AN 548: Nios II Compact Configuration System for Cyclone III

© November 2008 AN-548-1.0

Introduction

This application note discusses how you can use a Nios® II processor based configuration system in your design to actively manage one or more configurations from commodity flash. You can reduce the cost and simplify your FPGA based embedded system design by eliminating the need for an external configuration controller. You can use various configuration schemes with Altera Cyclone® III FPGAs. These schemes include active serial (AS), active parallel (AP), passive serial (PS), fast passive parallel (FPP), and Joint Test Action Group (JTAG). You can configure the Cyclone III devices using one of these schemes, depending on the device densities and package options.

Active configuration allows an intelligent controller within the FPGA (such as the Nios II processor) to perform configuration while passive configuration relies upon an external controller (a microprocessor, Complex Programmable Logic Device (CPLD) or EPCS configuration device) to configure the FPGA.

The example design demonstrates the use of the AP configuration scheme and Cyclone III remote system upgrade feature for reconfiguration purposes. To enable the design to fit easily into your existing system, it is optimized to achieve the smallest possible utilization of FPGA resources and software code footprint. You can easily adapt this design to any existing design to add reconfiguration capability to the application.

For more information on other configuration schemes, refer to the *Configuring Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

Active Parallel (AP) Configuration Scheme

In the AP configuration scheme, Cyclone III devices read configuration data via the parallel interface, which configures their SRAM cells. This scheme is referred to as active because the FPGA device controls the configuration interface. Depending on the available package options, some of the smaller Cyclone III devices do not support the AP configuration scheme. Table 1 shows which combinations of Cyclone III family devices and package options support the AP configuration scheme.

Table 1. Active Parallel (AP) Configuration Scheme Supported by Cyclone III Devices (Part 1 of 2)

Device	Package Options *(Note 1)*								
	E144	M164	Q240	F256	F324	F484	F780	U256	U484
EP3C5	NO	NO	N/A	NO	N/A	N/A	N/A	NO	N/A
EP3C10	NO	NO	N/A	NO	N/A	N/A	N/A	NO	N/A
EP3C16	NO	NO	NO	NO	N/A	YES	N/A	NO	YES
EP3C25	NO	N/A	NO	NO	YES	N/A	N/A	NO	N/A
EP3C40	N/A	N/A	NO	N/A	YES	YES	YES	N/A	YES
EP3C55	N/A	N/A	N/A	N/A	N/A	YES	YES	N/A	YES

Table 1. Active Parallel (AP) Configuration Scheme Supported by Cyclone III Devices (Part 2 of 2)

Device	Package Options *(Note 1)*								
	E144	M164	Q240	F256	F324	F484	F780	U256	U484
EP3C80	N/A	N/A	N/A	N/A	N/A	YES	YES	N/A	YES
EP3C120	N/A	N/A	N/A	N/A	N/A	YES	YES	N/A	N/A

Note to Table 1:

(1) N/A indicates that the device is not available with the specific package option.

The AP configuration scheme uses a parallel flash memory for storing configuration data. The flash memory provides a fast interface to access configuration data, mainly due to the parallel data bus. Since a flash memory has an inexpensive memory per cost ratio, it is feasible to store multiple configuration images in a single flash. This is useful when the application requires switching between a default factory image and application images.

To enable the AP configuration scheme for your Quartus® II design, perform the following steps in the Quartus II software:

1. On the Assignments menu, click **Device**.
2. On the **Device** page, click **Device and Pin Options**.
3. Click the **Configuration** tab and select **Active Parallel** under the **Configuration scheme** list.
4. Click **OK**.

For more information on Cyclone III AP configuration scheme, refer to the *Configuring Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

Cyclone III Remote System Upgrade Feature

Cyclone III devices that support active configuration schemes have dedicated built-in remote system upgrade circuitry. This circuitry allows you to upgrade the system from remote locations through a communication interface. You can add a factory image and multiple application images to your product. The factory image is a user-defined fall-back, or safe configuration that administers remote updates in conjunction with the dedicated circuitry. The application images implement user-defined functionality in the target Cyclone III device. You can include the default application image functionality in the factory image.

For example, by using remote system upgrade feature, you can update one of the application images stored in the flash memory of the embedded product through an Ethernet connection. After the update is complete, you can initiate a reconfiguration cycle to use the newly updated configuration image. The dedicated remote system upgrade circuitry detects any errors that occur during or after this cycle. If an error occurs, the remote system upgrade circuitry causes the device to automatically revert to the factory image. The factory image then performs error processing and recovery. While error processing functionality is limited to the factory configuration, both factory and application configurations can download and store remote updates and initiate system reconfiguration.

When the AP configuration scheme is powered up with remote system upgrade feature, the Cyclone III device loads the configuration image stored in the flash memory at offset of `0x010000` (for 16-bit word addressing) or `0x020000` (for 8-bit byte addressing). A default factory image is stored at this address so that the device is configured with this design once the configuration is powered up.

To use the remote system upgrade feature for your design, perform the following steps in the Quartus II software:

1. On the Assignments menu, click **Device**.
2. On the **Device** page, click **Device and Pin Options**.
3. Click the **Configuration** tab and select **Remote** under the **Configuration mode** list.
4. Click **OK**.

For more information on Cyclone III remote system upgrade feature, refer to the *Remote System Upgrade With Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

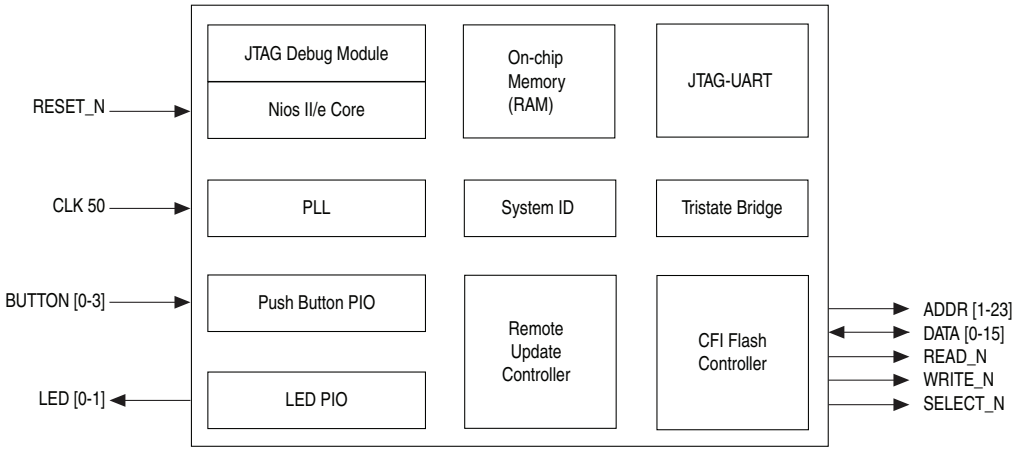
Example Design

This section demonstrates a compact example design that uses remote system upgrade feature with the AP configuration scheme for reconfiguration. To maintain its compactness, the design contains minimal number of SOPC Builder components. This design is targeted to the Cyclone III FPGA Starter Kit and the Nios II Embedded Evaluation Kit (NEEK), which uses the EP3C25 device. However, the design can be fitted into the smallest Cyclone III device that supports AP configuration scheme, which is the EP3C16 device (refer to Table 1).

Example Design Hardware

Figure 1 shows the block diagram of this example design.

Figure 1. Block Diagram of Nios II Compact Configuration Example Design



This design uses Nios II/e "economy" processor. The Nios II/e processor is chosen over other implementations of the Nios II processor because it requires the least amount of resources. The processor boots from an on-chip memory which holds the software program. A Common Flash Interface (CFI) flash controller is used for interfacing between the system and the flash memory. The remote update controller interfaces with the Cyclone III dedicated remote system upgrade circuitry for reconfiguration purpose.

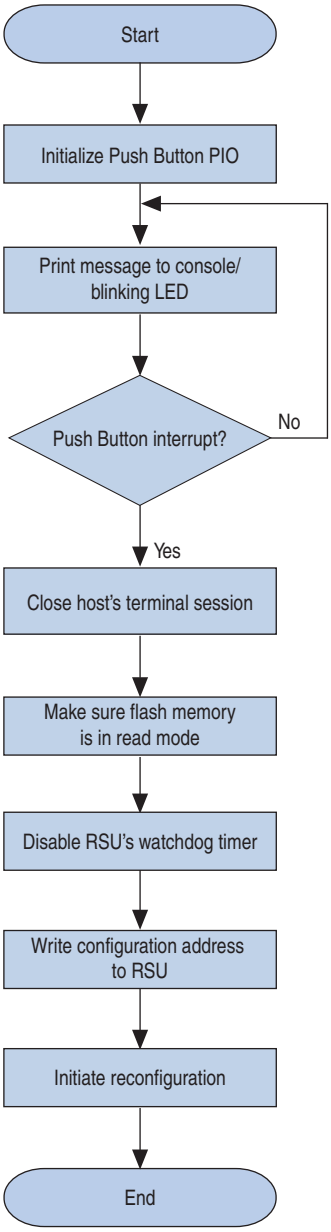
This system has a 50-MHz input clock fed to a Phase-Locked Loop (PLL). The PLL generates two output clocks—60 MHz and 40 MHz. The 60 MHz clock is used by all the components in the system, except the remote update controller that requires the 40 MHz clock as its supported frequency.

There are three user interactive peripherals in the system: the LED parallel input/output (PIO), the push button PIO, and the JTAG UART. The LEDs indicate which configuration image is in use. The push buttons allow you to choose a specific image for reconfiguration. You can display messages via the JTAG UART. You can add or remove these peripherals from the system according to your application needs. The peripherals are included in this example design for demonstration purpose.

Example Design Software

Figure 2 on page 5 shows the flowchart of the software program for this example design.

Figure 2. Software Program Flowchart for Nios II Compact Configuration Example Design



The push button PIO is initialized at the beginning of the software execution. The initialization process includes enabling the PIO interrupt, resetting the interrupt edge capture register, and registering the interrupt handler. The program prints messages to the host and blinks the LEDs, indicating which configuration image is in use.

For more information on the interrupt behavior of the PIO, refer to the *PIO Core* chapter in volume 5 of the *Quartus II Development Software Handbook*.

The LED keeps on blinking until a push button is pressed. Once a push button interrupt occurs, the reconfiguration process starts to take place in the following sequence:

1. The terminal on the host is closed.
2. The flash memory is set to Read mode.
3. The watchdog timer of the remote update controller is turned off.
4. The offset of the configuration image in flash memory is written to the remote update controller.
5. The reconfiguration is triggered.

For more information on the software programming model of the remote update controller, refer to the *Cyclone III Remote Update Controller Core* chapter in volume 5 of the *Quartus II Development Software Handbook*.

Tutorial

This section describes the steps to run the example design on the Cyclone III FPGA Starter Kit or the Nios II Embedded Evaluation Kit. This tutorial requires the following items:

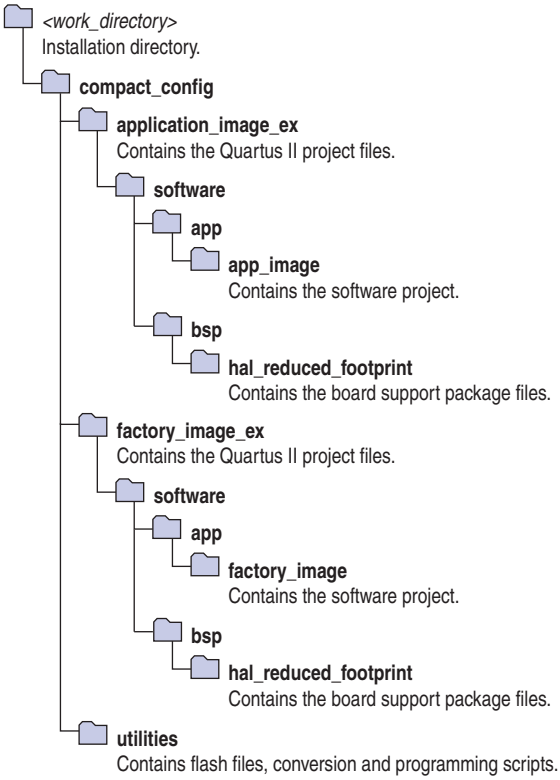
1. The Quartus II software version 8.1 or later.
2. The Nios II Embedded Design Suite (EDS) version 8.1 or later.
3. The Cyclone III FPGA Starter Kit or the Nios II Embedded Evaluation Kit.
4. The **compact_config.zip** file, which can be downloaded from www.altera.com/support/examples/download/compact_config.zip.

Getting Started

In order to install the example design, unzip the **compact_config.zip** file to a working directory in your hard disk. This location is referred to as <*work_directory*> in the rest of this document.

The directory structure appears as shown in Figure 3 on page 7.

Figure 3. Directory Structure of compact_config.zip

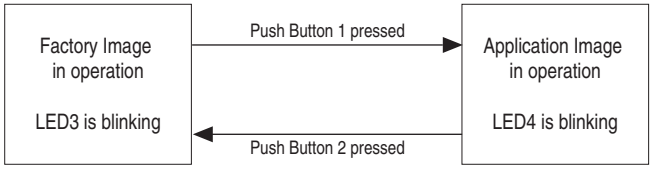


The Quartus II projects for both factory and application images are located in the **factory_image_ex** and **application_image_ex** directories respectively. In this example, the hardware designs for both images are the same—it is the software designs that differ. The factory image reconfigures the device with the application image and vice versa. You can implement other factory and application images if the need arises.

Running the Example Design

The operation flow of this example design is as shown in Figure 4.

Figure 4. Example Design Operation Flow



The following steps guide you through the process of configuring and running the example design:

1. Connect your board to your local PC using the provided USB cable and connect power to the board.

2. Open a Nios II command shell.
   a. On the Windows Start menu, point to **All Programs**, **Altera**, **Nios II EDS** <*version*>, and then click **Nios II** <*version*> **Command Shell**.
3. Set the directory to <*work_directory*>**/compact_config/utilities**.
4. To program the onboard flash memory with **output.flash** file, type the following in the command shell:

   `./program_flash.sh output.flash` <*programming hardware*>

   <*programming hardware*> is the name of the programming hardware that connects your board to the host. If you are not certain about the name, type `jtagconfig` in the command shell for a list of detected cables.

   Programming the flash memory on the Nios II Embedded Evaluation Kit erases the application selector program. Refer to "Restoring the Application Selector" on page 13 to restore the application selector on the Nios II Embedded Evaluation Kit.

5. Press the RECONFIGURE button on the board once the flash programming completes. Upon the completion of the configuration process, the factory image runs and LED3 on the board blinks.
6. Open a terminal session by typing `nios2-terminal` in the command shell. A message appears (Example 1) in the terminal.

Example 1. Message—Running Factory Image

```
Running Factory Image
Press Push Button 1 for App Image
```

7. Press BUTTON1 to reconfigure the device with application image. The following message (Example 2) appears before the terminal closes. The application image runs and LED4 on the board blinks.

Example 2. Message—Configuring to Application Image

```
Button1 is pressed, configuring to App Image
The nios2-terminal will now close. Please open a new one for App Image
```

8. Open a new terminal session by typing `nios2-terminal` in the command shell. The following message (Example 3) appears in the terminal.

Example 3. Message—Running Application Image

```
Running App Image
Press Push Button 2 for Factory Image
```

9. Press BUTTON2 to go back to the factory image. The following message (Example 4) appears before the terminal closes.

Example 4. Message—Configuring to Factory Image

```
Button2 is pressed, configuring to Factory Image
The nios2-terminal will now close. Please open a new one for Factory
Image
```

10. You can reconfigure back and forth with factory or application image by pressing the corresponding push buttons.

CAUTION

Remember to open a new terminal session every time before you reconfigure the device. The example software uses the reduced device driver for the JTAG UART, to achieve a small code footprint. The reduced device driver is a polling implementation that assumes the host is always connected. If the host is not connected, the system can hang after reconfiguration occurs. If you do not need the JTAG-UART component, remove it from the system and remove the printing commands from the software program.

For more information on the JTAG-UART software programming model, see the *JTAG-UART Core* chapter in volume 5 of the *Quartus II Development Software Handbook*.

Rebuilding the Example Design

Perform the following steps to create a complete design from the provided Quartus II project example:

1. Open the **niosII_cycloneIII_compact.qpf** file located in *<work_directory>***/compact_config/ factory_image_ex** with the Quartus II software.
2. On the Tools menu, click **SOPC Builder**.
3. In SOPC Builder, click **Generate** to generate the necessary files.
4. Click **Exit** once the generation successfully completes.
5. Perform the following steps to create the software project for the factory image:
   a. Open a Nios II command shell.
   b. Set the directory to *<work_directory>***/compact_config/factory_image_ex/software/app/factory_image**.
   c. Type the following command in the command shell (this may take some time):

```
./create-this-app
```

   You have successfully created an executable and linking format file (**.elf**) and a memory initialization file (**.hex**). The **.hex** file is used to initialize the on-chip memory with the software program. The Nios II processor then boots from this memory and executes the program.

If you wish to reduce the software code footprint, see "Software Board Support Package (BSP) Settings" on page 12.

6. In the Quartus II Processing menu, click **Start Compilation** to run a full compilation for this project. This may take some time.
7. Click **OK** when compilation completes. An SRAM object file (**.sof**) which includes the **.hex** file is created.
8. Repeat the same steps for application image using the project directory located at *<work_directory>***/compact_config/ application_image_ex**.
9. Set the directory to *<work_directory>***/compact_config/utilities** in the command shell.
10. To convert and combine the newly created **.sof** files for the factory and application projects into a single **.flash** file, type the following commands in the command shell:

```
./flash_convert.sh \
../factory_image_ex/niosII_cycloneIII_compact.sof \
../application_image_ex/niosII_cycloneIII_compact.sof \
0x20000 0xA00000
```

    The factory image is stored at offset 0x20000 of flash memory while the application image is stored at offset 0xA00000. You can change this offset value in the software code, see "Changing Configuration Image Offset Addresses" on page 11.
11. Upon completion of this script, an **output.flash** file is created in the utilities directory. Program the **.flash** file into the onboard flash memory, see "Running the Example Design" on page 7.

Adding the Example Design to an Existing Design

The following procedures depend on whether your design already contains an SOPC Builder system. This section describes how you can add the example design to systems both, with and without SOPC Builder systems.

Adding to a Design with No SOPC Builder system

If your existing design does not have an SOPC Builder system, you add the design by doing the following:

1. Copy the **niosII_cycloneIII_compact_sopc.bsf** and **niosII_cycloneIII_compact_sopc.sopc** files to your design project directory.
2. Open your top-level block diagram (**.bdf**) file in the Quartus II software.
3. On the Edit menu, click **Insert Symbol**.
4. In the **Symbol** page, expand the **Project** folder and click **niosII_cycloneIII_compact_sopc**.
5. Click **OK**.
6. Place the block anywhere you like in the block diagram file.
7. Double-click on the **niosII_cycloneIII_compact_sopc** block to open SOPC Builder.
8. In SOPC Builder, click **Generate** to generate the necessary files.
9. Click **Exit** once generation completes.

10. In the Quartus II software, right-click on the **niosII_cycloneIII_compact_sopc** block and click **Generate Pins for Symbols Ports** to connect the block with the input/output pins.

    The flash device that the CFI flash controller connects to may use word-addressing instead of byte-addressing. If this is the case, discard the lowest bit of the flash address bus when connecting it to the output pins.

11. Assign the pins according to the schematics of the board you are using.

    Make sure the clock connected to the system has a frequency of 50 MHz.

12. Recompile the design.

Make sure your design is targeted to a Cyclone III device and set to use the AP configuration scheme in **Remote** mode.

Refer to the steps given in "Active Parallel (AP) Configuration Scheme" on page 1 to enable the AP configuration scheme and "Cyclone III Remote System Upgrade Feature" on page 2 to set the **Remote** configuration mode.

Adding to a Design with SOPC Builder system

If your existing design already contains an SOPC Builder system, you can add the remote update controller to the system by doing the following:

1. To open your system in SOPC Builder, click **SOPC Builder** under the Tools menu in the Quartus II software.
2. In the SOPC Builder, expand **Peripherals** and expand **FPGA Peripherals**
3. Select **Remote Update Controller** (**Cyclone III**).
4. Click **Add**.

Make sure that the clock connected to this controller has a frequency of 40 MHz.

Changing Configuration Image Offset Addresses

This example requires the factory image to be stored at flash memory offset address of 0x20000 and the application image at offset 0xA00000. As the Cyclone III device by default uses 0x20000 (byte-addressing) for configuration data, it is recommended that the factory image be placed here. You can place your application image at any offset according to your need.

To change an image offset, you need to modify the **factory_image_main.c source** file. This file is located at *<work_directory>***/compact_config/factory_image_ex/software/app/factory_image**. Change the value of `hw_flash_offset` in the file to the desired offset.

Reconfiguration is not limited to a single application image. You can implement several application images and each image can configure the FPGA with any other image, provided the reconfiguration feature is enabled. All you need to do is to define the corresponding image offset address in your software routine.

To create a valid flash file that contains all your images, you can modify the **flash_convert.sh** script in the **utilities** folder. For more information on how to modify the script, refer to the comments inside the **flash_convert.sh** script.

Settings and Performance Results

This section shows the settings that apply to this example design and some expected performance results.

Quartus II Project Settings

This project uses the Quartus II **Area Optimization Technique**. This technique attempts to minimize logic usage. Table 2 shows the Fitter report on resources usage.

Table 2. Resources Used

Entity	Resources Usage	
	Actual Value	Percentage (%)
Total Logic Elements	2,100/24,624	9
Total Memory Bits	43,000/608,256	7
Total Pins	54/216	25

The results may vary depending on the version of the Quartus II software.

TimeQuest Timing Analyzer Settings

This design uses a constraints file for timing analysis purpose. For more information on the constraints, see the comments in the **niosII_cycloneIII_compact_constraints.sdc** file located in your project directory.

Software Board Support Package (BSP) Settings

In order to make the software code footprint small, several BSP settings are used, as shown in Table 3.

Table 3. BSP Settings

Settings ***(Note 1)***	**Selected Option**
Small C Library	Enabled
Program Never Exits	Enabled
Support C++	Disabled
Lightweight Device Driver API	Enabled
Clean Exit	Disabled
ModelSim® only, No Hardware Support	Disabled
Reduced Device Drivers	Enabled
Compiler Optimization Levels	Optimized size (Os)

Note to Table 3:

(1) You can view the BSP settings in the `create-this-bsp` script located in *<work_directory>***/compact_config/factory_image_ex/software/bsp/hal_reduced_footprint**.

The compiled software code size is about 3 Kbytes. This can be further reduced to a smaller size by turning off the message printing commands in the software code. You can do so by changing the value of `USE_JTAG_UART` from 1 to 0, located at the top section in the source files: **factory_image_main.c** and **app_image_main.c**.

Restoring the Application Selector

If you run the design, the application selector is erased from the Nios II Embedded Evaluation Kit. To restore the application selector on the kit, perform the following steps:

1. Using the Quartus II Programmer, configure the FPGA with the **.sof** file

   altera/<*version number*>**/kits/cycloneIII_3c25_niosII_eval/examples/application_selector/cycloneIII_embedded_evaluation_kit_application_selector.sof**

2. Open a Nios II command shell and set the directory to

   altera/<*version number*>**/kits/cycloneIII_3c25_niosII_eval/factory_recovery/flash_contents**

3. In the Nios II command shell, program the factory image into flash by typing the following command:

   ```
   nios2-flash-programmer --base=0x04000000 \
   restore_cycloneIII_3c25.flash
   ```

4. Reset the board to start the application selector.

Conclusion

This application note provides the information on how to take advantage of the built-in remote system update circuitry in Cyclone III and use a small Nios II processor system to manage multiple configuration images. This scheme eliminates the need for an external configuration controller in your design. You can extend this method to manage any number of user applications stored in inexpensive commodity flash. The accompanied example design shows one of the faster ways to get your application ready with the configuration features.

Revision History

Table 4 shows the revision history for this application note.

Table 4. Document Revision History

Date and Revision	Changes Made	Summary of Changes
November 2008, version 1.0	Initial Release.	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support