**Rev0-2020.1-Petalinux Project Setup Guide**

**Author(s):** J. Cronin, L. Bidulka

**Release:** 3.0

**Release Date:** 25/04/2021

**Document Type:** Guide

**Project:** NuPRISM

**Keywords:** Vivado, Petalinux, NuPRISM, Firmware, Guide, 2020.1

**History of Changes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Release No.** | **Date** | **Description** | **Authour(s)** |
| 1.0 | 21/08/2020 | Initial Release | R. Payne |
| 2.0 | 23/12/2020 | December Revision | L. Bidulka |
| 3.0 | 25/04/2021 | April ’21 Revision | J. Cronin |
|  |  |  |  |
|  |  |  |  |

# Introduction

This guide is intended to introduce the incoming firmware developer to the NuPRISM rev 1a board, TRIUMF issued desktop computer, as well as the software tools you will be using to develop the board’s firmware and programmable logic, mainly Vivado and the petalinux software development tools. Currently, there are two workspace setups available to you, a Windows with Linux virtual box and full Linux installation, each with advantages and disadvantages. The desktop computer you have been given already includes a windows 10 Pro installation dual-booted alongside an Ubuntu 18.04 Installation, during the Linux setup the windows workspace was reset so you will need to setup the virtual box and software yourself if you choose that route.

To access either operating system (OS), restart the machine and during boot-up, press “delete” when prompted to enter the UEFI BIOS Utility menu, from here you will see the boot priority in the bottom right side of the screen. Select “Boot Menu” to see all bootable partitions and choose either the Windows or Ubuntu partitions.

For the Windows 10 workspace, installing device drivers and tools like Vivado will be much easier but you will need to install and run petalinux out of a virtual machine (petalinux is Linux only), this makes filesharing and running midas (experiment frontend) very tedious and as mentioned earlier, you will need to set the workspace up yourself. If you choose to work in the native Linux environment, you will have no need for virtual machines and have access to all the same necessary features as in Windows, however if you are not familiar with Linux its **very important** to know that you can damage the OS irreversibly if you are not careful using your root privileges (i.e. sudo commands).

# Vivado and Vivado Project Setup

If you are like me and used to using Quartus for developing FPGA projects, Vivado may be overwhelming at first, this section will attempt to explain some of the features that you will be interacting with almost daily in your work.

1. If not installed, download Vivado 2020.1 from <https://www.xilinx.com/support/download.html>
   1. The only chip required is the ZynqMPSoC. To limit the download size, it is recommended that only that one is downloaded
2. Clone the project located [here](https://github.com/nuPRISM/iwcd-xu1-petalinux-rev0-2020-01/tree/canada-develop/documents), you will likely be using the rev1a-integration branch.
   1. Be sure to clone it as high as possible in your directory to avoid Vivado compilation errors due to length of path
      1. Ex: C:/Repos/ (Windows), ~/Source (Ubuntu)

### Block Design

Compared to Quartus, Vivado does a much better job integrating Intellectual Property (IP) into designs by providing a visual interface for organizing a design, IP refers to HDL modules developed by Xilinx that are general purpose, like clock dividers, buffers or modules that are static within the hardware package used. The NuPRISM utilizes a many of these modules inside the block diagram for defining processor, DMA and GPIO connections, you can open the block diagram by clicking “open block design” on the leftmost panel. The block design UI is useful for planning a design when the underlying HDL is out of scope, in this stage you can easily configure complex blocks, connect them using Xilinx’s AXI4 protocol, organize blocks and define input and output ports.

To use the block design in an HDL project, we will need to generate a “wrapper” file so we can instantiate the block design inside the top-level module (if you make changes to ports in the block design you will need to regenerate the wrapper). To (re)generate the wrapper, find the project source directory to the left of the block diagram view (the panel will have sources, design, signals tabs, you need to select sources), next, locate the **system\_top** entry and expand to see the modules instantiated, expand the MercuryXU1\_wrapper to show the MercuryXU1 block design. Finally, right-click on the block design source, select “create HDL wrapper and select “ok” at the prompt to regenerate, you can now click on MercuryXU1\_wrapper to see the new Verilog stub.

### Flow Navigator

To generate the bitstream for your design, follow these steps using the left-side panel:

1. Run synthesis
   1. If you get some errors, try regenerating output products and synthesizing again (this issue should be fixed now).
2. Run implementation
3. Generate Bitstream
4. Go to File -> Export -> Export Hardware
   1. Select Fixed platform type
   2. Select Include Bitstream
   3. Export to default folder
   4. Finish

If you wish to run all these steps at once, simply clock Generate Bitstream and all preceding steps will run if needed.

### Debugging

Vivado uses a module called an Internal Logic Analyzer (ILA) for on-target debugging essentially by defining one in the project it will synthesize into your design and take up a portion of the programmable logic dedicated to recording the signals you feed into it, this feature requires the Xilinx JTAG dongle be attached to the board. There are three ways of setting up the ILA, though I have never used the pre-synthesis option:

* Block design
  + In the open block design, right click on a bus you wish to debug and click the “debug” option, after a second a green bug icon will appear over the bus. Click the “Run Connection Automation” option inside the green banner above the block design and in the prompt, select a clock (most likely pl\_clk0) and click okay. Finally, regenerate the block design to create the ILA.
* Post-synthesis
  + This option requires first “marking” signals for debugging, use the syntax below.
  + In the .xdc constraints file
    - set\_property MARK\_DEBUG true [get\_nets {<module\_name>/<signal\_name>}]
    - Note <modue\_name> is with respect to the top level module
  + In the source code:
    - (\* mark\_debug = “true” \*) logic <signal\_name>
  + With the signals marked for debug, click “Set Up Debug” in the “Open Synthesized Design” in the flow navigator on the left. From here you can select the signals you want to debug and create the ILA.

After running the design flow descriped in “Flow Navigator”, you can open the hardware manager, auto connect to the target (NuPRISM board) and view the ILA data.

# Hardware Setup

Required Hardware:

* Mainboard
* Enclustra XU1
* Mainboard Power Cable
* Xilinx JTAG Debugger
* Power Supply
* USB A/B-2.0 cable
* Ethernet Cable

1. Ensure the XU1 is properly in place on the mainboard
2. Plug the JTAG Debugger into the Xilinx JTAG port of the mainboard
3. Use the USB cable to connect the board to your host computer
4. Use your DC power supply to deliver 12V to the mainboard

# 

# Petalinux Setup

**Note:** Please use [the Xilinx Petalinux Tools Guide PDF](https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf) in addition to this document if you run into issues, it has examples of expected outputs and instructions on how to use commands and how to build Petalinux. Please also reference [the Xilinx QEMU System Simulation Guide](https://www.xilinx.com/support/documentation/sw_manuals/petalinux2013_10/ug982-petalinux-system-simulation.pdf) for information on how to use the QEMU system simulator, where you can test your Petalinux image without hardware. QEMU runs entirely in a simulator through the terminal. I have created my own instructions focusing more on using the tools rather than setting them up, however the instructions left for me will be in version 2.0 of this document in the repository (note the in the old instructions, the Cypress USB to UART has been replaced by the USB connection and requires no additional steps).

1. Ensure that the [NuPRISM](https://github.com/nuPRISM/iwcd-xu1-petalinux-rev0-2020-01) repository is cloned (I left it in /home/edev/Source/”) and that you are on the branch “rev1a-integration”.
2. From the root directory, navigate to ./petalinux/nuprism/
   * cd ~/Source/iwcd-xu1-petalinux-rev0-2020-01/petalinux/nuprism
3. From time to time, you will need to modify the embedded Linux image running on the mainboard, you can do this by running the command “petalinux-config” with arguments. As an exercise, try running:
   * petalinux-config -c kernel
   * When the menu appears, navigate to Device Drivers->SPI Support and ensure “User mode SPI device driver support” is active, you’ll need it to communicate with the ADCs
     1. Whenever I run git clean and remove the build/ and components/ folders, this gets unchecked.
   * Exit the menu (be sure to save the config when asked)
4. To build and package the petalinux image, run the following:
   * If you have made changes to the Vivado project, or are building for the first time run this command:
     1. petalinux-config –silentconfig --get-hw-description <path-to-xsa-exported-from-vivado> (should be in the /Vivado\_NuPRISM)
   * To build the software, run:
     1. petalinux-build
     2. petalinux-build –c <application-name> –x compile to build an individual app
   * To package the binaries, run:
     1. petalinux-package --boot --u-boot --fpga images/linux/system.bit --format BIN –force
5. As an alternative to step 4), you can run the python script I created named “build\_package.py” located in petalinux/nuprism/, use the -g argument to locate the hardware description export.
6. Now you will need to copy the boot binaries and extract the root filesystem to an SD card (you should have been given a formatted one), with an SD card inserted, run the following:
   * Navigate to the image directory, “cd ~/Source/ iwcd-xu1-petalinux-rev0-2020-01/petalinux/nuprism/images/linux
   * To copy use: cp {BOOT.BIN,image.ub,boot.scr} /media/edev/boot
   * To format your SD card if needed, complete the tutorial in Appendix H of the Petalinux 2020.1 user guide: <https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf>
   * Extract rootfs with, “sudo tar -C /<destination directory, likely /media/<user>/root > -zxvf rootfs.tar.gz”
7. As an alternative to step 6), run “export\_image.py”, this will likely only work if you are using the SD card I formatted as it requires specific partition names and mount locations.
8. Remove the SD media, place the SD card into the NuPRISM board and power on the board (make sure the main board is in BOOT MODE 0, this is a switch located next to the SD insert).
9. Open up a serial connection using putty, for convenience you can run “python3 open\_serial\_con.py”
   * You will need to determine the dev path for the USB connection, the above script can do this if you pass the -a argument or you can do it manually running the command “ls /dev/ | grep “tty”” with the board turned off, and again with the board turned on, and seeing which entry has been added, likely this /dev/ttyACMx where x is a number.
10. When the putty terminal opens, you will likely see a blank screen, this is because the first stage bootloader has ran before the connection was established, hit enter and you will see a command prompt like this:
    * ZynqMP>
11. Enter “boot” to complete the boot process, you will then be prompted to login, use user: “root” and password: “root”.



# MIDAS Host Setup

The Midas host setup has been completed for the Linux workspace, if you intend to use Windows or need to refer to the guide see version 2.0 of this document.

1. On your host, navigate to ~/packages/midas/bin run in this order:
   1. ./odbedit
   2. ./mhttpd
   3. ./mlogger
   4. ./mserver
2. Add the ip of both the MIDAS host and the petalinux board to the list of allowed RPC hosts for MIDAS
   1. Connect to the localhost:8080 midas server in a browser on the VM
   2. Navigate to the ODB tab
   3. Navigate to /Experiment/Security/RPC hosts
   4. Add the ip of the VM and the petalinux board to the allowed hosts list
3. Once the firmware has booted, run (on the mainboard):
   1. mfe -h <host-ip-address>:<mserver-address> -e e777
      1. example: mfe -h 192.168.1.80:1175 -e e777
      2. you may need to add the mainboards ip address to a list of trusted ip addresses in the MIDAS control panel
4. Start a run by pressing the Start button on the MIDAS frontend panel start page
5. Stop the run and check to see that data has been acquired

**Quick Reference:**

1. Build sequence:
   1. petalinux-config --get-hw-description <path-to-xsa-exported-from-vivado, should be in the /Vivado\_NuPRISM folder of the git repo (shared folder)>
      1. If you used the shared folder, the directory will be something like: /media/sf\_<name of shared folder>/<path-to-xsa-exported-from-vivado>
      2. Exit the menu that opens (press e), to read more about this see the Petalinux 2020.1 guide
   2. petalinux-build
      1. This should generate files inside ~/project/nuprism/images/linux/
   3. petalinux-package --boot --u-boot --fpga images/linux/system.bit --format BIN --force
   4. cd into /images/linux
   5. be sure sd card is on usb
   6. Extract file system with: sudo tar -C /<destination directory, likely /media/<user>/root > -zxvf rootfs.tar.gz
   7. Copy boot files with: cp {BOOT.BIN,image.ub,boot.scr} /media/edev/boot
   8. Eject root drive