GRAND_V2

Board bring up and test procedure and building binaries from GIT repository

Prerequisites

1. Download Vivado Lab Solutions v. 2018.3 from:
   https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html
2. Download Analog Devices IIO scope (runs on Linux and macOS) from:
   https://wiki.analog.com/resources/tools-software/linux-software/iio_oscilloscope
3. GPS
   - Install Trimble's Visual Timing Studio (VTS) download from:
     https://timing.trimble.com/products/gnss-receivers/icm-smt-360-multi-gnss-timing-module/
     Or download TrimbleVTS.exe from: https://ftp.science.ru.nl/grand_v2/hardware/GPS/
   - Prepare a GPS antenna and TNC to SMB connector or cable
4. Power cable
   Make a cable to use with a power supply (12V, 3A) Use Molex Part Number: 39012060.
   Available at Mouser: 538-39-01-2060, Digi-Key: WM3702-ND
5. Power Chip configuration dongle and monitor
   - https://www.ti.com/tool/USB2ANY Part Number: USB2ANY Available at Mouser: 595-USB2ANY, Digi-Key: 296-49061-ND
   - Software: available at https://www.ti.com/tool/USB2ANY or https://ftp.science.ru.nl/grand_v2/hardware/Power
6. SiLabs program cable and software (runs only from Windows)
   - ClockBuilder Pro Field Programmer

   Available at Mouser: 634-CBPROG-DONLGE, Digi-Key: 336-3255-ND
7. UART to USB cable
   - https://www.ftdichip.com/
   - FTDI C232HD-EDHSP-0 Available at Mouser: 895-C232HD-EDHSP-0, Digi-Key: 768-1112-ND
   - Install your favorite serial terminal (picocom/minicom/putty).
8. USB to USB-micro (B)
   standard cable
9. Download the hardware folder with all the above software and tools from:
   https://ftp.science.ru.nl/grand_v2/hardware

10. Function generator
    The LNA in the antenna is supplied with 5V5 by the GRAND board. This is done by applying a DC offset on the analog input. To protect the (the output stage of the) function generator place a bias-t in the signal line or AC couple the line.
    - 14/16 bit
    - Test signal: 80MHz, 100mVpp

Introduction

To test GRAND v2 boards you need two PC's. One with Windows installed (to program the power chip, clocks and check the GPS) and one with Linux (Ubuntu18.04). After the initial programming of the clocks and checking the GPS functionality the Windows machine will not be used anymore. The Linux machine will used as the "host" to configure, control and check the GRAND v2 board, the "target". Because of the complexity of the board a lot of things must be configured before any signal can be measured. For instance the clock chip, the ADC, FPGA and the Linux image running on the CPU's embedded in the FPGA chip. The following steps guide you through this process. Copy and paste the terminal commands (written in "`courier`") directly in the terminal can result in very strange behavior and sometimes lock the UART completely. Copy/paste via a simple text program prevents this problem.

Build firmware from source

If you want to modify or develop firmware yourself you need to install Vivado 2018.3, PetaLinux and SDK locally or log in to the buildserver at the HEF department, 'keersop'. On this machine (running CentOS 7) all the necessary software is installed. If you use this document for board testing these chapters can be skipped and use the binaries created for testing available at the science.ru.nl ftp server: https://ftp.science.ru.nl/grand_v2/hardware. To start directly with testing the GRANDproto300 boards continue with chapter "Test procedure". To continue with the programming of the FPGA refer to: "Program QSPI / flash memory" for the exact steps.

Get source from GIT

1. Get the source files from the GIT repository
    a. `[rhabraken@keersop ~]$ mkdir GRAND`
    b. `[rhabraken@keersop ~]$ cd GRAND/`
    c. `[rhabraken@keersop GRAND]$ git init`
    d. `[rhabraken@keersop GRAND]$ git clone https://gitlab.science.ru.nl/rhabraken/grand_v1.git`
    e. `$ cd grand_v1/`
    grand_v1 is referenced as <gitdir> in the rest of this document.
2. Checkout the branch for the correct version of the board
    a. See board description above logo for the board version
    b. "GRAND PROTO300 V1.0" → `$ git checkout -track origin/flashboot`
    c. "GRAND PROTO300 V2" → `$ git checkout --track origin/GRANDv2`
3. Copy the board hardware files from the hardware folder in the repository to the Xilinx install directory.

```
<gitdir>$ sudo cp -rf hardware/BoardFiles/zu7cg/
/home/Xilinx/Vivado/2018.3/data/boards/board_files/
```

4. If needed, restart Vivado to refresh the board repository.

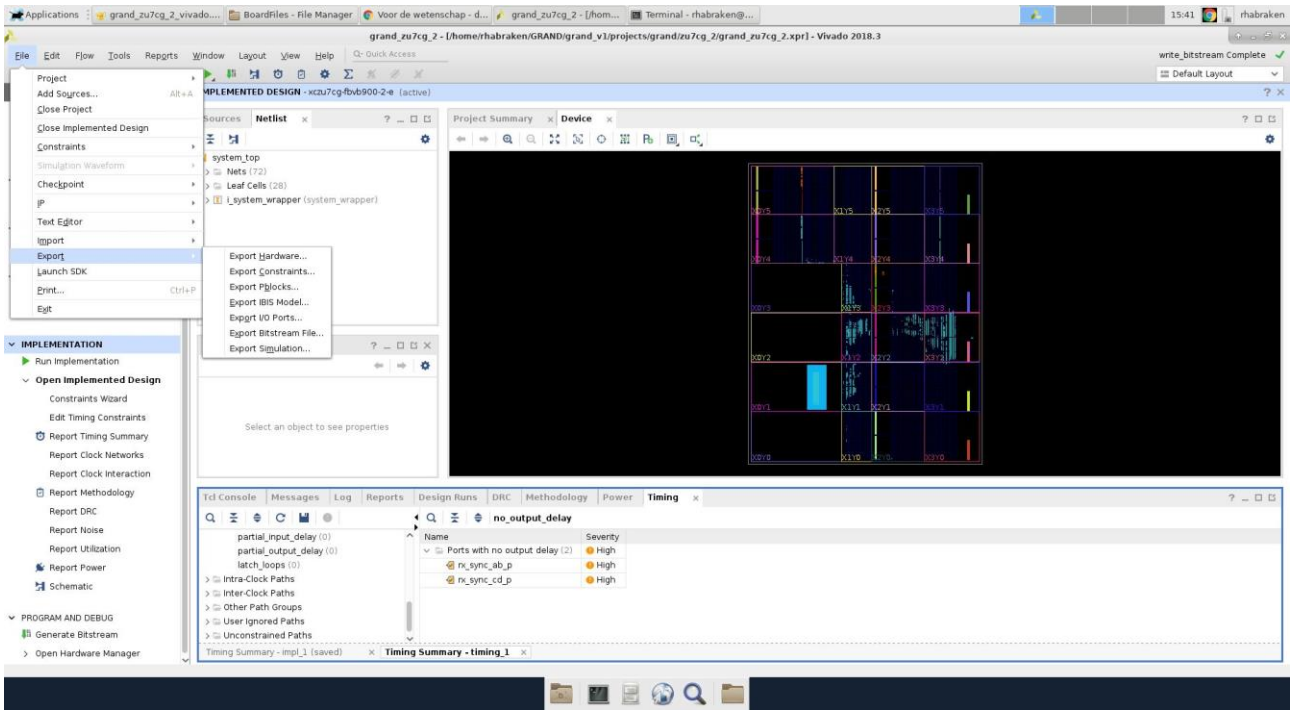
5. Create bitfile from the repository to modify or develop firmware
   GRAND V1 → `<gitdir>$ cd projects/grand/zu7cg/`
   GRAND V2 → `<gitdir>$ cd projects/grand/zu7cg_2`

Create Vivado project

1. Create a project by typing
   GRAND V1 → `<gitdir>/projects/grand/zu7cg/$ make`
   GRAND V2 → `<gitdir>/projects/grand/zu7cg_2/$ make`

Start Vivado and check if everything build correctly from source (does design meet timing, are the peripherals correctly defined). If so it is possible to add new functionality to the existing firmware. When done rebuild the firmware by clicking “Generate Bitstream” in the Vivado project.
After the creation of the bitfile the linux images must be rebuild if any modification is done on the registermap or any other interface with the CPU’s.

2. To continue with software and Linux developments do the following from within the Vivado project:
   a. Select **File**
   b. Select **Export**
   c. Export **Hardware**



3. Make sure to include the generated bitstream.

4. Select overwrite if a previous detected hardware is found.
5. Switch to a Linux terminal to continue with building the Linux images based on the newly created firmware.

Build Linux from source

Linux config

Create Linux config file with the correct hardware description file.

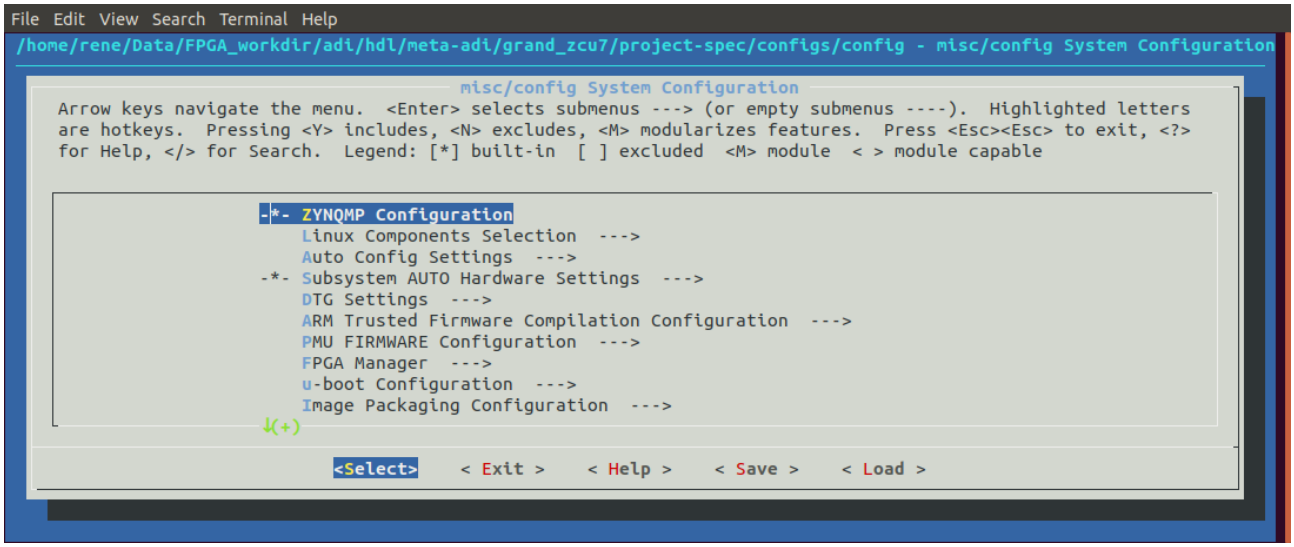
1. Go to the meta-adi folder where the Petalinux project will be created:
   `$ cd <gitdir>/meta-adi`

Because of the compressing strategy of git using the Board Support Package (BSP) file directly from the repository doesn't work. Therefor it must be downloaded manually.

2. Go to the gitlab page:
   https://gitlab.science.ru.nl/rhabraken/grand_v1/-/blob/GRANDv2/hardware/PetalinuxBSP/GRAND_V2.bsp
3. Click on **download** and note the download location.
4. Create a petalinux project with the downloaded BSP with the following command:
   `<gitdir>/meta-adi$ petalinux-create -t project -s <download folder>/GRAND_V2.bsp`
5. Import the exported hardware specification from Vivado
   GRAND_V1
   `petalinux-config –get-hw-description=/<path to GIT repo>/grand_v1/projects/grand/zu7cg/grand_zu7cg.sdk`

   GRAND_V2
   `petalinux-config –get-hw-description= <path to GIT repo>/grand_v1/projects/grand/zu7cg_2/grand_zu7cg_2.sdk`

6. If necessary make modification to the configuration Linux otherwise select **Exit** and press enter.



7. Check output in terminal:

```
grand_zcu7$ petalinux-config --get-hw-
description=/home/rene/Data/FPGA_workdir/adi/hdl/projects/grand/
zu7cg/grand_zu7cg.sdk
INFO: Getting hardware description...
INFO: Rename system_top.hdf to system.hdf
[INFO] generating Kconfig for project
[INFO] menuconfig project

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

[INFO] sourcing bitbake
[INFO] generating plnxtool conf
[INFO] generating meta-plnx-generated layer
[INFO] generating machine configuration
[INFO] generating bbappends for project . This may take time !
[INFO] generating u-boot configuration files
[INFO] generating kernel configuration files
[INFO] generating kconfig for Rootfs
[INFO] oldconfig rootfs
[INFO] generating petalinux-user-image.bb
```

Petalinux will create the configuration file for U-boot, the kernel and the root file system and sets up the bbappends Yocto layers.

Build Linux images

IMPORTANT!
After a clean checkout of the repository there is still one absolute path defined in https://gitlab.science.ru.nl/rhabraken/grand_v1/-/blob/GRANDv2/meta-adi/meta-adi-xilinx/recipes-bsp/device-tree/device-tree.bbappend#L54

1. Edit line L54 in the file below to match the correct path for your local repository: <path to GIT repo>/grand_v1/meta-adi/meta-adi-xilinx/recipes-bsp/device-tree/device-tree.bbappend

1. Start the creation of all the images by typing:
   ```
   grand_zcu7$ petalinux-build
   ```

Depending on the host machine (and internet connection) this process takes up quite some time. (On a Dell XPS 15, i7, 32GiB ram, SSD disc it takes 20 minutes).

2. Check output in terminal:
   ```
   grand_zcu7$ petalinux-build
   [INFO] building project
   [INFO] generating Kconfig for project
   [INFO] oldconfig project
   [INFO] sourcing bitbake
   [INFO] generating plnxtool conf
   [INFO] generating meta-plnx-generated layer
   [INFO] generating machine configuration
   [INFO] generating bbappends for project . This may take time !
   [INFO] generating u-boot configuration files
   [INFO] generating kernel configuration files
   [INFO] generating kconfig for Rootfs
   [INFO] oldconfig rootfs
   [INFO] generating petalinux-user-image.bb
   INFO: bitbake petalinux-user-image
   WARNING: Host distribution "ubuntu-18.04" has not been validated
   with this version of the build system; you may possibly
   experience unexpected failures. It is recommended that you use a
   tested distribution.
   Loading cache: 100%
   |#################################################################
   ####################| Time: 0:00:00
   Loaded 3465 entries from dependency cache.
   Parsing recipes: 100%
   |#################################################################
   ##################| Time: 0:00:04
   Parsing of 2578 .bb files complete (2532 cached, 46 parsed).
   3470 targets, 137 skipped, 0 masked, 0 errors.
   NOTE: Resolving any missing task queue dependencies
   ```

```
Initialising tasks: 100%
|#############################################################
################| Time: 0:00:06
Checking sstate mirror object availability: 100%
|#####################################################| Time:
0:00:17
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
WARNING: petalinux-user-image-1.0-r0 do_rootfs: [log_check]
petalinux-user-image: found 1 warning message in the logfile:
[log_check] warning: %post(sysvinit-inittab-2.88dsf-
r10.plnx_zynqmp) scriptlet failed, exit status 1

NOTE: Tasks Summary: Attempted 3399 tasks of which 2381 didn't
need to be rerun and all succeeded.

Summary: There were 2 WARNING messages shown.
INFO: Copying Images from deploy to images
INFO: Creating images/linux directory
NOTE: Successfully copied built images to tftp dir:  /tftpboot
[INFO] successfully built project
```

Build BOOT.bin

All the created files must be packaged into 1 binary the BOOT.BIN. This file will be programmed in the flash memory and with contain the fpga bitstream, firmware for the power management unit, first stage boot loader and the u-boot image.

1. Create the BOOT.BIN with the following command:
   ```
   grand_zcu7$ petalinux-package --boot --fsbl --fpga --u-boot –
   kernel --add images/linux/rootfs.jffs2 --offset 0x4240000 --
   force
   ```
2. Check output in terminal:
   ```
   INFO: Getting system flash information...
   INFO: File in BOOT BIN:
   "/home/rene/Data/FPGA_workdir/adi/hdl/meta-
   adi/grand_zcu7/images/linux/zynqmp_fsbl.elf"
   INFO: File in BOOT BIN:
   "/home/rene/Data/FPGA_workdir/adi/hdl/meta-
   adi/grand_zcu7/images/linux/pmufw.elf"
   INFO: File in BOOT BIN:
   "/home/rene/Data/FPGA_workdir/adi/hdl/meta-
   adi/grand_zcu7/project-spec/hw-description/system_top.bit"
   INFO: File in BOOT BIN:
   "/home/rene/Data/FPGA_workdir/adi/hdl/meta-
   adi/grand_zcu7/images/linux/bl31.elf"
   ```

```
INFO: File in BOOT BIN:
"/home/rene/Data/FPGA_workdir/adi/hdl/meta-
adi/grand_zcu7/images/linux/u-boot.elf"
INFO: File in BOOT BIN:
"/home/rene/Data/FPGA_workdir/adi/hdl/meta-
adi/grand_zcu7/images/linux/image.ub"
INFO: File in BOOT BIN:
"/home/rene/Data/FPGA_workdir/adi/hdl/meta-
adi/grand_zcu7/images/linux/rootfs.jffs2"
INFO: Generating ZynqMP binary package BOOT.BIN...


****** Xilinx Bootgen v2018.3
  **** Build date : Dec  6 2018-23:41:49
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: Binary is ready.
```

Test procedure

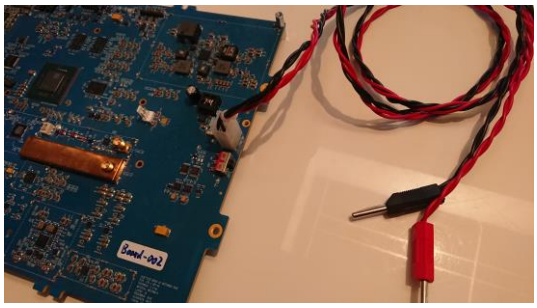
Preparation

The first part of the tests and setup steps must be performed on a Windows host because of support software from the different vendors of components. After the clocks, power supplies and GPS are configured and tested the rest of the board bring up continues on a Linux host.

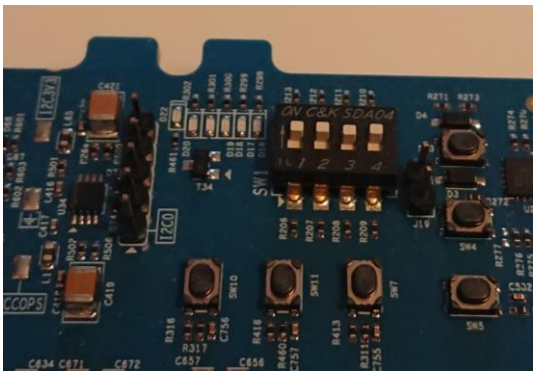
1. Visual board check
   a. Check silk screen
   b. Check component placement
   c. No jumpers should be placed
   d. Mounting of mechanical parts

2. Check if the problems that were seen with the first two version 2 boards are solved.
   a. Short in footprint of the FET for the fan control
   b. Replace U31 TPS6508641RSKT with TPS6508640RSKT
   c. Turn diode D28 180 degrees
   d. Check if capacitor C25 is mounted

3. Connect the power supply to the 12VDC IN connector (J6) on the board.



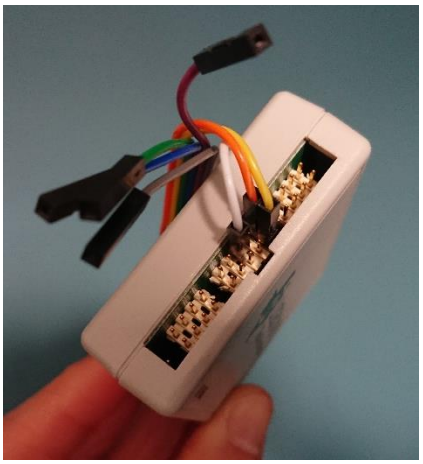
4. Set Boot switch (SW1) to JTAG mode
   a. All dip switches must be “up”, away from the numbers.



Part 1 on Windows host

Config power chip TPS650864

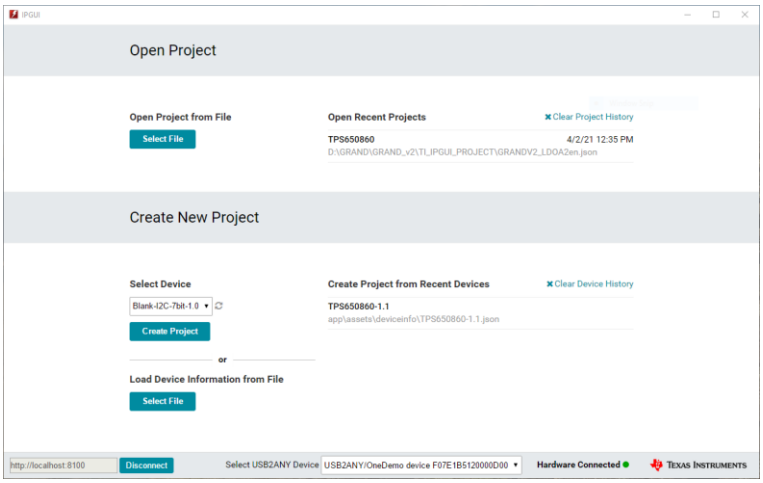
1. Connect the USB2ANY dongle to the I2C0 header (J24) in order to read out and configure the power chip TPS650864.



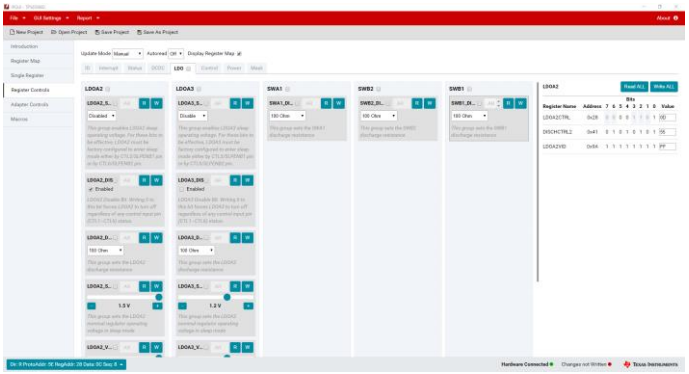
IMPORTANT!

The inrush current is very high so that limiting the power supply during the first power up doesn’t work. With a maximum limit on my power supply set to 2.78A did not result in a power up. The resettable fuse in the power supply trips and shuts down the power. Without the fuse set the board powers up and the current settles around 650mA, Without any activity going on in the readout and booting in JTAG mode (see above) the version 2 board consumes 7.8W: 12V, 650mA. (Version 1 board: 15.8W: 12V, 1.315A.)

2. Switch on the power supply
3. Set switch SW2 to “ON” position and monitor the power consumption. The current should stabilize within a second to approximately 650mA.
4. Start the IPG-UI program from Texas Instruments (download from https://ftp.science.ru.nl/grand_v2/hardware/Power) to readout the voltages generated on the TPS650864 power chip and check for errors.



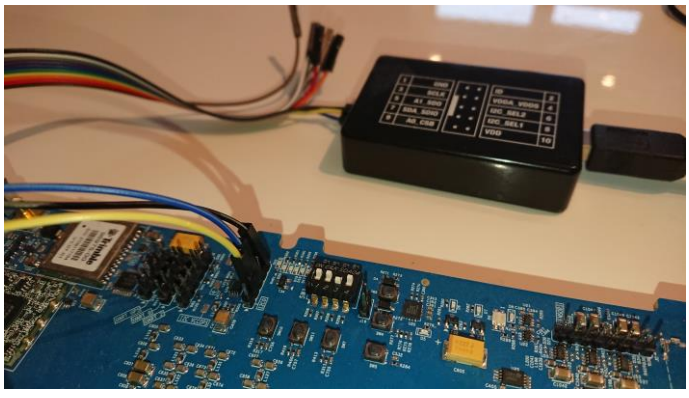
5. Click on **Open Project from File** and select ‘**GRANDV2_LDOA2en.json**’ available for download at https://ftp.science.ru.nl/grand_v2/hardware/Power.
6. Go to **Register Controls** and click on the **LDO** tab on the top of the page.



7. Check if **LDOA2_DIS** is enable or not.
8. Program the chip by pressing: RH??? (write all)
9. Check for warning or errors.

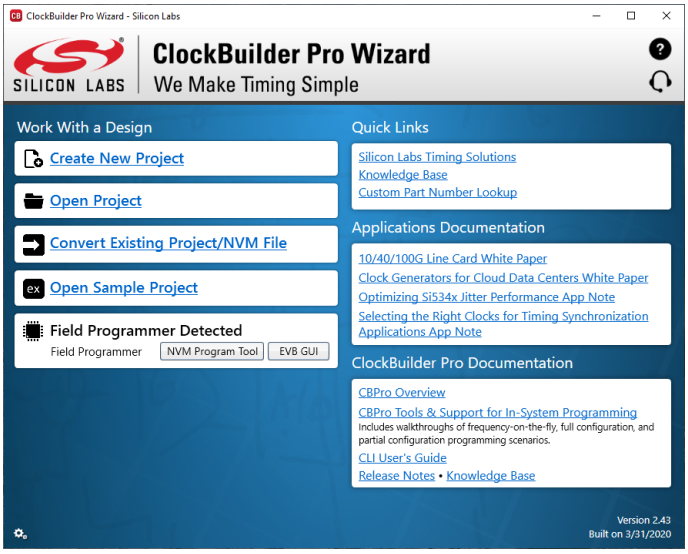
Program Clocks

1. Connect the cable of the Silicon Labs ClockBuilder Pro Field Programmer from the USB port on the Windows host to the I2C0 pin header (J24).

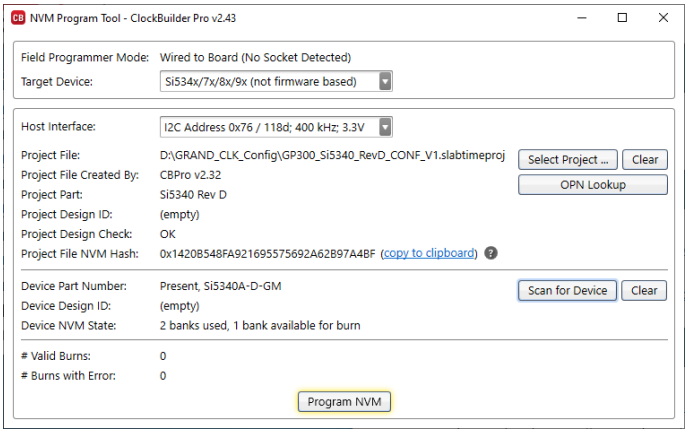


In the picture below: yellow (SCLK) to pin 2 (I2C0_SCL), blue (SDA_SDIO) to pin 3 (I2C0_SDA), black GND to pin 5,

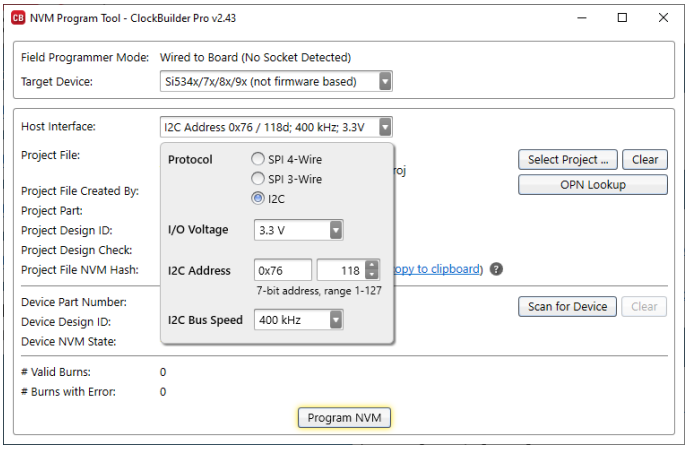
2. Download, extract and start Clock Builder Pro
   https://ftp.science.ru.nl/grand_v2/hardware/Clocks/ClockBuilder-Pro-Installer.zip



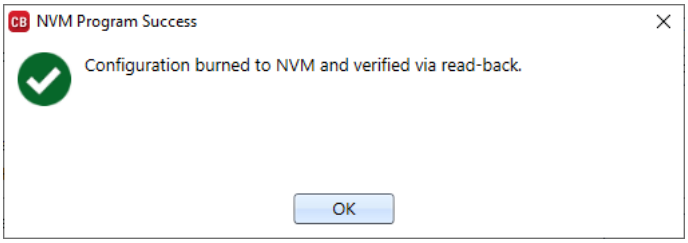
3. Click on **NVM Program Tool** to program the non-volatile memory of the Si5340 clock generator.
4. Select the project file '**GP300_Si5340_RevD_CONF_V1.slabtimeproj**' file available at:
   https://ftp.science.ru.nl/grand_v2/hardware/Clocks/GP300_Si5340_RevD_CONF_V1.slabtimeproj



5. Double check the I2C address and settings:



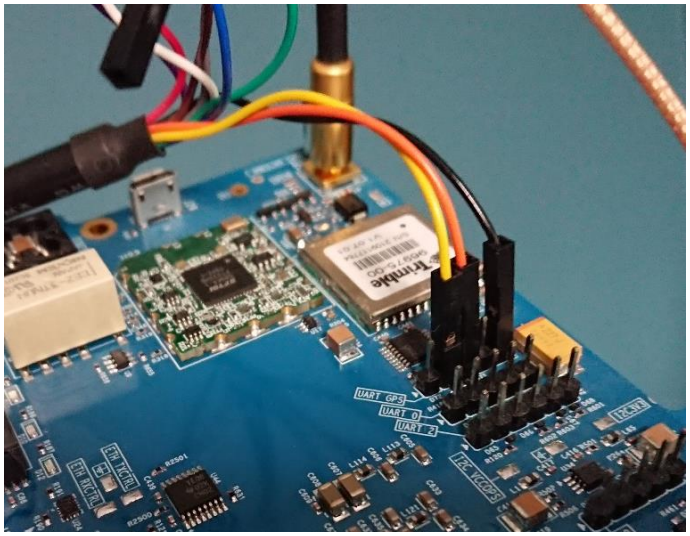
6. Click '**Program NVM**' and verify a successful writing to the chip



Test the GPS

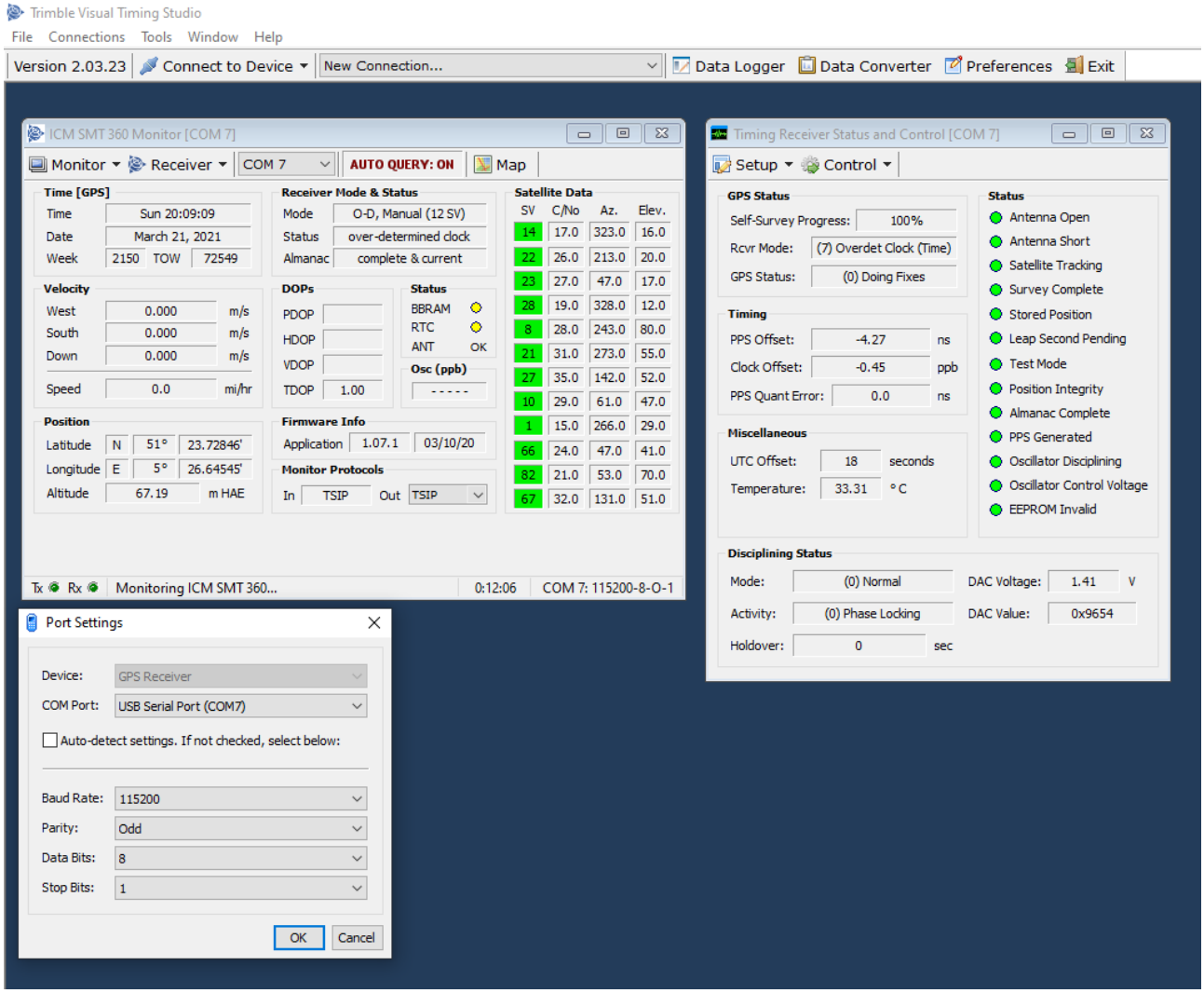
1. Prepare an adapter or cable to connect TNC cable (male) to SMB connector (male) on the board.
2. Connect the GPS to SMB connector J25 with the TNC cable

3. Connect the FTDI C232HD-EDHSP-0 from the host USB port to UART_GPS connector (J31) on the board.
   Yellow FTDI(RX) to pin 2 (UART0_TXD), orange FTDI (TX) to pin 3 (UART0_RXD) , black to pin 4 or 5 (GND).



4. Download TrimbleVTS.exe from https://ftp.science.ru.nl/grand_v2/hardware/GPS/.
5. Start the application to communicate with the GPS antenna and check the functionality.

Note: If the UART interface is connected correctly the TrimbleVTS starts automatically with the correct serial interface settings and opens the following windows.



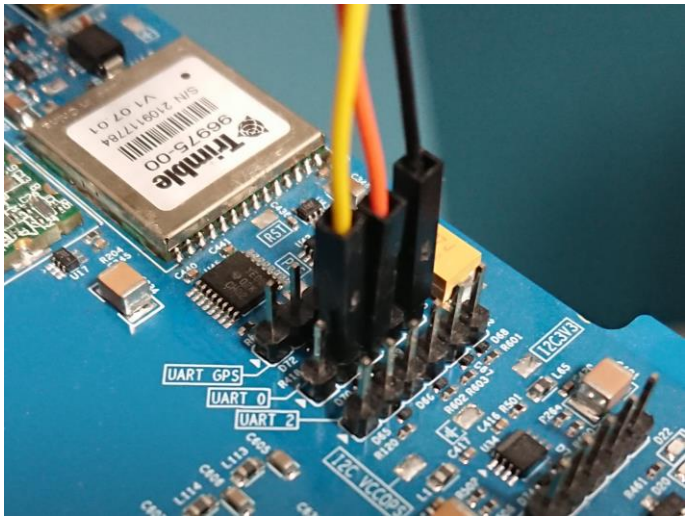
6. If everything is ‘green’ close the application and disconnect the FTDI cable from the UART_GPS.

These were all the tests and setup steps that have to be done on a Windows host machine. Continue on a Linux host machine.

Part 2 on Linux host

Preparation

1. Setup the serial connection with the board.
   a. Connect the FTDI C232HD-EDHSP-0 from the host USB port to UART0 connector (J30) on the board.
   Yellow FTDI(RX) to pin 2 (UART0_TXD), orange FTDI (TX) to pin 3 (UART0_RXD) , black to pin 4 or 5 (GND).



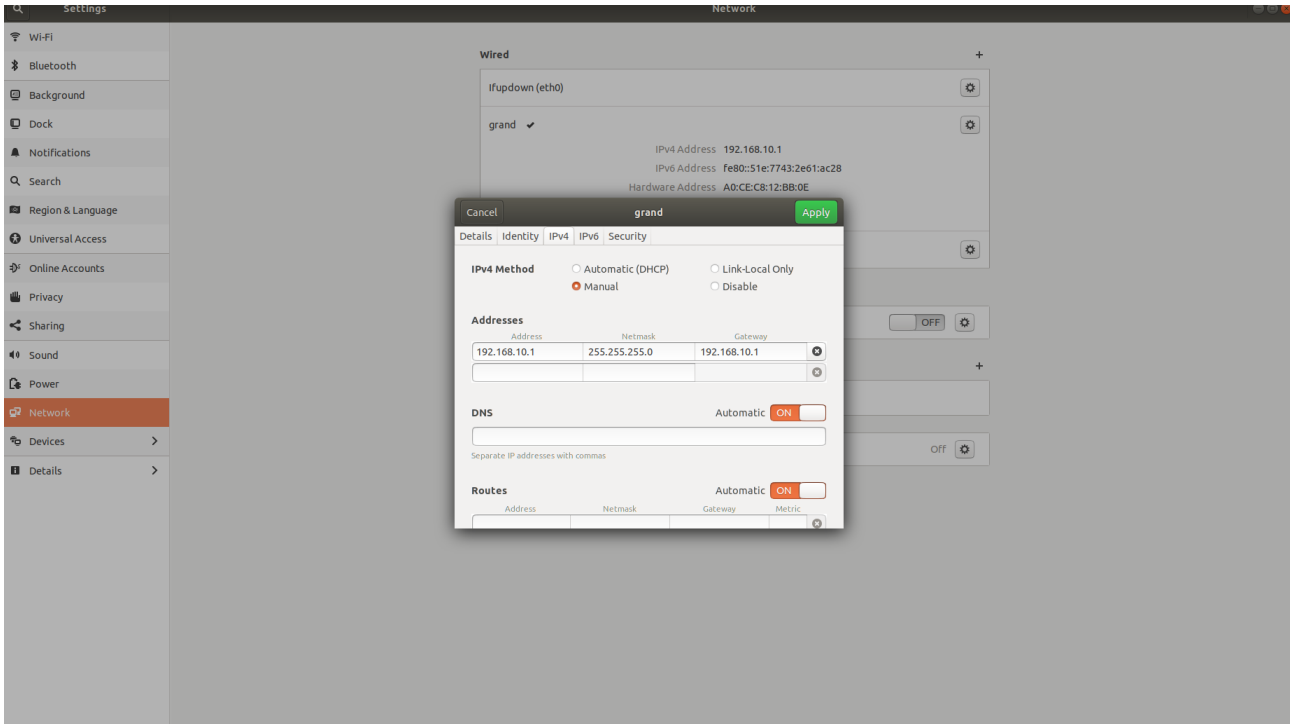
   b. Start serial connection with the board for:
   - picocom: `$ picocom -b  115200 /dev/ttyUSB0`
   - minicom: `$ sudo minicom -D /dev/ttyUSB0`

2. Connect JTAG
   a. Plug in USB to microUSB cable from PC to the connector J29 on the board.

Setup Ethernet connection

1. Plug in an ethernet cable from the PC to the ethernet port on the board.
2. Configure the network settings on the host PC.

Once correctly booted the GRAND board will be assigned a fixed IP-adress: 192.168.10.2. In a later stage of the project this will be changed but for testing and debugging a fixed IP-address has a lot of advantages.



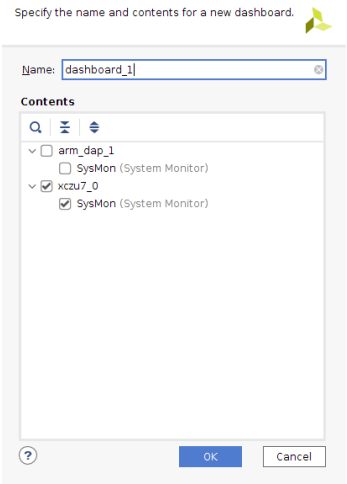
Program QSPI / flash memory

1. Start Vivado Lab Solutions and program QSPI / Flash memory
   a. Open a terminal and type:
   b. `source <vivadolabdir>/Vivado_Lab/2018.3/settings64.sh`
   c. Type the following command in the terminal to start Vivado Lab Edition:
   d. `vivado_lab`
2. Open the GUI by selecting '**Open Hardware Manger**',
3. In the green bar click on '**Open target**' and select '**auto connect**'.

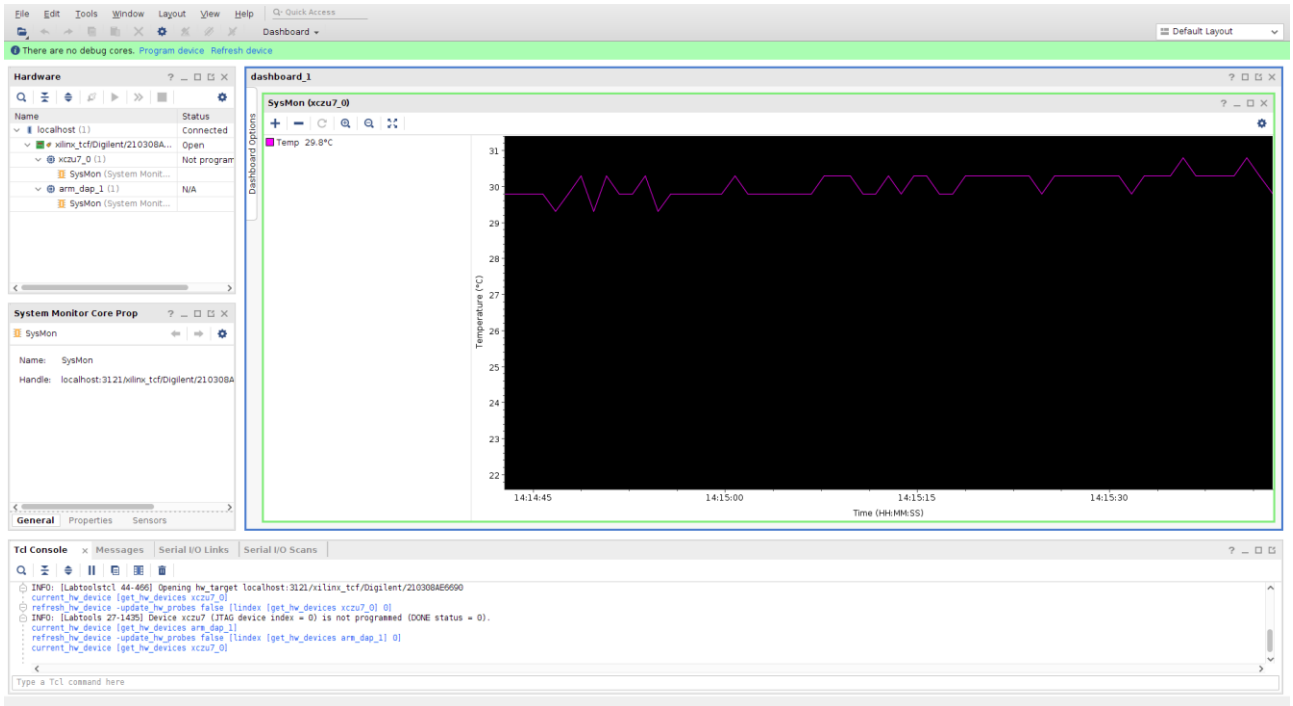
The FPGA should show up and by double clicking on **SysMon** a new Dashboard can be created to show the temperature of the FPGA. Leave just the **xczu7_0** selected and click **OK**.

Note: The FPGA should be recognized automatically by the hardware manager if this is not the case, install drivers.
<vivadolabdir>/2018.3/data/xicom/cable_drivers/lin64/install_script/install_drivers
In a terminal go to:

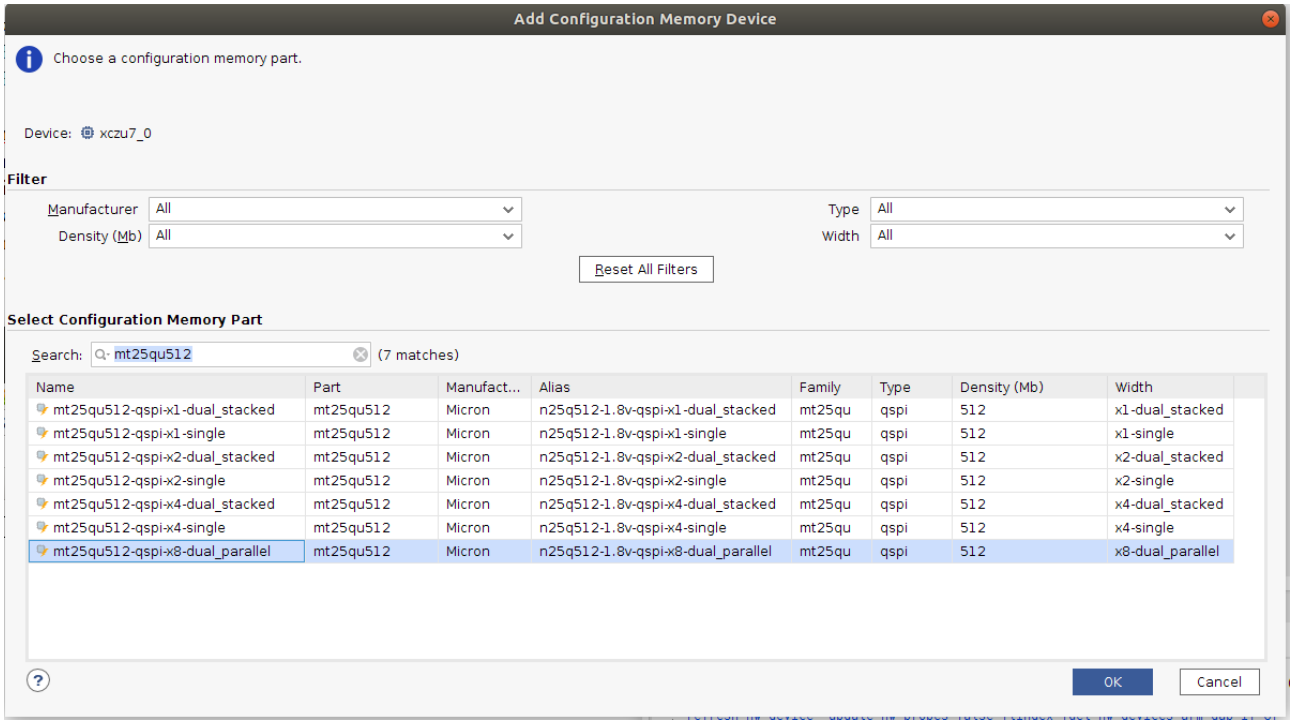
```
$ cd
<vivadolabdir>/2018.3/data/xicom/cable_drivers/lin64/install_script/install_drivers
$ sudo ./install_drivers
```



If the temperature start plotting and the correct FPGA is visible it means we can communicate with the FPGA and all the required power rails are configured correctly. If the SysMon does not show up try to connect again in (in the green bar) or restart the Vivado Lab Edition. If the xczu7_0 still does not show up in the hardware window something is wrong with the board and there is no point in continuing with the next steps.

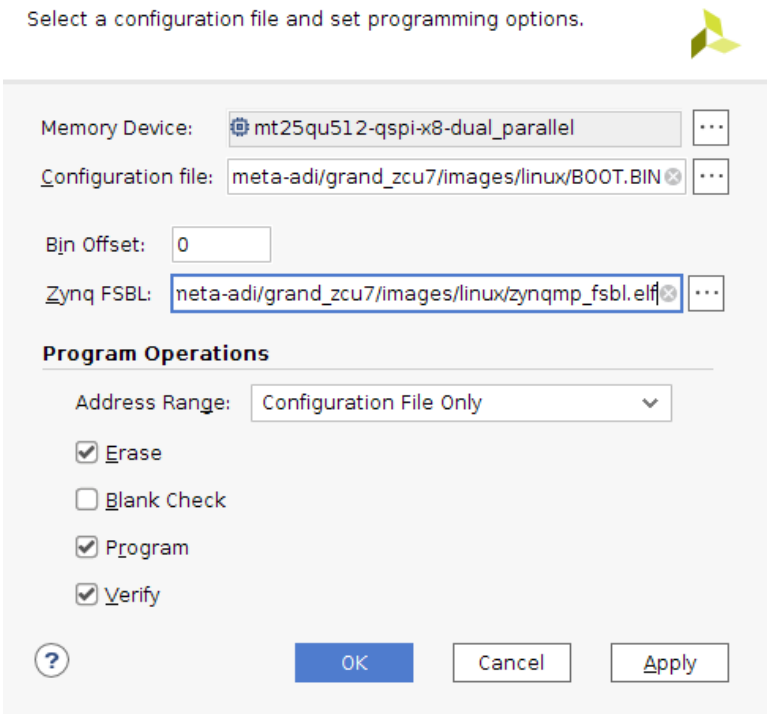


4. Choose the '**Tools**' menu bar tab, '**Add Configuration Memory device**' and select **xczu7_0**.
5. In the search bar type the qspi memory chip: '**mt25qu512**' and select the correct spi width **x8-dual_parallel**.



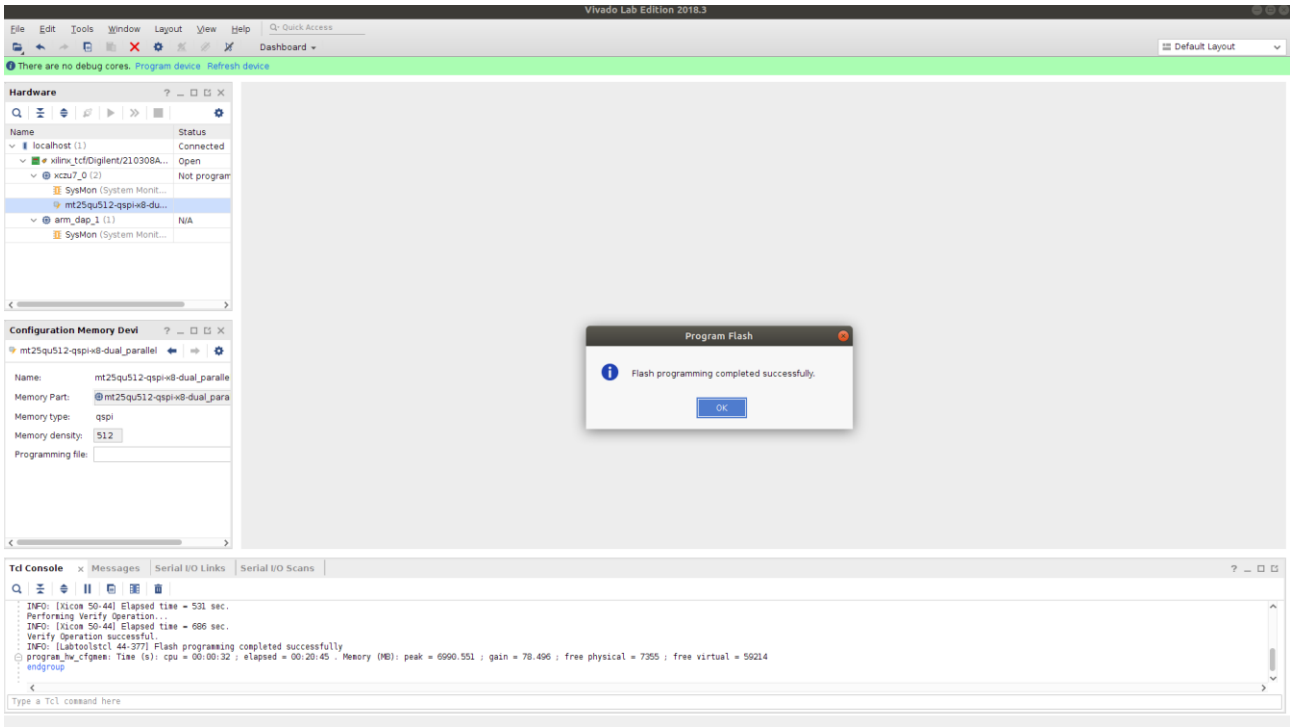
6. Then OK.

7. For the Configuration file select the correct **BOOT.BIN** file with the **...** button. This boot.bin is created earlier if the binaries are built from source. In that case the BOOT.BIN is in your Petalinux project folder under <gitdir>grand_zcu7/images/linux. The reference file for testing can be found on https://ftp.science.ru.nl/grand_v2/hardware/Images/2CHINA/linux/.
8. Select the FSBL **zynqmp_fsbl.elf** from the same folder.



9. Leave erase, program and verify checked and click **OK**.

10. Wait for the process to finish



The process of programming the flash takes up around 20 minutes to complete. You can check if the serial interface is working. It should show the first stage boot loader being executed preparing the system for programming the flash.
11. Check the UART for message:

```
Xilinx Zynq MP First Stage Boot Loader
Release 2018.3   Apr  1 2021  -  10:14:55
PMU-FW is not running, certain applications may not be supported.
```

If you started Vivado Lab Edition on the host PC from a Linux terminal it should show the programming process being executed.

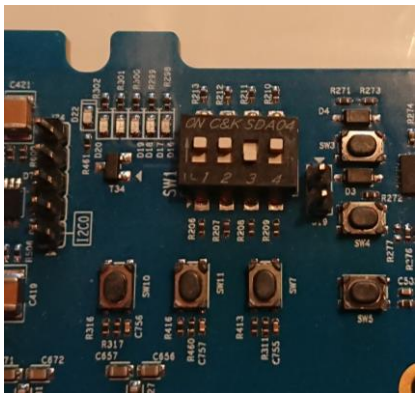
```
start_gui
===== mrd->addr=0xFF5E0204, data=0x00000000 =====
BOOT_MODE REG = 0x0000
Downloading FSBL...
Running FSBL...
Finished running FSBL.


U-Boot 2018.01-00073-g63efa8c-dirty (Oct 04 2018 - 08:27:12 -0600)

Model: ZynqMP MINI QSPI
Board: Xilinx ZynqMP
DRAM:  256 KiB
EL Level: EL3
Using default environment
```

```
In:    dcc
Out:   dcc
Err:   dcc
ZynqMP> sf probe 0 0 0
SF: Detected n25q512a with page size 512 Bytes, erase size 128
KiB, total 128 MiB
ZynqMP> Sector size = 131072.
sf erase 0 4C80000
SF: 80216064 bytes @ 0x0 Erased: OK
ZynqMP> sf write FFFC0000 0 20000
device 0 offset 0x0, size 0x20000
SF: 131072 bytes @ 0x0 Written: OK
ZynqMP> sf write FFFC0000 20000 20000
device 0 offset 0x20000, size 0x20000
SF: 131072 bytes @ 0x20000 Written: OK
ZynqMP> sf write FFFC0000 40000 20000
device 0 offset 0x40000, size 0x20000
SF: 131072 bytes @ 0x40000 Written: OK
ZynqMP> sf write FFFC0000 60000 20000
device 0 offset 0x60000, size 0x20000
           ............
           ... snip ...
           ............
ZynqMP> cmp.b FFFC0000 FFFD0000 10000
Total of 65536 byte(s) were the same
ZynqMP> sf read FFFC0000 4C70000 10000
device 0 offset 0x4c70000, size 0x10000
SF: 65536 bytes @ 0x4c70000 Read: OK
ZynqMP> cmp.b FFFC0000 FFFD0000 10000
Total of 65536 byte(s) were the same
ZynqMP>
```

Boot Linux

Set Boot switch (SW1) to QSPI mode (boot from flash)

1. Power off the board
2. Set SW1 to QSPI mode.
   Dip switches 1, 2 and 4 must be “up”, away from the numbers. Switch 3 must be “down”.



3. Power on again and closely monitor the serial interface. The hard-core CPU inside the FPGA fabric starts booting and must be stopped before loading the Linux kernel.
4. Boot Linux until U-boot and stop boot process by pressing a key

```
Xilinx Zynq MP First Stage Boot Loader
Release 2018.3   Jan 14 2021  -  21:53:37
NOTICE:  ATF running on XCZU7CG/silicon v4/RTL5.1 at 0xfffea000
NOTICE:  BL31: Secure code at 0x0
NOTICE:  BL31: Non secure code at 0x8000000
NOTICE:  BL31: v1.5(release):xilinx-v2018.2-919-g08560c36
NOTICE:  BL31: Built : 21:52:50, Jan 14 2021
PMUFW:  v1.1


U-Boot 2018.01 (Jan 14 2021 - 22:10:03 +0000) Xilinx ZynqMP
ZCU102 rev1.0

I2C:   ready
DRAM:  3 GiB
EL Level:    EL2
Chip ID:     zu7cg
SF: Detected n25q512a with page size 512 Bytes, erase size 128
KiB, total 128 MiB
In:    serial@ff000000
Out:   serial@ff000000
Err:   serial@ff000000
Model: ZynqMP ZCU102 Rev1.0
Board: Xilinx ZynqMP
Net:   ZYNQ GEM: ff0e0000, phyaddr c, interface rgmii-id
PHY is not detected
GEM PHY init failed
No ethernet found. (RH: Normal during first boot)
Hit any key to stop autoboot:  0
```

5. Set boot arguments, root, kernel debug parameters, serial interface

```
$ ZynqMP> setenv bootargs "console=ttyPS0,115200 earlyprintk
clk_ignore_unused root=mtd:jffs2 rw rootfstype=jffs2"
```

6. Save settings to flash
   ```
   $ ZynqMP> saveenv
   ```
7. Reboot
   ```
   $ ZynqMP> reset
   ```

8. Login with:
   a. $ user → root
   b. $ password → root

IMPORTANT!
The power supply for the ethernet chip is enabled during the boot by the device driver of the TPS650864 chip. However, in the first boot stage the status of the ethernet communication is checked by the processor and if it is not powered up, the ethernet socket is completely disabled in the processor. If the processor is restarted (without power cycling the board!) the ethernet interface will already be powered up before booting the processor and will be loaded correctly.

---

9. Reset board with a soft reset to enable ethernet power supply.
   It is not necessary to stop the boot process during u-boot a second time. Linux should boot automatically now with the ethernet interface enabled.
   ```
   $ shutdown -r now
   ```

```
U-Boot 2018.01 (Apr 01 2021 - 10:31:31 +0000) Xilinx ZynqMP ZCU102 rev1.0

I2C:   ready
DRAM:  3 GiB
EL Level:       EL2
Chip ID:        zu7cg
SF: Detected n25q512a with page size 512 Bytes, erase size 128 KiB, total 128
MiB
In:    serial@ff000000
Out:   serial@ff000000
Err:   serial@ff000000
Model: ZynqMP ZCU102 Rev1.0
Board: Xilinx ZynqMP
Net:   ZYNQ GEM: ff0e0000, phyaddr c, interface rgmii-id
eth0: ethernet@ff0e0000
```

Setup ADC test

1. Login (2nd time) with:
   a. $ user → root
   b. $ password → root

2. Check boot log with reference boot log (can be found here: https://ftp.science.ru.nl/grand_v2/hardware/Images/2CHINA/bootlog_1Apr2021.txt)

3. Check ADC to FPGA links with:
   ```
   $ grep "" /sys/bus/platform/devices/*.axi-jesd*/status*
   ```

4. Check status:

```
root@grand_zcu7:~# grep "" /sys/bus/platform/devices/*.axi-
jesd*/status*
Link is enabled
Measured Link Clock: 250.250 MHz
Reported Link Clock: 250.000 MHz
Lane rate: 10000.000 MHz
Lane rate / 40: 250.000 MHz
LMFC rate: 15.625 MHz
Link status: DATA
SYSREF captured: disabled
SYSREF alignment error: disabled
```

Note: If needed more detailed (debugging) information about the ADC lanes can be displayed with:

```
root@grand_zcu7:~# grep "" /sys/bus/platform/devices/*.axi-jesd*/lane*
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:Errors: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:CGS state: DATA
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:Initial Frame
Synchronization: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:Lane Latency: 1 Multi-frames
and 29 Octets
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:Initial Lane Alignment
Sequence: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:DID: 0, BID: 0, LID: 0, L:
2, SCR: 1, F: 2
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:K: 32, M: 2, N: 16, CS: 0,
N': 16, S: 1, HD: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:FCHK: 0xE0, CF: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:ADJCNT: 0, PHADJ: 0, ADJDIR:
0, JESDV: 1, SUBCLASS: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane0_info:FC: 10000000
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:Errors: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:CGS state: DATA
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:Initial Frame
Synchronization: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:Lane Latency: 1 Multi-frames
and 29 Octets
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:Initial Lane Alignment
Sequence: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:DID: 0, BID: 0, LID: 1, L:
2, SCR: 1, F: 2
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:K: 32, M: 2, N: 16, CS: 0,
N': 16, S: 1, HD: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:FCHK: 0xE1, CF: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:ADJCNT: 0, PHADJ: 0, ADJDIR:
0, JESDV: 1, SUBCLASS: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane1_info:FC: 10000000
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:Errors: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:CGS state: DATA
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:Initial Frame
Synchronization: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:Lane Latency: 1 Multi-frames
and 27 Octets
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:Initial Lane Alignment
Sequence: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:DID: 0, BID: 0, LID: 0, L:
2, SCR: 1, F: 2
```

```
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:K: 32, M: 2, N: 16, CS: 0,
N': 16, S: 1, HD: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:FCHK: 0xE0, CF: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:ADJCNT: 0, PHADJ: 0, ADJDIR:
0, JESDV: 1, SUBCLASS: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane2_info:FC: 10000000
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:Errors: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:CGS state: DATA
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:Initial Frame
Synchronization: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:Lane Latency: 1 Multi-frames
and 30 Octets
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:Initial Lane Alignment
Sequence: Yes
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:DID: 0, BID: 0, LID: 1, L:
2, SCR: 1, F: 2
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:K: 32, M: 2, N: 16, CS: 0,
N': 16, S: 1, HD: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:FCHK: 0xE1, CF: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:ADJCNT: 0, PHADJ: 0, ADJDIR:
0, JESDV: 1, SUBCLASS: 0
/sys/bus/platform/devices/84aa0000.axi-jesd204-rx/lane3_info:FC: 10000000
root@grand_zcu7:~#
```

5. Enable PG_VCCANA_12 and PG_VCCANA_34

```
$ cd /sys/class/gpio/
$ echo 83 > export
$ echo 84 > export
$ echo 500 > export
$ echo 501 > export

$ echo out > gpio83/direction
$ echo out > gpio84/direction
$ echo out > gpio500/direction
$ echo out > gpio501/direction

$ echo 1 > gpio83/value
$ echo 1 > gpio84/value
$ echo 1 > gpio500/value
$ echo 1 > gpio501/value
```

6. Set gains to of front-end to 0dB

```
$ cd /sys/bus/iio/devices/iio:device2/
$ echo 612 > out_voltage0_raw
$ echo 612 > out_voltage1_raw
$ echo 612 > out_voltage2_raw
$ echo 612 > out_voltage3_raw
```

Start IIO Scope

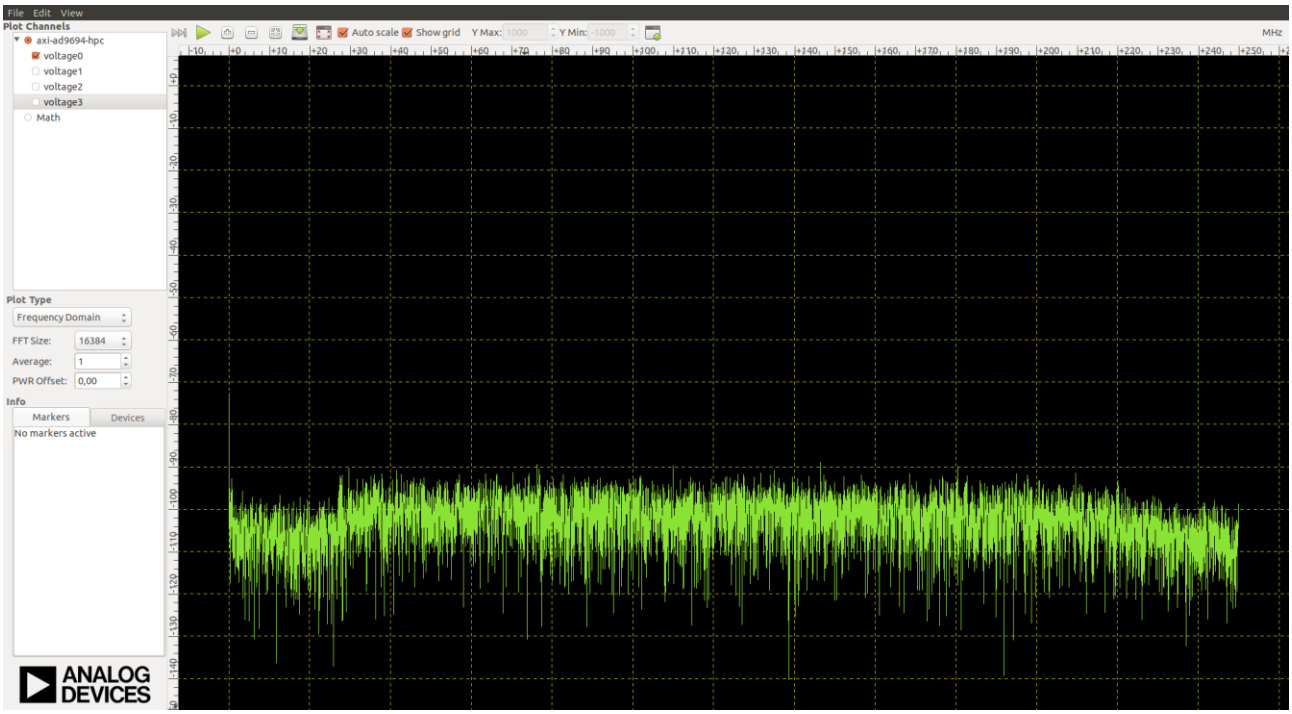
1. On the host PC (!) start the IIO Oscilloscope program by typing

```
$ osc
```

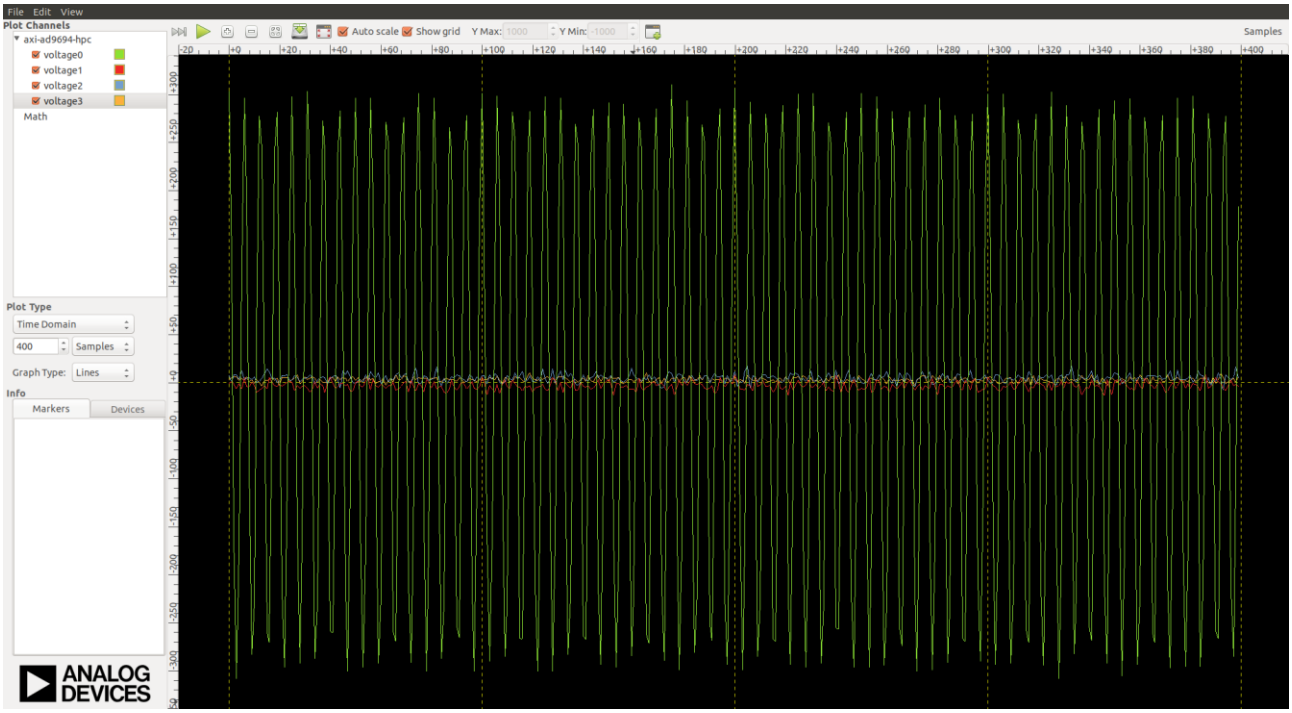
If the ethernet connection is setup correctly the program will startup automatically with the viewer enabled to look at the ADC data. This data comes from device axi-ad9694-hpc. (Devices ad5593 are the general purpose GPIO chips that can also be monitored from within the IIO Scope program).

2. Take data

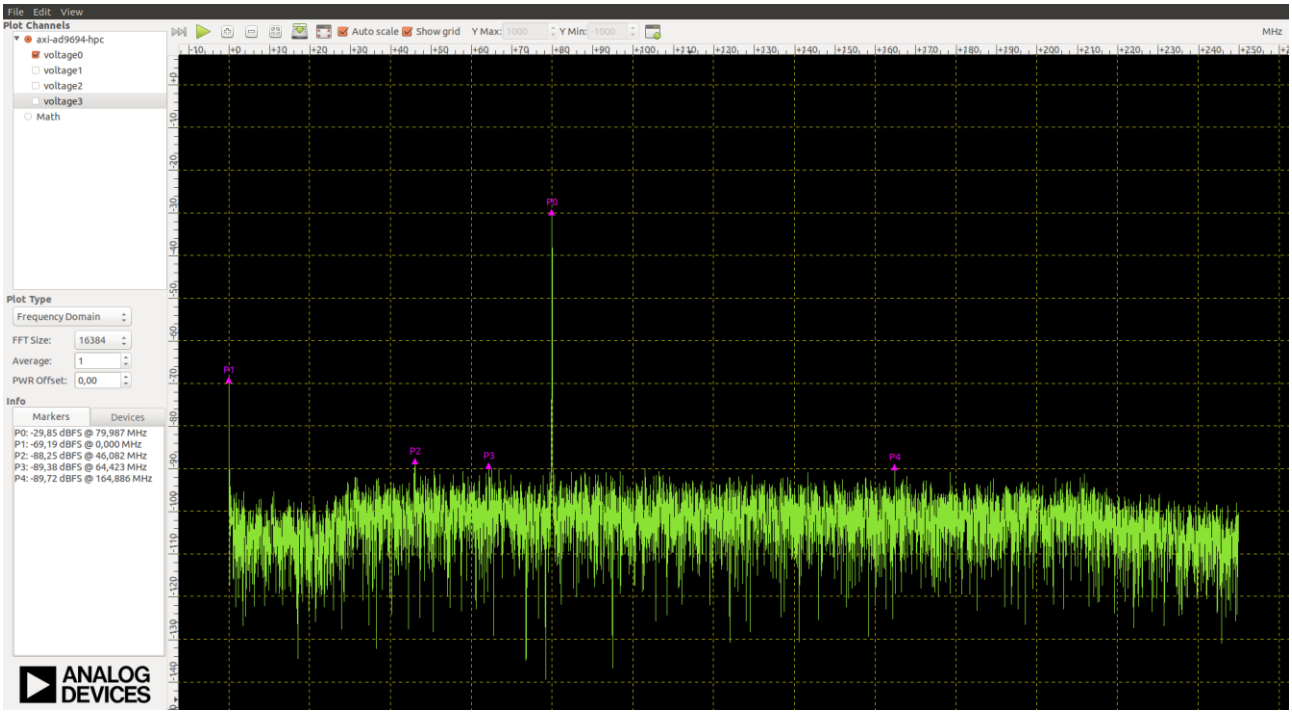
Without any input signal it should result in a clean spectrum with the shape of a band filter visible in the frequency response coming from the input filter. Enable only 1 ADC in the **Plot Channel** window and select **Frequency Domain** in **Plot Type** to generate the following figure.



Enable the ADC's that you want to readout (in the **time domain** it is possible to select multiple channels by checking on **voltage0**, **voltage1**, **voltage2** and **voltage3**) and supply a 80MHz 100mVpp sine wave. The result should look like this in the time domain:



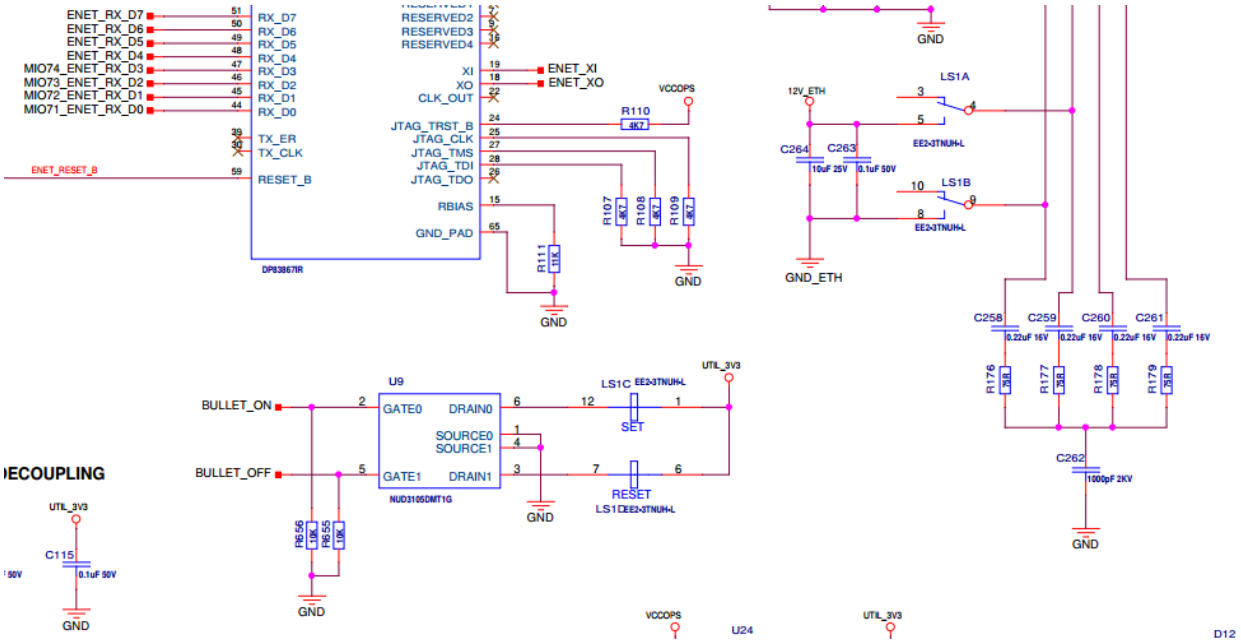
… and in the frequency domain (to place markers, **right click** in the graph and select **peak markers**):



3. Save the data from the time domain to a CSV file. (This function doesn't work for the frequency domain plot.)
4. Save the frequency domain plot as .png file.

Bullet test

The test of the bullet consist of powering the bullet and see if it lights up and if the ethernet interface comes up again. Therefor the relay for power over ethernet (PoE) must be set correctly. The default state of relay LS1 is OFF.



1. Unplug the ethernet cable from the host PC and plug it in the back of the bullet.

To power on the bullet BULLET_ON must be set to ‘1’ and BULLET_OFF must be set to ‘0’.

2. Type the following commands in the terminal:

```
root@grand_zcu7:~# cd /sys/class/gpio/
root@grand_zcu7:/sys/class/gpio# echo 61 > export
root@grand_zcu7:/sys/class/gpio# echo 62 > export
root@grand_zcu7:/sys/class/gpio# echo out > gpio61/direction
root@grand_zcu7:/sys/class/gpio# echo out > gpio62/direction
root@grand_zcu7:/sys/class/gpio# echo 0 > gpio62/value
root@grand_zcu7:/sys/class/gpio# echo 1 > gpio61/value
root@grand_zcu7:/sys/class/gpio# [ 2669.246105] macb
ff0e0000.ethernet eth0: link up (100/Full)
```

After setting gpio61 to ‘1’ the relays should switch and the bullet gets powered up. Also the ethernet connection should come up again.

3. To switch the bullet OFF again

```
root@grand_zcu7:/sys/class/gpio# echo 1 > gpio62/value
root@grand_zcu7:/sys/class/gpio# echo 0 > gpio61/value
```