

Introduction

The Excalibur™ bootloader is one of the system integration tools included with the Quartus® II software. This tool, invoked using the **makeprogfile** utility, creates all the software necessary to boot from flash memory and configure Excalibur devices. The **makeprogfile** utility merges the hardware design, the software design, and the embedded stripe configuration data to create a single bootable image. The Excalibur bootloader can also load memory such as SDRAM or on-chip SRAM with user-application software plus a branch to that section of code. In most cases, this is advantageous because executing code from flash is generally much slower than other types of memory.

Run-from-Flash Mode

In some circumstances, you might want to run application code from flash memory. For example, where a non-speed-critical system has no SDRAM, or in a system where the application code copies itself to the appropriate destination in memory, and then branches to it. In cases such as these, it is beneficial to use run-from-flash mode.

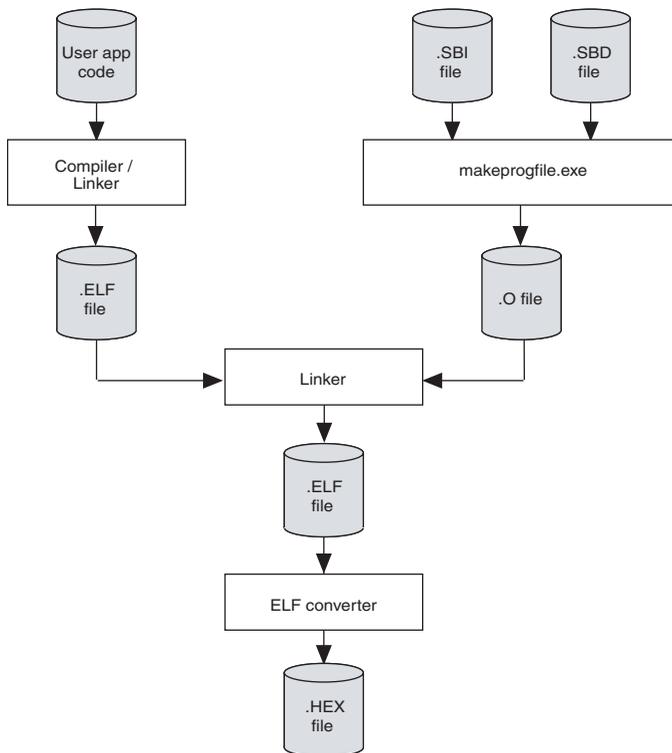
Procedure

Figure 1 on page 2 shows the procedure for using run-from-flash mode with the Excalibur bootloader.



EXCALIBUR™



Figure 1. Using Run-from-Flash Mode with the Excalibur Bootloader

There are three steps involved in running application code from flash memory using the Excalibur bootloader:

1. At the beginning of the application code, define the symbol `__altera_user_start`. This is where the bootloader will branch after it has completed configuring the system.
2. Compile, assemble, and partially link the application software code to create the application `.elf` file.
3. Create the bootloader object file by running the `makeprogfile` utility: specify the project's `.sbd` file and the `.sbi` hardware image.



It is not necessary to submit an `.sbi` hardware image file to the `makeprogfile` utility. If a `.sbi` file is not specified, the bootloader will not configure the FPGA, although it will set up the embedded stripe configuration registers as specified in the `.sbd` file.



Do not specify a `.hex` file, because the presence of a `.hex` filename instructs **makeprogfile** to generate memory-loading routines that are not required if you want to run the application from flash memory.

4. Link the bootloader object file with the application `.elf` file to create an `.elf` file that comprises both the bootloader and application code.
5. Convert the final `.elf` file to a bootable `.hex` file using a conversion utility such as **fromelf** or **elf-objcopy**. The hex file can be programmed into a flash memory and executed. When the bootloader has finished configuring the system, it branches to `__altera_user_start` and begins to execute the application software code.

Command Sequence Example

The following examples demonstrate how a simple assembly file is integrated using the Excalibur bootloader in run-from-flash mode. The first uses the GNUPro Tools for ARM; the second uses the ARM ADS assembler and linker, but any software development suite that supports the ARM922T™ processor can be used.

Using the GNUPro Tools for ARM

The source code below demonstrates how GNUPro tools are used to integrate a simple assembly file using the Excalibur bootloader in run-from-flash mode.

```
#### - Run C Pre-Processor on Assembly - ####
arm-elf-cpp my_soft_app.s -o my_soft_app.i

#### - Assemble Source - ####
arm-elf-as --MD my_soft_app.d --gdwarf2 -L -EL -marm920t -o my_soft_app.o my_soft_app.i

#### - Link Source - ####
arm-elf-ld -p -EL -r -o my_soft_app.elf my_soft_app.o

#### - Create Bootloader object file - ####
makeprogfile -b my_bootdata.o my_system.sbd my_system.sbi

#### - Link Source with Bootloader - ####
arm-elf-ld -p -EL -T armelf.x -entry=__altera_entry -o my_bootable_system.elf my_soft_app.elf
my_bootdata.o %QUARTUS_ROOTDIR%\libraries\software\boot\libboot_xa_ads.a

#### - Convert .ELF file to an Intel-format .HEX file - ####
arm-elf-objcopy -O ihex my_bootable_system.elf my_bootable_system.hex
```

Using the ADS Tools

The source code below demonstrates how ADS tools are used to integrate a simple assembly file using the Excalibur bootloader in run-from-flash mode.

```
#### - Assemble Source - ####
armasm -g my_soft_app.s my_soft_app.o

#### - Link Source - ####
armlink my_soft_app.o -partial -o my_soft_app.elf

#### - Create Bootloader object file - ####
makeprogfile -b my_bootdata.o my_system.sbd my_system.sbi

#### - Link Source with Bootloader - ####
armlink my_soft_app.elf my_bootdata.o
%QUARTUS_ROOTDIR%\libraries\software\boot\libboot_xa_ads.a -ro 0 -o my_bootable_system.elf

#### - Convert .ELF file to an Intel-format .HEX file - ####
fromelf -i32 -o my_bootable_system.hex my_bootable_system.elf
```



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
<http://www.altera.com>
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com

Copyright © 2002 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. 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 Corporation. 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.



I.S. EN ISO 9001