How to use the Project Template

Table of Contents

Introduction

The provided zip, *CreateProject_[date].zip*, file contains:

```
.       (This is usualy the /templates directory)
├─ /VhdlProjects
│   ├─ /FileHeaders              File headers for all created files.
│   ├─ /Library_Name             Library template directory
│   │   ├─ /Documents
│   │   ├─ /SimScripts
│   │   ├─ /Simulation
│   │   └─ Vhdl
│   ├─ /Project_Name             Project template directory
│   │   ├─ /Constraints
│   │   ├─ /Documents
│   │   ├─ /Libraries
│   │   │   ├─ /Common_Lib
│   │   │   ├─ /HierarchyName_Lib
│   │   │   └─ /IpCoreName_Lib
│   │   ├─ /SimScripts
│   │   ├─ /Simulation
│   │   ├─ /Vhdl
│   │   ├─ /Vivado
│   │   └─ /ZipFiles
│   │   README_FIRST.md
│   │   README_FIRST.txt
│   ├─ CreateProject.desktop
│   ├─ CreateProject.sh          Bash script for project creation, discussed below.
│   └─ ProjectTemplateUse.pdf      PDF version of this document
│
```

Each above presented directory contains one or more files. These files are copied and correctly renamed into the project that need or is going to be created. The directories and their files are discussed in this document.

Remark:
All files in this zip file are delivered as is and can be modified to the needs of the user.

Linux Installation:

- Copy the provided zip file in the directory you use to store templates.
    - For Ubuntu or Linux-Mint users the default folder is /home//Templates
    - When storing templates somewhere else, use that folder.
    - When not having a folder for templates create one or use the default, in the home directory created templates directory
- Unzip the contents of the zip file in the chosen *Templates* folder.
- When having unzipped the zip file you get a directory/file structure as showed above.

For Linux do:

This is written, and it works, for Ubuntu - Gnome and Linux-Mint - Cinnamon Linux distributions.
In principle it should work on all systems running “the Bourne Again SHell” (Bash).
When done as in above described “Linux Installation”, continue doing following.

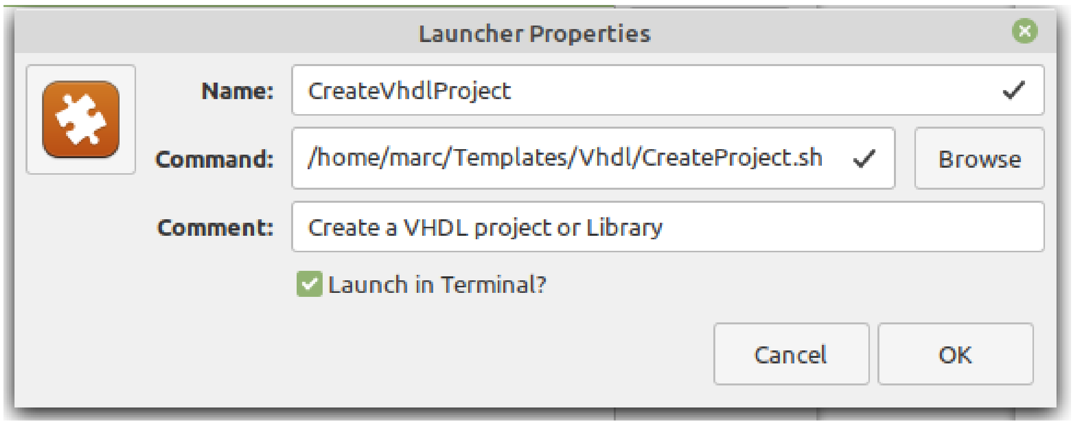
- Copy the *CreateProject.desktop* file onto the desktop or into the `.local/share/applications` directory.
    - Open the file with a text editor and change the word “**user**” in the **Exec=** line in YOUR user home directory.
    - It is also possible that the zip file is unzipped into a complete other directory than the default provided Templates directory in the home directoy. Then make sure the path behind **Exec=** reflects the correct place of the “CreateProject.sh” file
- Make sure both the .desktop and .sh file are executable!

Now it's possible to just double click the icon to create a new project or library with all its directories
and files. Running the *.sh* script will setup a project structure like showed below.

A second way to get easy access to the project generator is *(This works in Linux-Mint)*:

- Right click the Linux-Mint **Menu** icon. By efault located in the left-bottom corner of the screen.
- Select [Configure]
  A new window will pop-up.
- Select in this window, on the top, [Menu]
- Select now the large, in the middle, button [Open the menu editor]
- Select an application, like “Electronics” in the left pane of the new popped window.
- Hit now at the right side the button [New Item]

- Create a new launcher



Launcher Properties

Name: CreateVhdlProject

Command: /home/marc/Templates/Vhdl/CreateProject.sh

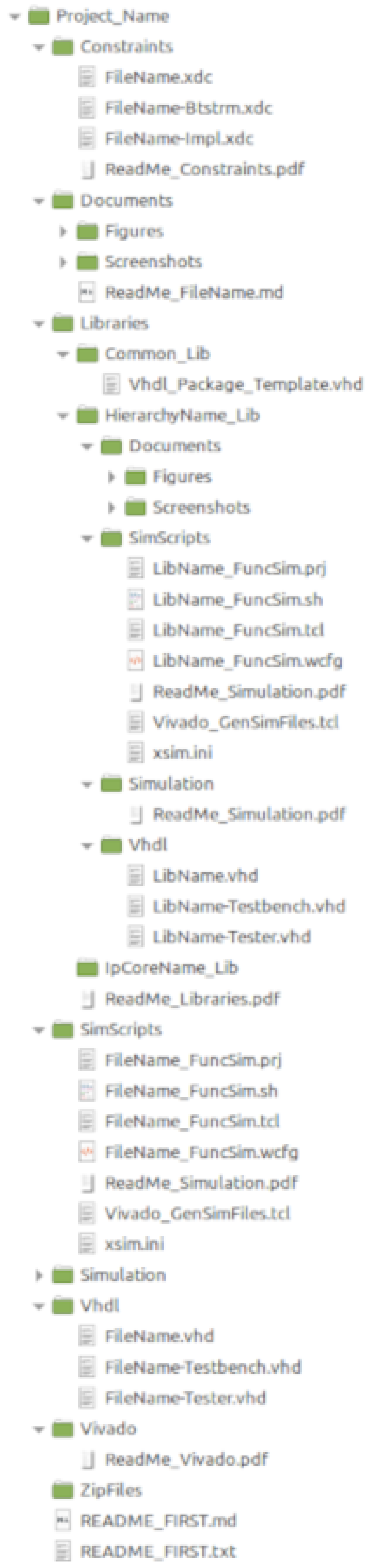
Comment: Create a VHDL project or Library

✓ Launch in Terminal?

Project Setup

This is the layout of the project setup used as base for new projects.
This directory with sub-directories and files is placed (read only) in the templates directory and when a new project is created it’s copied to the new project while all names are updated following the data supplied while the project information was asked.

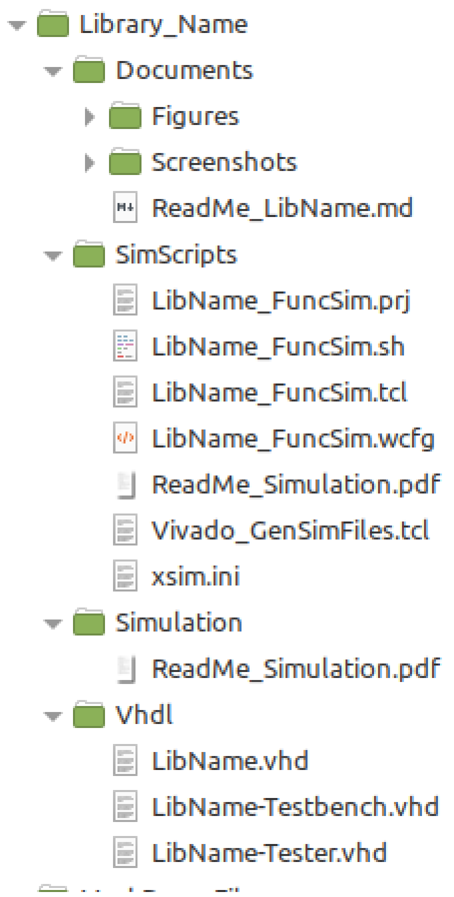
Find more about the project setup in **Project and Library Template Setup** below.



```
Project_Name
    Constraints
        FileName.xdc
        FileName-Btstrm.xdc
        FileName-Impl.xdc
        ReadMe_Constraints.pdf
    Documents
        Figures
        Screenshots
        ReadMe_FileName.md
    Libraries
        Common_Lib
            Vhdl_Package_Template.vhd
        HierarchyName_Lib
            Documents
                Figures
                Screenshots
            SimScripts
                LibName_FuncSim.prj
                LibName_FuncSim.sh
                LibName_FuncSim.tcl
                LibName_FuncSim.wcfg
                ReadMe_Simulation.pdf
                Vivado_GenSimFiles.tcl
                xsim.ini
            Simulation
                ReadMe_Simulation.pdf
            Vhdl
                LibName.vhd
                LibName-Testbench.vhd
                LibName-Tester.vhd
        IpCoreName_Lib
        ReadMe_Libraries.pdf
    SimScripts
        FileName_FuncSim.prj
        FileName_FuncSim.sh
        FileName_FuncSim.tcl
        FileName_FuncSim.wcfg
        ReadMe_Simulation.pdf
        Vivado_GenSimFiles.tcl
        xsim.ini
    Simulation
    Vhdl
        FileName.vhd
        FileName-Testbench.vhd
        FileName-Tester.vhd
    Vivado
        ReadMe_Vivado.pdf
    ZipFiles
    README_FIRST.md
    README_FIRST.txt
```

Library Setup

This is the layout of the project setup used as base for new projects.
This directory, sub-directories and files is placed (read only) in the templates directory.
Find more about the project setup in **Project and Library Template Setup** below.



```
Library_Name
    Documents
        Figures
        Screenshots
        ReadMe_LibName.md
    SimScripts
        LibName_FuncSim.prj
        LibName_FuncSim.sh
        LibName_FuncSim.tcl
        LibName_FuncSim.wcfg
        ReadMe_Simulation.pdf
        Vivado_GenSimFiles.tcl
        xsim.ini
    Simulation
        ReadMe_Simulation.pdf
    Vhdl
        LibName.vhd
        LibName-Testbench.vhd
        LibName-Tester.vhd
```

Using the Linux Script

Click the icon set by *CreateProject.desktop*, chose [CreateVhdlProject] in teh applications menu structure or open a terminal/command window and type:
.\Path\To\The\Project_Template\CreateProject.sh.

This is what the script will show and ask:

Whne the script is started, below text is displayed for a while without possibility to enter any value.
The text tells what's going to happen.

```
==============================================================================
 This script creates a new project or new library to be used with the Vivado
 tool (simulation and implementation).
 The directory setup created by the script assumes that the Vivado simulator
 (xsim) is going to be run in standalone mode. The Vivado tool is used for
 synthesis, inplementation and bit streaam generation.

 Each of the created directories in a project or library has a PDF document
 with instructions and guidelines about how to use the directory, tools and
 files.

 The documentation on how to use or modify this script can be found in the
 directory used to store the script.

 To exit the script at any moment: hit CTRL-C or close the terminal window.
 The file headers add to the different basic files in the created project or
 library are formatted such that they can be used with the 'TerosHDL'
 extension of the 'Visual Studio Code' editor. The 'TerosHDL' extension can
 among a lot of things auto-generate markdown documentation from VHDL files.
==============================================================================
```

Next is a text and question about what must be created:

```
==========================================================================
 Single character answers are case independent (y/n can also be Y/N).
 All other answers are case sensitive (Author, Project , ...).
==========================================================================
 What needs to be created?
  A project = p :
  A library = l :
  or exit the script = e :
==========================================================================
```

Answer **P** when a project need to be generated or answer **L** for a library. The third option is **E** to exit the script. The letters entered to make a selection are not case sensitive.
Remark:
    One can exit the script, at any moment, by hitting CTRL-C.

Assume a project has to be created, answer thus **P**.
When a library, answer **L**, is the correct answer continue reading at: **"Using the Linux Script - Library"**.

```
======================================================================
 Is this the path where the new project needs to be created?
      /home/<user>
======================================================================
 y/n :
```

When chosen to create a new project the question is asked **where** the project **should** be created.
    When answering *yes*, the project will be created later on in the presented directory.
    If the answer is *no*, the next question is asked.

```
 Provide the path of the directory where the new project
 needs to be created. Example: /home/marc/Projects
 Don't use '-' in directory or file names! :
```

Arriving here means that the answer to the previous question was no.
Provide the path to the directory were the new project needs to be created.
The new project will be created as directory with all it's sub-directories and files under the given path.

The path information can be absolute or can be relative. When relative the directory where the script is started is taken as base.

Assume the entered path is: */home/user/Documents*

When the provided path already exist, what mostly will be the case, next is asked.

```
 Directory, /home/marc/Templates, already exists!
 Use (u) the given path anyway or change (c) the path?
 u/c :
```

When using a directory a hoem for all projects, the answer will be u/U because a new project must be created near all other already existing projects. When the answer is c/C the previous question will be asked again.
Assume we answer **u**
The script will confirm the directory name and path that will be used as hoem for the new project and will at the same time ask the name of the new to create project.

```
 new project will be created in /home/marc/Documents
======================================================================
 What is the name of the new project to create?
======================================================================
 :
```

Provide the name of the project to create (like: MyNewProject).
The project will be created as directory with all it's sub-directories and files under the previous given path.

A check on the given project name will be done to detect if the name is not already used by another project. When that is the case, previous question will be asked again.

```
 Checking if the project already exist in the provided path.
/home/marc/Templates/Vhdl/CreateProject.sh: line 163: [-d: command not found

 The project MyNewProject will be created in: /home/marc/Documents

=======================================================================
 Required Project Information

 What is the target family for this project?
 Use , to separate different FPGA used in the project
 Example: Ultrascale, Ultrascale+, Zync-Ultrasacle+   .  .  .  .  :
```

Before the project can be really created in the given directory and with the given name, questions about the used FPGA and tools are asked.

```
=======================================================================
 Required Project Information

 What is the target family for this project?
 Use , to separate different FPGA used in the project
 Example: Ultrascale, Ultrascale+, Zync-Ultrasacle+   .  .  .  .  : 7-Series
 Used Vivado version for the project   .  .  .  .  .  .  .  .  .  :
Vivado_2020.2
 Used Simulator and version (ex: QuestaSim_10.7)   .  .  .  .  .  : Xsim_2020.2
 Name (without extension) of the top level VHDL file? .  .  .  .  : MyTopLevel
 Is the project generated for the one running this script? (y/n)  : n
 What is the author reference for all file headers?           : ZirconfleX
 Company/Vendor name creating this Library?  .  .  .  .  .  .  .  : ZirconfleX
bv
==========================================================================
```

Before creating the project structure on disk, a summary of all given answers is showed.
Answer *y/Y* when all data is correct and let the script generate the project or answer *n/N* wen some or all data is wrong. In this case all questions will be asked again.
The summay looks like:

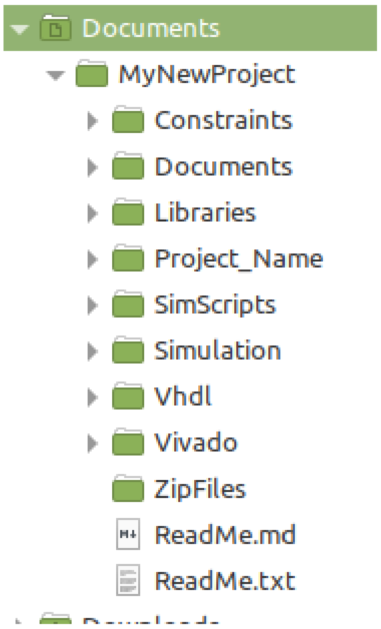
```
===========================================================================
 Summary of Project Information before creating the project:

 This script is stored in:
/home/marc/Templates/Vhdl
 The script is run from:                                    /home/marc
 The project directory is created in:                       /home/marc/Documents
 The name of the new project is:                            MyNewProject
 If
    a project is created it will fit in:
/home/marc/Documents/MyNewProject
    a library is created it fits in:
 The FPGA(s) used in this project are:                      7-Series
 The top level VHDL file is:                                MyTopLevel
 The Vivado version used is:                                Vivado_2020.2
 The Simulator and version used is:                         Xsim_2020.2
 The project or design author is:                           ZirconfleX
 The company/vendor creating the project or Library is:     ZirconfleX bv
 Today is:                                                  29 sep 2021
===========================================================================
 Is the displayed information correct? (y/n) :
```

If the above information is correct answer yes and the project will be created. When answering no the process of project information gathering is started again.
The final output of the script is this:

```
 Create the project.
 Add file headers to the basic template files.
 Name the files.
 Replace the keywords in all files of the created project.
sed: no input files
 Create .mdf project file.
===========================================================================
 Project MyNewProject is created in /home/marc/Documents
===========================================================================
Press enter to continue
```

When hitting [Enter] the command/terminal window will close.
The project is now created and all files are in place, ready to be used.



Using the Linux Script - Library

A library is normally created in teh */Libraries* directory of an existing project.
Using the script to do this is explained here.

Click the icon set by *CreateProject.desktop*, chose [CreateVhdlProject] in teh applications menu structure or open a terminal/command window and type:
.\Path\To\The\Project_Template\CreateProject.sh.
As when a project is created, the script will run in a terminal/command window and display at the start the same text as is displayed when a project is created.
The question wether a project, **p** or a library **l** must be create is asked.

Answer **L**

The script now asks for the path where to create the new library

```
=====================================================================
 A new library can be created in a existing project or library.
 Provide the path to the root directory of an existing project
 in '/home/marc' or to an existing library in a exitsting
 project under '/home/marc'.
 Examples:
  /home/marc/<Projects_Path>/<ProjectName>/
  /home/marc/<Projects_Path>/<ProjectName>/Libraries/
  /home/marc/<Projects_Path>/<ProjectName>/Libraries/<LibraryName>/
 DO NOT use '-' in directory or file names!
=====================================================================
 :
```

Since a new project is just created, let’s add a new library to it.
Libraries are stored in teh */Libraries* directory of a project. The script will check if a Libraries directory is available in teh given path. When there is a */Libraries* directory, that directory will be used as home for the new library. When where is no */Libraries* directory, one will be created.

The answer will thus be: `/home/user/Documents/MyNewProject` because the new library should fit into the existing */Libraries* directory.

The script outcome looks as:

```
 A check in '/home/marc/Documents/MyNewProject' for a 'Libraries' directory

 will be done. If there is no 'Libraries' directory in the
 provided project path one will be created.

 Libraries directory is available.
======================================================================
 What is the name of the library that needs to be created?
 REMARK: The entered name will be extended with '_Lib'
======================================================================
 :
```

In case there is no */Libraries* directory or the */Libraries* directory is included (by accident) in teh path where the library should be created the script return is: *(The given path was: /home/user/Documents/MyNewProject/Libraries)*

```
 A check in '/home/marc/Documents/MyNewProject/Libraries' for a 'Libraries'
directory
 will be done. If there is no 'Libraries' directory in the
 provided project path one will be created.

 Libraries directory is available.
======================================================================
 What is the name of the library that needs to be created?
 REMARK: The entered name will be extended with '_Lib'
======================================================================
```

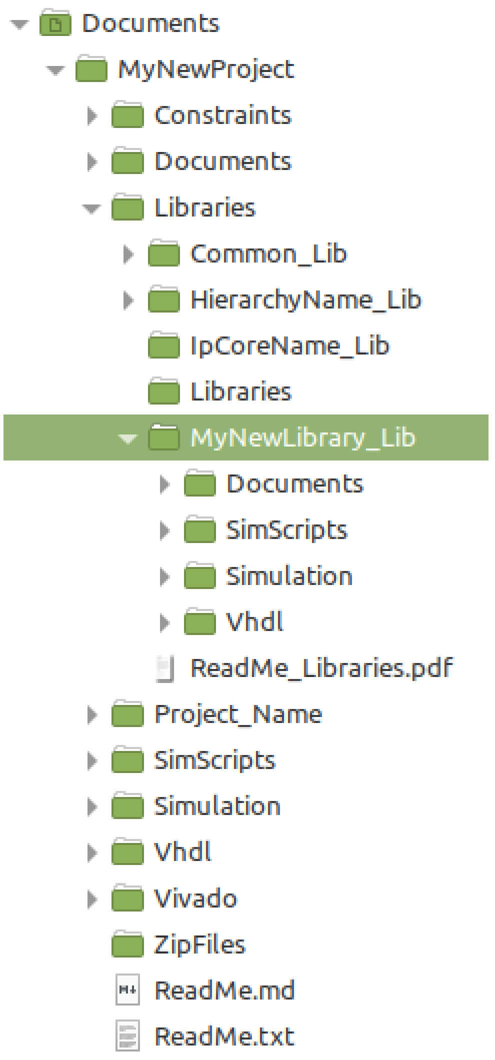
 before creating the library, the script needs to know the name o the library.
The project rules, like they are set now, require that a library directory has a name ending in *_Lib*. As the script also tells, provide the name of teh library **without** _Lib. The _Lib is add automatically by the script. The answer to above script question can be *MyNewLibrary*

All questions about used FPGA family, tools, author, and ... are asked again. The questions are asked to autofill the file headers in teh VHDL and other files. Every answer is refected back to teh script user and when evrything is correct the user should hit **y/Y**.

The library is created and the script returns following text:

```
 Create the library:
 Add file headers to the basic template files.
 Name the files.
 Replace the keywords in all files of the created project.
sed: no input files
==========================================================================
 Library MyNewLibrary_Lib is created in
/home/marc/Documents/MyNewProject/Libraries
==========================================================================
Press enter to continue
```

Hit [Enter] to close the terminal/command window and go to teh project location to check the creating of the new library



```
Documents
    MyNewProject
        Constraints
        Documents
        Libraries
            Common_Lib
            HierarchyName_Lib
            IpCoreName_Lib
            Libraries
            MyNewLibrary_Lib
                Documents
                SimScripts
                Simulation
                Vhdl
            ReadMe_Libraries.pdf
        Project_Name
        SimScripts
        Simulation
        Vhdl
        Vivado
        ZipFiles
        ReadMe.md
        ReadMe.txt
```

Project and Library Template Setup.

The script file copies one of the directories, Library_name or Project_name, including sub-directories and files to a new given, in the script file, directory. The script file then renames the whole directory and file tree and changes keywords in the files so that headers and other settings reflect the correct names, dates and used tools.
If wanted one can modify the setup of the *Project_Name* and/or *Library_Name* directory, respecting the names of directories and files, to reflect a different project setup.

- FileHeaders
  This directory contains the text of the legal headers visible in VHDL and other text files.
  One can adjust, adapt these files to his/her needs.
- Project_Name
  Root directory of the project.
  This will be created in the location given at the Project path question.
  This will get the name of requested Project Name.
  The script will create all sub-directories and necessary files.
  - Constraints
    Shows three base XDC files; one for bit stream constraints
    one for implementation constraints and one for global design
    constraints (timing, pin lock, ...)

- Documents
  This folder is the home of all documentation of the project.
  The *Figures* sub-directory hold figures used in the documentation
  Markdown text file. The *Screenshots* directory is there to keep
  screenshots all kind. Give figures and screenshots a meaningful name
- Libraries
  Each hierarchical level in the design is represented as a library.
  This allows easy re-use of previous or parts of previous projects.
  A Library directory can thus contain a full project, used in this project
  as instantiated component in the top level or as instantiated component
  in a sub-level. The minimum a library must contain is a VHDL or verilog
  source code file.

  IP cores are created as libraries because they act in the design is hierarchical
  levels. As with normal hierarchical levels, this setup makes it easy to re-use
  previous created IP cores. Read the **ReadMe_Libraries.pdf** on how to create
  IP-cores and get them in a library
- SimScripts
  Folder containing .do files to run with QuestaSim.
  The *.do* files are in fact TCL files that control the simulator.
  Functional simulation uses VHDL testbench and tester files in the /VHDL directory.
  All simulation is pure VHDL based.

  Timing simulation uses TCL scripts to run. This is done because for a timing simulation
  the design is implemented a FPGA and the FPGA is simulated as some kind of back
  box with inputs and outputs.
  Read the **ReadMe_Simulation.pdf** to grasp how simulations can be run from here.
  The *transcript* file is kind of a simulator log file.
  The *Vivado_GenSimFiles.tcl* is a TCL file that need to be run from the Vivado-TCL
  command line. It generates the necessary files for timing simulation. The generated
  files are dropped in the *Simulation* directory.
- Simulation
  Directory used by the simulator for functional and timing simulations.
  The .do and Vivado_GenSimFiles craete files that are stored in this directory.
- Vhdl
  This directory holds the top level or top levels ofthe design and project.
  The directory also holds the functional simulation testbench and tester files.
- Vivado
  This is the directory holding the Vivado implementation project of the project.
  Read the **ReadMe_Vivado.pdf** document on how to proceed.
- ZipFiles
  Any zip files having some kind of band/bond with the project go here.
  Example: as version of the project to be kept/archived can be stored here.