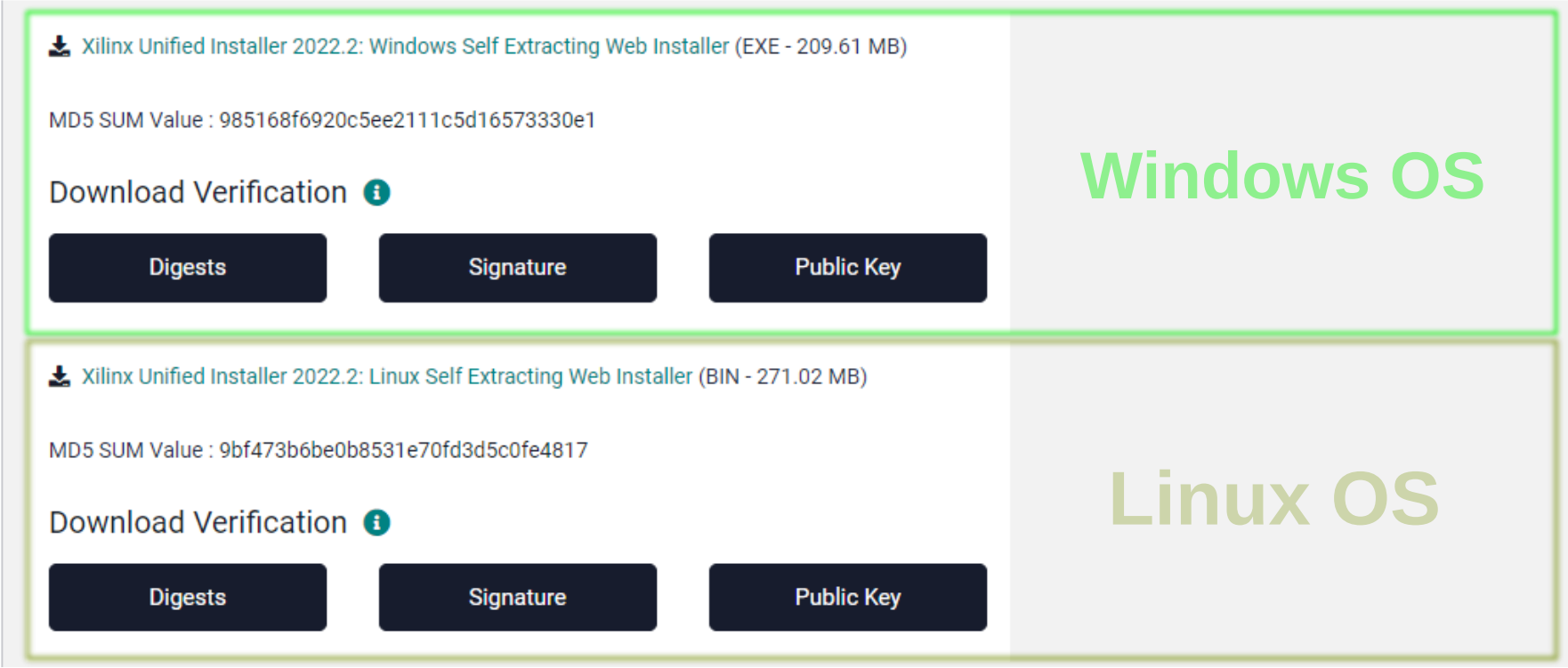
HDL – Agenda

1. Setup – Prerequisites
2. Setup - Knowledge
3. HDL introduction
4. Project description
5. Tasks
6. Extra resources & References

Setup – Prerequisites

HDL software installation Xilinx Vivado and Vitis 2022.2

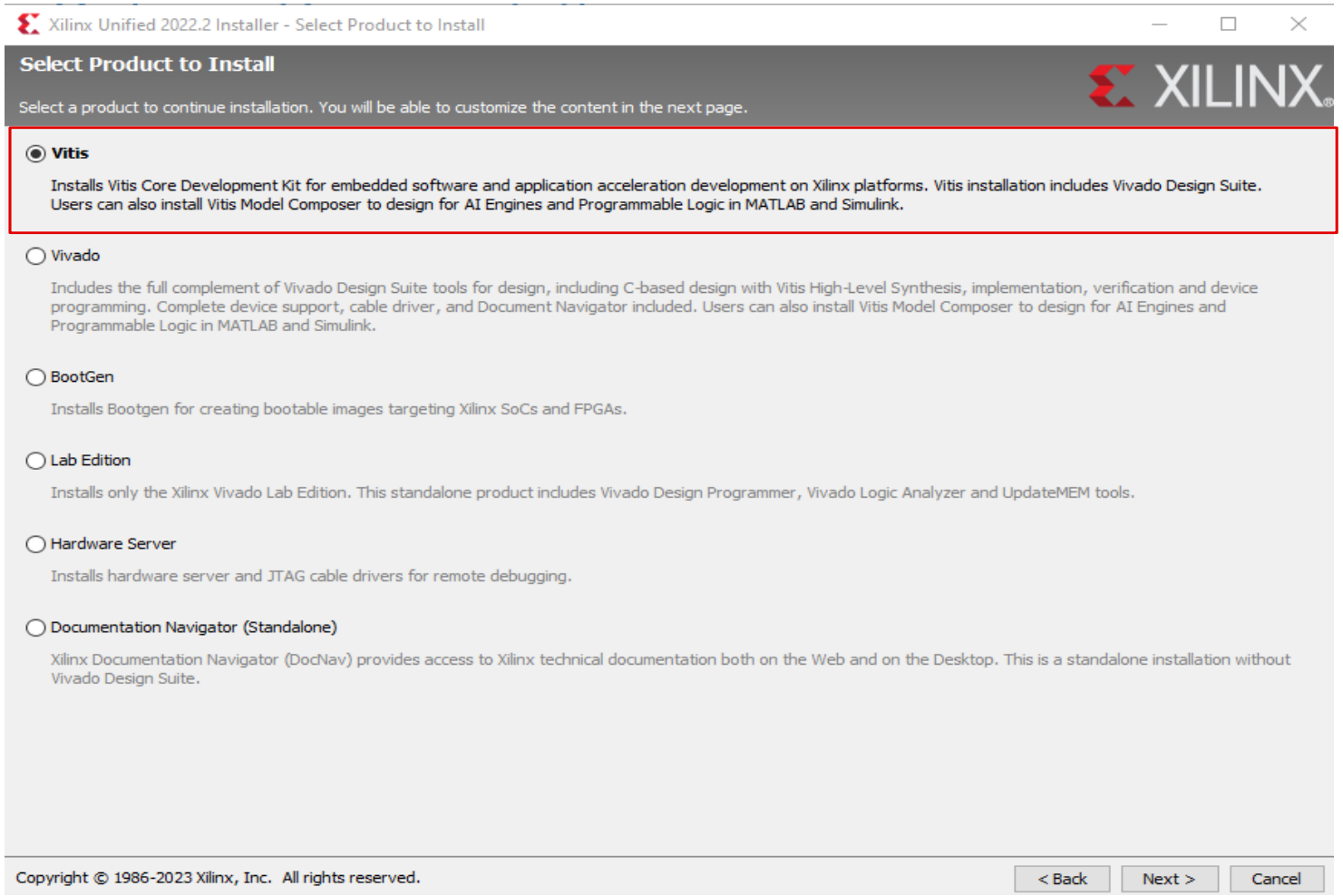
- Choose the appropriate Self Extracting Web Installer



Setup – Prerequisites

HDL software installation Xilinx Vivado and Vitis 2022.2

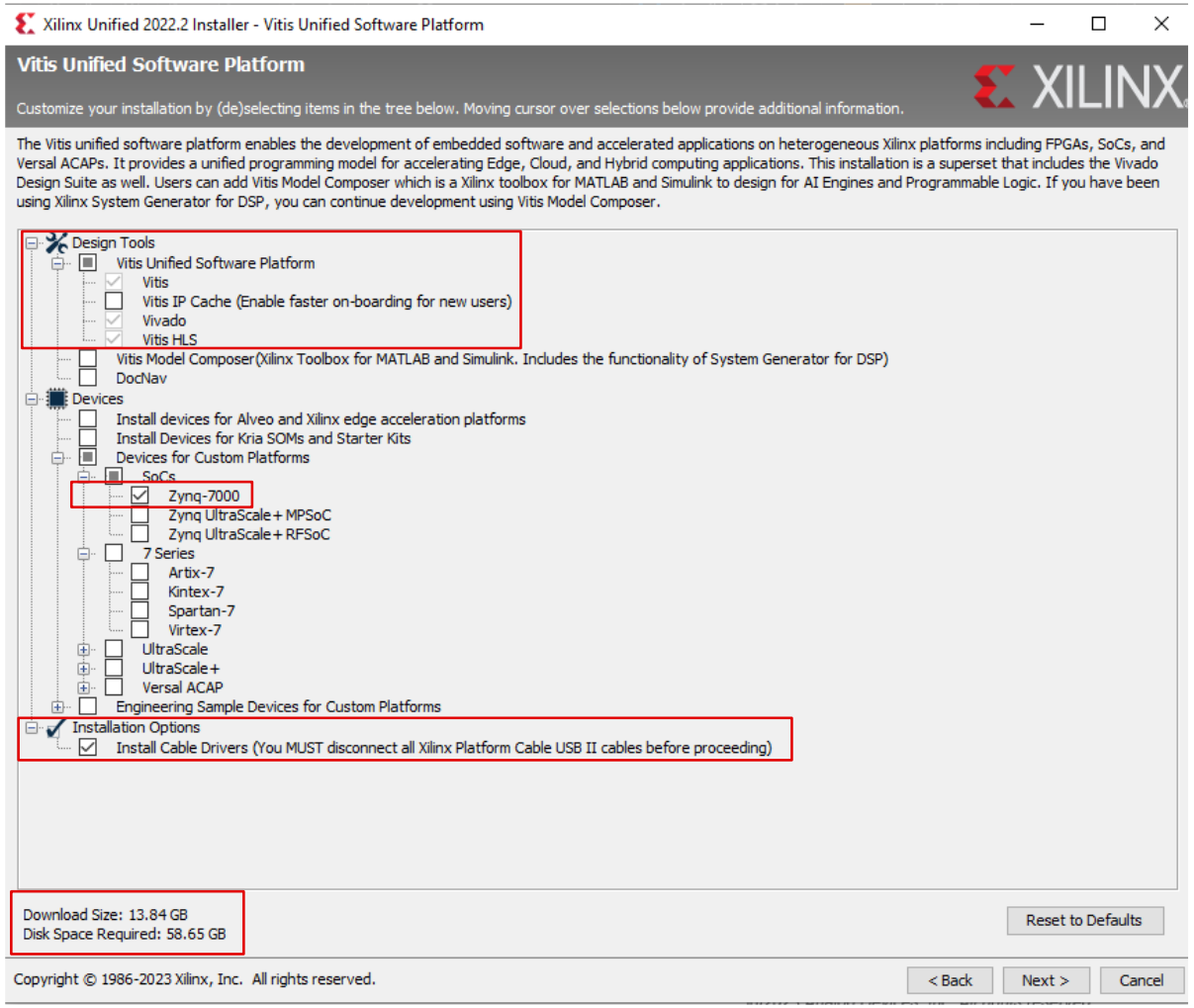
- Make sure you select the **Vitis** option:



Setup – Prerequisites

HDL software installation Xilinx Vivado and Vitis 2022.2

- Select **ONLY** the following plug-ins and make sure that you have the required free space:



Setup - Prerequisites

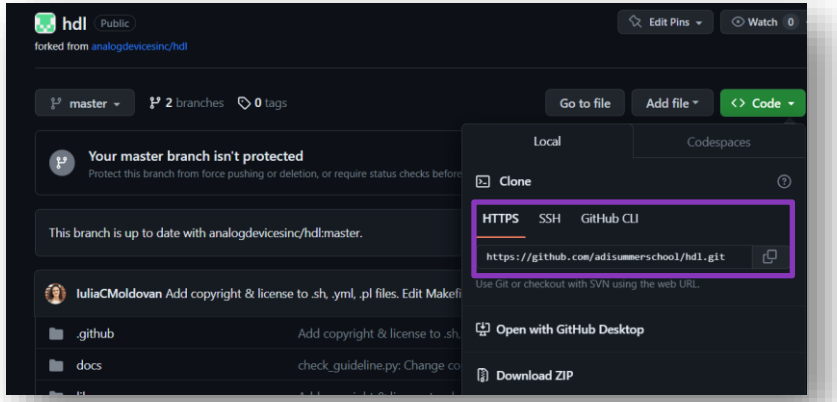
Tools

- TeraTerm (Windows only) or Putty – to create UART connections with the devices and communicate with them
- Cygwin – Linux terminal for Windows
  - Make sure to have the latest versions for the following packages: attr, automake, bash, cmake, git, grep, gvim, make, ssh, perl, python, run, sed, vim, wget, which, xlsclients.
- Text editors: Visual Studio Code or Notepad++ on Windows, Visual Studio Code or VIM on Linux
- WinScp – connect to the SD card from device to process files using a GUI
- Kuiper Imager to write SD cards with a Kuiper image
- IIO Oscilloscope

Setup – Prerequisites

Environment preparation - **GitHub setup**

- To use Git on your laptop, firstly you must do a First-time Git setup
- Follow the Creating a personal access token (classic) tutorial
- Copy the Personal access token
- Get the HTTPS links for both HDL and Linux repository from the ADISummerSchool organization



- Edit the paths:
  - https://**[copied_personal_access_token]**@github.com/adisummerschool/hdl.git
  - https://**[copied_personal_access_token]**@github.com/adisummerschool/linux.git
- Run the following commands in the workspace folder:
  - **git clone** https://**[copied_personal_access_token]**@github.com/adisummerschool/hdl.git
  - **git clone** https://**[copied_personal_access_token]**@github.com/adisummerschool/linux.git

Setup – Prerequisites

Environment preparation - **Setup Cygwin/Linux terminal**

- Exporting the paths to the installed tools and HDL repository:
  - **echo $PATH** to see how your $PATH looks like before exporting anything
  - In Cygwin/Linux terminal, type **vim ~/.bashrc** (VIM must be installed as a package for Cygwin prior to this) and write the following paths, modifying them accordingly to your system and preference (if you don't use Cygwin, then **do not** put */cygdrive/*!)
  - **export PATH=$PATH:/cygdrive**/partition/path/to**/Xilinx/Vivado/2022.2/bin**
  - **export PATH=$PATH:/cygdrive**/partition/path/to**/Xilinx/Vitis/2022.2/bin**
  - **export PATH=$PATH:/cygdrive**/partition/path/to**/hdl/bin**
  - now you can do again *echo $PATH* to see how the $PATH changed

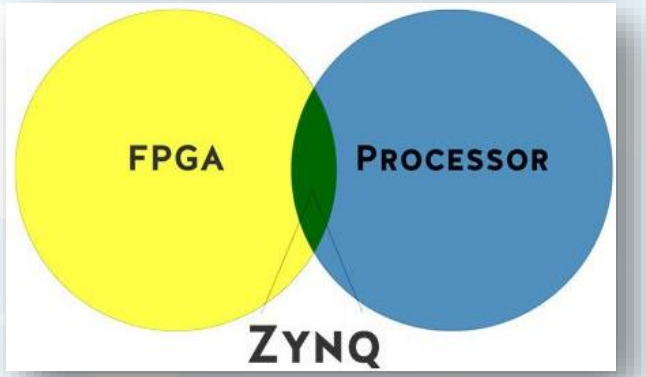
Setup – Knowledge

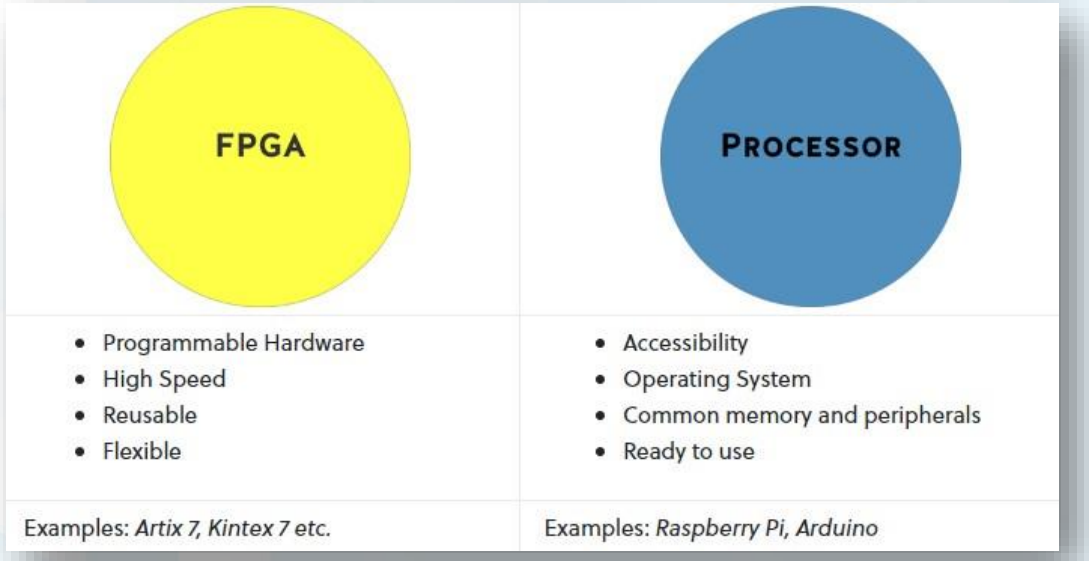
Prerequisites – Learning Materials

1. Git:
   a. Learn Git Branching interactively, with animations
   b. How to write a good commit message
   c. For more details, check the Git Book
2. VIM basics:
   a. Shortcuts
   b. Cheat sheet and another cheat sheet
3. Linux basics:
   a. Frequent terminal commands
   b. VIM text editor cheat sheet
   c. edX CS course: Introduction to Linux* OS, files, etc. (takes about 1 month to complete). Check the syllabus for more details.
4. Linux*: a few useful resources for a better understanding of the Linux Kernel drivers and device trees:
   a. [ADI] building the ADI Linux kernel
5. Coding guidelines*:
   a. HDL – on GitHub, used for the Guideline Checker GitHub action

HDL – About FPGAs

- **Field-Programmable Gate Array**
  - meaning the firmware can be modified without disassembling it or returning it to the manufacturer
- **Semiconductor devices**
  - matrix of CLBs and programmable interconnects; reconfigurable
  - Distinguished from ASICs which are custom manufactured for specific design tasks
  - The most detailed and enthusiastic presentation about FPGA can be found here*



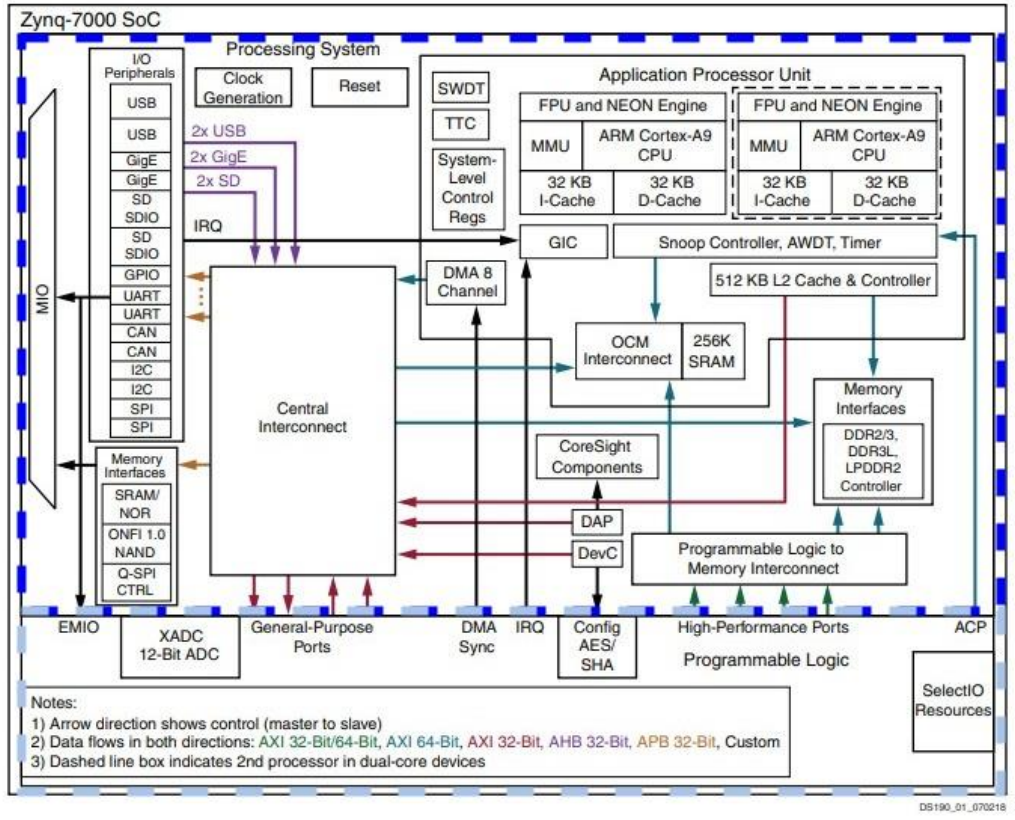


HDL – Zybo Z7-10 architecture

Zybo Z7-10 FPGA

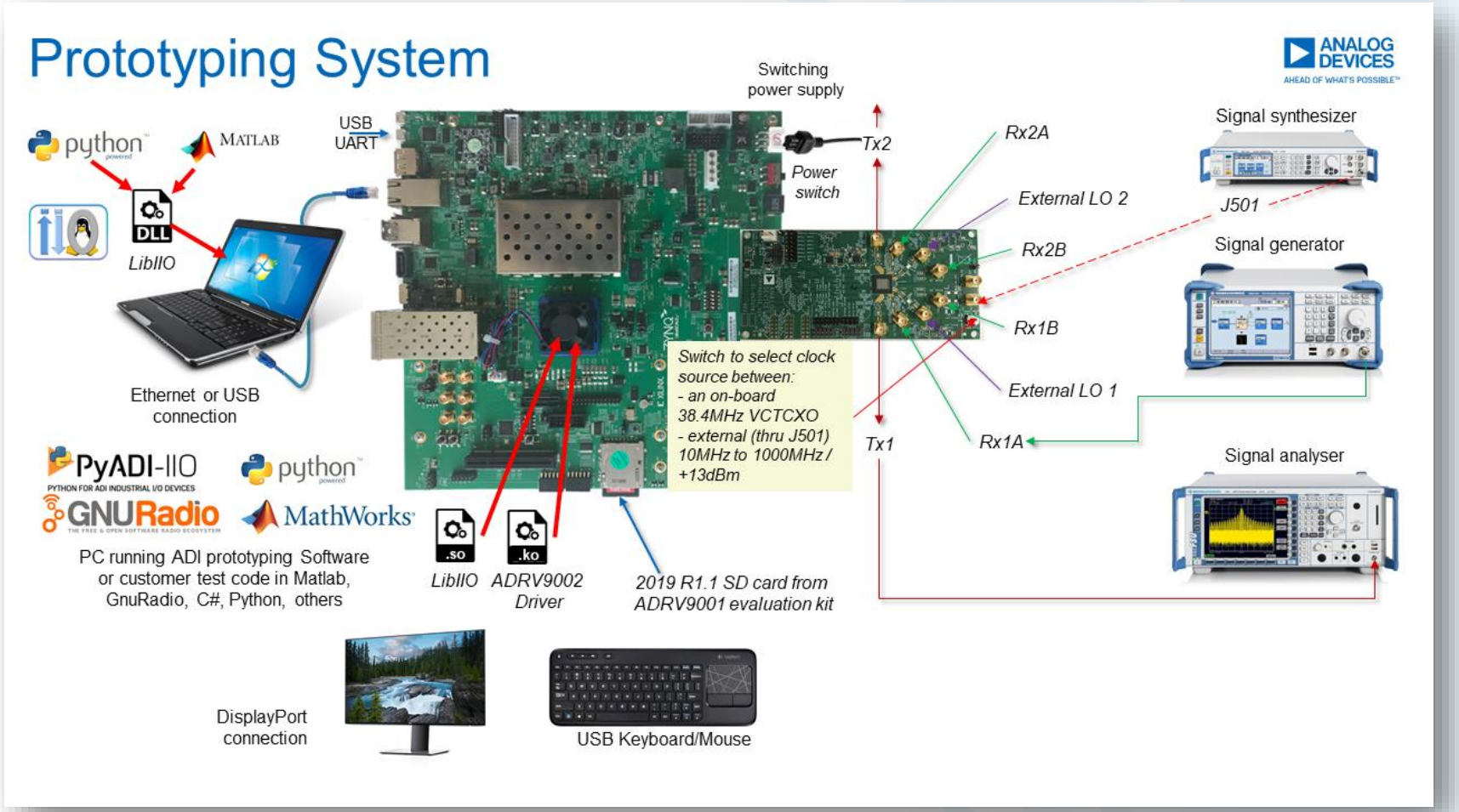
PS

PL



- **PS – Processing System** (the blue component)
- **PL – Programmable Logic** – all the other components that are added by the user (it can be seen better in the Block Diagram in Vivado)

HDL – Introduction

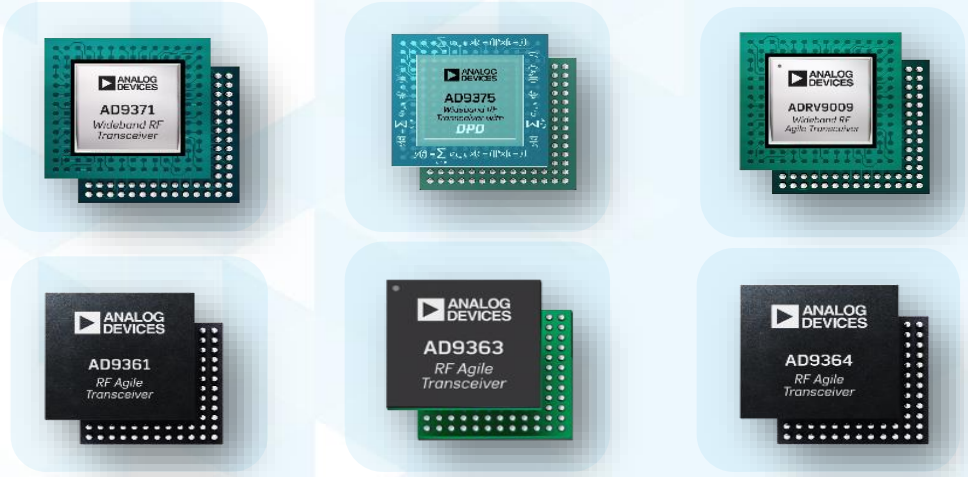


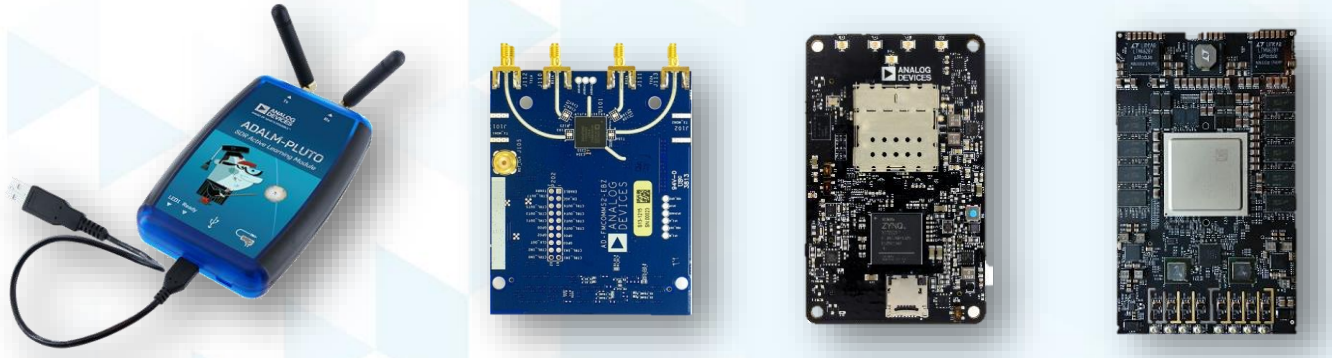
AMD/Xilinx	AC701	KC705	VC707	ZC702	ZC706	ZCU102	Zed	CoraZ7	Microzed	ADRV9009-EG11-SOM	ADRV936x-SOM	VCK190	KCU105	VCU118	VCU128
Intel/Altera	A10SOC	C5SOC	DE10 Nano	A10GX	A5GT	A5SOC									

* supported in previous releases

HDL – Enabled Designs

- **Open-source reference designs**
  - Precision Converters (AD40xx, AD4630-24…)
  - High Speed Converters (AD9081, AD9144, AD9680…)
  - Transceivers (AD9361, ADRV9371, ADRV9009…)
- **Prototyping boards**
  - AD-FMCOMMS2/3/4/5-EBZ
  - AD-FMC-DAQ2/3-EBZ
  - AD-FMCOMMS11-EBZ
  - AD-FMCOMMS8-EBZ
  - AD-FMCLIDAR1-EBZ
- **Standalone SOMs**
  - ADRV9361-Z7035/ADRV9364-Z7020
  - ADRV9009-ZU11EG
- **Educational Systems**
  - ADALM1000/ADALM2000
  - ADALM-Pluto



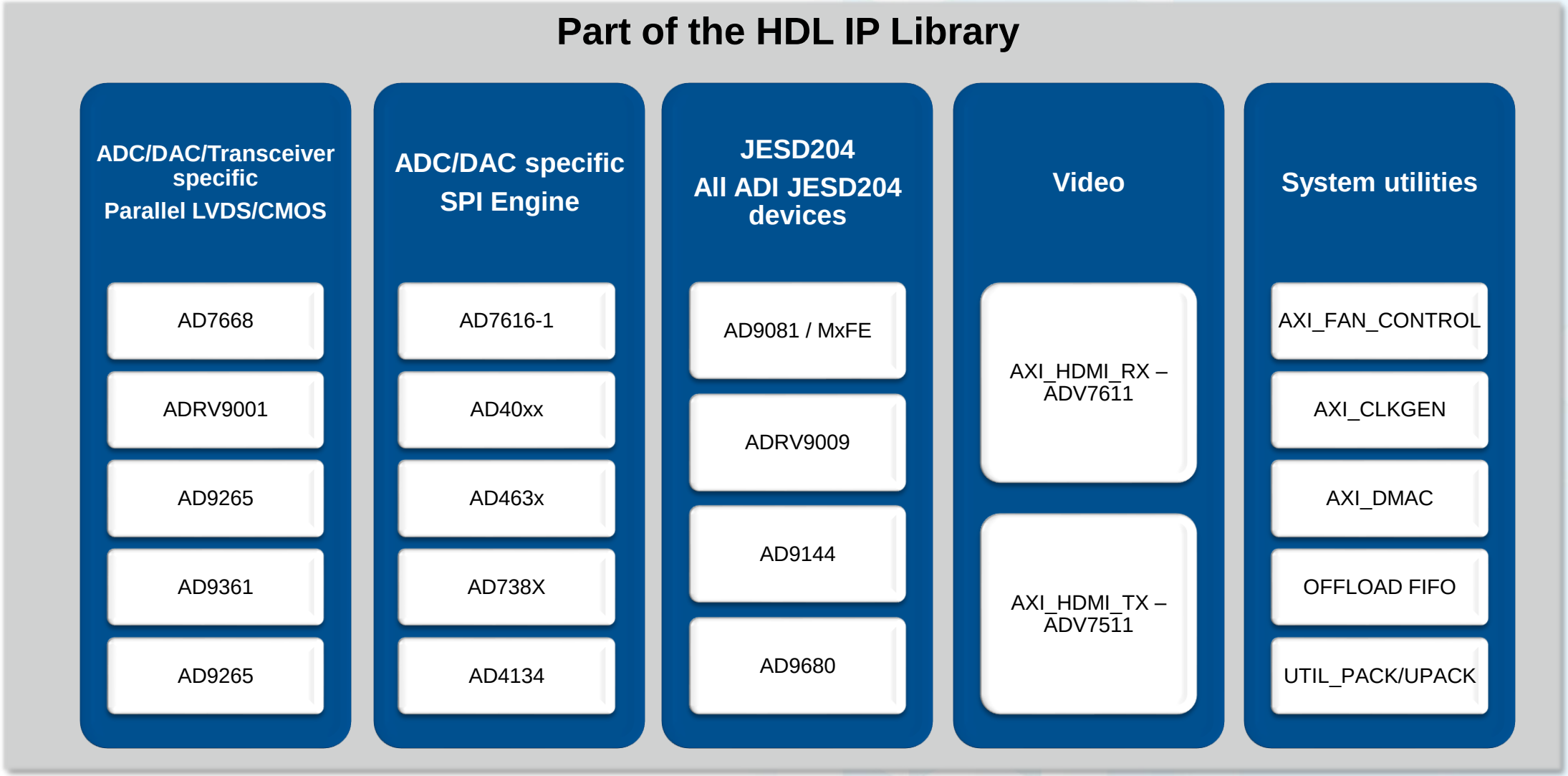


HDL – GitHub

- Map of the HDL repository:

  - ***/.github***: contains the resources needed for a GitHub action to be run: the scripts that are run in the workflows and the workflow code itself
  - ***/docs***: contains the coding guideline by which the guideline checker script is correcting the files and raising warnings
  - ***/library/scripts*** and ***/projects/scripts***: are the scripts that can be run on them
  - ***/library/Makefile***: contains all the IPs that exist there (what is written here will be built when running *make lib-all*)
  - ***/library/IP_name/Makefile***: contains all the file dependencies which are needed for this IP to be built
  - ***/projects/project_name/carrier/Makefile***: contains all the IP dependencies which are needed for this project to be built
  - More details about the structure of the repository can be found on the Git repository wiki page and about the projects' structure can be found here

HDL – IP Library



Github repository: HDL

HDL – Knowledge

- For **Zynq** architecture, the boot files are **uImage** + **devicetree.dtb** and **BOOT.BIN**
- For **ZynqMP** architecture, the boot files are **Image** + **system.dtb** and **BOOT.BIN**

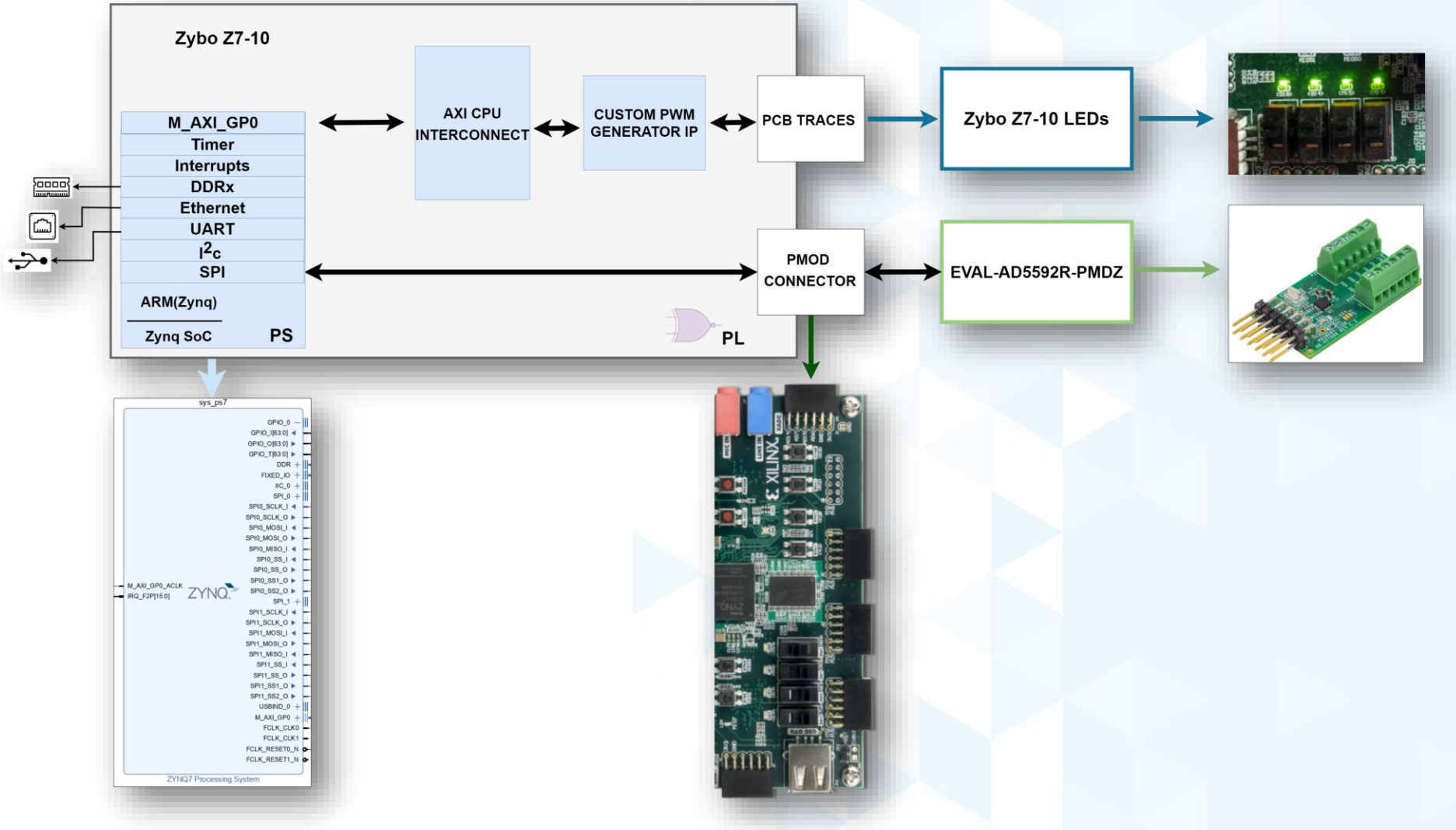
Booting with a custom design

- To understand better the structure of ADI's project framework, check Project files for Xilinx boards. There you will find out the purpose of each file.
- Useful information for generating the boot files:
  - Boot image: BOOT.BIN is the HDL part. It requires the files specified in Project files for Xilinx boards; using these, the project must be built (see How to build an HDL project), to have the **.xsa**; afterwards, a **u-boot.elf** file should be taken from the SD card for the BOOT.BIN to be generated (see How to build the boot image BOOT.BIN)
  - Linux Kernel image: depending on the configuration, the commands differ (see *How to build the Linux Kernel image and Devicetree Blob from source*); it will result an *image* file
  - Devicetree Blob: it is generated from a Devicetree Structure file (**.dts**) (see *How to build the Linux Kernel image and Devicetree Blob from source*); it will result a **.dtb** file

Booting with the reference design

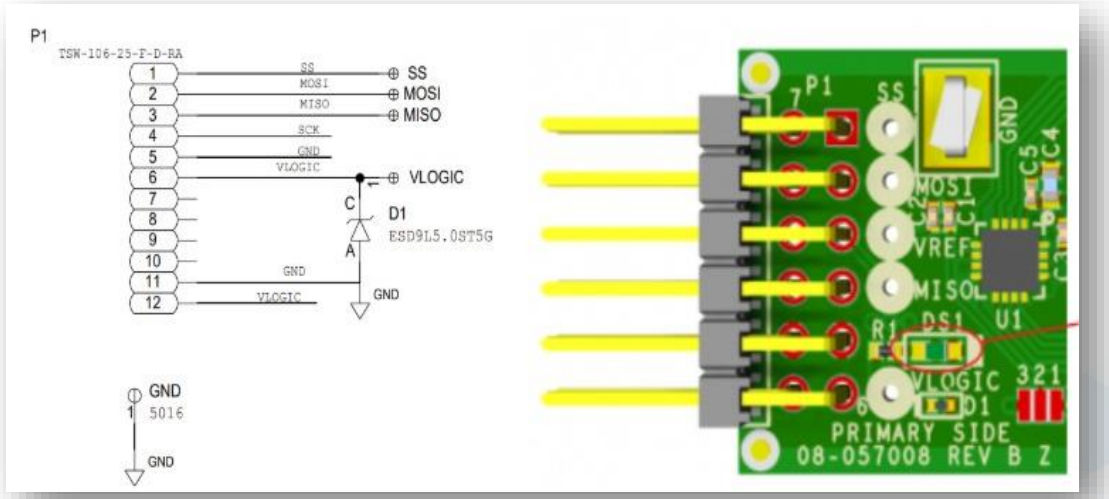
- The alternative to building these from scratch, is to take them from the SD card, where the reference designs are (after step How to prepare an SD card):
  - BOOT.BIN and the Devicetree Blob (**.dtb**) can be taken from the folders that have your desired configuration as a name
  - The Linux Kernel image can be taken from the common folder for the supported architectures (**zynq-common / zynqmp-common**)

HDL – Project diagram

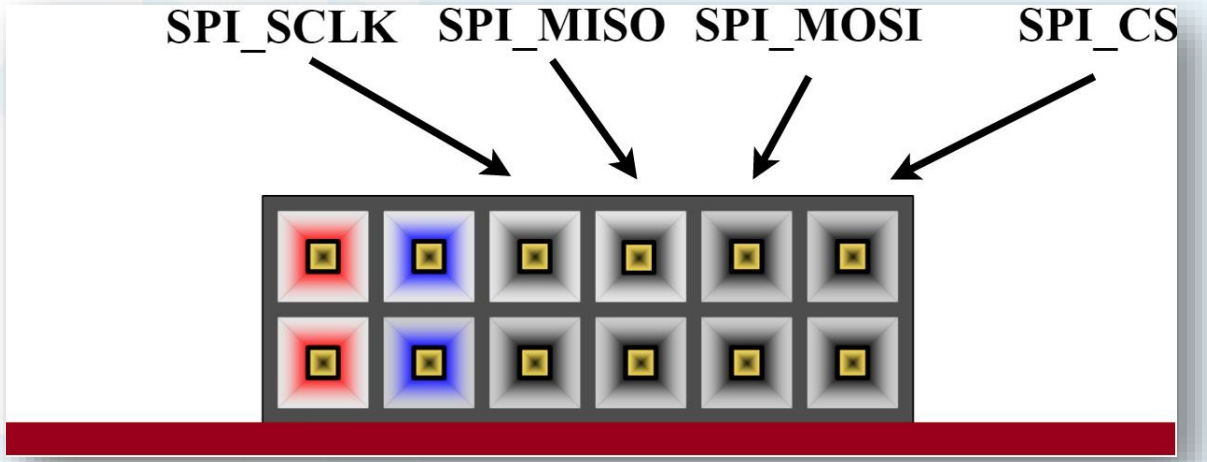


HDL – PMOD connector

Positioning of SPI pins on the EVAL-AD5592R-PMDZ board



SPI pins on the PMOD connector from the Zybo Z7-10 board



HDL – Start-up and Resources

A. Start-up

ADI HDL User Guide is a starting point to get more information about our projects and structure.

HDL coding guideline - to maintain things in an ordered manner, when submitting a Pull Request (PR) a script is run in a GitHub action, and it will raise warnings regarding one's code if some rules from this document are not respected.

B. Resources

The following links are meant as a support for the *Knowledge* chapter, and they are written in the proper order of execution as steps (the ones with * will be of good use for the next teams)

1. How to prepare an SD card with a Kuiper image
2. How to build an HDL project
3. How to build the boot image BOOT.BIN (two ways):
   - Using Vivado Tcl console, by running **adi_make::boot_bin** (for this option, *u-boot.elf* should be taken from the appropriate folder from the SD card that now has a Kuiper image) – check the steps from here
   - For Zynq architecture or for ZynqMP architecture
4. How to build the Linux Kernel image and Devicetree Blob from source for Zynq architecture* or for ZynqMP architecture*
5. How to build the No-OS with GNU make*

HDL – Specific prerequisites

1. **Xilinx Vivado and Vitis** – free 2022.2 version
2. **Verilog/VHDL training**: for the passionate and curious ones about this side, the following training will give you a glimpse of the basics – towards intermediate difficulty projects
   - Verilog beginner's tutorial which contains theory, hands-on examples, etc.
   - An intermediate tutorial* (it's still work in progress, but it will touch great topics)
   - HDL labs from Xilinx*(requires a few days to complete): takes you through several digital design concepts and helps you get accustomed to VHDL/Verilog if your background is with Verilog/VHDL
   - An introduction to formal methods* from zipcpu.com
3. **Tcl scripts**: Tcl or *Tool Command Language*, is an open-source multi-purpose C library which includes a powerful dynamic scripting language; in the HDL team, we use **.tcl** scripts to describe our project such that Xilinx Vivado understands it and can build it
   - More about Tcl can be found on Tcler's Wiki and on Tcl Introduction and Tutorial*
4. **Makefile**: contains all the dependencies needed for an IP/project to be built. The projects need several IPs to be previously built, and sometimes some IPs include other IPs or source files which they depend on
   - More about Makefiles*
5. **BOOT.BIN**: it's the result (boot image) after building the HDL project with a few extra files; this will be programmed into the FPGA, either through SD card or via JTAG
   - Xilinx Confluence Documentation* about it
6. **u-boot.elf**: Universal Boot Loader that is frequently used in the Linux community, and it is needed for the BOOT.BIN generation
   - More about it on the Xilinx Confluence Documentation about U-Boot*

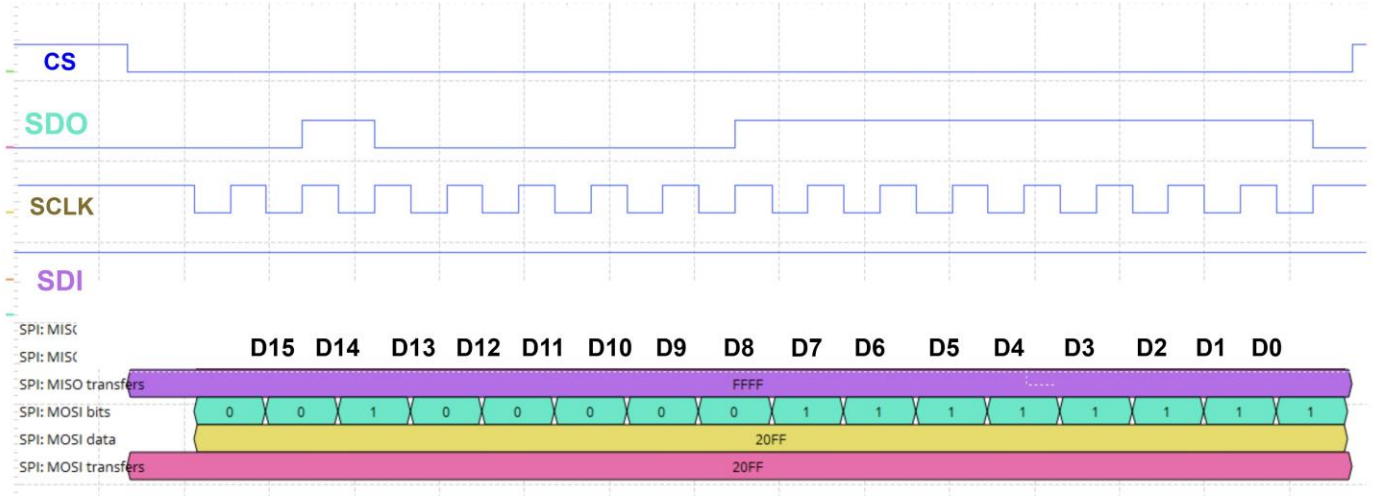
HDL – Knowledge - SPI communication

SPI COMMAND FOR CONFIGURING THE CHIP AS AN ADC

MSB															LSB
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	Register address				Reserved			ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

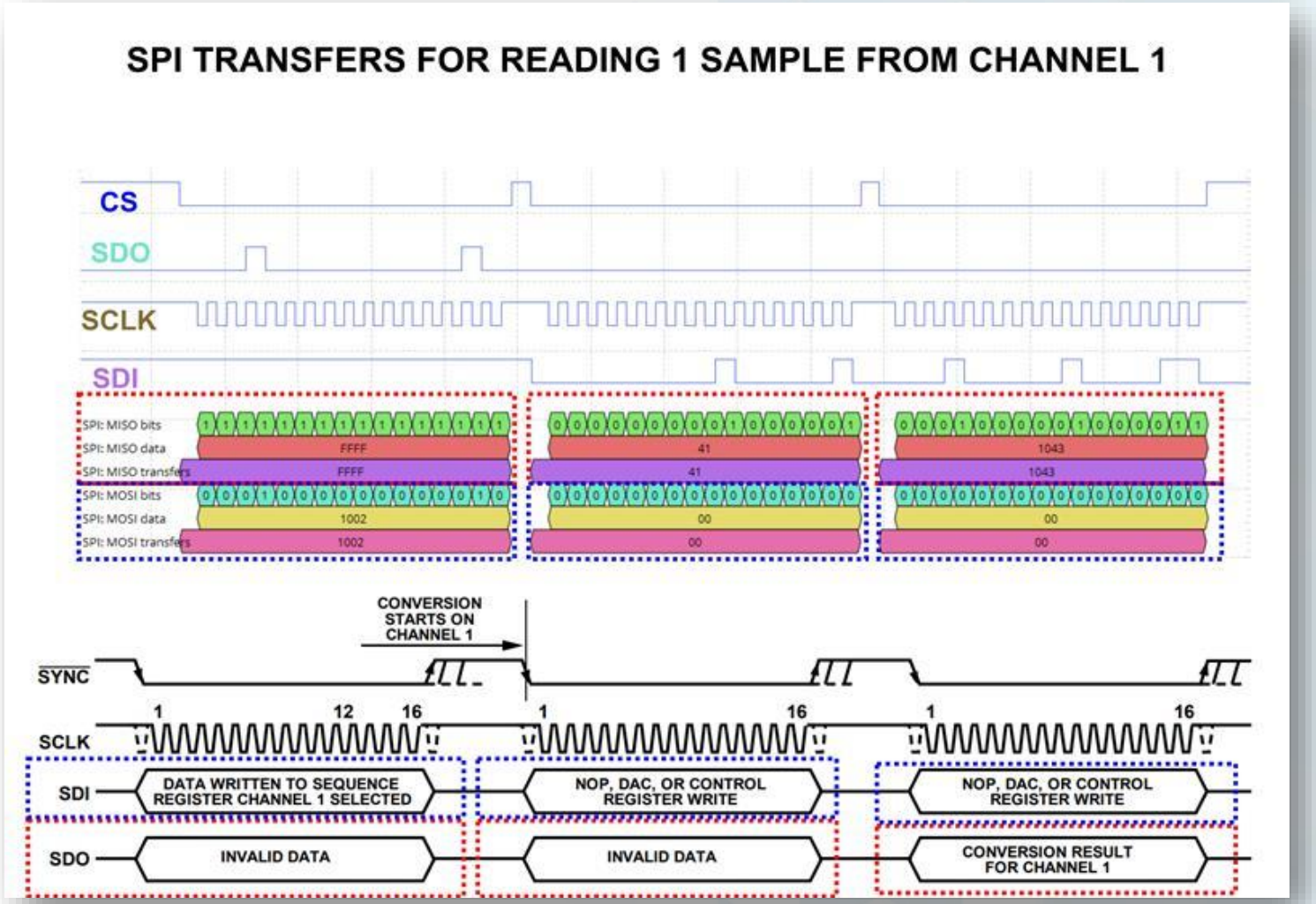
Table 26. Bit Descriptions for the ADC Pin Configuration Register

Bit(s)	Bit Name	Description
D15	MSB	Set this bit to 0.
D14 to D11	Register address	Set these bits to 0b0100.
D10 to D8	Reserved	Reserved. Set these bits to 0.
D7 to D0	ADC7 to ADC0 Text	Select I/Ox pins as ADC inputs. 1: I/Ox is an ADC input. 0: I/Ox function is determined by the pin configuration registers (default).



[Introduction to SPI interface]

HDL – Knowledge - SPI communication



[Introduction to SPI interface]

HDL – Project Structure

- We will create the described project for the Zybo Z7-10 FPGA board using the EVAL-AD5592R-PMDZ ADC.
- For the following files, there are already templates, and they can be found at [hdl/projects/ad5592r_pmdz]

system_constr.xdc :

- FPGA pins are connected to INPUT/OUTPUT ports

system_top.v:

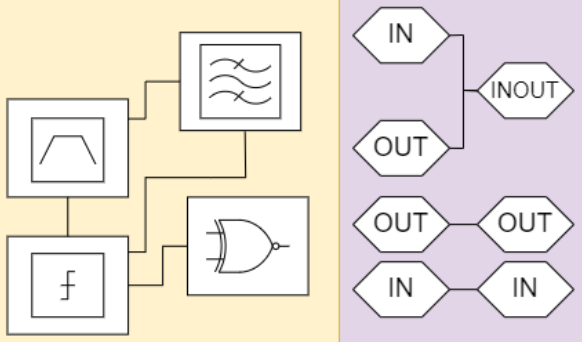
-INPUT/OUTPUT ports are connected to block design ports

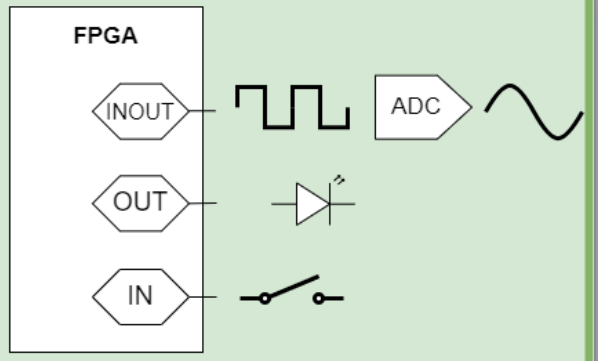
system_bd.tcl :

modules(IPs) and **ports** are added in:

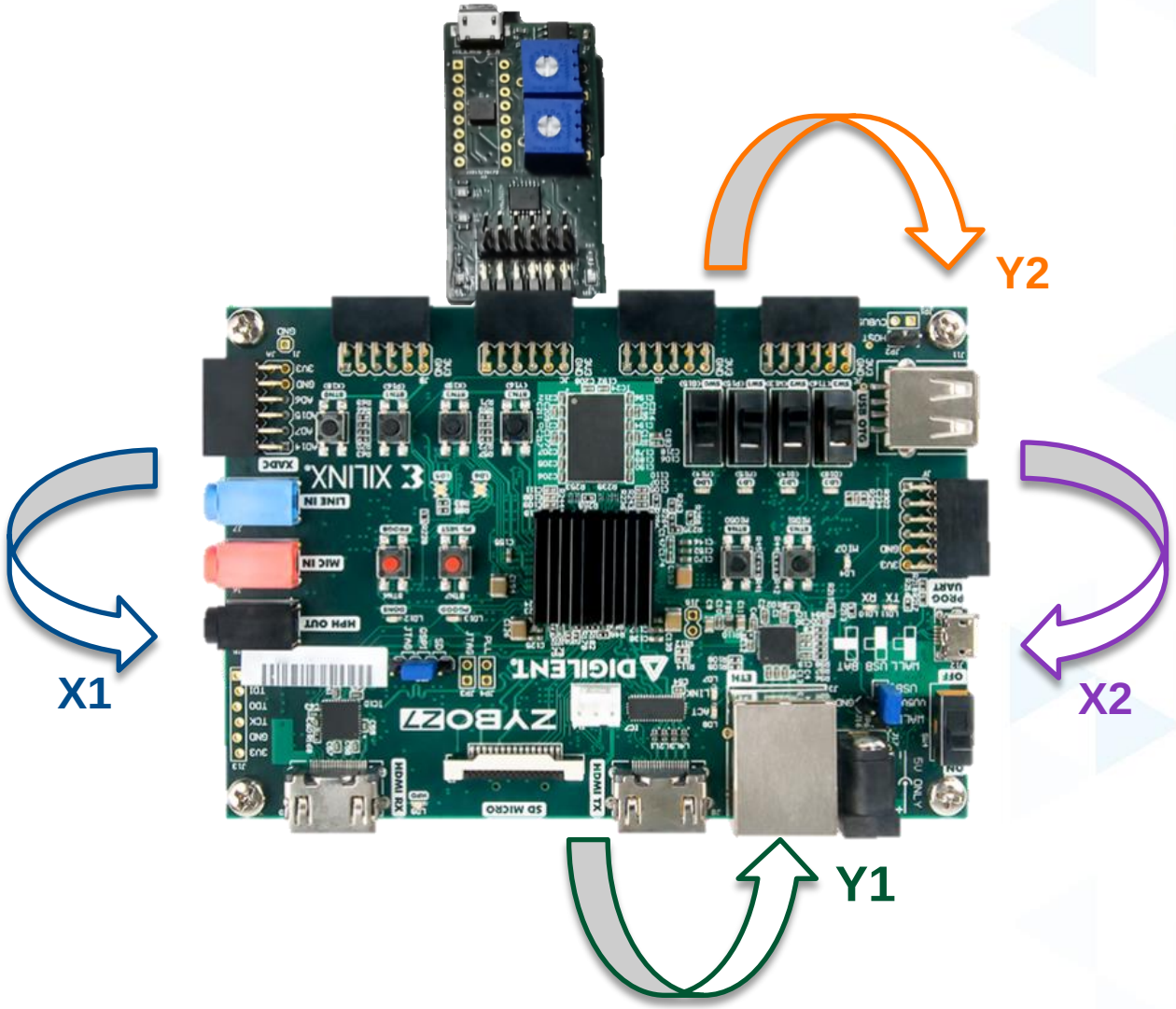
-projects/common/zyboz7/**zyboz7_system_bd.tcl**

-projects/ad5592r_pmdz/common/**ad5592r_pmdz_bd.tcl**



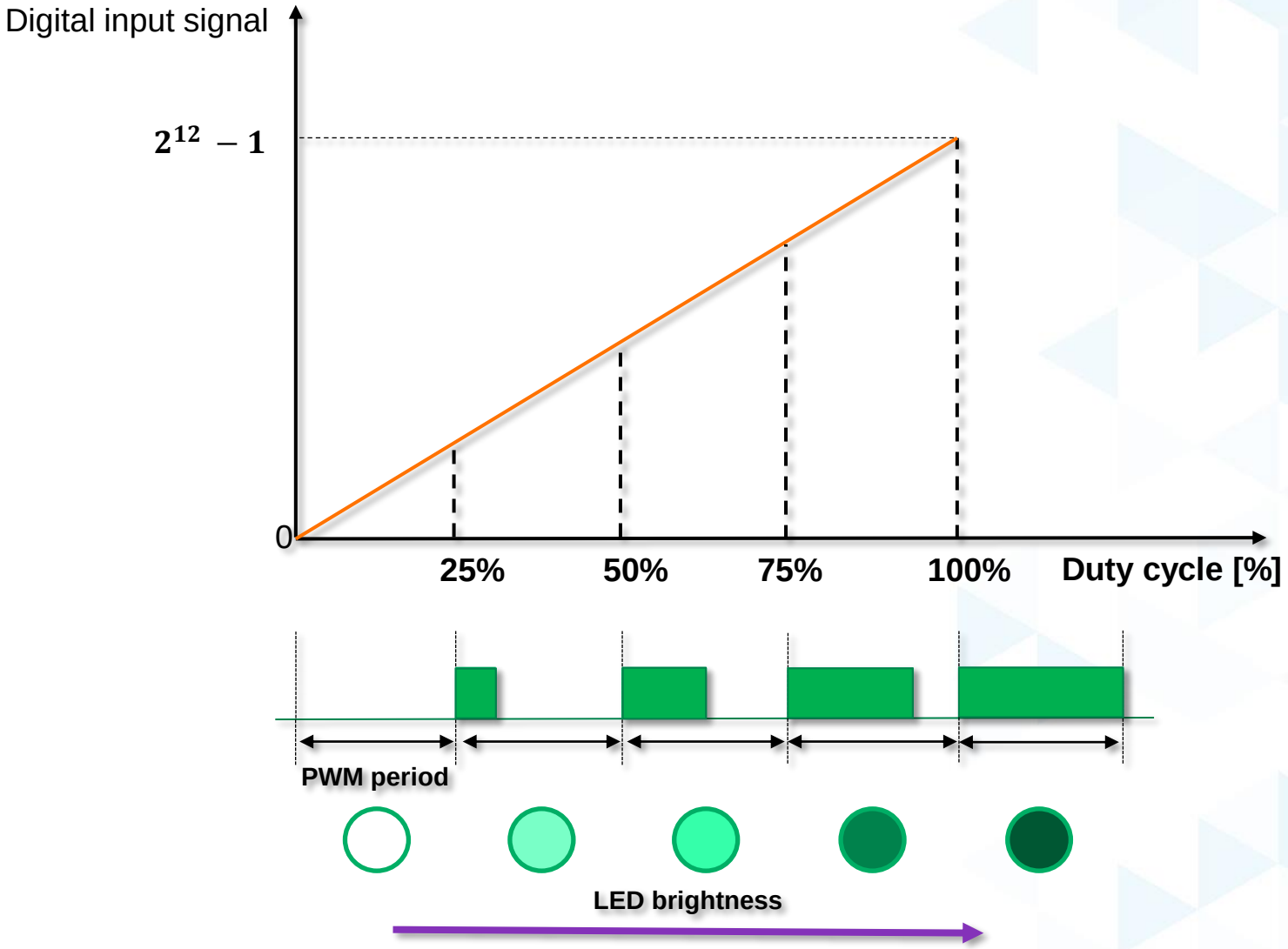


HDL – IP Task



Rotation Axis	Zybo-Z7	ADXL327
X1	LD0	D1
X2	LD1	D2
Y1	LD2	D3
Y2	LD3	D4

HDL – IP Task

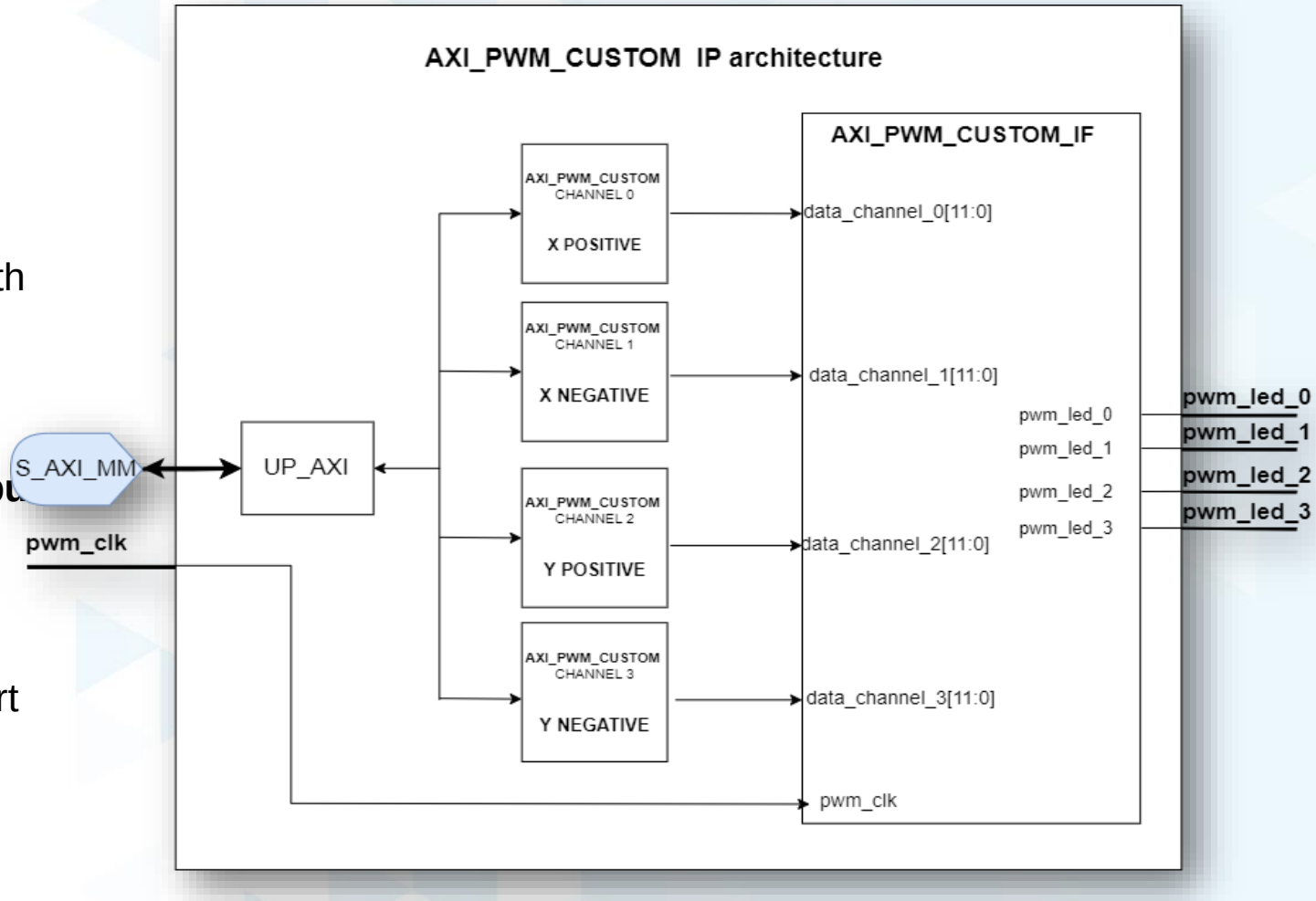


(1) $f_{\text{ref\_clk}} = 100\text{MHz}$

(2) $T_{\text{PWM}} = 2^{12} * T_{\text{ref\_clk}}$

HDL – IP Task

1. Starting from the block design of the IP the **axi_pwm_custom.v** file should be completed with the module instantiation
2. The interface module should generate a PWM signal with **variable duty cycle** based on the **input data magnitude**
3. The data wires should be connected to the **up_adc_channel** modules **conversion_data** port



HDL – Project Task

1. Starting from the schematic of the board, the constraints file is written
    a) For this, the pinout of the board is needed as well as the template for the constraints
2. After identifying the I/Os, the board design of the project will be written
3. The top file will be written with regards to the SPI interface
4. The custom PWM generator should be created in the **library/axi_pwm_led** folder and added in the **projects/ad5592r_pmdz/common/ ad5592r_pmdz_bd.tcl**
5. The Makefile can be written, with the dependencies and the needed sources
6. In order to generate the BOOT.BIN and to program the FPGA, we need all the files previously created, and an **u-boot.elf** and **zynq_fsbl.elf** (which will be provided to you)

HDL – 1. Constraints (1)

- It’s the connection between the physical pins of the FPGA that you want to use and the HDL code that describes the behavior of the FPGA
- Here you define the board specific pins, clocks, etc.
- Contents [1] in order suggested by Xilinx:
  - Timing assertions section:
    - Primary clocks, virtual clocks, generated clocks, clock groups
    - Bus skew constraints
    - Input and output delay constraints
  - Timing exceptions section:
    - False paths, max/min delay, multicycle paths
    - Case analysis, disable timing
  - Physical constraints section:
    - Setting-up variables for easy configuration
    - Pin specifications
  - Learn more here [2]

HDL – 1. Constraints (2)

1. Start by searching for the schematic of the FPGA and additional devices
2. Take the default constraints file and leave uncommented the pins that you will be using
   1) We will need the Pmod Headers **JC** and **JD** for the SPI connections and the **LEDs pins** for the PWM generator. Uncomment those lines and remove the others
   2) These pins have default names, so change the names to be suggestive (the names are the ones specified with get_ports, in curly braces) and close to the ones from the schematic
3. This file will be put at ***hdl/projects/ad5592r_pmdz/zyboz7/system_constr.xdc***

Example of pin constraint:

```
set_property -dict {PACKAGE_PIN Y17 IOSTANDARD LVCMOS33 DIFF_TERM TRUE} [get_ports pin_name]
```

- PACKAGE_PIN: the pin number found in the FPGA's schematic
- IOSTANDARD: defines what Voltage Level your interface operates on and what kind of signaling is;
  - Low-Voltage CMOS transistor is used on this pin with a voltage of 3.3V
  - There are many others in [3]
- DIFF_TERM: internal differential termination that can be activated; used for the signal not to reflect, when having LVDS pairs of signals

HDL – 2. Base/board design file

- **Base design** file of the **carrier**: projects/common/zyboz7/**zyboz7_system_bd.tcl**
- **Board design** file: projects/ad5592r_pmdz/common/**ad5592r_pmdz_bd.tcl**
- **Design** file of the **project**: projects/ad5592r_pmdz/ zyboz7/**system_bd.tcl**

- In all design files (of a project), firstly the base design must be sourced, and then the board design.

Example:

source $ad_hdl_dir/projects/common/ zyboz7/ zyboz7_system_bd.tcl

source ../common/ ad5592r_pmdz_bd.tcl

HDL – Project files

3. Top file

- Top wrapper file, in which the *system_wrapper.v* module is instantiated
- *system_wrapper.v* is a tool generated file and can be found at <project_name>.srcs/sources_1/bd/system/hdl/system_wrapper.v
- The I/O ports of this Verilog module will be connected to actual I/O pads of the FPGA

4. Project design

- This Tcl script is creating the actual Vivado project and runs the synthesis and implementation of the design
- It needs all files used in the project to be specified

5. Makefile

- Contains all the dependencies needed for the project to be built

6. BOOT.BIN

- It's the result (boot image) after building the HDL project with the previously described files
- This will be programmed into the FPGA (through SD card or via JTAG)
- To learn more about the structure, check this*

HDL – Extra resources

- AD Circuits from the Lab videos
- Digital Design and Computer Architecture course – ETH Zürich, with Onur Mutlu (and all his courses from his YouTube page)
- Vivado QuickTake Tutorials
- FPGA design presentations from Intel

HDL – References

- [1] UG903 (v2022.1): *Ordering Your Constraints – Recommended Constraints Sequence*
- [2] UG949 (v2022.1): Defining Timing Constraints – Defining Timing Constraints in Four Steps
- [3] UG471 (v1.10, 2018): Supported I/O Stadards and Terminations