 devices to Selecting an SPI Flash . Added trace propagation delay to Equation 1 . In Configuration Clock Calculation Example , updated SPI flash clock to out values, added paragraph about trace propagation delays, and updated clock frequency from 93.9 MHz to 83.3 MHz. In References , added UG908 and Micron Serial NOR Flash Memory N25Q128A13, and updated URLs to ISE Design Suite documentation and Cypress Semiconductor.
08/20/2020	1.4	In Figure 6 , updated OP CODE to 0BH. Removed obsolete reference to Micron Serial NOR Flash Memory N25Q128A13 data sheet from References .
12/09/2022	1.5	Updated Summary and Figure 1 to replace reference to ISE® Design Suite to XILINX tools. Updated Introduction to include a reference to the Vivado Design Suite and added additional section references. Updated SPI Flash Basics reference to Xilinx Design Suite tools from ISE Design Suite and added reference to XAPP1328. Added 7-Series to Figure 3 title and Table 2 . Added voltage to SPI Flash Configuration Interface . Updated the configuration option name in Configuring the FPGA from SPI Flash . Updated configuration option title in SPI Flash Configuration Time . Added SPI Flash Using Xilinx Vivado Design Suite and Appendix .

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which

can be viewed at <https://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

NOTICE: This pre-release document contains confidential and proprietary information of Xilinx, Inc. and is being disclosed to you as a participant in an early access program, under obligation of confidentiality. You may print one (1) copy of this document for evaluation purposes. You may not modify, distribute, or disclose this material to anyone, including your employees, coworkers, or contractors (these individuals must apply for separate access to the program). This document contains preliminary information and is subject to change without notice. Information provided herein relates to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of the products and/or services referred to herein.

This document contains preliminary information and is subject to change without notice. Information provided herein relates to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of the products and/or services referred to herein.

© Copyright 2012-2022 Advanced Micro Devices, Inc. Xilinx, the Xilinx logo, Alveo, Artix, ISE, Kintex, Kria, Spartan, Versal, Virtex, Vitis, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.