LEON-CERTUS

LEON-CERTUS Bitstreams Quick Start Guide

USER MANUAL

RELEASED JULY 2023

Frontgrade Gaisler AB | Kungsgatan 12 | SE-411 19 | Göteborg | Sweden | frontgrade.com/gaisler

Table of Contents

1. Introduction

1.1. Overview

This document is a quick start guide for the LEON-CERTUS example bitfile.

The purpose of this document is to get users quickly started using the example bitfiles.

1.2. Prerequisites

To use the provided bitstream, the user needs:

- Certus-NX Versa Evaluation Board
- Lattice Radiant (to program the FPGA), available at https://www.latticesemi.com/latticeradiant
- On a Linux system, the user needs to install D2XX proprietary FTDI drivers (for Lattice Radiant to connect the board), available at https://ftdichip.com/drivers/d2xx-drivers/. On a Windows system this should be taken care automatically by the system.
- GRMON3 Hardware Debugger, latest available version, available at https://www.gaisler.com/index.php/downloads/debug-tools

1.3. References

Table 1.1. References

RD-1	Certus-NX Versa User's Manual [https://www.latticesemi.com/products/developmentboardsandkits/certus_nx_versa_eval]
RD-4	GRMON User's Manual [https://www.gaisler.com/doc/grmon3.pdf]
RD-10	Bare C Cross-Compilation System [https://www.gaisler.com/index.php/products/operating-systems/bcc]
RD-11	BCC User's Manual [https://www.gaisler.com/doc/bcc2.pdf]

2. LEON-CERTUS System-On-Chip

2.1. Overview

The LEON-CERTUS example bitstream is based on an architecture centered around the AMBA Advanced High-speed Bus (AHB), to which the processor and other high-bandwidth units are connected. Low-bandwidth units are connected to the AMBA Advanced Peripheral Bus (APB) which is accessed through an AHB to APB bridge. The architecture for the basic design is shown in Figure 2.1.

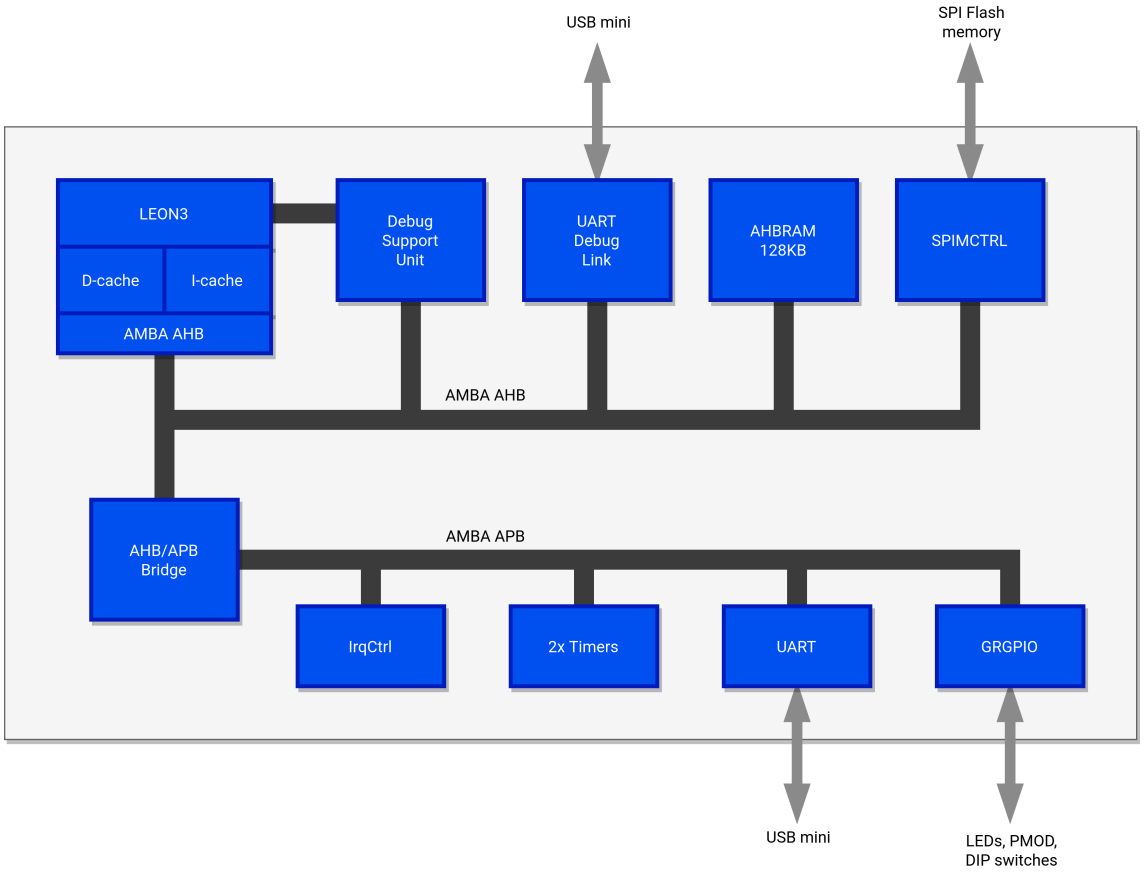

Figure 2.1. Architectural block diagram of LEON-CERTUS

The LEON-CERTUS architecture includes the following modules:

- LEON3 SPARC processor with 2 KiB instruction cache and 2 KiB data cache
- Debug Support Unit with trace buffer
- UART Debug Link
- 128 KiB of on-chip RAM
- Timer unit with two 32-bit timers
- Interrupt controller for 15 interrupts in two priority levels
- UART with FIFO and separate baud rate generator
- SPI Flash memory controller
- General purpose I/O port (GPIO)

In the following sections we'll go through each IP. For a more thorough description of each IP, please refer to the GRLIB IP Library User's Manual available at https://gaisler.com/products/grlib/grip.pdf

2.2. System Configuration

In this section we describe the configuration of the design and of the IP blocks instantiated. Table 2.1 provides an overview of the processor configuration.

Table 2.1. System configuration of provided bitstream

Configuration name	Value
Processor	LEON3
Processor configuration	MIN
Number of processor cores	1
Level-1 cache	2+2 KiB
Floating Point Unit	No
Debug Support Unit	Yes
RAM on-chip memory	128 KiB
UART Debug Link	Yes

2.2.1. Reset and LED status

For basic operations with the design, some signals are tied to in/outputs devices on the board:

- `J2`: USB UART interface via FTDI with mini-USB connector
- `SW2` (push button): Main reset signal
- `LED_0`: PROC_ERRORN
- `LED_1`: DSU_ACTIVE

More in/outputs are accessible through the GRGPIO core (see Section 2.3.3).

2.3. Cores and Memory Map

We present here the cores instantiated in the design with their AHB/APB memory mapping and interrupt. For more information on the operations of each individual core, please refer to the GRLIB IP Library User's Manual available at https://gaisler.com/products/grlib/grip.pdf

Table 2.2. Cores and their AHB/APB memory mapping

Core	Function	Memory mapping	IRQ
AHBCTRL	AHB Arbiter & Decoder	-	-
LEON3	LEON3 SPARC 32-bit processor	-	-
AHBRAM	On-chip RAM memory	AHB: `0x40000000-0x40100000`	-
APBCTRL	AHB/APB Bridge	AHB: `0x80000000-0x80100000`	-
DSU	LEON3 Debug support unit	AHB: `0xD0000000-0xE0000000`	-
SPIMCTRL	Controller for SPI Flash memory Register area SPI Flash memory area	 AHB: `0xFFF00000-0xFFF00100` AHB: `0x00000000-0x01000000`	10
APBUART	8-bit UART with FIFO	APB: `0x80000100-0x80000200`	2
IRQMP	LEON3 Interrupt controller	APB: `0x80000200-0x80000300`	-
GPTIMER	Modular timer unit with watchdog	APB: `0x80000300-0x80000400`	8
AHBUART	Serial/AHB debug interface	APB: `0x80000400-0x80000500`	-
GRGPIO	General purpose I/O port	APB: `0x80000500-0x80000600`	4
AHBSTAT	AHB failing address register	APB: `0x80000800-0x80000C00`	-

2.3.1. LEON3

LEON3 is a highly configurable 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. The LEON3 is instantiated in the MIN configuration. For further information about the MIN configuration, see the LEON/GRLIB Configuration Guide available at https://gaisler.com/products/grlib/guide.pdf

2.3.2. SPIMCTRL

This IP block allows the user to access the SPI Flash memory featured on the board. The user can set the IP and the memory to work in Extended, Dual or Quad SPI. By issuing `spim flash detect`, grmon can recognise the memory and therefore load the needed settings to operate

```
grmon3> spim flash detect
    Got manufacturer ID 0x20 and device ID 0xbb18
    Detected device: Micron N25Q128A
```

For more information about how to use grmon with SPI Flash memory, please refer to the GRMON3 User's Manual available at https://www.gaisler.com/doc/grmon3.pdf

2.3.3. GRGPIO

Through this IP block, the user can control the following components:

- `LED_[4..7]`: Connected to GPIO outputs [0..3] (active LOW).
- `PMOD0_[1..8]`: Connected to GPIO in-outputs [4..11]. They can be configured as input (0) or output (1) through the value assigned to `I/O port direction register` (can be set through GRMON).
- `DIP_SW[1..4]`: Connected to GPIO inputs [12..15]

For more information about its functionalities, please refer to the GRLIB IP Library User's Manual available at https://gaisler.com/products/grlib/grip.pdf

3. Using the board

3.1. Programming the FPGA

To program the Certus-NX, after downloading the bitstream from the webpage:

1. Connect the Certus-NX Versa Development Board to a USB port on the PC by using a mini-USB cable connected to the `J2` connector on the board. Also, power on the board using the provided power supply.
2. If on a Linux system, unload the `ftdi_sio` kernel module by issuing in a terminal (it may require root privileges)

   ```
   modprobe -r ftdi_sio
   ```

   Remember that every time the usb cable is connected to the PC (or in case the board is powered off and on) the `ftdi_sio` module needs to be unloaded again.
3. Launch Radiant. Once it's open, click on Radiant Programmer
4. In the pop-up window, create a project by writing a "Project Name", pointing to a folder in the "Project Location" and selecting "Create a new project from a scan". Check that the "Cable" selected is the FTDI one and the "Port" is FTUSB-0 (press on "Detect Cable" to be sure that all drives and modules are working). Press "Ok"
5. After the scan, you should have all the information regarding the device listed in a row. Under "File Name" you can nagivate to the bitstream file you downloaded from the webpage. Once back to the main window, press the "Program Device" button. The process may take around 1 minute.
6. Once the programming process is done, you can close Radiant Programmer and Radiant.
7. If on a Linux system, you need to reload the `ftdi_sio` module by issuing (root privileges may be required)

   ```
   modprobe ftdi_sio
   ```

3.2. Connecting with GRMON

Once the steps in Chapter 3 are done, you can then connect to the SoC with the GRMON hardware debugger. On Linux, you need to issue

```
grmon -uart /dev/ttyUSB1 -u
```

where `ttyUSB1` can be different but in any case it corresponds to the second `ttyUSB` entry of the connected devices (in this case being `ttyUSB0` and `ttyUSB1`).

On Windows, you need instead to use the second `COM` port related to the connected device. In case the COM ports are, for example, `COM5` and `COM6`, then you would need to issue

```
grmon -uart COM6 -u
```

For complete information about grmon and how to use it, please refer to Chapter 4 and to the GRMON3 User's Manual available at https://www.gaisler.com/doc/grmon3.pdf.

4. GRMON3 hardware debugger

4.1. Overview

GRMON3 is a debug monitor used to develop and debug GRLIB/LEON systems. The target system, including the processor and peripherals, is accessed on the AHB bus through a debug-link connected to the host computer. GRMON3 has GDB support which makes C/C++ level debugging possible by connecting GDB to the GRMON3's GDB socket. With GRMON3 one can for example:

- Inspect LEON and peripheral registers
- Upload applications to RAM with the **load** command.
- Program the FLASH with the **flash** command.
- Control execution flow by starting applications (**run**), continue execution (**cont**), single-stepping (**step**), inserting breakpoints/watchpoints (**bp**) etc.
- Inspect the current CPU state listing the back-trace, instruction trace and disassemble machine code.

The first step is to set up a debug link in order to connect to the board. The following section outlines which debug interfaces are available and how to use them on the Certus-NX Versa Development Board. After that, a basic first inspection of the board is exemplified.

GRMON3 is described on the homepage [https://www.gaisler.com/index.php/products/debug-tools] and in detail in [RD-4].

It is recommended to use version GRMON3 3 or later with leon-certus.

4.2. Debug-link alternatives

4.2.1. Connecting via the serial UART

Please see GRMON User's Manual for instructions how to connect GRMON3 to a board using a serial UART connection. The PC is connected using a standard RS232 cable (or via USB to serial converter) to the USB-MINI connector, J2 connector and then starting GRMON3 without debug-link option (default is UART) or by specifying which PC UART using the `-uart COMPORT_NAME` command line switch. For example:

```
grmon -uart /dev/ttyUSB0
```

4.3. First steps

The previous sections have described which debug-links are available and how to start using them with GRMON3. The subsections below assume that GRMON3, the host computer and the Certus-NX Versa board have been set up so that GRMON3 can connect to the board.

When connecting to the board for the first time it is recommended to get to know the system by inspecting the current configuration and hardware present using GRMON3. With the **info sys** command more details about the system is printed and with **info reg** the register contents of the I/O registers can be inspected. Below is a list of items of particular interest:

- AMBA system frequency is printed out at connect, if the frequency is wrong then it might be due to noise in auto detection (small error). See `-freq` flag in the GRMON User's Manual [RD-4].
- Memory location and size configuration is found from the **info sys** output.

4.4. Connecting to the board

The transcript below shows an example session with GRMON3. GRMON3 is started with the `-u` flag in order to redirect UART output to the GRMON3 terminal.

```
cg@hwlin0:~$ grmon -uart /dev/ttyUSB2 -u

        GRMON debug monitor v3.3.3-25-g4491e81 64-bit

        Copyright (C) 2023 Frontgrade Gaisler - All rights reserved.
        For latest updates, go to https://www.gaisler.com/
        Comments or bug-reports to support@gaisler.com

        Parsing -uart /dev/ttyUSB2
        Parsing -u
```

```
Commands missing help:
echotrace
package

using port /dev/ttyUSB2 @ 115200 baud

Device ID:          0x800
GRLIB build version: 4283
Detected frequency:  50.0 MHz

Component                            Vendor
LEON3 SPARC V8 Processor             Frontgrade Gaisler
AHB Debug UART                       Frontgrade Gaisler
LEON3 Debug Support Unit             Frontgrade Gaisler
AHB/APB Bridge                       Frontgrade Gaisler
SPI Memory Controller                Frontgrade Gaisler
Single-port AHB SRAM module          Frontgrade Gaisler
LEON3 Statistics Unit                Frontgrade Gaisler
Generic UART                         Frontgrade Gaisler
Multi-processor Interrupt Ctrl.      Frontgrade Gaisler
Modular Timer Unit                   Frontgrade Gaisler
General Purpose I/O port             Frontgrade Gaisler

Use command 'info sys' to print a detailed report of attached cores

grmon3>
```

5. Bare C Cross-Compiler System

5.1. Overview

The Bare C Cross-Compiler (BCC for short) is a GNU-based cross-compilation system for LEON processors. It allows cross-compilation of C and C++ applications for LEON2, LEON3 and LEON4. This section gives the reader a brief introduction on how to use BCC together with the Certus-NX Versa Development Board. It will be demonstrated how to build an an example program and run it on the Certus-NX Versa using GRMON3.

The BCC toolchain includes the GNU C/C++ cross-compiler 7.2.0, GNU Binutils, Newlib embedded C library, the Bare-C run-time system with LEON support and the GNU debugger (GDB). The toolchain can be downloaded from [RD-10] and is available for both Linux and Windows. Further information about BCC can be found in [RD-11].

The installation process of BCC is described in [RD-11]. The rest of this chapter assumes that **sparc-gaisler-elf-gcc** is available in the PATH variable.

5.2. Compiling with BCC

The following command shows an example of how to compile a typical *hello, world* program with BCC.

```
$ cat hello.c
#include <stdio.h>

int main(void)
{
        printf("hello, world\n");
        return 0;
}

$ sparc-gaisler-elf-gcc -msoft-float -mcpu=leon3v7 -O2 -g hello.c -o hello.elf
```

All GCC options are described in the gcc manual. Some of the most common options are:

Table 5.1. BCC's GCC compiler relevant options

`-g`	generate debugging information - recommended for debugging with GDB
`-msoft-float`	emulate floating-point - must be used if no FPU exists in the system
`-O2`	optimize for speed
`-Os`	optimize for size
`-Og`	optimize for debugging experience
`-qsvt`	use the single-vector trap model
`-mflat`	enable flat register window model. The compiler will not emit SAVE and RESTORE instructions.

It is recommended to use the options

```
-msoft-float -mcpu=leon3v7
```

with leon-certus. For more details, see [RD-10].

5.3. Running and debugging with GRMON3

Once your application is compiled, connect to your Certus-NX Versa with GRMON3. The following log shows how to load and run an application. Note that the console output is redirected to GRMON3 by the use of the `-u` command line switch, so that the application standard output is forwarded to the GRMON3 console.

To debug the compiled program you can insert breakpoints, step and continue execution directly from the GRMON3 console. Compilation symbols are loaded automatically by GRMON3 once you load the application. An example is provided below.

```
grmon3> load hello.elf
  40000000 .text                    23.6kB /  23.6kB   [===============>] 100%
  40005E70 .data                     2.7kB /   2.7kB   [===============>] 100%
  Total size: 26.29kB (806.59kbit/s)
```

```
  Entry point 0x40000000
  Image hello.elf loaded

grmon3> bp main
  Software breakpoint 1 at <main>

grmon3> run

  CPU 0:  breakpoint 1 hit
          0x40001928: b0102000  mov  0, %i0  <main+4>
  CPU 1:  Power down mode

grmon3> step
  0x40001928: b0102000  mov  0, %i0  <main+4>

grmon3> step
  0x4000192c: 11100017  sethi  %hi(0x40005C00), %o0  <main+8>

grmon3> cont
hello, world

  CPU 0:  Program exited normally.
```

Alternatively you can run GRMON3 with the `-gdb` command line option and then attach a GDB session to it. For further information see Chapter 3 of [RD-11].

6. Support

For support contact the Frontgrade Gaisler support team at support@gaisler.com.

When contacting support, please identify yourself in full, including company affiliation and site name and address. Please identify exactly what product that is used, specifying if it is an IP core (with full name of the library distribution archive file), component, software version, compiler version, operating system version, debug tool version, simulator tool version, board version, etc.

There is also an open forum available at https://grlib.community.

Frontgrade Gaisler AB
Kungsgatan 12
411 19 Göteborg
Sweden
frontgrade.com/gaisler
sales@gaisler.com
T: +46 31 7758650
F: +46 31 421407