Simulation starts here.

Simulation is run using Xilinx Vivado Xsim and the how, what and where is what is described in this document. Using a different simulator is possible, it probably will require the same steps as described in this document but most likely other command/scripts/TCL files.
REMARK:

> The best, most complete and comprehensive simulation of the design is obtained
> by implementing a minimal version of the design and control it through the VIO while gathering visual, wave forms, information in a ILA.

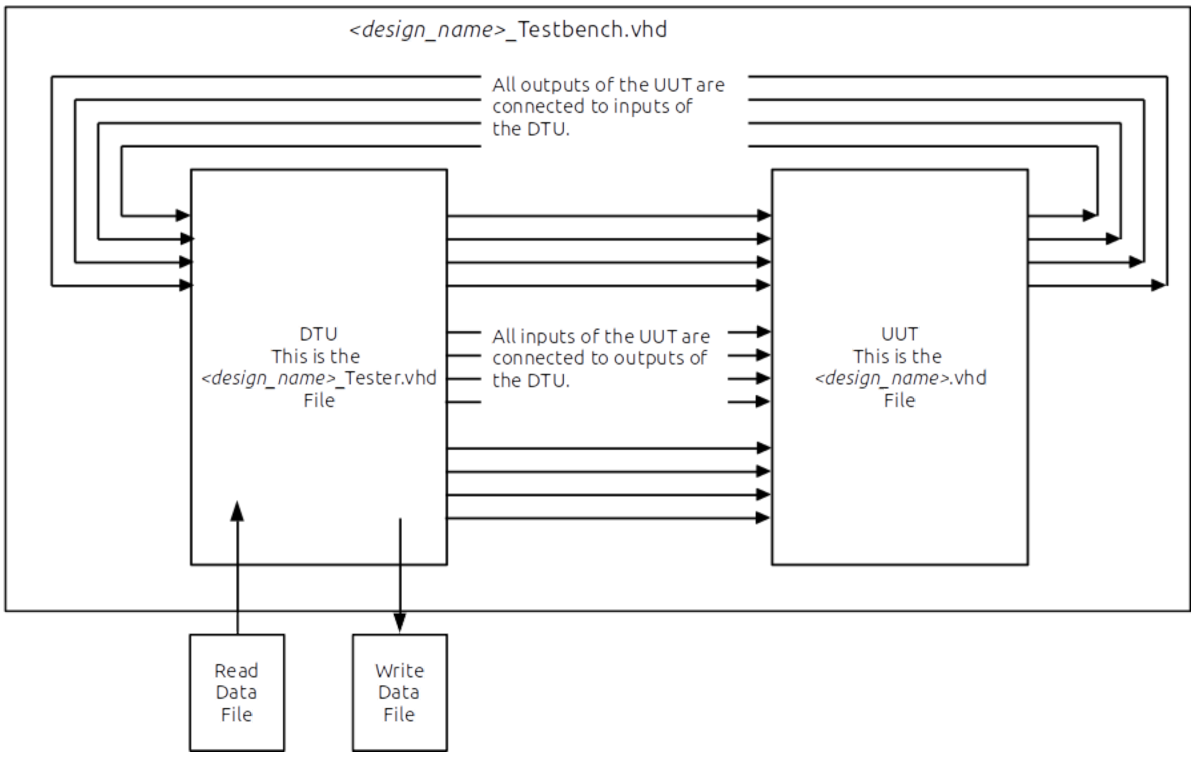
Functional Simulation.

Xsim

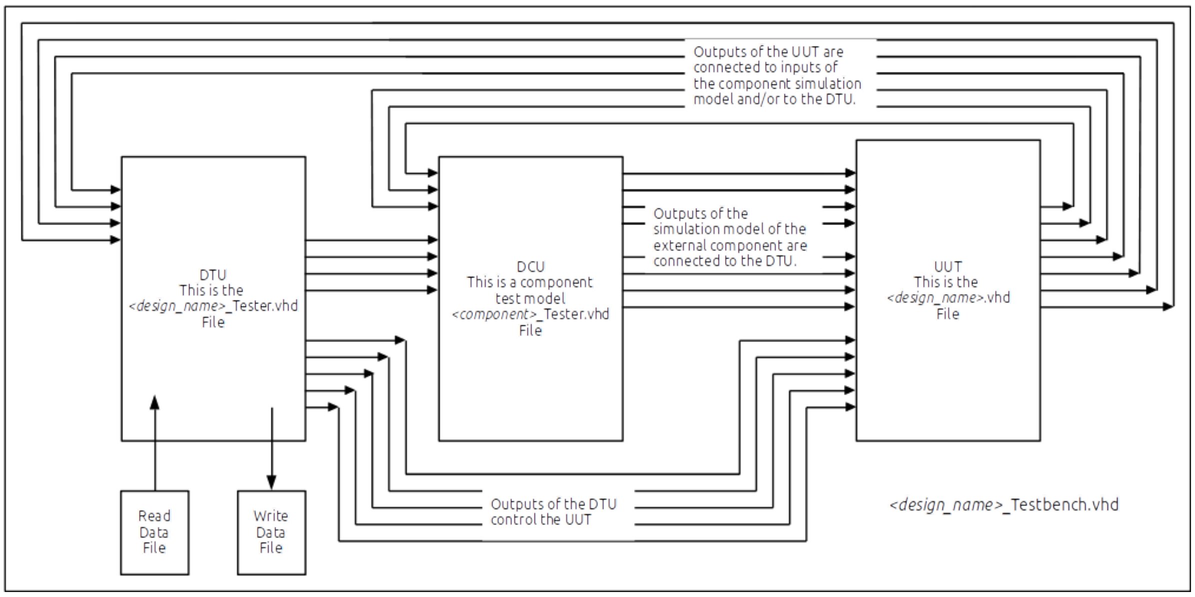
Xsim is part of the Xilinx Vivado software tools and as such it can be launched from within Vivado. Start Vivado, setup a project, insert, write, copy, synthesise and/or implement HDL and/or constraints files. Then to simulate (behavioural, functional after synthesis or timing) the design, hit the Xsim button in the Vivado cockpit.
This is one way to use and work with the Xsim simulator.
When running the tools that way the entire design control is left over to the Xilinx Vivado tool and be left over at the mercy of the Xilinx software developers and their idea how projects should be handled. This is, when browsing though such a created project, not the most user friendly and understandable way of human project handling.
If you want to run Xsim the way you or your company thinks projects should be setup and run, follow the guideline below. If you wan to run Xsim and projects a human understandable way follow the guide below.

Detailed Description of a simulation setup

Functional simulation uses the HDL source level code instantiated in a HDL test bench file together with at least one HDL stimulus source file and eventually data and or text files. In shorter wordings: Simulation is/can be done completely in VHDL. View below two examples of a simulation test bench setup:



<design_name>_Testbench.vhd
All outputs of the UUT are connected to inputs of the DTU.
DTU
This is the
<design_name>_Tester.vhd
File
All inputs of the UUT are connected to outputs of the DTU.
UUT
This is the
<design_name>.vhd
File
Read Data File
Write Data File



Outputs of the UUT are connected to inputs of the component simulation model and/or to the DTU.
DTU
This is the
<design_name>_Tester.vhd
File
DCU
This is a component test model
<component>_Tester.vhd
File
Outputs of the simulation model of the external component are connected to the DTU.
UUT
This is the
<design_name>.vhd
File
Read Data File
Write Data File
Outputs of the DTU control the UUT
<design_name>_Testbench.vhd

The top level of the design (UUT) is tested using a VHDL stimulus file (DTU) generating stimuli for the UUT and collecting data from the UUT. Generation of stimuli can use an external data file while collecting results can write data into an external text file. All is called from within a top level test bench. Waveforms from the simulator can easily be displayed, and all signals of both UUT and DTU can easily be made visible.

Bend Xsim to your will

Each HDL project and/or library is setup so that it contains, beside other directories, a **Simulation** and a **SimScripts** directory. The Simulation directory contains files generated or created by the simulator and waveform viewer. The SimScripts directory contains all scripts and other files in order to run a simulation.

Each HDL project and or library has a Vhdl and/or Verilog directory where the top level source of the project or library element is stored and were also the;

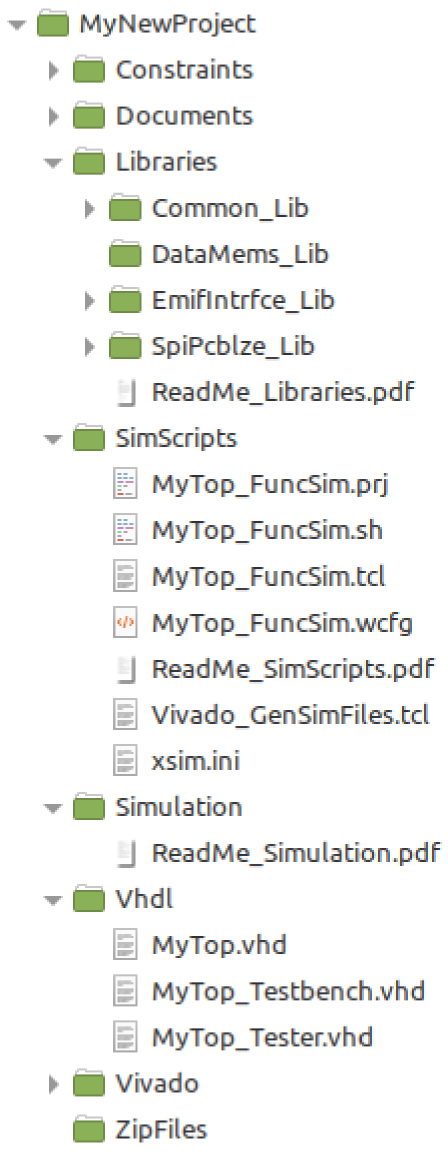
- *{design_name}.vhd*
- *{design_name}_Tester.vhd*
- Optional *{component_name}*_Tester.vhd

and

- *{design_name}*_Testbench.vhd

files are stored.

The setup of a project is shown below:



```
MyNewProject
    Constraints
    Documents
    Libraries
        Common_Lib
        DataMems_Lib
        EmifIntrfce_Lib
        SpiPcblze_Lib
        ReadMe_Libraries.pdf
    SimScripts
        MyTop_FuncSim.prj
        MyTop_FuncSim.sh
        MyTop_FuncSim.tcl
        MyTop_FuncSim.wcfg
        ReadMe_SimScripts.pdf
        Vivado_GenSimFiles.tcl
        xsim.ini
    Simulation
        ReadMe_Simulation.pdf
    Vhdl
        MyTop.vhd
        MyTop_Testbench.vhd
        MyTop_Tester.vhd
    Vivado
    ZipFiles
```

SimScript Files:

The /SimScripts directory should contain below files.
The .txt files are optional as those are the files used to read data from and write data to.

> /SimScripts
> |-- {name}_DataFile.txt
> |-- Result_{name}File.txt
> |-- {design name}_FuncSim.prj
> |-- {design name}_FuncSim.sh
> |-- {design name}_FuncSim.tcl
> |-- {design name}_FuncSim.wcfg
> |-- Vivado_GenSimFiles.tcl
> |-- xsim.ini

{name}_DataFile.txt *(optional)* and Result{name}_File.txt *(optional)*

Files loaded with all sorts of data that can be read and or written by the simulation stimulus {design name}_Tester.vhd file. This file contains the data used by the simulation to provide valid input to the design under simulation.

{design name}_FuncSim.prj

Use this file to create a compilation setup of the project.
This file contains all source code, used libraries and other in the order it needs to be compiled by xvhdl or xvlog. The syntax of the file is:

- Comments are expressed when line starts with a '**#**'.
- verilog <work_library> <file_names>... [-d]...[-i <include_path>]...
  vhdl <work_library> <file_name>
  vhdl2008 <work_library> <file_name>
    - <work_library>: Is the library into which the HDL files on the given line are to be compiled.
      This library is created as sub-directory in an xsim.dir directory created by xvhdl.
      In the flow used here, that xsim.dir directory is created as sub-directory in the Simulation directory
    - <file_name>: Is a VHDL or Verilog source file; specify only one VHDL file per line.

Example:

> vhdl work \
> "../Vhdl/{design name}.vhd" \
> "../Vhdl/{design name}*Tester.vhd" \
> "../Vhdl/{design name}*Testbench.vhd" \
> Do not sort the compile order
> nosort

The `nosort` option mean that xvhdl will handle the files as they are presented in the {design name}_FuncSim.prj file.

{design name}_FuncSim.sh

This is the script handling all simulation items.
It is written so that it can be used, re-used, copied, or other into all possible designs.
Use this script in a terminal and as:

> {design name}_FuncSim.sh [-help]
>   -- Display help information for this script in the terminal.
> {design name}_FuncSim.sh [-compile]
>   -- Compile only of the design files to simulate.
> {design name}_FuncSim.sh [-elaborate]
>   -- Elaborate only the result of the compile step.
> {design name}_FuncSim.sh [-simulate]
>   -- Simulate the with elaborate created snapshot of the design.
> {design name}_FuncSim.sh [-all]
>   -- run compile, elaborate and simulate.
> {design name}_FuncSim.sh [-reset_run]
>   -- Remove files from the previous runs and set everything up for a clean new run.
> {design name}_FuncSim.sh [-noclean_files]
>   -- Reset previous run, but do not remove simulator files from the previous run.

{design name}_FuncSim.tcl

File containing tcl simulator commands.
To activate thsi file in a simulation use the `-tclbatch {filename}` command option with xsim.
When the xsim command gets the option -tclbatch, the filename given with this command option is executed and xsim runs tcl commands.

As example, this file can have following tcl commands:
  run 20ns to make the simulator when invoked run 20ns.

{design name}_FuncSim.wcfg

This is the file being the home of all waveform window settings (colors, column widths, signals, busses, ...).
The file is copied to the */Simulations* directory before it is used.
THUS! : Do not delet the *{design name}_FuncSim.wcfg* in teh */Simulation* directory!

Vivado_GenSimFiles.tcl

This file is not use for or during simulation!
It can be source from withing the tcl console of Vivado and is used to create timing aware vhdl files of an implemented design.
Thus: Source this file from the Vivado tcl console after place and route of a design to create files that can be used by the simulator for performing a timing simulation.

xsim.ini

The HDL compile programs, xvhdl, xvlog, and xelab, use the xsim.ini configuration file to find the definitions and physical locations of VHDL and Verilog logical libraries.
The compilers attempt to read xsim.ini from these locations in the following order:

1. xsim.ini in current working directory
2. User-file specified through the -initfile switch. If -initfile is not specified, the program searches for xsim.ini in the current working directory
3. <Vivado_Install_Dir>/data/xsim

In short; This file tells where XSIM can find libraries and where it needs to generate libraries.
Find libraries of VHDL, Verilog, Xilinx Primitives and IP-cores.
Generate libraries for the simulation in progress.

REMARK:

1. Make sure there is always a xsim.ini file in the current working directory (the directory from where the {design name}_FuncSim.sh script is invoked).
2. Make sure that the xsim.ini file contains all libraries that are going to be used and also all libraries that need to be generated by the different compiler tools.
3. The xelab tool is using, and you cannot change this, always a directory xsim.dir. When it's not present it will be generated. xvhdl and/or xvlog generate libraries specified in the xsim.ini file. Make sure those libarries end up in the xsim.dir directory to get a clean and understandable simulation setup.

HDL source code files

> /Vhdl
> |-- {design name}.vhd
> |-- {design name}*Testbench.vhd*
> |-- *{design name}*Tester.vhd
> |-- {component name}*Tester.vhd* ***

> |-- *{design name}*Checker.vhd ***
> *** = optional

{design name}_TestBench.vhd

To run the custom design in simulation, one needs to have a simulation test bench.
The simulation test bench must reflect the behaviour of the external world to the real design.
In case this test bench is created for the top level of the design add `_Top_` to the file name;
like `{design name}_Top_Testbench.vhd`
This test bench contains minimal two components:

- The design, called when used Unit Under Test (UUT)
- The tester of the design, called when used Design Test Unit (DTU)

{design name}_Tester.vhd

This file is the Design Test Unit (DTU and contains the stimuli that needs to be applied to the design under test (UUT) and/or other stimuli files as simulation models for external components, it controls inputs and outputs of the design under test and/or used component models.

{design name}.vhd

This is the design or part of the design that will be implemented in the FPGA.
It is instantiated in the test bench as Unit Under Test (UUT).

{component name}_Tester.vhd *(optional)*

It is possible that the {design name}.vhd that needs to be implemented in the FPGA is connected to an external component as a ADC, DAC, sensor or other component. A simulation model for that external component can be created, in most cases using the components data sheet. For a valid simulation the HDL model of the external component can be used and is called as {component name}_Tester.vhd.

{design name}_Checker.vhd *(optional)*

When a design becomes large it is possibly a good idea to split the {design name}*Tester.vhd in a part the controls the inputs of the design under test and a part gathering the data from the outputs of the design under test. The part gathering data from the design under test can be named {design name}*Checker.vhd

REMARK:
It is always possible that there are multiple versions of one or more file types used in the same simulation setup.

Timing Simulation.

Vivado_GenSimFiles.tcl

- Vivado must have successfully finished design synthesis and implementation.
- Switch in Vivado to the TCL command window.
- Type on the command line:
  - cd ../SimScripts.
  - source Vivado_GenSimFiles.tcl.
    - Before doing this, modify the tcl file with a text editor to generate the correct timing files for the simulation. Timing following SLOW or FAST corner timing models.

        - To do this enable/disable one line or the other.
    - Vivado will now generate two files in the /Simulation directory.
        - Time_Impl.sdf : The file containing all timing information.
        - Time_Impl.v : Verilog source generated from implemented design.
- REMARK: Timing simulations can only run in verilog.

-

REMARK_1:

- If a simulation contains verilog coded files, then: to get simulation working a special file must be included in the compiler file list of the **_FuncSim** file.
- Make sure all necessary Xilinx Vivado environments are set. Example below:

> # Xilinx Tools
> #. /opt/Xilinx/Vivado/$XILINX_VERSION/settings64.sh
> export XILINX_VERSION=2020.1
> #
> export PATH=$PATH:/opt/Xilinx/DocNav
> export PATH=$PATH:/opt/Xilinx/Vivado/$XILINX_VERSION/bin
> export XILINX_VIVADO=/opt/Xilinx/Vivado/$XILINX_VERSION
> export RDI_DATADIR=/XILINX_VIVADO/data

    - On stand alone Linux systems it is possible that the "XILINX_VIVADO" environment variable
      is set when the Vivado tools are started, but that it's unset when the Vivado tools are closed or not running.
    - On networked Linux systems where the Vivado software is installed central, it is very likely
      that the environment "XILINX_VIVADO" is not set at all on a user machine or as in previous
      point the environment variable is set when the Vivado tools are started/running.
- In both above cases it might be a good solution to find out where and what Vivado tools are used and then leave a hard coded path to the "glbl.v" library.

REMARK_2:

- Xsim runs default in a 'ns' setup. Simulation allows that a clock is modified
  with a deviation on/of the normal clock cycle. The deviation can be specified in 'fs'.
  In order to let the waveform window display the deviation of the clock the simulation
  must be run and displayed in 'fs'.

REMARK_3:

- Xsim uses its own pre-compiled set of the Xilinx component/primitive libraries. It is thus not necessary, as is with other simulators, to pre-compile all used Xilinx component libraries for simulation.

REMARK_4:

- Whenever changes are done to any of the VHDL files, the syntheses run for simulation must be redone.
- ## REMARK_5:
- All scripts invoking Xilinx tools create log files in the Simulation directory.
  When errors or warnings appear in the terminal window, go check the log files for their meaning.