Toolflow for ARM-Based Embedded Processor PLDs

December 2000, ver. 1 Application Note

Introduction

The Excalibur embedded processor devices achieve a new level of system integration from the inclusion of an embedded processor system within a programmable logic device (PLD). Such an integration increases the demands placed on the system development tools and the resulting programming files. To utilize the Excalibur embedded processor device fully, both programmable logic development tools and embedded software development tools are used.

This document describes the tools available to designers using ARM-based embedded processor devices, and explains how to use the tools to generate programming files.

EXCALIBUR™

There are two mechanisms for programming ARM-based embedded processor PLDs: the first facilitates booting from an external flash device; the second is used to boot from an external configuration device, e.g. an EPC2 serial EPROM. The tools described in this document are used to build the various types of programming images for both mechanisms.

Preliminary Information

An Excalibur-based system contains three sections, as follows:

- The digital logic design of the PLD
- The configuration of the embedded processor stripe (see the Excalibur hardware manual for detailed configuration requirements)
- The embedded software application.

The development of digital logic for the programmable logic section of the devices follows the same flow as the design for Altera APEX devices. Typically, the Altera Quartus software development tools are used in conjunction with third-party synthesis tools and third-party hardware simulation tools. Designs can be entered in VHDL or Verilog; or schematic-based designs can be used. A variety of simulation models are provided, depending on the stage of design development reached.

A-AN-136-01

The first tool in the tool chain is the Altera® Excalibur™ MegaWizard® Plug-In. This is a graphical user interface (GUI) utility, which allows designers to create a system build descriptor (SBD, or .**sbd**) file which describes the set-up of the device, including the following characteristics:

- Whether the device boots from an external configuration device or an external flash device
- Which device peripherals are enabled
- Peripheral input-voltage levels
- Peripheral output configurations
- Processor endianness
- Whether the bridges between the stripe and the PLD are used
- The frequencies of the phased-lock loops (PLLs)
- The device memory map

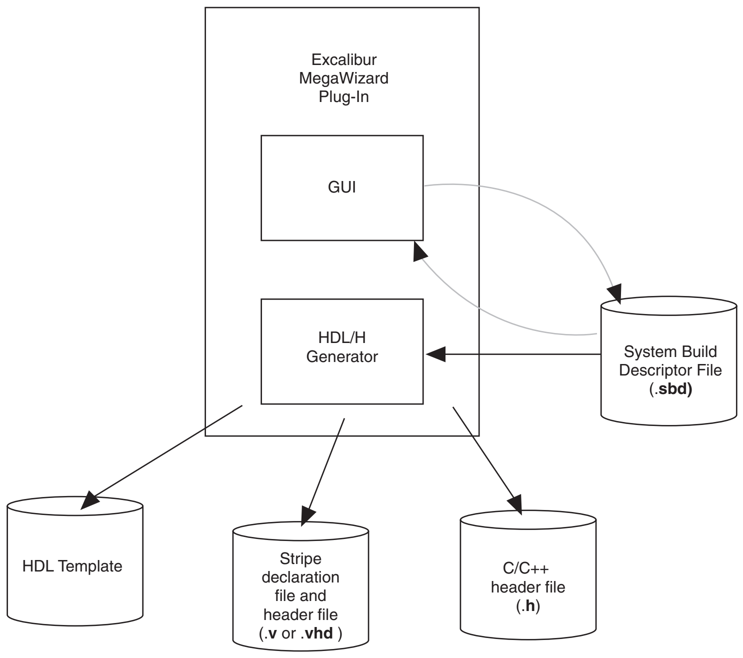
The .**sbd** file produced by the Altera MegaWizard Plug-In is the basis for files which are used to configure the hardware and software design flow. The files produced are as follows:

- **.v** or .**vhd** files containing instantiations of the embedded processor and dual-port RAM blocks and header files, as follows:
  - For Verilog—module instance containing stripe structural code, plus an include file
  - For VHDL—entity instance containing stripe structural code, plus .**vhd** package, plus additional template component declaration (VHDL 87 only)
- A C header file containing definitions of the memory map

Every time the Excalibur MegaWizard Plug-In updates the SBD file, it automatically recreates these files.

Figures 1 shows the MegaWizard process.

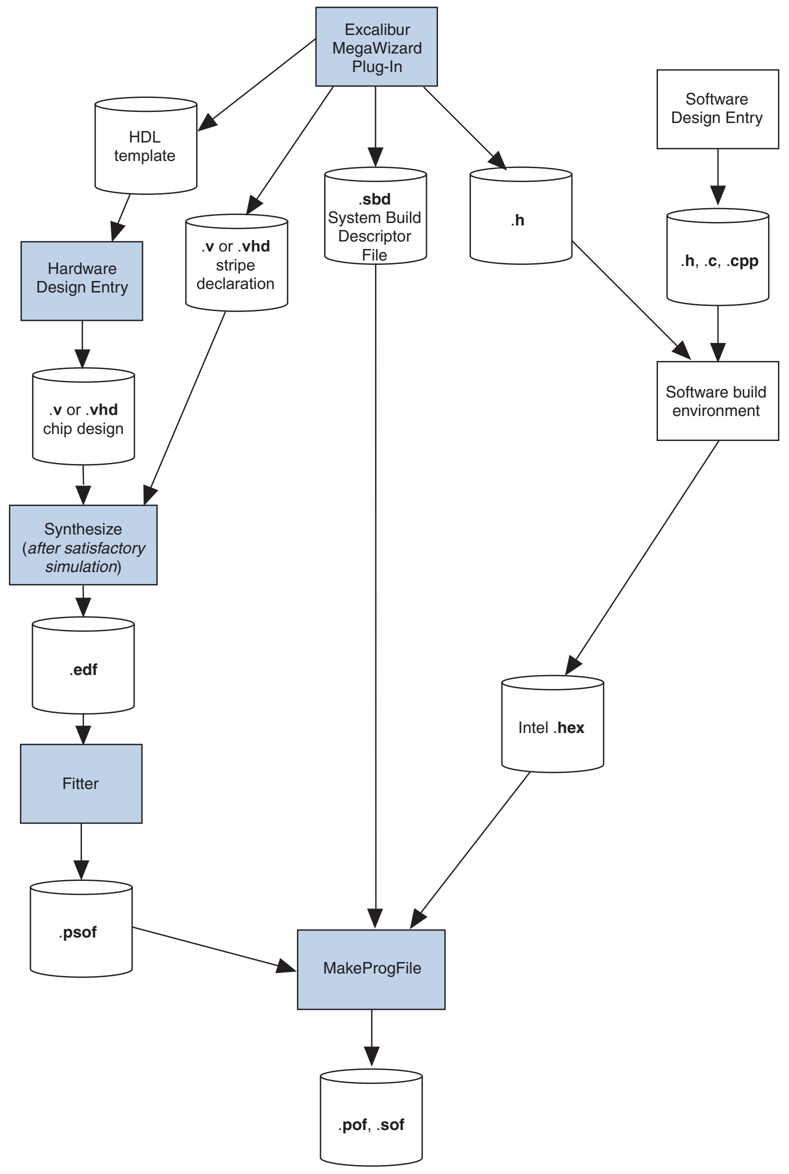
Figure 1. Excalibur MegaWizard Process


Excalibur MegaWizard Plug-In
GUI
HDL/H Generator
System Build Descriptor File (.**sbd)**
HDL Template
Stripe declaration file and header file (.**v** or .**vhd**)
C/C++ header file (.**h**)

Configuration from an External Source

Figure 2 on page 4 shows the tool flow for configuration from an external configuration device, via passive configuration schemes.

Figure 2. Configuration by Passive-Serial or Passive-Parallel Configuration Schemes


Excalibur MegaWizard Plug-In
HDL template
Software Design Entry
.sbd System Build Descriptor File
.h
.v or .vhd stripe declaration
.h, .c, .cpp
Hardware Design Entry
.v or .vhd chip design
Software build environment
Synthesize (after satisfactory simulation)
.edf
Intel .hex
Fitter
.psof
MakeProgFile
.pof, .sof

When a device is configured using a passive-serial or passive-parallel configuration scheme, the required output at the end of a successful hardware compilation is one or more of the following file types: .**pof**, .**sof**, .**rbf**, .**ttf**, .**hexout.**

The following steps explain how to create a programming file of the hardware design:

1. Run the Excalibur MegaWizard Plug-in to configure the embedded logic.

2. Create and synthesize the RTL, using either the Quartus II™ software or third-party hardware development tools.

3. Specify a software image (in Intel .**hex** format) to be merged into the programming file at the fitting stage.

The Quartus II software always produces a .**pof** and a .**sof** file. Optionally, .**rbf**, .**ttf**, and .**hexout** files are also produced.

To generate a configuration file for the software design, proceed as follows:

1. Create an Intel .**hex** file for the software image using either the compiler/linker provided with the Quartus II software in software mode or a preferred utility.

   If the .**hex** file does not specify an entry point, it is assumed to be the first address in the .**hex** file.

   ☞ The ARM **FromElf** utility does not specify an entry point in the .**hex** file, even if it is non-zero, so the first address is always used.

2. Use the Quartus II **MakeProgFile** command-line utility to merge the .**hex** file, the .**sbd** file, and the partial SRAM object file (.**psof**) PLD image into the appropriate types of programming file.

   In software mode, you can specify the .**psof** PLD image to be merged with the .**hex** file. The Quartus II software then produces .**pof**, .**sof**, .**rbf**, and .**ttf** files.

After the programming file has been loaded into the device, and at the instant when execution is transferred to the user's code, the device is configured as follows:

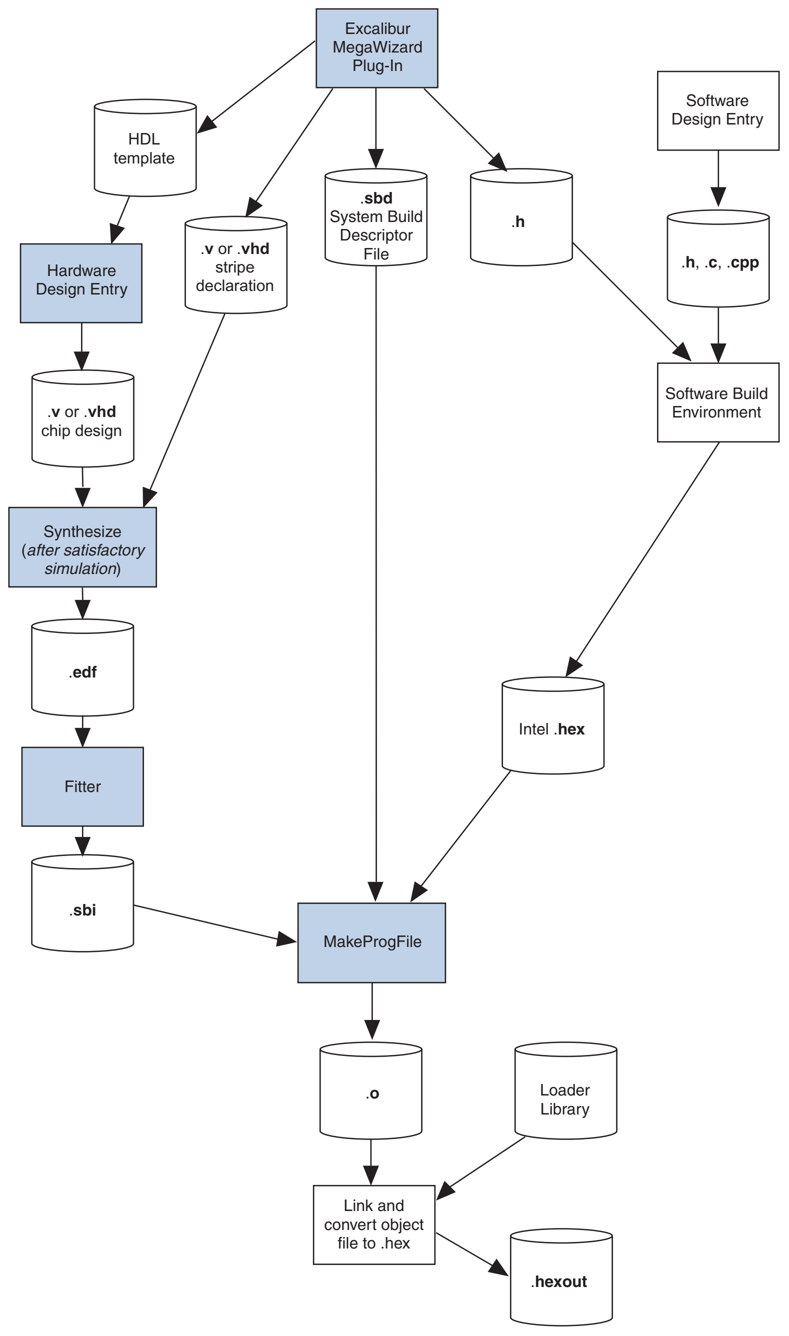
- The device registers have been initialized to the configuration requested in the Excalibur MegaWizard Plug-In
- If no application has been loaded, the processor is held in reset; if an application is present, the processor is released from reset with the following characteristics:
  - The embedded processor is in supervisor (SVC) mode
  - `IRQ` and `FIQ` are disabled; the status flags are undefined
  - The processor is executing ARM code
  - The instruction and data caches, and the MMU are disabled; the cache contents are invalid
  - The embedded processor's registers are undefined
- The watchdog timer is running, unless `DEBUG_EN` is asserted. See the *ARM-Based Embedded Processor PLDs* data sheet for more information about this

Configuration from Flash Via the Altera Bootloader

Altera provides a bootloader for use when booting from external flash memory. The bootloader initializes the device registers according to the MegaWizard output, including setting up the memory map of the device; and then loads the software into RAM. It resets the watchdog timer and finally sets the endianness of the processor, before passing control to the user's code.

Figure 3 on page 7 shows the tool flow for configuration from flash memory.

Figure 3. Configuration from Flash


Excalibur MegaWizard Plug-In
HDL template
.v or .vhd stripe declaration
.sbd System Build Descriptor File
.h
Software Design Entry
.h, .c, .cpp
Hardware Design Entry
.v or .vhd chip design
Software Build Environment
Synthesize (after satisfactory simulation)
.edf
Intel .hex
Fitter
.sbi
MakeProgFile
.o
Loader Library
Link and convert object file to .hex
.hexout

When a device is configured using the Altera flash bootloader, the required output at the end of a successful design compilation is an Intel .**hex** file.

To run a hardware compilation and produce a .**hex** file for configuring a device from flash memory, proceed as follows:

1. Run the Excalibur MegaWizard Plug-In to configure the embedded logic.

2. Create an Intel .**hex** file for the software image using either the compiler/linker provided with the Quartus II software in software mode or a preferred utility.

   If the .**hex** file does not contain an entry point, it is assumed to be the first address in the .**hex** file.

   ☞ The ARM **FromElf** utility does not specify an entry point in the .**hex** file, even if it is non-zero, so the first address is always used.

3. Use the Quartus II software to compile the design, generating a slave binary image (.**sbi**) file.

4. Use the **MakeProgFile** command-line utility to merge the .**hex** file, the .**sbd** file, and the .**sbi** PLD image into an object file.

5. Using **Armlink** or a similar utility, link the object file produced by **MakeProgFile** with the boot library **boot.a** to produce an executable and linkable format (.**elf**) file.

6. Use **FromElf** or a similar application to create a .**hex** programming file.

☞ Steps 4 to 6 can be placed into a makefile to be called from the Quartus command line after the build.

After the programming file has been loaded into the device, and at the instant when execution is transferred to the user's code, the device is configured as follows:

- The device registers have been initialized to the configuration requested in the Excalibur MegaWizard Plug-in
- The embedded processor is in SVC mode; in addition:
  - `IRQ` and `FIQ` are disabled; the status flags are undefined
  - The processor is executing ARM code
  - The instruction and data caches, and the MMU are disabled; the cache contents are invalid
  - The embedded processor's registers are undefined
- The watchdog timer is running, unless `DEBUG_EN` is asserted. See the *ARM-Based Embedded Processor PLDs* data sheet for more information about this.

☞ The watchdog is reset immediately before execution is transferred to the user's code.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com
Applications Hotline:
(800) 800-EPLD
Customer Marketing:
(408) 544-7104
Literature Services:
lit_req@altera.com