LEON3-XCKU-NANDFCTRL2

FRONTGRADE Gaisler

USER MANUAL

RELEASED SEPTEMBER 2023

LEON3-XCKU-NANDFCTRL2
Quick Start Guide

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 LEON3-XCKU-NANDFCTRL2 example bitstream.

The purpose of this document is to get users quickly started with the bitstream on a Xilinx KCU105 board together with a UT81NDQ512G8T NAND FLASH FMC mezzanine board.

This quick start guide does not contain many technical details and is instead how-to oriented. However, to make the most of the guide the user should have glanced through the LEON3-XCKU-NANDFCTRL2-EX User Manual and should ideally also be familiar with the GRMON debug monitor.

1.2. Availability

To get access to the evaluation bitstreams, contact sales@gaisler.com

More information about NANDFCTRL2 can be found on the products web page https://www.gaisler.com/NANDFCTRL2.

1.3. Prerequisites

To use the provided bitstream, the user needs:

- Xilinx KCU105 board.
- Frontgrade UT81NDQ512G8T NAND FLASH FMC mezz board: `PN UT81NDQ512G8T-KU060-EVB`. Contact your local Frontgrade representative for inquiries.
- Xilinx Vivado Design Suite (to program the FPGA). Vivado is available at https://www.xilinx.com/products/design-tools/vivado.html.
- GRMON3, available at https://www.gaisler.com/grmon.
- Workstation with GNU/Linux or Microsoft Windows.
- Example applications, provided together with the bitstream located in the systest folder.
- Bare-C Cross Compilation system (BCC) is required to update the available example applications, available at https://gaisler.com/index.php/downloads/compilers.

1.4. References

Table 1.1. References

RD-1	The SPARC Architecture Manual, Version 8, Revision SAV080SI9308
RD-2	GRMON User's Manual [https://www.gaisler.com/doc/grmon3.pdf]
RD-3	Bare C Cross-Compilation System [https://www.gaisler.com/index.php/products/operating-systems/bcc]
RD-4	BCC User's Manual [https://www.gaisler.com/doc/bcc2.pdf]
RD-5	GRLIB User's Manual [https://gaisler.com/products/grlib/grlib.pdf]
RD-6	GRLIB IP Cores Manual [https://gaisler.com/products/grlib/grip.pdf]
RD-7	LEON3-XCKU-NANDFCTRL2-EX User Manual; [https://www.gaisler.com/NANDFCTRL2]
RD-8	KCU105 Board User Guide [https://www.xilinx.com/support/documentation/boards_and_kits/kcu105/ug917-kcu105-eval-bd.pdf]
RD-9	RTEMS homepage [https://www.rtems.org]
RD-10	RTEMS User Manual [https://docs.rtems.org/branches/master/user/index.html]
RD-11	LEON/ERC32 RTEMS Cross Compilation System (RCC) [https://www.gaisler.com/index.php/products/operating-systems/rtems]
RD-12	RCC User's manual [https://gaisler.com/anonftp/rcc/doc]
RD-13	Frontgrade Gaisler RTEMS driver documentation [https://gaisler.com/anonftp/rcc/doc]

2. Overview

2.1. Board

The LEON3-XCKU-NANDFCTRL2 example bitstream should be used with the following boards:

- Xilinx KCU105
- UT81NDQ512G8T NAND FLASH FMC mezzanine board.

2.2. The design

The SoC system is described in [RD-7], available at https://www.gaisler.com/NANDFCTRL2. For details about the the interfaces' connections in the board see (Chapter 3).

2.3. GRMON Hardware Debugger

Non-intrusive debugging of the design and application execution can be performed using the GRMON hardware debugger. Please see (Chapter 5) for further information regarding GRMON and the available debug links.

2.4. Example Applications

The bitstream comes together with a set of example applications available both in .c and .exe format. Those are located in the systest folder. The recommended method to load software onto LEON3-XCKU-NANDFCTRL2-EX is by connecting to a debug interface of the board through the GRMON hardware debugger (Chapter 5). The applications can be updated and rebuilt using BCC. Please see (Section 6.1) for further information regarding BCC.

There are several basic applications that perform a basic setup, reset of NANDFCTRL2 and the FLASH memory device, followed by a read id command. These basic applications include one unique program per flash device ce_n signal, i.e. 4 programs. There is also one basic program using linked list mode.

The systest.exe application performs all ONFI mandatory commands as well as read and write between ram memory and the nand flash device. This includes fill ram memory with data, setup and reset NANDFCTRL2 and the FLASH memory device, read id, read parameter page, block erase, read status, page program including change write column, and page read including change read column.

The log below show the print of the systest_basic.exe application loaded and executed with GRMON.

```
grmon3> load systest_basic.exe
  40000000 .text             65.0kB /  65.0kB   [===============>] 100%
  400103d0 .rodata            2.3kB /   2.3kB   [===============>] 100%
  40010cf0 .data              1.6kB /   1.6kB   [===============>] 100%
Total size: 68.80kB (28.18Mbit/s)
  Entry point 0x40000000
  Image systest/systest_basic.exe loaded

grmon3> run
INF : main enter
INF : Setting up IRQ handler for IRQ 4
INF : Setting up unmask IRQ 4 - res 0
INF : OK IRQ unmask for IRQ 4
INF : IRQ handler OK.
-----------------------------------
NAND flash controller 2 - test start
-----------------------------------
INF : Reset nandfctrl2 registers using soft reset
INF : Assure DA = 0
     core status 0 = 0
INF : Setup SDR Timing Mode 0
 1) Issue a reset via command to one target
     core status 1 = 3

    Disable WP
     core status 0 = 0
 2) Command Read ID 0x90 0x20
     datain_0_reg = 4f4e4649
-----------------------------------
NAND flash controller 2 - test end
-----------------------------------
  Program exited normally

grmon3>
```

3. Board Configuration

This chapter describes board items as used by the LEON3-XCKU-NANDFCTRL2-EX design.

3.1. Buttons and switches

- CPU_RST button: Main reset to the FPGA design.
- SW12[1:4]: 4-Pole DIP Switch GPIO0 connected to inputs 0-1-2-3.
- SW7, SW9, SW10, SW8 buttons: connected to GPIO0 inputs 4-5-6-7.
- SW6 button: Connected to DSUBREAK signal. Push to break software execution.

The Switch SW12.1 is also connected to DSU_SEL and acts as select signal for the UART interface.

- When set to "ON" the UART interface is connected to the UART debug link (AHBUART).
- When set to "OFF" the UART interface is connected to the APBUART.

3.2. Connectors

- J87: USB JTAG interface via Digilent module with micro-B USB connector. Connector USB-JTAG. See (Chapter 5).
- J4:USB UART interface. Connector USB-UART. AHBUART debug link or APBUART function selectable by SW12.1.
- Ethernet PHY SGMII interface with RJ-45 connector. See (Chapter 5).
- J52: PMOD Connector J52: GPIO1 inputs 0-7.
- J53: PMOD Connector J53 pin 1 and 3: GPIO1 inputs 8-9.

3.3. LEDs

- LED[0]: Connected to DSU_Active.
- LED[1]: When ON indicates that the CPU is in error mode.
- LED[5]: Connected to DSU_SEL signal.
- LED[6..7]: When ON they indicate that the memory controller calibration is complete and the FPGA design has access to the on-board SDRAM.

3.4. UT81NDQ512G8T NAND FLASH FMC mezzanine board

- LED: Indicates that the board is powered.
- Switch: The Switch on mezzanine is used to set the address to the eeproms and should be set to pin 1.

3.5. Memories

The board is equipped with 2 GB DDR4 component memory (four [256 Mb x 16] devices).

3.6. Programming the bitstream

A vivado script to program the FPGA is provided in the bitstream folder. It has been tested using Vivado 2018.1. The .mcs file should be used to program the configuration memory. Follow the instructions below to program the FPGA:

1. Assure the NAND FLASH mezzanine board is unconnected during programming.
2. Connect the PC and the board using a standard micro-USB cable into the connector USB-JTAG J87.
3. Make sure that Vivado is added to your path variables.
4. Open a terminal in the downloaded folder and issue the following command to launch Vivado:

```
vivado -mode tcl -notrace -source doprog.tcl
```

5. To program the configuration memory, run in the Vivado console:

```
doprog
```

The expected output should be similar as below:

```
vivado -mode tcl -notrace -source doprog.tcl

****** Vivado v2018.1 (64-bit)
  **** SW Build 2188600 on Wed Apr  4 18:39:19 MDT 2018
  **** IP Build 2185939 on Wed Apr  4 20:55:05 MDT 2018
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

source doprog.tcl -notrace
Vivado% doprog
-----------------------------------------------------
Programming the FPGA configuration memory
-----------------------------------------------------
INFO: [Labtools 27-2285] Connecting to hw_server url TCP:localhost:3121
INFO: [Labtools 27-2222] Launching hw_server...
INFO: [Labtools 27-2221] Launch Output:

****** Xilinx hw_server v2018.1
  **** Build date : Apr  4 2018-18:56:09
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.


INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210308A7A4F5
INFO: [Labtools 27-2302] Device xcku040 (JTAG device index = 0) is programmed with a design
that has 1 SPI core(s).
INFO: [Labtools 27-3164] End of startup status: HIGH
Mfg ID : 20   Memory Type : bb   Memory Capacity : 19   Device ID 1 : 0   Device ID 2 : 0
Performing Erase Operation...
Erase Operation successful.
Performing Program and Verify Operations...
Program/Verify Operation successful.
INFO: [Labtoolstcl 44-377] Flash programming completed successfully
program_hw_cfgmem: Time (s): cpu = 00:00:00.85 ; elapsed = 00:05:57 . Memory (MB):
peak = 1827.641 ; gain = 32.000 ; free physical = 11683 ; free virtual = 23616
INFO: [Labtools 27-2302] Device xcku040 (JTAG device index = 0) is programmed with a design
that has 1 SPI core(s).
****** Webtalk v2018.1 (64-bit)
  **** SW Build 2188600 on Wed Apr  4 18:39:19 MDT 2018
  **** IP Build 2185939 on Wed Apr  4 20:55:05 MDT 2018
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: [Common 17-206] Exiting Webtalk at Wed Sep  6 10:32:24 2023...
Vivado%
```

6. Turn off the KCU-105 board.
7. Connect the FMC mezzanine to J22. Some of the early revisions of the NAND FLASH mezzanine board have some "overhang" over some of the jumpers on the KCU105 board. If using one of those boards assure the board is mounted properly and there is no risk of shorts, i.e. add some isolation between the mezzanine and the jumpers.
8. Turn on the KCU-105. Wait a few seconds an press the clk_sys (SW-5) button.

Alternatively, the bitstream file (.mcs) can be programmed to the Xilinx KCU105 using the Vivado design suite in GUI mode. Start vivado and Open the Hardware Manager, open the target, add configuration memory device mt25qu256-spi-x1_x2_x4 and program the configuration memory using the bitstream file (the *.mcs file). Close the hardware manager.

4. Software Development Environment

4.1. Overview

Frontgrade Gaisler provides a comprehensive set of software tools to run several different operating systems.

For the latest status of NANDFCTRL2 drivers for the different operating systems, contact support@gaisler.com.

LEON3 supports the following operating systems:

BCC	the Bare C Cross-Compiler System is a toolchain to compile bare C or C++ applications directly on top of the processor without the servises provided by an operating system.
RTEMS	a hard Real Time Operating System. Frontgrade Gaisler provides, for LEON3, a preliminary toolchain and kernel to develop and compile RTEMS applications.
Linux	the open source operating system. Board Support Packages and tools to ease the compilation and deployment of the kernel are provided.
VxWorks	an embedded real-time operating system developed by WindRiver. Frontgrade Gaisler provides a LEON architectural port (HAL) and a Board Support Package (BSP) in full source code.

Frontgrade Gaisler also provides debug tools. The LEON3-XCKU-NANDFCTRL2-EX is supported by the following:

GRMON	Used to run and debug applications on LEON3-XCKU-NANDFCTRL2-EX hardware. See (Chapter 5).

The recommended method to load software onto LEON3-XCKU-NANDFCTRL2-EX is by connecting to a debug interface of the board through the GRMON hardware debugger (Chapter 5).

5. GRMON hardware debugger

5.1. Overview

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

- Inspect LEON3 and peripheral registers
- Upload applications to RAM with the **load** 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 LEON3-XCKU-NANDFCTRL2 example bitstream. After that, a basic first inspection of the board is exemplified.

GRMON is described on the homepage [https://www.gaisler.com/index.php/products/debug-tools] and in detail in [RD-2].

5.2. Debug-link alternatives

5.2.1. Connecting via the Ethernet debug interfaces

If another address is wanted for the Ethernet debug link then one of the other debug links must be used to connect GRMON to the board. The EDCL IP address can then be changed using GRMON's **edcl** command. This new address will persist until next system reset.

With the Ethernet Debug Communication Link 0 address set to 192.168.0.187 the GRMON command to connect to the board is:

```
grmon -eth 192.168.0.187
```

5.2.2. Connecting via the UART debug link

Make sure that the switch `SW12.1` select the UART debug link (ON position). Connect the PC and the board using a standard micro-USB cable into the connector USB-UART J4 and issue the following command:

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

5.3. First steps

The previous sections have described which debug-links are available and how to start using them with GRMON. The subsections below assume that GRMON, the host computer and the LEON3-XCKU-NANDFCTRL2-EX board have been set up so that GRMON 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 GRMON. 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-2].
- Memory location and size configuration is found from the **info sys** output.

5.4. Connecting to the board

In the following example the Ethernet debug-link is used to connect to the board. The auto-detected frequency, memory parameters and stack pointer are verified by looking at the GRMON terminal output below.

```
grmon -u -eth 192.168.0.187
```

```
  GRMON debug monitor v3.3.4.1-35-g5ecb169 64-bit internal version

  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

  This internal version will expire on 05/09/2024

Parsing -u
Parsing -eth 192.168.0.187

     Commands missing help:
      echotrace
      package

 Ethernet startup...
WARNING! NANDF: PHY timing is based on APB clock (100000000 Hz) and not PHY clock
  Device ID:           0xA705
  GRLIB build version: 4283
  Detected frequency:  100.0 MHz

  Component                            Vendor
  LEON3 SPARC V8 Processor             Frontgrade Gaisler
  JTAG Debug Link                      Frontgrade Gaisler
  AHB Debug UART                       Frontgrade Gaisler
  GR Ethernet MAC                      Frontgrade Gaisler
  NAND Flash Controller 2              Frontgrade Gaisler
  AHB/APB Bridge                       Frontgrade Gaisler
  LEON3 Debug Support Unit             Frontgrade Gaisler
  Xilinx MIG Controller                Frontgrade Gaisler
  Single-port AHB SRAM module          Frontgrade Gaisler
  Generic AHB ROM                      Frontgrade Gaisler
  Generic UART                         Frontgrade Gaisler
  Multi-processor Interrupt Ctrl.      Frontgrade Gaisler
  Modular Timer Unit                   Frontgrade Gaisler
  LEON3 Statistics Unit                Frontgrade Gaisler
  On chip Logic Analyzer               Frontgrade Gaisler
  General Purpose I/O port             Frontgrade Gaisler
  General Purpose I/O port             Frontgrade Gaisler
  XILINX SGMII Interface               Frontgrade Gaisler

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

grmon3>
```

5.5. Get system information

One can limit the output to certain cores by specifying the core(s) name(s) to the **info sys** and **info reg** commands. As seen below the memory parameters, first UART and first Timer core information is listed.

```
grmon3> info reg uart0
  Generic UART
      0x80000104  UART Status register                    0x00000086
      0x80000108  UART Control register                   0x80000003
      0x8000010c  UART Scaler register                    0x00000145
grmon3> info sys gptimer0
  gptimer0  Frontgrade Gaisler  Modular Timer Unit
            APB: 80000300 - 80000400
            IRQ: 8
            16-bit scalar, 2 * 32-bit timers, divisor 100
```

5.6. Load a RAM application

An application linked to RAM can be loaded directly with the **load** and run with **run**.

```
grmon3> load systest.exe
  40000000 .text             70.2kB /  70.2kB   [===============>] 100%
  400118d0 .rodata            2.9kB /   2.9kB   [===============>] 100%
  40012490 .data              1.6kB /   1.6kB   [===============>] 100%
Total size: 74.70kB (26.61Mbit/s)
  Entry point 0x40000000
  systest.exe loaded

grmon3> run
INF : main enter
...
```

6. Toolchains

6.1. Bare C Cross-Compiler System

6.1.1. 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 LEON3-XCKU-NANDFCTRL2 example bitstream. It will be demonstrated how to build an an example program and run it on the LEON3-XCKU-NANDFCTRL2-EX using GRMON.

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-3] and is available for both Linux and Windows. Further information about BCC can be found in [RD-4].

The installation process of BCC is described in [RD-4]. The rest of this chapter assumes that **sparc-gaisler-elf-gcc** is available in the PATH variable.

6.1.2. 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 -qbsp=gr712rc -mcpu=leon3 -mfix-gr712rc -O2 -g hello.c -o hello.elf
```

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

Table 6.1. 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

```
-qbsp=gr712rc -mcpu=leon3 -mfix-gr712rc
```

with LEON5. For more details, see [RD-3].

6.1.3. Running and debugging with GRMON

Once your application is compiled, connect to your LEON3-XCKU-NANDFCTRL2-EX with GRMON. The following log shows how to load and run an application. Note that the console output is redirected to GRMON by the use of the `-u` command line switch, so that the application standard output is forwarded to the GRMON console.

To debug the compiled program you can insert breakpoints, step and continue execution directly from the GRMON console. Compilation symbols are loaded automatically by GRMON 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 GRMON with the `-gdb` command line option and then attach a GDB session to it. For further information see Chapter 3 of [RD-4].

7. 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