Syfala project: how to install and run Syfala toolchain on Linux

The Syfala Team

26 septembre 2022

1 Installation instruction of syfala v7 toolchain

The Syfala toolchain is a compilation toolchain of Faust program on FPGA. This document explains how to install and run the toolchain v7 (version without petalinux), on a linux[1] machine. In practice, installing the Syfala tool-chain means :

- Installing the Faust compiler, see section 2 below.
- Creating a Xilinx account and downloading/installing a version 2020.2 of Xilinx `vivado` toolchain : `vitis_hls`, `vivado` and `vitis`. See section 3 below.
- Installing Vivado Board Files for Digilent Boards, see section 4
- Installing udev rules to use JTAG connection, see section 4
- Cloning the Syfala directory and running a simple example as explained in Section 5.

Section 6 explains the hardware configuration of the Zybo board for Syfala and Section 7 list all the important bug encountered during Syfala building, If you encounter a bug during the installation, please see Section 7.

Ubuntu dependencies : Syfala dependencies on Linux Ubuntu are the following :
`sudo apt install libncurses5 libtinfo-dev g++-multilib gtk2.0`

Warning : You need approximately 50GB of disk space to install the tool chain, and a good connection. The installation take several hours.

2 Installing Faust

It is recommanded to clone Faust from the github repository : https://github.com/grame-cncm/faust :

```
git clone https://github.com/grame-cncm/faust faust
cd faust
make
sudo make install
```

If you are using older version of Syfala, you might need to use older version of Faust (see `version` files in Syfala directory). the procedure is to get the commit number of the

1. tested on Ubuntu 18.04 and Ubuntu 20.04 and arch linux

version you need here : https://github.com/grame-cncm/faust/releases. For instance, if you use Syfala v5.4, it requires Faust version 2.31.1 (at least), it commit number is : 32a2e92c955c4e057d424ab69a84801740d37920, then execute :

```
cd faust
git checkout  32a2e92c955c4e057d424ab69a84801740d37920
make
sudo make install
```

3 Installing `Vivado`, `Vitis` and `Vitis_hls`

- Open an account on https ://www.xilinx.com/registration
- The Xilinx download page (https ://www.xilinx.com/support/download.html) and browse to the 2020.2 version. The page contains links for downloading the "Xilinx_Unified_2020.2_1118_1232_Lin64.bin" (It is available for both Linux and Windows but Syfala compiles only on Linux).
  - Download the Linux installer `Xilinx_Unified_2020.2_1118_1232_Lin64.bin`
- execute `chmod a+x Xilinx_Unified_2020.2_1118_1232_Lin64.bin`
- execute `./Xilinx_Unified_2020.2_1118_1232_Lin64.bin`
  - We suggest to use the "Download Image (Install Separately)" option. It creates a directory with a xsetup file to execute that you can reuse in case of failure during the installation
- execute `./xsetup`
  - Choose to install **Vitis** (it will still install **Vivado**, **Vitis**, and **Vitis HLS**).
  - It will need 110GB of disk space : if you uncheck *Ultrascale*, *Ultrascale+*, *Versal ACAP* and *Alveo acceleration platform*, it will use less space and still work.
  - Agree with everything and choose a directory to install (e.g. ~/Xilinx)
  - Install and wait for hours. . .
- Setup a shell function allowing to use the tools when necessary (add this to your `~/.bashrc`, `~/.zshrc` or whatever you're currently using, replacing `$XILINX_ROOT_DIR` by the directory you chose to install all the tools)
  - `export XILINX_ROOT_DIR=$HOME/Xilinx`

Then Install missing Vivado board files for Digilent boards and drivers for linux (explained in Section 4 below).

You HAVE to read sections 7.1 (locale setting) and 4.3 (vivado 2022 bug patch). If you do not, you might end up with unpredictible behaviour of Vivado.

4 Installing Vivado Board Files and Linux drivers

4.1 Cable drivers (Linux only)

- go to :
  `$XILINX_ROOT_DIR/`

`Vivado/2020.2/data/xicom/cable_drivers/lin64/install_script/install_drivers` directory
— run `./install_drivers`
— run `sudo cp 52-digilent-usb.rules /etc/udev/rules.d`, this allows **JTAG** connection through **USB**.

4.2 Vivado Board Files for Digilent Boards

Important : This step is needed to enable vivado to generate code for the Zybo Z10
— download :
https://github.com/Digilent/vivado-boards/archive/master.zip
— Open the folder extracted from the archive and navigate to its `new/board_files` folder. You will be copying all of this folder's subfolders
— go to `$XILINX_ROOT_DIR/Vivado/2020.2/data/boards/board_files`
— **Copy** all of the folders found in vivado-boards `new/board_files` folder and **paste** them into this folder

4.3 Installing the 2022 patch

— Follow this link : https ://support.xilinx.com/s/article/76960?language=en_US
— Download the file at the bottom of th page and unzip it in `$XILINX_ROOT_DIR`
— run `cd $XILINX_ROOT_DIR`
— run (in one single command line) :

```
export LD_LIBRARY_PATH=$PWD/Vivado/ \
2020.2/tps/lnx64/python-3.8.3/lib/ \
Vivado/2020.2/tps/lnx64/python-3.8.3/bin/python3 y2k22_patch/patch.py
```

5 Use Syfala (clone and launch)

The syfala repository is freely accessible (reading only) on github (`https://github.com/inria-emeraude/syfala`), you have to have a github account of course to clone it. As mentionned before, there may be several sub-directories with different version of Syfala (i.e. different interface for Faust hardware IP). Here are the step needed to run Syfala (after having following the installation instruction of Sections above) :
— Clone the Syfala github repository.
— install the `syfala.tcl` script
— Run the script

5.1 Clone the Syfala repository

to clone the version needed and compile a first architecture you can use the following commands :

```
git clone https://github.com/inria-emeraude/syfala mysyfala
cd mysyfala/
./syfala.tcl install
syfala examples/virtualAnalog.dsp
```

or if you have installed your ssh key on github :

```
git git@github.com:inria-emeraude/syfala.git mysyfala
cd mysyfala/
./syfala.tcl install
syfala examples/virtualAnalog.dsp
```

5.2 Use the `syfala.tcl` script

the command :
`$ ./syfala.tcl install`
will install a **symlink** in **/usr/bin**. After this you'll be able to just run :
`$ syfala myfaustprogram.dsp`
You'll also have to **edit** your shell **resource file** (~/.**bashrc** / ~/.**zshrc**) and set the following environment variable :

```
export XILINX_ROOT_DIR=/my/path/to/Xilinx/root/directory
```

`XILINX_ROOT_DIR` is the root directory where all of the Xilinx tools (Vivado, Vitis, Vitis_HLS) are installed.

5.2.1 Major Syfala commands

```
$ syfala examples/virtualAnalog.dsp
# -> runs full toolchain on the virtualAnalog.dsp Faust dsp file, which will be
    ready to be flashed afterwards on a Zybo Z710 board (by default)

$ syfala examples/virtualAnalog.dsp --board GENESYS --sample-rate 96000
# -> runs full toolchain for the Genesys board, with a sample-rate of 96000Hz

$ syfala examples/phasor.dsp --export phasor-build
# -> runs full toolchain on 'phasor.dsp', automatically exporting the build to
# export/phasor-build.zip

$ syfala examples/fm.dsp --arch --hls --report
# -> only run 'arch' & 'high-level synthesis' (HLS) step on 'fm.dsp', and show
    the report afterwards.

$ syfala examples/fm.dsp --board Z20 --arch --hls --export z20-fm-hls-build
# -> only run 'arch' & HLS step on 'fm.dsp' for Zybo Z20 board, and export the
    build.
```

5.2.2 Additional Syfala 'one-shot' commands

name	description	arguments
`install`	installs this script as a symlink in /usr/bin/	none
`clean`	deletes current build directory	none
`export`	exports current build in a .zip file located in the 'export' directory	name of the build
`report`	prints HLS report of the current build	none
`demo`	fully builds demo based on default example (virtualAnalog.dsp)	none
`flash`	flashes current build onto target device	none
`gui`	executes the Faust-generated gui application	none
`rebuild-app`	rebuilds the host control application, without re-synthesizing the whole project	none

Syfala 'one-shot' command examples

```
$ syfala clean
$ syfala demo
$ syfala export my-current-build
$ syfala rebuild-app
$ syfala flash
```

5.2.3 General Options to Syfala command

option	accepted values	description
`-c --compiler`	`HLS - VHDL`	chooses between Vitis HLS and faust2vhdl for DSP IP generation.
`--reset`	/	resets current build directory before building (**careful**! all files from previous build will be lost)

5.2.4 Controling Syfala Run steps

Note : the `--all` is not necessary if you wish to run all steps, just run :
`syfala myfaustdsp.dsp`

`--all`	runs all toolchain compilation steps (from `--arch` to `--gui`)
`--arch`	uses Faust to generate ip/host cpp files for HLS and Host application compilation
`--hls --ip`	runs Vitis HLS on generated ip cpp file
`--project`	generates Vivado project
`--synth`	synthesizes full Vivado project
`--host --app`	compiles Host application, exports sources and .elf output to `build/sw_export`
`--gui`	compiles Faust GUI controller
`--flash`	flashes boot files on device at the end of the run
`--report`	prints HLS report at the end of the run
`--export`	`<id>` exports build to export/ directory at the end of the run

5.2.5 Controlling the architecture build by Syfala

parameter	accepted values	default value
`--nchannels, -n`	an even number (2, 4, 6, etc.)	2
`--memory, -m`	DDR - STATIC	DDR
`--board, -b`	Z10 - Z20 - GENESYS	Z10
`--sample-rate`	48000 - 96000 - 192000 - 384000 - 768000	48000
`--sample-width`	16 - 24 - 32	24
`--controller-type`	DEMO - PCB1 - PCB2 - PCB3 - PCB4	PCB1
`--ssm-volume`	FULL - HEADPHONE - DEFAULT	DEFAULT
`--ssm-speed`	FAST - DEFAULT	DEFAULT

Here is the description of these parameters :

parameter	description
`--nchannels, -n`	sets the project's number of channels, it is equal to the **maximum number of input/output channels** rounded to the superior even number (e.g : if 3 channels : `nchannels` would be 4)
`--memory, -m`	selects if **external DDR3** is used. Enable if you use some delay, disable if you do want any memory access (should not be disabled)
`--board`	Defines target board. **Z10** ,**Z20** and **GENESYS** only. If you have a VGA port (rather than 2 HDMI ports), you have an old Zybo version, which is not supported.
`--sample-rate`	Changes **sample rate** value (Hz). Only 48kHz and 96kHz is available for **SSM** embeded codec. 192000 (**ADAU1777** and **ADAU1787** only) 384000 (**ADAU1787** only) 768000 (**ADAU1787** only and with `--sample--width 16` only)
`--sample-width`	Defines **sample bit depth** (16|24|32)
`--controller-type`	Defines the controller used to drive the controls when **SW3** is **UP**. (**SW3 DOWN** for **software** control), **SEE BELOW** for details on each value
`--ssm-volume`	Chooses audio codec to use. For now, it only changes the scale factor. **FULL** : Maximum (**WARNING** : for speaker only, do not use with headphones). **HEADPHONE** : Lower volume for headphone use. **DEFAULT** : Default value +1dB because the true 0dB (`0b001111001`) decreases the signal a little bit.
`--ssm-speed`	Changes **SSM ADC/DAC** sample rate. **DEFAULT** : 48kHz sample rate. **FAST** : 96Khz sample rate

6 Hardware configuration (Zybo Z7-10/20)

— Jumper **JP5** should be on *JTAG*
— **Power select** jumper should be on *USB*

— **Switches** SW0, SW1, SW2, SW3 should be **down** (i.e. toward the opposite side of the ethernet connector

— The **audio input** is **LINE IN** (blue), not MIC IN

— The **audio output** is the black **HPH OUT** jack

6.1 Control of the Syfala IP

To control your DSP, you can either use a Hardware Controller Board or a GUI on your computer. Beguinner should use GUI control.

GUI (SW3 DOWN) **SW3** should be **down** (0).
If you use GUI, open the GUIcontroller after booting with the following command :

```
make gui
```

Syfala Hardware Controller Board (SW3 UP) **SW3** should be **up** .

If you use a Hardware Controller Board, please set the `--controller-type` command-line parameter to the proper value (see below)

Controller-type values description

— **DEMO** : Popophone demo box
— **PCB1** : Emeraude PCB config 1 : 4 knobs, 2 switches, 2 sliders (default)
— **PCB2** : Emeraude PCB config 2 : 8 knobs
— **PCB3** : Emeraude PCB config 3 : 4 knobs, 4 switches
— **PCB4** : Emeraude PCB config 4 : 4 knobs above, 4 switches below

You can swap from hardware to software controller during DSP execution by changing SW3.

6.1.1 Switch description

```
  SW3   SW2    SW1    SW0
+-----+-----+-------+------+
| Hard| ADAU| BYPASS| MUTE |
|     |     |       |      |
|     |     |       |      |
| GUI | SSM |USE DSP|UNMUTE|
+-----+-----+-------+------+
```

— **SW3** : Controller type select : hardware (Controller board) or software (GUI). Default : **GUI**
— **SW2** : Audio codec input select (ADAU=external or SSM=onboard). Does not affect output. Default : **SSM**
— **SW1** : Bypass audio dsp. Default : **USE DSP**
— **SW0** : Mute. Default : **UNMUTE**

6.1.2 Status LEDs

The RGB led indicate the program state :

— **BLUE** = WAITING
— **GREEN** = ALL GOOD
— **ORANGE** = WARNING (Bypass or mute enable)
— **RED** = ERROR (Config failed or incompatible). Could happen if you select SSM codec with incompatible sample rate.

The 4 LEDs above the switches indicate the switches state. If one of them blink, it indicates the source of the warning/error.

6.1.3 SD card files

You can put the program on an SD card (if you want something reproductible and easily launchable, for the demos...).
After a `make` command, you should see a `BOOT.bin` file in SW_export (or you can build it with `make boot_file`).
Put the file on the root of SD card. And don't forget to put JP5 on 'SD' position !

7 Known bugs : Important "tricks" to be known ! !

This section regroups all the tricks that can result in unlimited waste of time if not known. These *known bugs* have been kept as they have been initially written, even if some of them do not occur anymore in more recent tool version.

7.1 Locale setting on linux

it is a known bug that `vivado` is sensible to the "locale" environment variable on linux, hence you have to set these variables in your `.bashrc` file :

```
export LC_ALL=en_US.UTF-8
export LC_NUMERIC=en_US.UTF-8
```

If you do not, you might end up with unpredictible behaviour of Vivado.

7.2 Patch 2022 date bug

Vivado and Vitis tools that use HLS in the background are also affected by this issue. HLS tools set the ip_version in the format YYMMDDHHMM and this value is accessed as a signed integer (32-bit) that causes an overflow and generates the errors below (or something similar).

Follow this link : https://support.xilinx.com/s/article/76960?language=en_US

Download the file at the bottom of the page and unzip it in your Xilinx base install directory (Xilinx file where you have your Vitis,Vitis_HLS and Vivado files).

DONT FOLLOW THE README... Just check the "Known Issues :" section on the Xilinx page which takes over the readme.

From the Xilinx directory, run :

— export LD_LIBRARY_PATH=$PWD/Vivado/2020.2/tps/lnx64/python-3.8.3/lib/
— Vivado/2020.2/tps/lnx64/python-3.8.3/bin/python3 y2k22_patch/patch.py

7.3 Save the Vivado Install file in case of installation failure

Vivado installation tends to fail. To avoid having to redownload the installation file each time you try , we suggest to use the " Download Image (Install Separately)" option. It creates a directory with a xsetup file to execute for installing. But don't forget to duplicate the installation file, because Vivado will delete the xsetup installation file you use if you choose to let him delete all files after the installation failed.

7.4 Vivado Installation stuck at "final processing : Generating installed device list"

If the install of Vivado is stuck at "final processing : Generating installed device list", cancel it and install the libncurses5 lib :

```
sudo apt install libncurses5
```

7.5 Installing Vivado Board Files for Digilent Boards

It is necessary, once Vivado install, to add support for new digilent board. the content of directory `board_files` has to be copied in `$vivado/2019.2/data/boards/board_files` (see

```
https://reference.digilentinc.com/learn/programmable-logic/tutorials/\
    zybo-getting-started-with-zynq/start?redirect=1#
```

Or directly here : https://github.com/Digilent/vivado-boards

7.6 Cable drivers (Linux only)

For the Board to be recognized by the Linux system, it is necessary to install additional drivers. See https://digilent.com/reference/programmable-logic/guides/install-cable-drivers

7.7 Digilent driver for linux

On some linux install, programming the Zybo board will need to install an additionnal "driver" : Adept2 https://reference.digilentinc.com/reference/software/adept/start?redirect=1#software_downloads

7.8 Vitis installation

Warning Apparently the installation process does not end correctly if the `libtinfo-dev` package is not correctly installed (https://forums.xilinx.com/t5/Installation-and-Licensing/Installation-of-Vivado-2020-2-on-Ubuntu-20-04/td-p/1185285. In case of doubt, execute these commands (april 2020) :

```
sudo apt update
sudo apt install libtinfo-dev
sudo ln -s /lib/x86_64-linux-gnu/libtinfo.so.6 /lib/x86_64-linux-gnu/libtinfo.so.5
```

7.9 "'sys/cdefs.h' file not found" during vitis_HLS compilation

If Vitis HLS synthesis fails with the following error :

```
'sys/cdefs.h' file not found: /usr/include/features.h
```

You have to install the g++-multilib lib

```
sudo apt-get install g++-multilib
```

8 The syfala team

Here is a list of person that have contributed to the Syfala project :

— Tanguy Risset
— Yann Orlarey
— Romain Michon
— Stephane Letz
— Florent de Dinechin
— Alain Darte
— Yohan Uguen
— Gero Müller
— Adeyemi Gbadamosi
— Ousmane Touat
— Luc Forget
— Antonin Dudermel
— Maxime Popoff
— Thomas Delmas
— Oussama Bouksim
— Pierre Cochard