Generic Nios II Booting Methods User Guide

2016.05.24

UG-20001 Subscribe Send Feedback

Overview

Altera® Nios® II processor is a soft processor that supports all Altera System on Chip (SoC) and Field Programmable Gate Array (FPGA) families. Developing Nios II embedded programs can be based on Altera hardware abstraction layer (HAL). The creation and management of software projects based on the HAL is integrated tightly with the Nios II Software Build Tools (SBT).

The boot memory could be the Compact Flash Interface (CFI) flash, User Flash Memory (UFM) flash, Altera Serial Flash (EPCS)/Altera Quad Serial Flash (EPCQ) configuration device or on-chip RAM (OCRAM). Regardless of the nature of the boot memory, HAL-based systems are constructed so that the reset vector and all program and data sections are initially stored in the boot memory. The HAL provides a small boot loader program (also known as boot copier) that copies these sections to their run time location at boot time. You can specify the run time locations for program and data memory by manipulating the Nios II BSP settings.

This document describes:

- The Nios II processor boot copier that boots your Nios II system according to the boot memory selection
- Nios II processor booting options and general flow
- Nios II programming solutions for selected boot memory

Prerequisites

You are required to have knowledge in instantiating and developing a system with a Nios II processor. Altera recommends that you go through the following online tutorials and training materials before using this application note:

- *Nios II Classic Software Developer's Handbook*
- *Nios II Flash Programmer User Guide*
- *AN736: Nios II Processor Booting from Altera Serial Flash (EPCQ)*
- *AN730: Nios II Processor Booting Methods in MAX 10 Devices*
- *AN370: Using the Serial Flash Loader with the Quartus Prime Software*
- *Parallel Flash Loader IP Core User Guide*
- *AN458: Alternative Nios II Boot Methods*

Related Information

- **Nios II Classic Software Developer's Handbook**

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

- **AN736: Nios II Processor Booting from Altera Serial Flash (EPCQ)**
- **AN730: Nios II Processor Booting Methods in MAX 10 Devices**
- **Nios II Flash Programmer User Guide**
- **AN370: Using the Serial Flash Loader with the Quartus Prime Software**
- **Parallel Flash Loader IP Core User Guide**
  Formerly, AN386: Using the Parallel Flash Loader with the Quartus Prime Software
- **AN458: Alternative Nios II Boot Methods**

Acronym

Acronym	Definition
Avalon MM	Avalon Memory Map
CFI	Compact Flash Interface
EPCQ	Altera Quad Serial Flash
EPCS	Altera Serial Flash
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
OCRAM	On-chip RAM
RAM	Read Access Memory
SBT	Software Build Tools
SoC	System on Chip
UFM	User Flash Memory
XiP	Execute-in-place

Nios II Processor Boot Copier

The Nios II processor boot copier has the following features:

- Locates the software application in memory software application in memory
- Unpacks and copies software application image to Read Access Memory (RAM)
- Automatically switches to application code in RAM after copy completes

The boot copier is placed at the reset address if the runtime location of the .text section is outside of the boot memory.

However, if the runtime location of the .text section is in the boot memory, the system does not need a separate loader. Instead the `_reset` entry point in the HAL executable program is called directly. The function `_reset` initializes the instruction cache and then calls `_start`. This initialization sequence lets you develop applications that boot and execute directly from flash memory.

You may enable the `alt_load()` function in the HAL code. This function operates as a mini boot copier that copies only the writable memory sections to RAM based on the BSP settings. It optionally copies sections from the boot memory to RAM. It can copy data sections (**.rodata**, **.rwdata**, **.exceptions**) to RAM

but not the code sections (**.text**). The code section (.text) section is a read-only section and remains in the booting flash memory region. This partitioning helps to minimize the RAM usage but may limit the code execution performance because accesses to flash memory are slower than accesses to the on-chip RAM. The `alt_load()` function can be enabled in the BSP Settings.

BSP Settings	Functions[1]
`hal.linker.enable_alt_load`	Enable `alt_load()` function.
`hal.linker.enable_alt_load_copy_rodata`	`alt_load()` copies **.rodata** section to RAM.
`hal.linker.enable_alt_load_copy_rwdata`	`alt_load()` copies **.rwdata** section to RAM.
`hal.linker.enable_alt_load_copy_exceptions`	`alt_load()` copies **.exceptions** section to RAM.

For more information about the Nios II default sections and the `alt_load()` function, please refer to the "Nios II Software Build Tools" chapter in the *Nios II Classic Software Developer's Handbook*.

Besides the `alt_load()` function, there are two types of Nios II boot copiers to support Nios II system booting:

- Memcpy-based boot copier
- EPCS Controller- based boot copier

Related Information

Nios II Classic Software Developer's Handbook

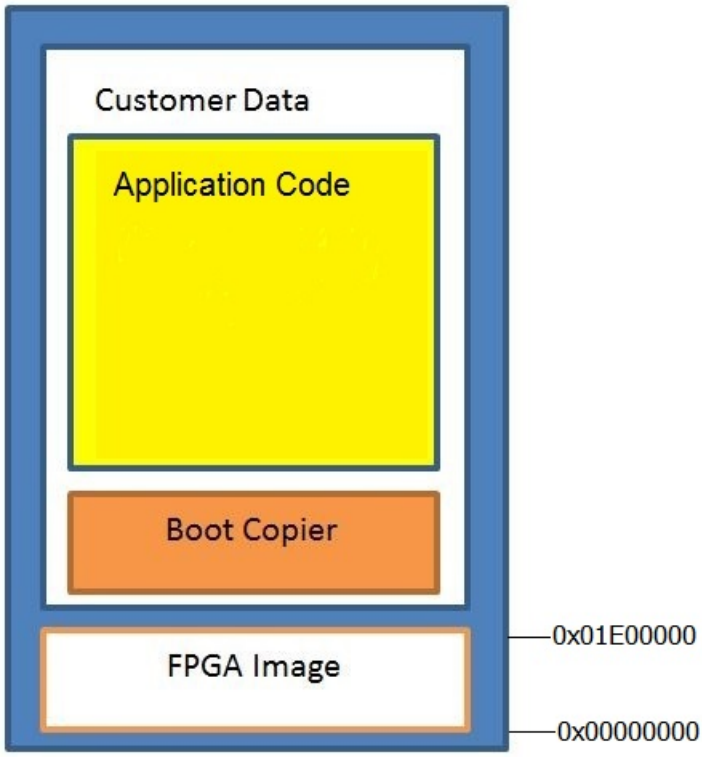
Memcpy-Based Boot Copier

Memcpy-based boot copier supports MAX® 10 UFM, EPCQ and CFI flash memory. The boot copier is stored within the flash memory as the flash controller supported Execute-in-Place (XiP). Subsequent boot image is located just right after the boot copier. You need to ensure the Nios II reset vector offset points to the start of the boot copier.

The following figure shows the flash memory map for EPCQ and CFI flash when using a boot copier. This memory map assumes a FPGA image is stored at the start.

[1] **.text**, **.rodata**, **.rwdata** and **.exceptions** are part of the Nios II default section names.

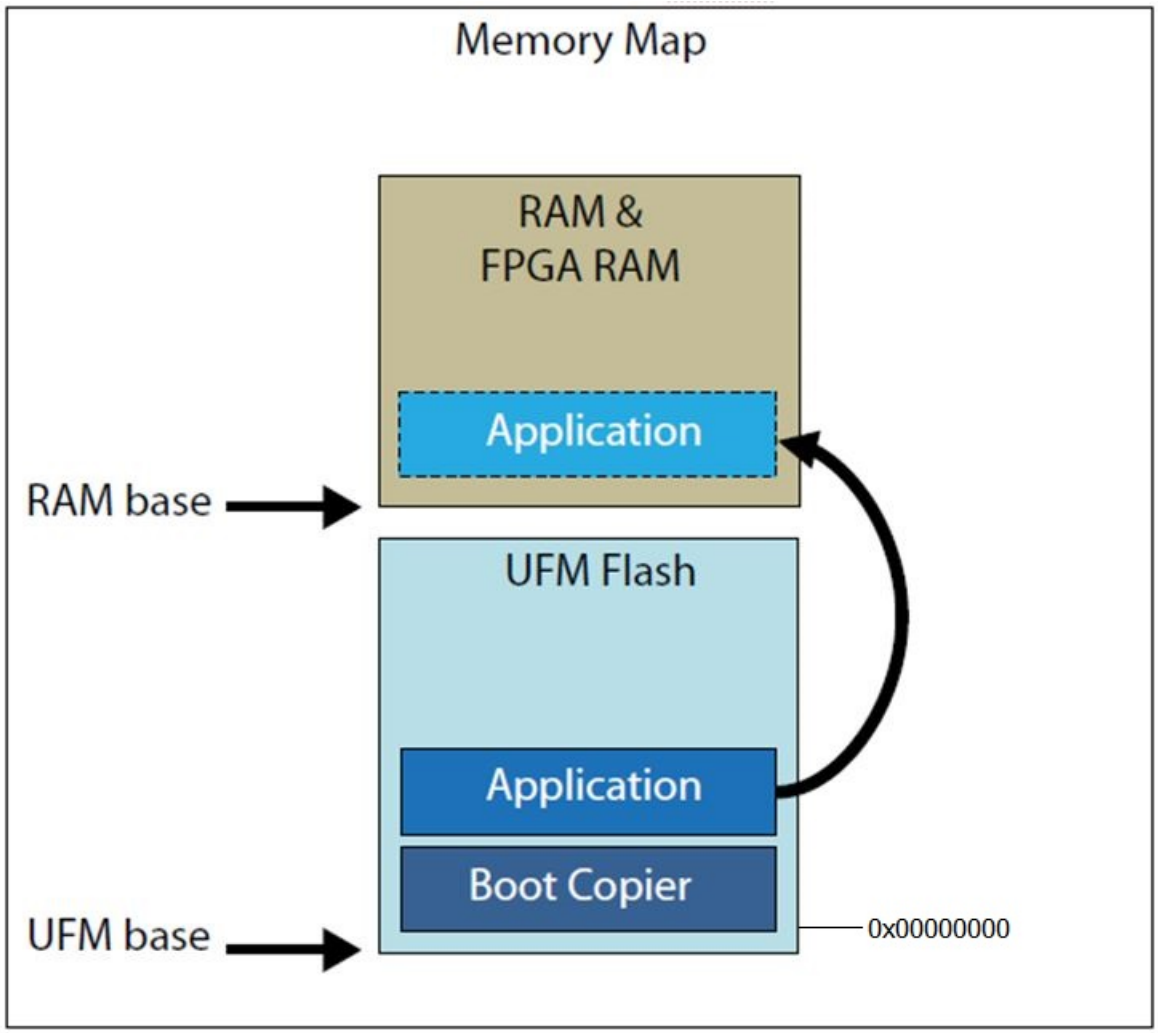
Figure 1: Memory Map for EPCQ and CFI Flash with Memcpy-based Bootcopier



Note: At the start of the memory map is the FPGA image, followed by the customer data which consists of boot copier and application code. The size of the FPGA image is unknown and the exact size can only be known after the Quartus Prime project compilation. The Nios II reset vector offset must be set in Qsys and must point to the start of the boot copier which is located after the FPGA image. You will have to determine an upper bound for the size of the FPGA image. For instance, if the size of the FPGA image is estimated to be less than `0x01E00000`, you can set the Nios II Reset Vector offset to `0x01E00000` in Qsys, which is also the start of the boot copier.

The following diagram shows the memory map of a system using UFM flash and the memcpy-based controller. Since the FPGA image (***.sof**) is stored in MAX10 CFM section, the boot copier is located at the base address of UFM, followed by the application code. Hence, the Nios II reset vector offset can be set to address **0x00000000** in Qsys.

Figure 2: Memory Map of a System Using UFM with memcpy-based Boot Copier



The memcpy-based boot copier is automatically incorporated as part of the:

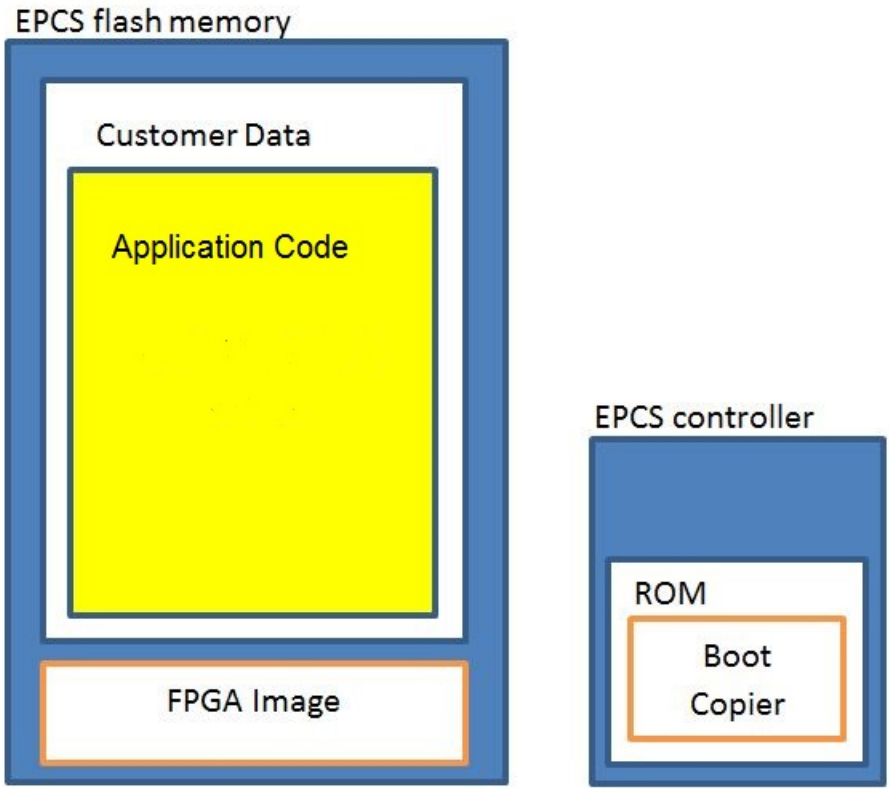
- HEX file during memory initialization file generation ("`mem_init_generate`" target) in the Quartus Prime Programming Flow method
- SREC image (***.flash**) file during elf2flash utility execution in the Nios II Flash Programming Flow method

EPCS Controller-Based Boot Copier

EPCS controller-based boot copier supports EPCS memory only. Boot copier is stored in the ROM within the EPCS flash controller. The boot copier is included during Qsys and Quartus Prime project compilation time.

The next stage boot image is located in the EPCS memory flash. The EPCS controller-based boot copier is automatically appended into the SREC image (*.flash) file during the **elf2flash** utility execution in the Nios II Flash Programming flow method.

Figure 3: Memory Map for EPCS Flash with EPCS Controller-based Bootcopier



Note: Quartus Prime and Nios II Flash programming flow is briefly described in the Nios II Booting General Flow diagram.

Related Information

Nios II Booting General Flow on page 8

Nios II Processor Booting Methods

There are a few methods to boot up Nios II processor in Altera devices. The methods to boot up Nios II processor varies according to the flash memory selection.

The following is the supported flash memories with respective boot options:

Supported Flash Memories	Nios II Booting Options	Boot Copier
CFI Flash	Nios II processor application execute-in-place from CFI flash	Enable `alt_load ()`
	Nios II processor application copied from CFI flash to RAM using boot copier	Memcpy-based

Supported Flash Memories	Nios II Booting Options	Boot Copier
EPCS Flash	Nios II processor application copied from EPCS flash to RAM using boot copier	EPCS controller-based
UFM (MAX 10 only)	Nios II processor application executes in-place from Altera On-chip Flash (UFM)	Enable `alt_load ()`
	Nios II processor application copied from UFM to RAM using boot copier	Memcpy-based
EPCQ Flash	Nios II processor application execute-in-place from EPCQ flash	Enable `alt_load ()`
	Nios II processor application copied from EPCQ flash to RAM using boot copier	Memcpy-based
Altera On-chip Memory (OCRAM)	Nios II processor application executes in-place from Altera OCRAM	No boot copier required

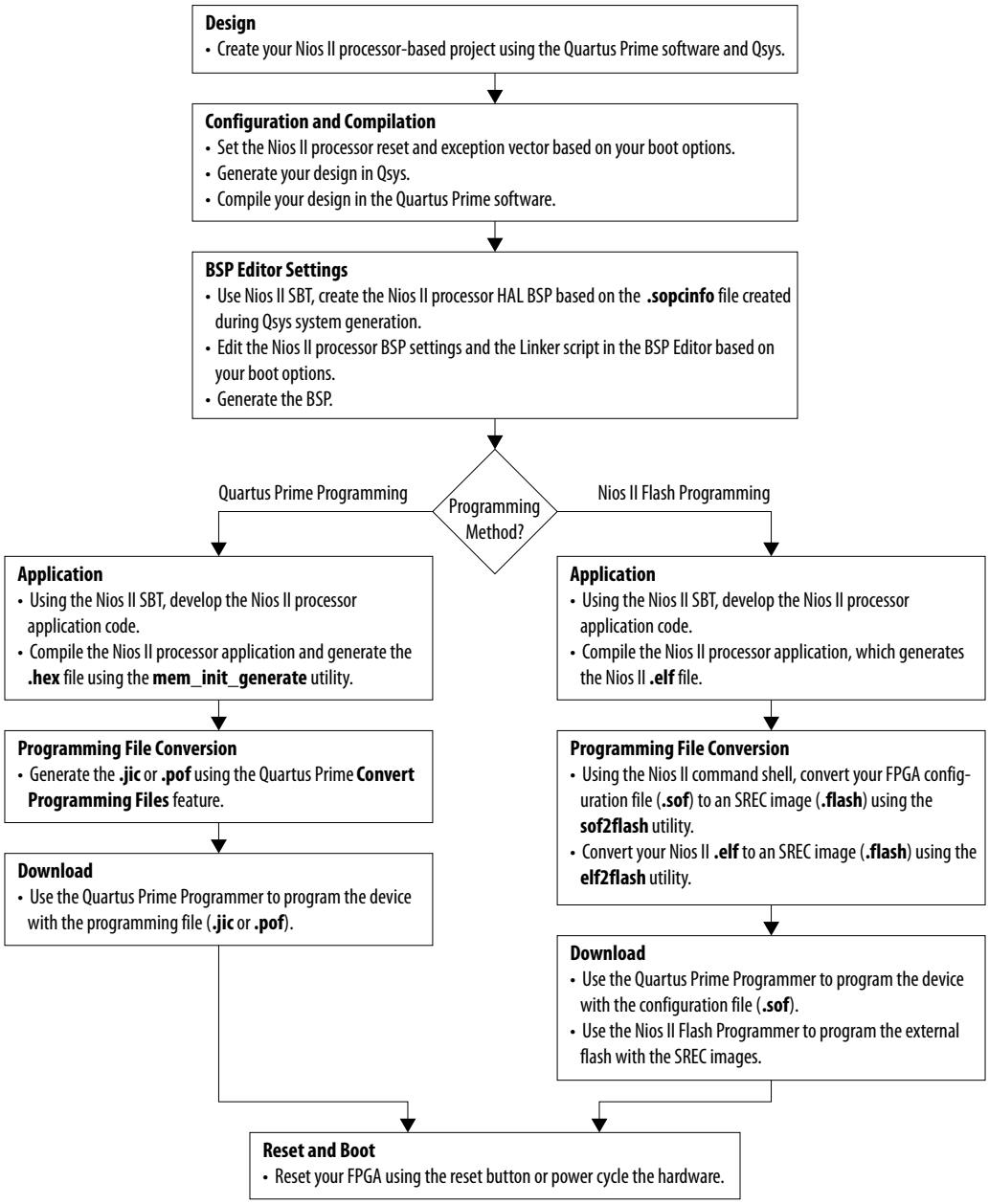
For more information about Nios II Processor booting from UFM and execute-in-place from Altera OCRAM, refer to *AN730: Nios II Processor Booting Methods in MAX 10 Devices*.

For more information about Nios II Processor booting from Altera EPCQ flash, refer to *AN736: Nios II Processor Booting from Altera Serial Flash (EPCQ)*.

Related Information

- **AN736: Nios II Processor Booting from Altera Serial Flash (EPCQ)**
- **AN730: Nios II Processor Booting Methods in MAX 10 Devices**

Nios II Booting General Flow



Summary of Nios II Processor Vector Configurations and BSP Settings

Table 1: Summary of Nios II Processor Vector Configurations

Boot Option	Reset Vector Configuration	Exception Vector Configuration
Nios II processor application execute-in-place from CFI flash	CFI flash	Choose between: • OCRAM/ External RAM • CFI Flash

Boot Option	Reset Vector Configuration	Exception Vector Configuration
Nios II processor application copied from CFI flash to RAM using boot copier	CFI flash	OCRAM/ External RAM
Nios II processor application copied from EPCS flash to RAM using boot copier	EPCS controller	OCRAM/ External RAM
Nios II processor application executes in-place from Altera On-chip Flash (UFM)	UFM	Choose between: • OCRAM/ External RAM(2) • UFM
Nios II processor application copied from UFM to RAM using boot copier	UFM	OCRAM/ External RAM
Nios II processor application execute-in-place from Altera On-chip Memory (OCRAM)	OCRAM	OCRAM

Table 2: Summary of Nios II Processor BSP Settings

Boot Option	BSP Settings: Settings.Advanced.hal.linker	BSP Editor Setting: Linker Script
Nios II processor application execute-in-place from CFI flash	If the exception vector memory is set to OCRAM/ External RAM, enable the following settings in **Settings.Advanced.hal.linker:** • `allow_code_at_reset` • `enable_alt_load` • `enable_alt_load_copy_rodata` • `enable_alt_load_copy_rwdata` • `enable_alt_load_copy_exceptions` If the exception vector memory is set to CFI Flash, enable the following settings in **Settings.Advanced.hal.linker:**	• Set `.text` Linker Section to Altera On-chip Flash • Set other Linker Sections (**.heap, .rwdata, .rodata, .bss, .stack**) to OCRAM/ External RAM

(2) Setting exception vector memory to OCRAM/ External RAM is recommended to make the interrupt processing faster.

Boot Option	BSP Settings: Settings.Advanced.hal.linker	BSP Editor Setting: Linker Script
	• `allow_code_at_reset` • `enable_alt_load` • `enable_alt_load_copy_rodata` • `enable_alt_load_copy_rwdata`	
Nios II processor application copied from CFI flash to RAM using boot copier	All settings in **Settings.Advanced.hal.linker** are left unchecked.	All Linker Sections are set to OCRAM/ External RAM
Nios II processor application copied from EPCS flash to RAM using boot copier	All settings in **Settings.Advanced.hal.linker** are left unchecked.	All Linker Sections are set to OCRAM/ External RAM
Nios II processor application executes in-place from Altera On-chip Flash (UFM)	If the exception vector memory is set to OCRAM/ External RAM, enable the following settings in **Settings.Advanced.hal.linker:** • `allow_code_at_reset` • `enable_alt_load` • `enable_alt_load_copy_rodata` • `enable_alt_load_copy_rwdata` • `enable_alt_load_copy_exceptions` If the exception vector memory is set to UFM, enable the following settings in **Settings.Advanced.hal.linker:** • `allow_code_at_reset` • `enable_alt_load` • `enable_alt_load_copy_rodata` • `enable_alt_load_copy_rwdata`	• Set `.text` Linker Section to Altera On-chip Flash • Set other Linker Sections (**.heap, .rwdata, .rodata, .bss, .stack)** to OCRAM/ External RAM
Nios II processor application copied from UFM to RAM using boot copier	All settings in **Settings.Advanced.hal.linker** are left unchecked.	All Linker Sections are set to OCRAM/ External RAM
Nios II processor application execute-in-place from Altera On-chip Memory (OCRAM)	Enable **allow_code_at_reset** in **Settings.Advanced.hal.linker** and leave the other settings unchecked.	All Linker Sections are set to OCRAM

Nios II Processor Application Execute-In-Place from CFI Flash

Altera Avalon CFI Flash maps Avalon Memory Map (Avalon MM) signals into CFI signals which the entire CFI address space is mapped into Altera CFI Flash's Avalon MM slave interface. Hence, CFI devices are immediately accessible by Nios II processor upon system reset without CSR initialization.

For these reasons, Nios II can execute application stored on CFI devices directly without a boot copier.

When a boot copier is not used, the Nios II application `.text` linker sections are set to CFI memory region while `.bss`, `.rodata`, `.rwdata`, `.stack` and `.heap` linker sections are set to RAM memory region. Enable the **alt_load()** function in the BSP Settings to copy the data sections (**.rodata**, **.rwdata**, **.exceptions**) to the RAM upon system reset. The code section (**.text**) remains in the CFI memory region.

For more information, refer to the "Summary of Nios II Processor Vector Configurations" chapter and the "BSP Settings" table for detailed BSP Settings.

Related Information

Summary of Nios II Processor Vector Configurations and BSP Settings on page 8

Nios II Processor Application Copied from CFI Flash to RAM Using Boot Copier

Using the boot copier to copy a Nios II processor application from CFI flash to internal or external RAM for execution helps to improve the execution performance.

The Nios II SBT tool automatically adds the Nios II processor memcpy-based boot copier to the system when the executable file (**.elf**) is converted to one of the following:

- SREC image (***.flash**), if you are using Nios II flash programming method
- Memory initialization file (**.hex**), if you are using the Quartus Prime programming method

The boot copier is located at the base address of the image or file, followed by the application.

For this boot option, the Nios II processor starts executing the boot copier software upon system reset. The software copies the application from the CFI flash to the internal or external RAM. Once this is complete, the Nios II processor transfers the program control over to the application.

Note: Quartus Prime and Nios II Flash programming flow is briefly described in Nios II Booting General Flow diagram.

Related Information

Nios II Booting General Flow on page 8

Nios II Processor Application Copied from EPCS Flash to RAM Using Boot Copier

The EPCS address space is not mapped into the Altera Avalon EPCS Flash Controller's Avalon MM slave interface. Instead, read or write accesses are done through CSRs. Upon system reset, the EPCS device needs to be initialized before usage.

For these reasons, the EPCS controller-based boot copier is required for initializing the EPCS device and copying the Nios II application to RAM for execution.

The EPCS controller instantiates a block of boot ROM internally at its base address (offset `0x0`, just before EPCS controller's CSRs) for storing the boot copier. Nios II reset vector offset must set to EPCS controller base address, such that upon system reset the boot copier is executed first to initialize the EPCS controller and device.

Nios II Booting Elements

Nios II SBT Makefile "mem_init_generate" Target

The Nios II SBT application makefile "`mem_init_generate`" target is responsible for generating memory initialization files using various file conversion tools. The files generated include a HEX file for the UFM data, a HEX file for initialization of the on chip RAM in the SOF and a DAT file for initializing the on chip flash model for simulation.

When required, the Nios II SBT tool automatically adds the Nios II processor default boot copier to the system when the executable file (**.elf**) is converted to memory initialization file (**.hex**). This operation takes place whenever the **.text** section is located in a different memory that the reset vector points to, which indicates a code copy is required. The file conversion happens during execution of "`make mem_init_generate`" command. The "`make mem_init_generate`" command generates different HEX file content based on the specified boot options.

The `mem_init_generate` target also generates a Quartus Prime IP file (**meminit.qip**). Quartus Prime software refers to the **.qip** file for the location of the initialization file. This **.qip** file has to be added into Quartus Prime project if "Initialize flash content" is enabled in Altera On-chip Flash (UFM) or Altera On-chip Memory (OCRAM) IP.

This makefile target is an important element for Quartus Prime programming flow method.

Convert Programming Files Option

Convert Programming Files option in Quartus Prime software is used to convert programming files from one file format to another. This tool is used for combining a ***.sof** and a ***.hex** file into a single ***.pof** or ***.jic** file for programming into the configuration or booting device.

This tool is important for Quartus Prime programming flow method.

Related Information

Quartus Prime Handbook, Volume 3: Verification
For more information about the convert programming files option, refer to the "Quartus Prime Programmer" chapter in volume 3 of the *Quartus Prime Handbook*.

Elf2flash Utility

The **elf2flash** utility generates SREC image (***.flash** extension) by extracting ELF loadable sections. The SREC image can then be programmed into flash devices supported by Nios II Flash Programmer.

For CFI flash devices, both the optional CFI bootcopier and ELF loadable sections are stored in the device. When the CFI bootcopier is not used, the SREC image only contains loadable section data. When the CFI bootcopier is used, the SREC image contains a CFI bootcopier and the ELF payload, where the bootcopier is precompiled to assume the payload is linked immediately after it.

For EPCS/EPCQ flash devices, only the ELF payload is stored in the device. EPCS bootcopier is always required, but is stored within the EPCS controller boot ROM initialized after POF configuration, and thus the SREC image only contains the ELF payload.

When either CFI and EPCS bootcopier is used, **elf2flash** extracts ELF loadable sections as the bootcopier's payload in the form of payload header (4 bytes load address + 4 bytes section size) and payload data (n-

bytes section data) for each loadable sections .**Elf2flash** also generates a last, extra section for storing the `.text` section load address to allow bootcopier to transfer control to the program.

This **elf2flash** utility is an important element for Nios II Flash Programming flow method.

Nios II Programming Solutions

The following table lists the available Nios II processor booting methods with the respective programming solution:

Booting Memory Flash	Programming Solution 1	Programming Solution 2	References
CFI	Nios II Flash Programmer	Quartus Prime Programmer	*Nios II Flash Programmer User Guide* *Parallel Flash Loader IP Core User Guide*
EPCS	Nios II Flash Programmer	Quartus Prime Programmer	*Nios II Flash Programmer User Guide* *AN370: Using the Serial Flash Loader with the Quartus II Software*
MAX10 UFM	Quartus Prime Programmer		*AN730: Nios II Processor Booting Methods in MAX 10 Devices*
EPCQ	Quartus Prime Programmer		*AN736: Nios II Processor Booting from Altera Serial Flash (EPCQ)*
OCRAM	Quartus Prime Programmer		*AN730: Nios II Processor Booting Methods in MAX 10 Devices*

Related Information

- **AN736: Nios II Processor Booting from Altera Serial Flash (EPCQ)**
- **AN730: Nios II Processor Booting Methods in MAX 10 Devices**
- **Nios II Flash Programmer User Guide**
- **AN370: Using the Serial Flash Loader with the Quartus Prime Software**
- **Parallel Flash Loader IP Core User Guide**
  Formerly, AN386: Using the Parallel Flash Loader with the Quartus Prime Software

Recommended Programming Solution

Altera is moving forward with the Quartus Prime programming solution for all Nios II system programming because there is no enhancement made on Nios II Flash Programmer to support new Altera devices. Based on the table in the "Nios II Programming Solutions" chapter, the Quartus Prime programmer is able to support all range of booting flash memories. It is recommended to use the Quartus Prime programming solution for all Nios II system programming.

Related Information

Nios II Programming Solutions on page 13

Document Revision History for Generic Nios II Booting Methods

2016.05.24

UG-20001 Subscribe Send Feedback

Date	Version	Changes
May 2016	2016.05.24	• Renamed Quartus II to Quartus Prime. • Updated the "Summary of Nios II Processor Vector Configurations and BSP Settings" section so that it will be in alignment with AN730.
September 2015	2015.09.11	Removed the Nios II processor boot methods list.
June 2015	2015.06.29	Initial Draft

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered