Initializing the UniPHY Nios II Sequencer in HardCopy Devices using FPP Configuration Scheme

AN-650-1.0 Application Note

This application note describes the reuse of the fast passive parallel (FPP) configuration interface, which is more commonly used in the Altera® FPGA configuration, to initialize the UniPHY Nios® II sequencer for HardCopy® device migration.

There are different methods to initialize the Nios II sequencer in HardCopy devices. Altera recommends re-using the existing FPGA configuration interface to initialize the Nios II sequencer to save resources and avoid a board re-spin.

For more information about the Nios II sequencer initialization method, refer to the *HardCopy Migration* chapter in volume 3 of the *External Memory Interface Handbook*.

For more information about FPP configuration scheme for each FPGA family, refer to the configuration chapter of the respective device handbook.

This application note contains the following sections:

- "Introduction to the Nios II Sequencer" on page 2
- "Functional Description" on page 2
  - "MAX II Controller" on page 3
  - "Sequencer ROM Controller" on page 5
- "Design Verification in FPGAs" on page 7
  - "Software and Hardware Requirements" on page 8
  - "Converting Nios II Sequencer Instruction Code" on page 9
  - "Programming Flash Memory with PFL Megafunction" on page 9
  - "Verifying the MAX II Controller and Sequencer ROM Controller" on page 10

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Introduction to the Nios II Sequencer

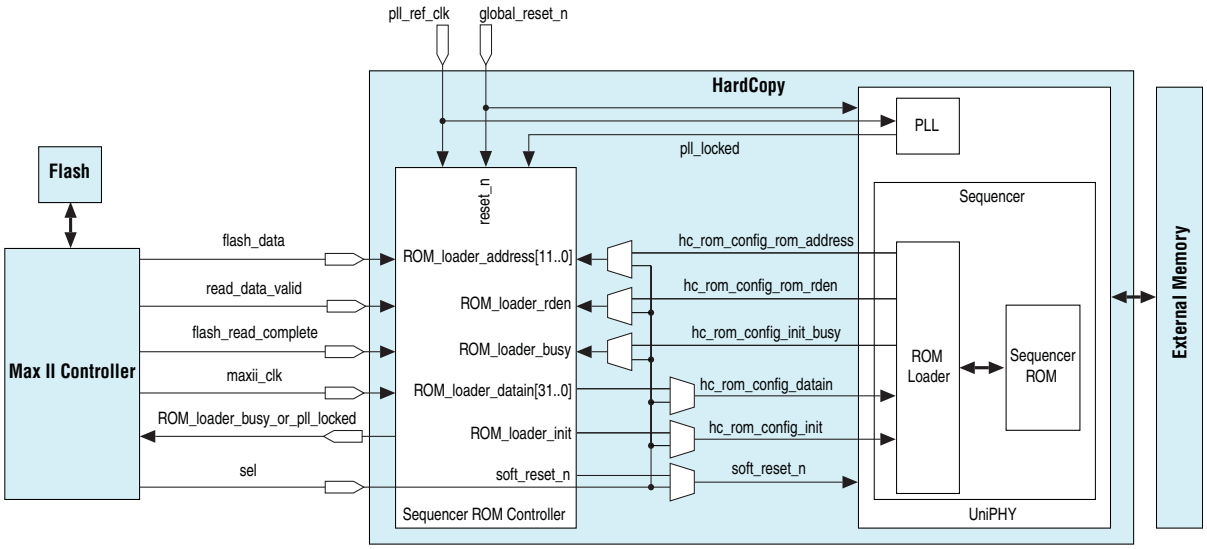
The Nios II sequencer is integrated into the UniPHY intellectual property (IP) to replace the register transfer level (RTL) sequencer to improve the performance of the UniPHY IP and to make the calibration process more robust. The Nios II system boot-up is required after the integration of the Nios II sequencer into the UniPHY IP. The Nios II instruction code must be loaded into the Nios II system during the system boot-up to start the calibration process. The instruction code is stored in the FPGA internal memory and is initialized during FPGA power up. However, the RAM cannot be initialized during power up in HardCopy devices, thus the Nios II instruction code cannot be stored in the RAM of HardCopy devices. You may store the instruction code in the ROM of HardCopy device, but you cannot make any changes after that. To provide the flexibility to change the instruction code in future, you need to store the Nios II instruction code in an external non-volatile memory and initialize the Nios II sequencer through the ROM loader in the UniPHY IP.

Functional Description

The two main blocks in the reference design are the MAX II controller and the sequencer ROM controller.

Figure 1 shows the connection of the MAX II controller and the sequencer ROM controller in the reference design.

Figure 1. MAX II Controller and Sequencer ROM Controller Connections



MAX II Controller

In the FPGA FPP configuration scheme, the MAX II controller reads the programming file in the flash device and configures the FPGA through the configuration interface pins.

In this reference design, the MAX II controller is in user mode to read the Nios II instruction code from the flash device. It then sends the code read to the sequencer ROM controller to initialize the Nios II sequencer. The MAX II controller reference design is only applicable with the 512-Mb Numonyx P30 Flash, which has a 16-bit data bus and 20-bit address bus. The controller runs at 50 MHz.

Figure 2 shows the top level of the MAX II controller.

Figure 2. MAX II Controller Top Level Block Diagram

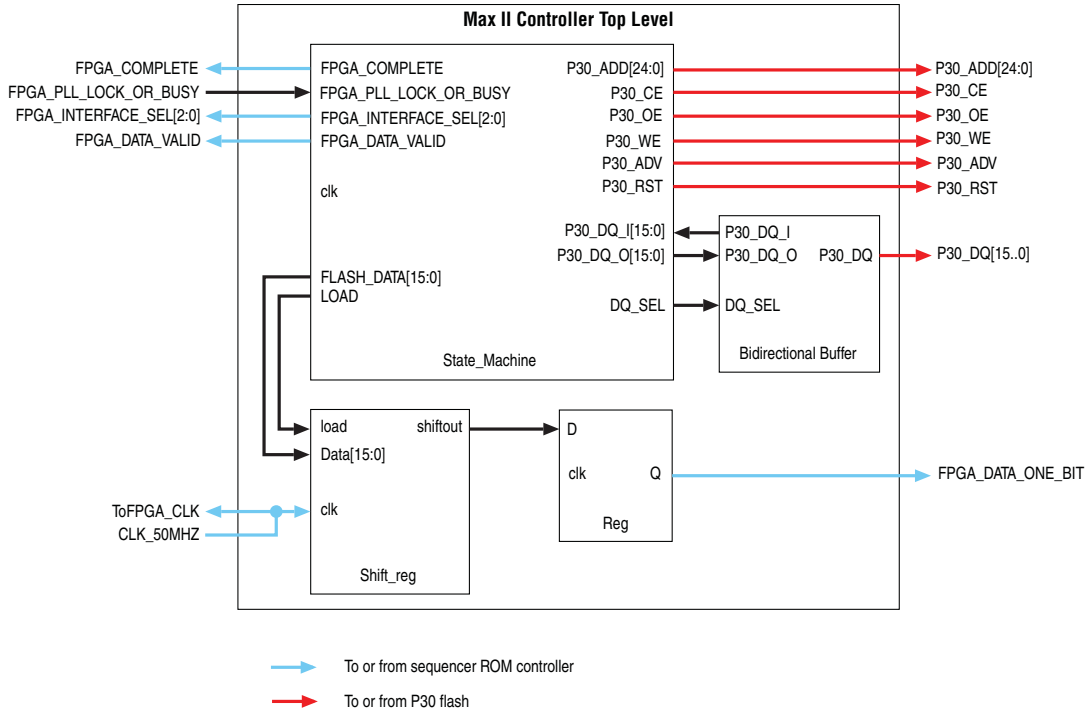


Table 1 describes the functions of the MAX II controller input and output ports.

Table 1. MAX II Controller Input and Output Ports Descriptions

MAX II Controller Input and Output Ports	Description
CLK_50MHZ	To source the clock from the external oscillator.
FPGA_PLL_LOCK_OR_BUSY	This port has two functions: ■ Initial stage ■ **1**—State machine of the MAX II controller starts to run. ■ **0**—Idle. ■ During state machine progression ■ **1**—Sequencer ROM controller is busy. Remains idle after shifting out the 16-bit data. ■ **0**—Sequencer ROM controller is idle. Shifts the next 16-bit data.
FPGA_COMPLETE	To inform the sequencer ROM controller that all 4096 x 32 bits of data is completely shifted.
FPGA_DATA_ONE_BIT	One bit flash data bus.
FPGA_DATA_VALID	To inform the sequencer ROM controller that the 16-bit data has started to shift. ■ Rising edge—first bit of the 16-bit data has started to send. ■ Falling edge—last bit of the 16-bit data is completely sent.
FPGA_INTERACE_SEL	To select an interface from multiple interfaces for loading the Nios II instruction code.
ToFPGA_CLK	Clock to the FPGA controller for data synchronization.
P30*	Ports which interface with the P30 flash.

The MAX II controller starts to read the flash data when the HardCopy PLL is locked and the FPGA_PLL_LOCK_OR_BUSY signal is high. In the multiple interfaces design, the controller continues to read the subsequent pages of the flash data when the FPGA_PLL_LOCK_OR_BUSY signal is low in the following cycles. The controller issues the FPGA_INTERAFACE_SEL signal to select the interface of the Nios II sequencer to be initialized.

Because of I/O pin limitation, the controller serializes the 16-bit flash data before shifting the data to the sequencer ROM controller in the HardCopy device. The FPGA_DATA_VALID signal remains high for 16 clock cycles during the shift of data to the sequencer ROM controller for every single read of flash data to show that the data read is valid. After reading the whole page of flash data (4096 x 32 bits), the FPGA_COMPLETE signal goes high to indicate to the sequencer ROM controller that the MAX II controller has completed reading a single page of data.

Sequencer ROM Controller

The sequencer ROM controller is a block used to interface with the MAX II controller. It deserializes the Nios II instruction code from the MAX II controller before the ROM loader starts loading the code to the UniPHY sequencer ROM. The sequencer ROM controller consists of four main blocks:

- DeserializeRAM

  As a result to I/O count limitation, the Nios II instruction code read from the flash device is serialized by the MAX II controller before sending it to the HardCopy device. The sequencer ROM controller stores the single bit data in the DeserializeRAM block. When all the single bit data is stored (a total of 4096 x 32 bits), the ROM loader loads the data in 32-bit format from the DeserializeRAM block into the UniPHY sequencer ROM.

- `ROM_loader` init generator

  The init generator requests the ROM loader to load the instruction code from the DeserializeRAM block when the MAX II controller has completed reading the code.

- `soft_reset` control block

  This control block generates the `soft_reset_n` signal to reset the UniPHY sequencer. It ensures the `soft_reset_n` signal is low during data transaction from the MAX II controller to the sequencer ROM, so that the PHY is in reset state. Upon completion of the data transaction from the MAX II controller to the sequencer ROM, the `soft_reset_n` signal is pulled to high. The Nios II sequencer starts to retrieve data from the sequencer ROM when all the data is ready in the sequencer ROM.

- `ROM_loader_busy_or_pll_locked` control block

  This block generates the `ROM_loader_busy_or_pll_locked` signal to the MAX II controller. The MAX II controller starts to read the flash data when PLL is locked. It requests the MAX II controller to start reading the data from the subsequent page when the ROM loader has completed loading the data from one page to the sequencer ROM.

Figure 3 illustrates the interconnections of the sequencer ROM controller.

Figure 3. Sequencer ROM Controller Block Diagram

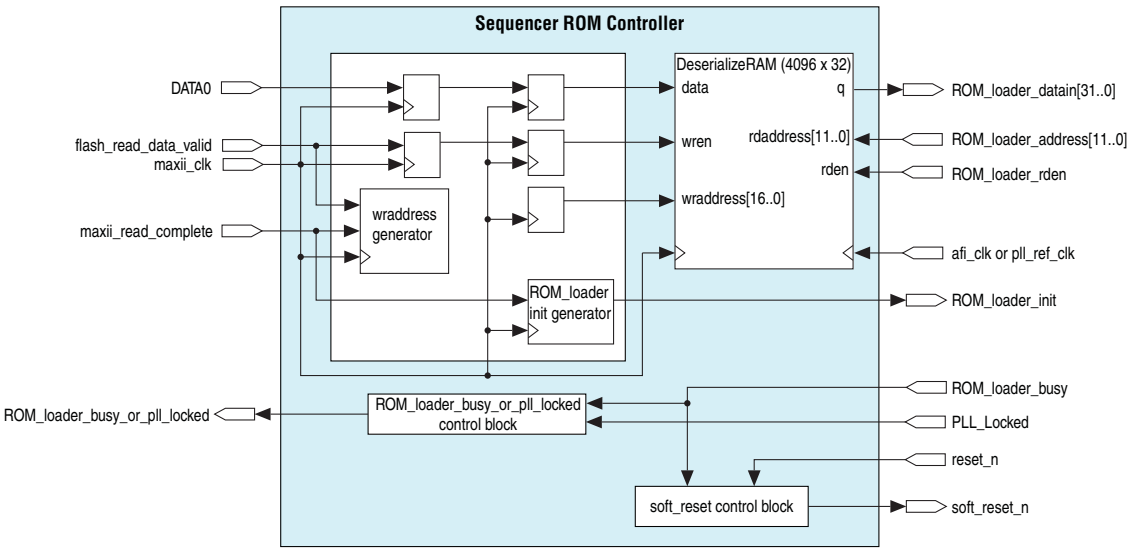


Table 2 shows the signals descriptions for the sequencer ROM controller.

Table 2. Sequencer ROM Controller Signals Descriptions

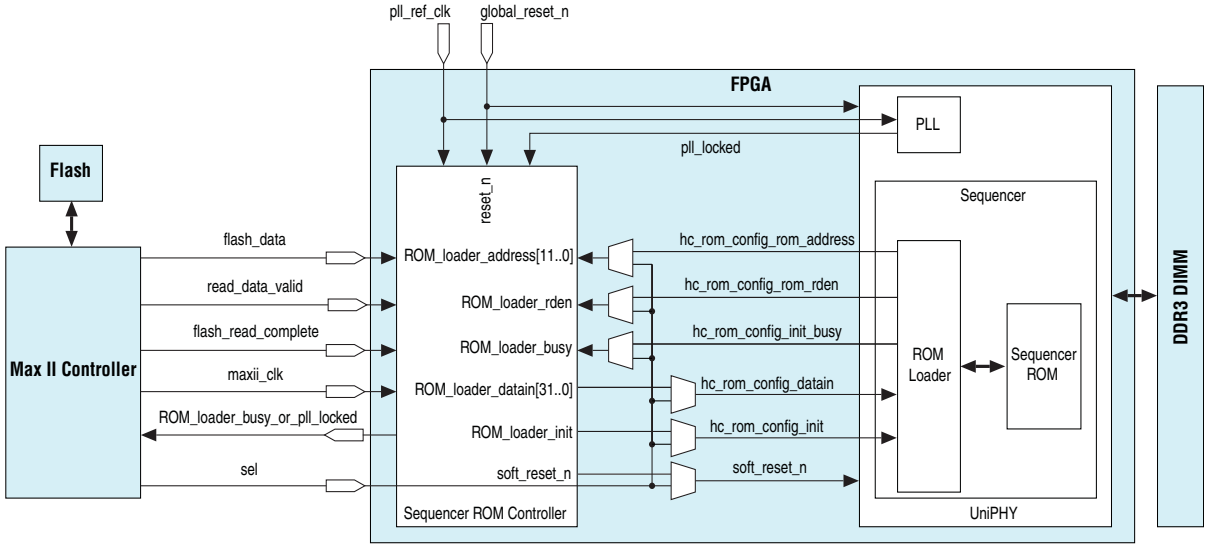
Signal	Description
DATA0	Serial flash data from the MAX II controller.
flash_read_data_valid	The signal indicates the validity of the read data. The signal sets to high, which indicates that the read data is valid.
maxii_clk	Read clock from the MAX II controller.
maxii_read_complete	Signal from the MAX II controller indicates the completion of reading one page of 4096 x 32-bit flash data.
ROM_loader_busy_or_pll_locked	The signal is sent to the MAX II controller to indicate the PLL is locked or the ROM loader is busy.
ROM_loader_datain[31..0]	Parallel data to the ROM loader.
ROM_loader_address[31..0]	Read address from the ROM loader.
ROM_loader_rden	Read enable signal from the ROM loader.
ROM_loader_init	The signal to the ROM loader to request the ROM loader to start reading data from the sequencer ROM controller.
ROM_loader_busy	The signal shows the status of the ROM loader. The signal sets to high, which indicates that the ROM loader is reading the data.
PLL_locked	PLL locked signal from the PLL.
reset_n	Global reset signal.
soft_reset_n	Reset signal that resets the UniPHY sequencer without resetting the PLL.
afi_clk or pll_ref_clk	Options to use the signal from the UniPHY or PLL reference clock.

Design Verification in FPGAs

The MAX II controller and the sequencer ROM controller are verified with a DDR3 SDRAM full system on the Stratix IV GX checkout board. The sequencer ROM controller is integrated into the reference design created in the DDR3 SDRAM UniPHY IP. The DDR3 SDRAM reference design runs at 400 MHz with a half-rate controller and a 72-bit interface to the FPGA.

Figure 4 shows the DDR3 SDRAM UniPHY full system connection together with the MAX II controller and the sequencer ROM controller.

Figure 4. MAX II Controller and Sequencer ROM Controller Connections with UniPHY IP on a FPGA



The MAX II controller starts to load the Nios II instruction code from the flash device to the UniPHY sequencer ROM when the FPGA PLL is locked. When the Nios II instruction code is successfully loaded from the external flash into the UniPHY sequencer ROM, the `soft_reset_n` signal is deasserted to allow the Nios II sequencer to read the instruction code from the sequencer ROM. The DDR3 SDRAM UniPHY reference design starts the calibration when the Nios II sequencer is booting up.

No extra connection is required between the MAX II device and the FPGA. The MAX II controller and the sequencer ROM controller interface signals are connected through the eight FPGA configuration data pins (`DATA[0..7]`).

Table 3 shows the connection between the FPGA and the MAX II device through the configuration data pins.

Table 3. FPGA Configuration Data Pins and Interface Signals for FPGA and MAX II Device Connection (Part 1 of 2)

FPGA Configuration Data Pin	Interface Signal	Description
DATA0	flash_read_complete	The signal from the MAX II controller indicates the completion of reading one page of 4096 x 32-bit flash data.
DATA1	flash_data	Serial flash data from the MAX II controller.

Table 3. FPGA Configuration Data Pins and Interface Signals for FPGA and MAX II Device Connection (Part 2 of 2)

FPGA Configuration Data Pin	Interface Signal	Description
DATA2	flash_data_valid	The signal indicates the validity of the data read. The signal sets to high, which indicates that the data read is valid.
DATA[3..5]	sel[0..2]	Interface selection signal from the MAX II controller for multiple interfaces. Select the Nios II sequencer interface to boot up.
DATA6	ROM_loader_busy_or_pll_locked	The signal issued by the sequencer ROM controller to the MAX II controller to indicate that the PLL is locked or the ROM loader is busy.
DATA7	maxii_clk	The signal that reads the clock from the MAX II controller.

Ensure that the dual purpose DATA[0] pin is set to use as **Regular I/O** in the device setting before compilation.

Software and Hardware Requirements

The reference design folder contains the following reference designs :

- **UniPHY_HCX_Migration_DDR3_Example_Design.zip**—The DDR3 SDRAM UniPHY reference design where the sequencer ROM is integrated.
- **MaxII_Controller.qar**—The MAX II controller which is used to convey the Nios II instruction code to the Nios II sequencer.
- **MaxIIPFL.qar**—The parallel flash loader (PFL) megafunction which is used to program the Nios II instruction code into the flash.

The reference designs require the following software and hardware:

- Quartus® II software version 11.0
- Stratix IV FPGA checkout board (F1517) with FPP configuration interface
- Stratix IV device: EP4SGX230KF40C3
- MAX II device: EPM2210F324C3 (with JTAG port connection)
- Numonyx P30 flash device
- DDR3 SDRAM DIMM (MT9JSF12872AY-1G1BZES)

Converting Nios II Sequencer Instruction Code

To convert the UniPHY Nios II sequencer instruction code, follow these steps:

1. From the reference design folder, open the **IntelFormatConversionScript.tcl** script. In the **.tcl** script, change the variables in the following commands according to your design.
   - `set fileid [open "<project directory>/<variable_name>_sequencer_rom.hex" r+]`
   - `set new_fileid [open "<project directory>/Converted_<variable_name>_sequencer_rom.hex" w+]`
2. The UniPHY Nios II sequencer instruction code is stored as **<*variable_name*>_sequencer_rom.hex** file. Run the **.tcl** script to convert the **<*variable_name*>_sequencer_rom.hex** file to the **.hex** file in Intel format.

   The conversion enables the change of the addressing method from one address increment to four address increments because 4 bytes of instruction codes are being written to the flash in each row of the **.hex** file.
3. Convert the **Converted_<*variable_name*>_sequencer_rom.hex** file in Intel format to the **.pof** file format with the Quartus II software.
   a. On the File menu, select **Convert Programming Files…**.
   b. In **Configuration device**, select the option **CFI_512Mb**.
   c. For **Mode**, select **Fast Passive Parallel x8.**
   d. Rename the output **.pof** file.
   e. Remove **SOF Data**. Select **Add Hex Data** to add the **Converted_<*variable_name*>_sequencer_rom.hex** file.
   f. Click **Generate**.

   The conversion is required to program the flash device.

Programming Flash Memory with PFL Megafunction

The PFL megafunction is used to program the flash memory devices through the MAX II devices using the JTAG interface. A reference design is provided for the flash memory programming mode with the PFL megafunction. You can find the reference design in the **MaxIIPFL.qar** file. The reference design is applicable to the 512-Mbit common flash interface (CFI) parallel flash, which is used in the Stratix IV checkout board.

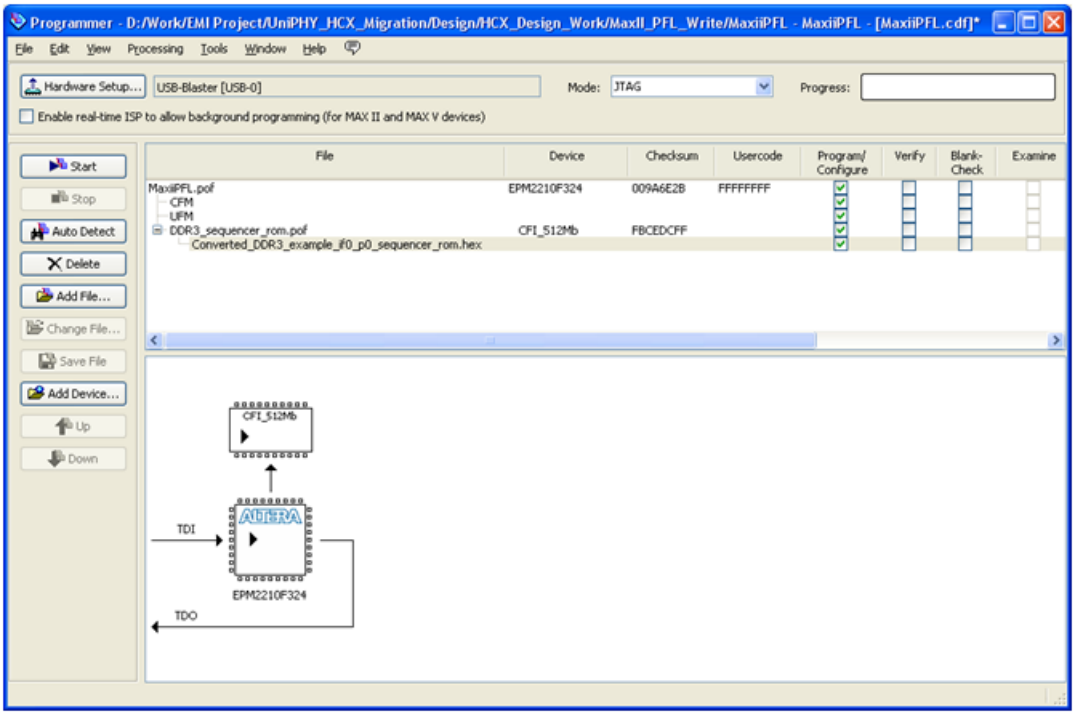
For more information about the PFL megafunction, refer to the *Parallel Flash Loader Megafunction User Guide*.

To program the flash memory device, follow these steps:

1. Un-archive the **MaxIIPFL.qar** file and recompile the design.
2. Open the Quartus II Programmer, add the **MaxIIPFL.pof** file, and attach the sequencer ROM **.pof** file to the **MaxIIPFL.pof** file.

   Figure 5 shows the Quartus II Programmer interface for the design setup.

Figure 5. Design Setup for Flash Memory Programming



3. Program the converted sequencer ROM **.pof** file into the flash device as provided in the **MaxIIPFL.qar** reference design with the MAX II device through the JTAG interface.

Verifying the MAX II Controller and Sequencer ROM Controller

To verify the MAX II controller and the sequencer ROM controller, follow these steps:

1. Un-archive the **MaxII_Controller.qar** file and re-compile the design.
2. After compilation is completed, program the MAX II device with the Quartus II Programmer.
3. Extract the **UniPHY_HCX_Migration_DDR3_Example_Design.zip** file and re-compile the design.
4. Configure the FPGA.
5. On the Tools menu, select **SignalTap II Logic Analyzer** to verify the design.
6. On the Tools menu, select **In-System Sources and Probes Editor** to control the `global_reset_n` signal. Choose **global reset** to start the design.

Document Revision History

Table 4 shows the revision history for this document.

Table 4. Document Revision History

Date	Version	Changes
November 2011	1.0	Initial release.