RANDOM BIT FLIPS

HOME ABOUT

December 6, 2017

Installing Xilinx Vivado (2016.4) and Intel Modelsim Starter Edition (16.1) on 64-bit Ubuntu (16.10)

The problem with Linux

So you've got your new FPGA board(s) and want to setup a nice and clean development environment but you got sick of Windows and its terrible command line and want access to a multitude of freely available scientific/engineering tools. Moreover most researchers/hobbyists distribute their work mainly for Linux, plus all major EDA vendors these days support Linux as well. This looks like the perfect OS for your FPGA projects!

The problem is, even in 2017 installing tools in Linux is nowhere near as smooth as it is in Windows. Linux is by nature fragmented, different distributions have different kernels, libraries, desktop environments, etc. and on top of that users may configure their operating system in a myriad of different ways. EDA vendors cannot support every single configuration, only a handful of distributions are officially supported - usually the enterprise editions (RedHat, Suse) plus a few stable/major ones (eg. CentOS) - and only specific releases.

Xilinx Vivado v2016.4 **officially supports the following distributions**:

- Red Hat Enterprise Workstation/Server 7.1 and 7.2 (64-bit)
- Red Hat Enterprise Workstation 6.7 and 6.8 (64-bit)
- Red Hat Enterprise Workstation 5.11 (64-bit)
- SUSE Linux Enterprise 11.4 and 12.1 (64-bit)
- Cent OS 6.8 (64-bit)
- Ubuntu Linux 16.04 LTS (64-bit)

while Intel Modelsim **officially only supports**

- Red Hat Enterprise Linux 5.10 (64-bit)
- Red Hat Enterprise Linux 6.5 (64-bit)
- Red Hat Enterprise Linux 7.2 (64-bit)

Of course *officially*, doesn't mean they can't be installed in other distributions provided we have the right setup. There are various sources on the web where people have successfully installed those tools in Ubuntu [1] [2] [3].

However those solutions didn't work for me out of the box, probably because they refer to different tool/OS versions. I managed to make it work using various bits and pieces from different guides/forums. The following guide is for:

- Ubuntu 16.10 64-bit (yakkety)
- Xilinx Vivado 2016.4 WebPack
- ModelSim-Intel FPGA Starter Edition 16.1*

* actually Modelsim version is **10.5b**, 16.1 is the version of the complete Intel Quartus Prime Standard Edition suite

Install Xilinx Vivado

1) Go to https://www.xilinx.com/support/download.html and download the linux self-extracting web installer. You need to create a free Xilinx account first (if you don't already have one), but it will prompt you so

2) Open a terminal in the folder you downloaded the file and run

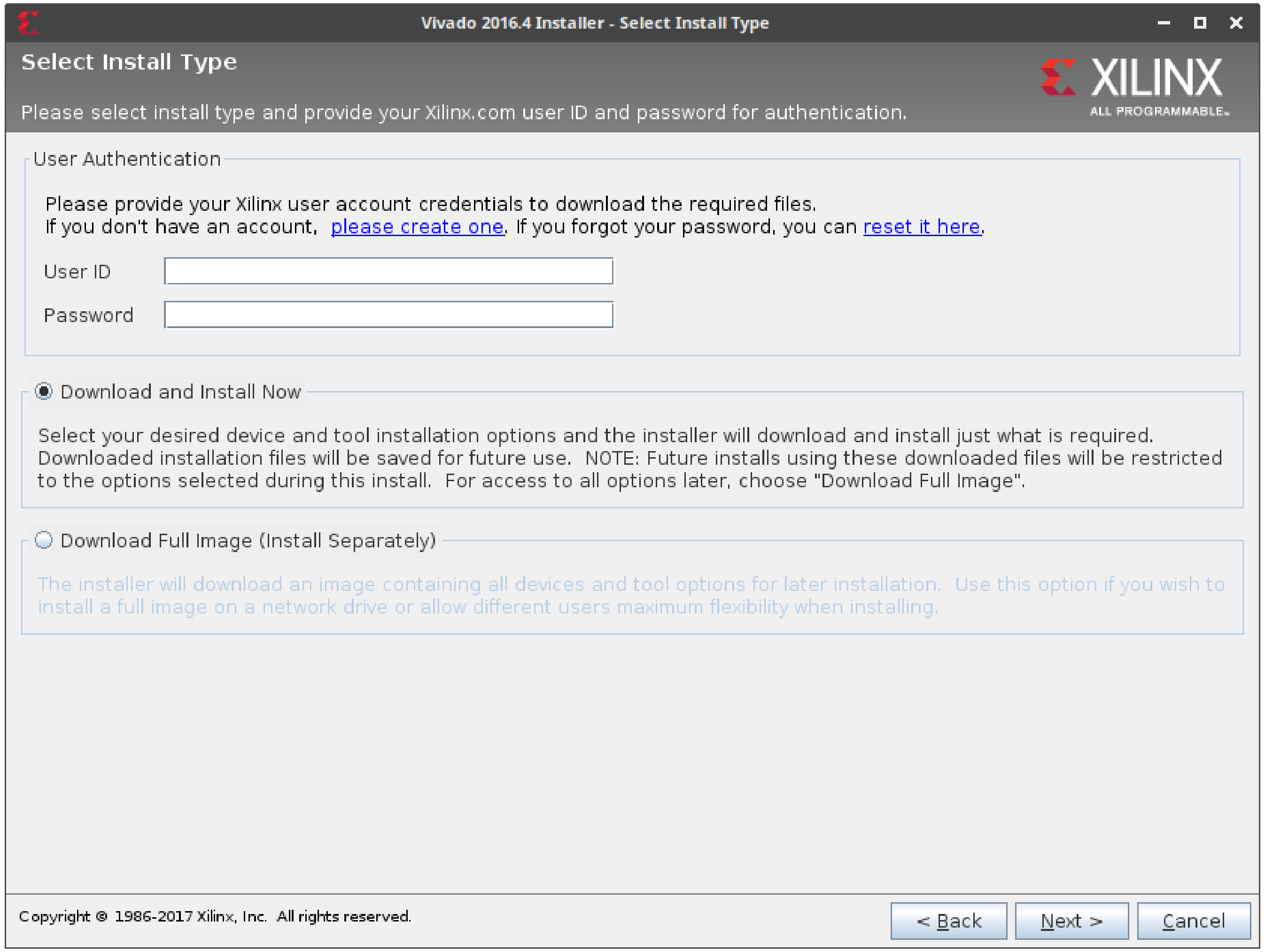
```
$ chmod +x Xilinx_Vivado_SDK_2016.4_0124_1_Lin64.bin
```

to make it executable

3) If you install Vivado in the default path (**/opt/Xilinx**) you need to run the installer with root privileges. That's not necessary if you install in your *home* directory. I personally install my tools in /opt in my home PC (single user environment) so I need to launch the installer with `sudo`

```
$ sudo ./Xilinx_Vivado_SDK_2016.4_0124_1_Lin64.bin
```

4) In the first screen click next. In the second screen enter your Xilinx user ID and password, select Download and Install Now and click next.



Select Install Type

Please select install type and provide your Xilinx.com user ID and password for authentication.

User Authentication

Please provide your Xilinx user account credentials to download the required files.
If you don't have an account, please create one. If you forgot your password, you can reset it here.

User ID

Password

[✓ Download and Install Now]

Select your desired device and tool installation options and the installer will download and install just what is required. Downloaded installation files will be saved for future use. NOTE: Future installs using these downloaded files will be restricted to the options selected during this install. For access to all options later, choose "Download Full Image".

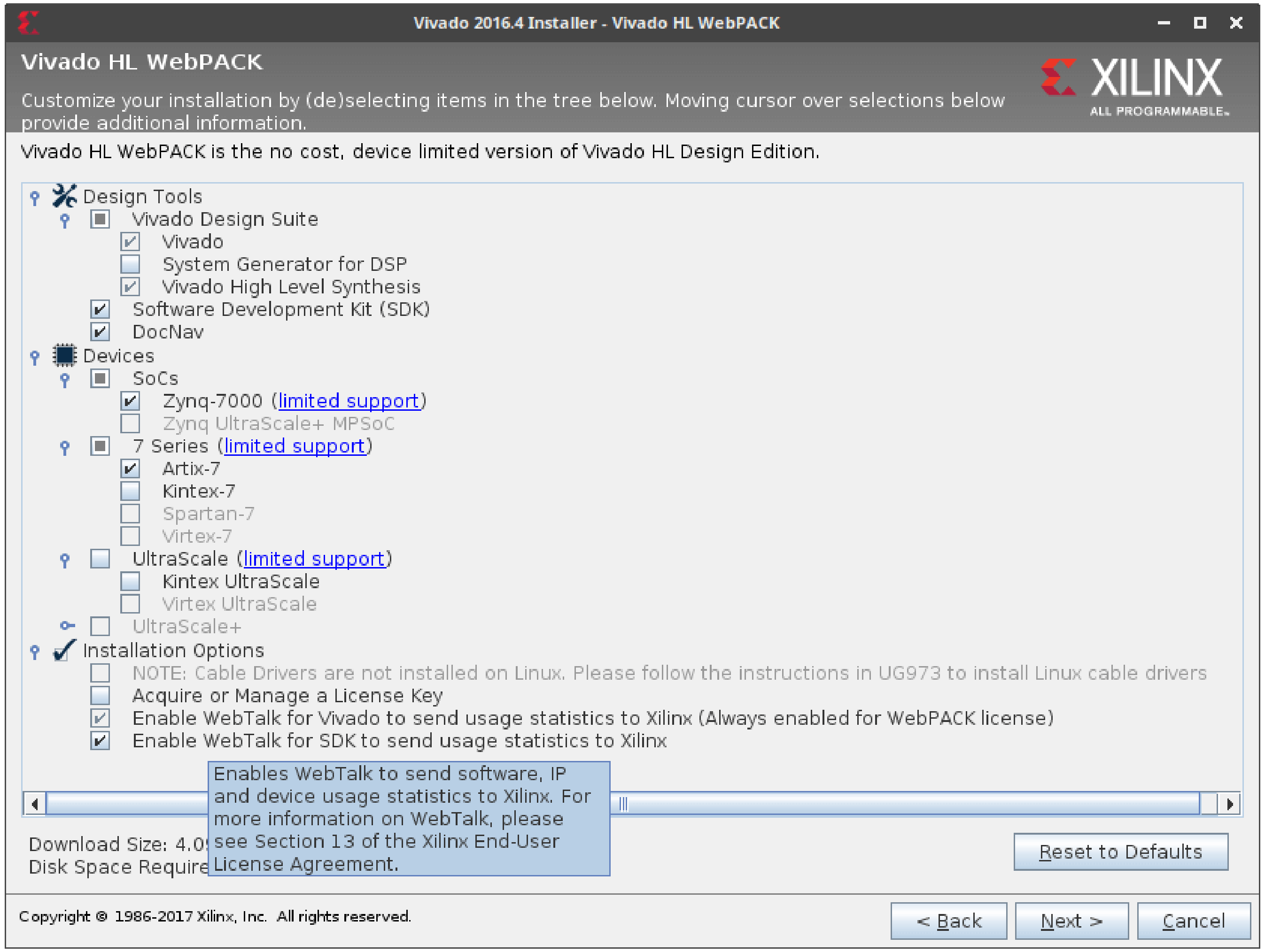
Download Full Image (Install Separately)

The installer will download an image containing all devices and tool options for later installation. Use this option if you wish to install a full image on a network drive or allow different users maximum flexibility when installing.

Copyright © 1986-2017 Xilinx, Inc. All rights reserved.

In the third screen check all boxes and click next. In the following screen select Vivado HL WebPACK and click next

5) The fifth screen lets you select the devices and tools you want to install. Personally I only install the devices that I own, to save space. Note that the WebPACK edition doesn't support all devices, but thankfully it supports the most popular ones used by hobbyists (visit https://www.xilinx.com/products/design-tools/vivado/vivado-webpack.html for a full list of supported devices). I also don't install System Generator for DSP, but that's just me.



Vivado 2016.4 Installer - Vivado HL WebPACK

Vivado HL WebPACK

Customize your installation by (de)selecting items in the tree below. Moving cursor over selections below provide additional information.

Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition.

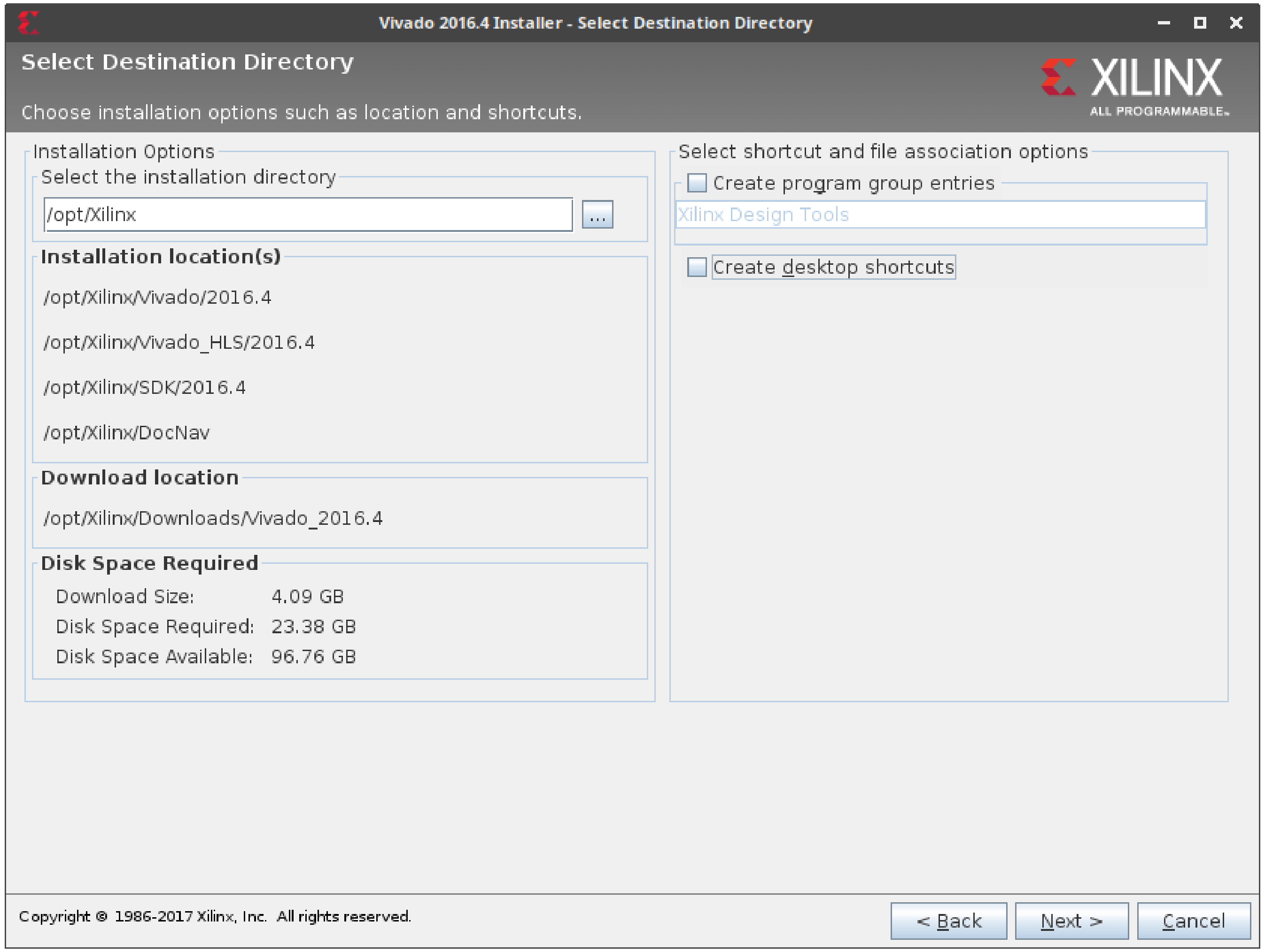
- Design Tools
  - ▣ Vivado Design Suite
    - ☑ Vivado
    - ☐ System Generator for DSP
    - ☑ Vivado High Level Synthesis
  - ☑ Software Development Kit (SDK)
  - ☑ DocNav
- Devices
  - ▣ SoCs
    - ☑ Zynq-7000 (limited support)
    - ☐ Zynq UltraScale+ MPSoC
  - ▣ 7 Series (limited support)
    - ☑ Artix-7
    - ☐ Kintex-7
    - ☐ Spartan-7
    - ☐ Virtex-7
  - ☐ UltraScale (limited support)
    - ☐ Kintex UltraScale
    - ☐ Virtex UltraScale
  - ☐ UltraScale+
- Installation Options
  - ☐ NOTE: Cable Drivers are not installed on Linux. Please follow the instructions in UG973 to install Linux cable drivers
  - ☐ Acquire or Manage a License Key
  - ☑ Enable WebTalk for Vivado to send usage statistics to Xilinx (Always enabled for WebPACK license)
  - ☑ Enable WebTalk for SDK to send usage statistics to Xilinx

Enables WebTalk to send software, IP and device usage statistics to Xilinx. For more information on WebTalk, please see Section 13 of the Xilinx End-User License Agreement.

Download Size: 4.0
Disk Space Require

Copyright © 1986-2017 Xilinx, Inc. All rights reserved.

Click next. In the next screen select the installation directory. I leave it to the default and also un-check the boxes for desktop shortcuts and group entries (I always launch vivado from the command line)


Vivado 2016.4 Installer - Select Destination Directory

Select Destination Directory

Choose installation options such as location and shortcuts.

Installation Options

Select the installation directory: /opt/Xilinx

Installation location(s)

/opt/Xilinx/Vivado/2016.4

/opt/Xilinx/Vivado_HLS/2016.4

/opt/Xilinx/SDK/2016.4

/opt/Xilinx/DocNav

Download location

/opt/Xilinx/Downloads/Vivado_2016.4

Disk Space Required

Download Size:	4.09 GB
Disk Space Required:	23.38 GB
Disk Space Available:	96.76 GB

Select shortcut and file association options

☐ Create program group entries: Xilinx Design Tools

☐ Create desktop shortcuts

Copyright © 1986-2017 Xilinx, Inc. All rights reserved.

Click next and then Install. Depending on your internet bandwidth this may take some time

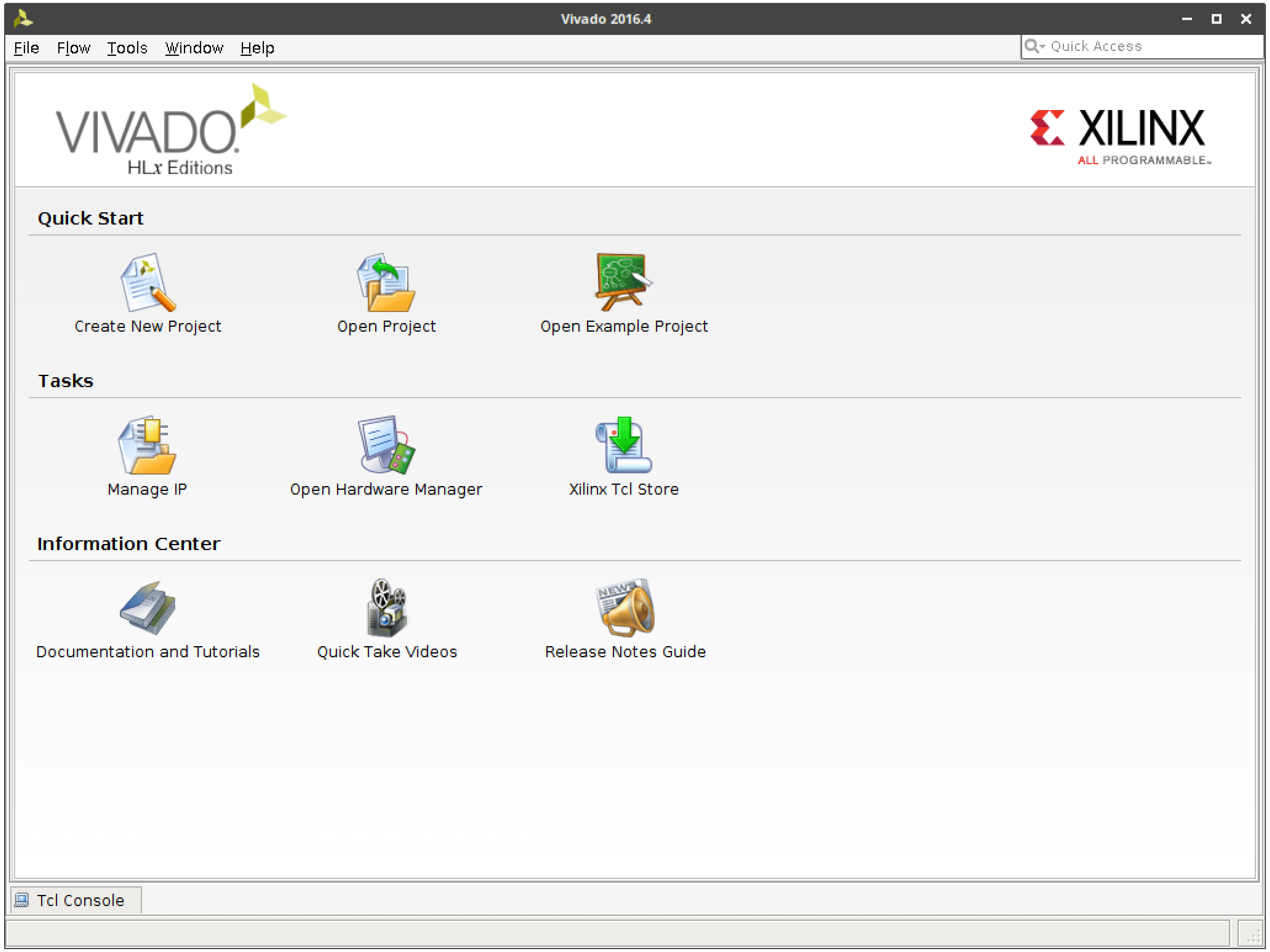
6) After installation is finished, open **~/.bashrc** in a text editor and add the following line

```
$ source /opt/Xilinx/Vivado/2016.4/settings64.sh
```

then in the terminal run

```
$ source ~/.bashrc
vivado &
```

If everything went well you should be seeing the Vivado main window.



The WebPACK versions of Vivado after 2016.x don't require a license any more, so everything should be working fine.

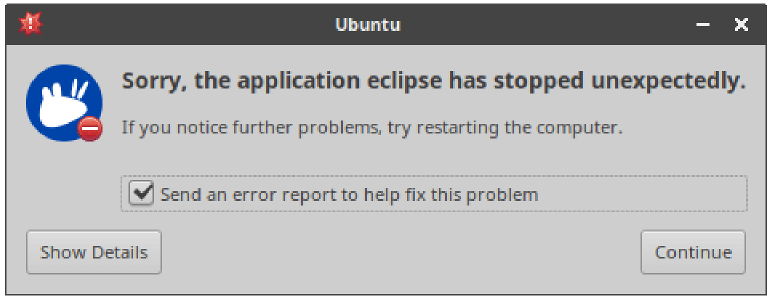
7) You may have noticed that during installation (see step 5 above) the option to install cable drivers is grayed out, with a note to check guide UG973. If you own a Xilinx Platform USB cable you will need to manually install them:

```
$ cd /opt/Xilinx/Vivado/2016.4/data/xicom/cable_drivers/lin64/install_script/install_drivers
$ sudo ./install_drivers
```

You may need to reboot.

Do not follow any older guides that tell you to download and install the drivers from the Xilinx website, those drivers won't work on Ubuntu 16.10

8) An extra step is needed to make Xilinx SDK work. If you run `xsdk` from the command line you get the following error:


Ubuntu

Sorry, the application eclipse has stopped unexpectedly.

If you notice further problems, try restarting the computer.

[✓] Send an error report to help fix this problem

Open the file **/opt/Xilinx/SDK/2016.4/bin/xsdk** in a text editor (vi, nano, etc.)

```
$ sudo nano /opt/Xilinx/SDK/2016.4/bin/xsdk
```

and add the following line:

```
$ export SWT_GTK3=0
```

9) You may also get this error when running SDK

```
$ /opt/Xilinx/SDK/2016.4/lib/lnx64.o/libstdc++.so.6: version `GLIBCXX_3.4.20' not found (required by /usr/lib/x86_64-lin
Failed to load module: /usr/lib/x86_64-linux-gnu/gio/modules/libgiolibproxy.so
```

It's probably because libstdc++.so.6 shipped with SDK is an older/incompatible version. The solution is to use your system's library instead. In a terminal execute the following commands

```
$ sudo mv /opt/Xilinx/SDK/2016.4/lib/lnx64.o/libstdc++.so.6 /opt/Xilinx/SDK/2016.4/lib/lnx64.o/libstdc++.so.6.orig
$ sudo ln -s /usr/lib/x86_64-linux-gnu/libstdc++.so.6 /opt/Xilinx/SDK/2016.4/lib/lnx64.o/libstdc++.so.6
```

Solution taken from here

Execute `xsdk &` and it should now run without errors

Install Intel Modelsim

Optionally you can install Intel (ex Altera) Modelsim Starter Edition for simulations. The Vivado simulator is great for most designs, but I find Modelsim more versatile and I'm more used to it to be honest (used it professionally for many years). Alternatively you can use freeware simulators like ghdl (for VHDL) and Icarus (for Verilog) but I'm not familiar with either tool.

The Starter Edition has a limitation of 10,000 lines and 3,000 instances but it's free (the full featured one costs ~2000$). **Unfortunately it's also a pain to install in Ubuntu...**

1) First go to https://www.altera.com/ and create an account

2) Go to http://dl.altera.com/?product=modelsim_ae#tabs-2 and click the *Individual Files* tab. Click the download button for *ModelSim-Intel FPGA Edition*.

3) After download finishes, open a terminal where you downloaded the file and enter

```
$ chmod +x ModelSimSetup-16.1.0.196-linux.run
$ sudo ./ModelSimSetup-16.1.0.196-linux.run
```

(`sudo` is not necessary if you're going to install in your home folder)

4) Click next in the first two screens, accept the license agreement and click next. Choose installation path and click next. My installation directory is **/opt/intelFPGA/16.1** so the rest of the guide will assume this path and use `sudo` where necessary. Click next to start the installation and then Finish.

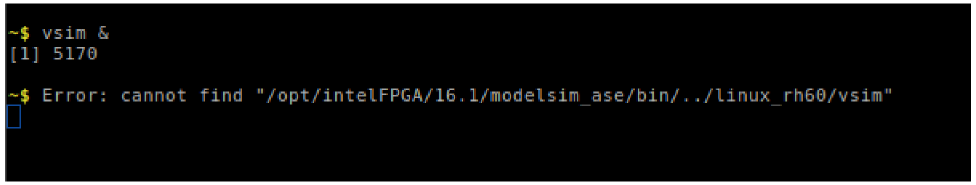
5) Open **~/.bashrc** and add this line

```
$ export PATH=$PATH:/opt/intelFPGA/16.1/modelsim_ase/bin
```

then in the terminal run

```
$ source ~/.bashrc
vsim &
```

Oops..



```
~$ vsim &
[1] 5170

~$ Error: cannot find "/opt/intelFPGA/16.1/modelsim_ase/bin/../linux_rh60/vsim"
```

Well, that's one of a dozen errors you may get until you have everything right so from now on I'll just describe all the necessary steps without showing the intermediate error messages

6) Open **/opt/intelFPGA/16.1/modelsim_ase/bin/vsim** in a text editor

- Search for `linux_rh60` and change it to `linux`
- Search for `MTI_VCO_MODE:-""` and change it to `MTI_VCO_MODE:-"32"`
- Search for ``dir=`dirname "$arg0"` `` and under that line add `export LD_LIBRARY_PATH=${dir}/lib32`

save and close

7) Modelsim is a 32-bit application and needs some 32-bit libraries. In the terminal run

```
$ sudo dpkg --add-architecture i386
$ sudo apt-get update
$ sudo apt-get install libxft2:i386 libxext6:i386 libncurses5:i386
```

8) We also need **libpng12-0:1386** but in Ubuntu 16.10 that library is not in the repository. We can download it from this link: http://launchpadlibrarian.net/233197172/libpng12-0_1.2.54-1ubuntu1_i386.deb

Go to where you downloaded the file and run

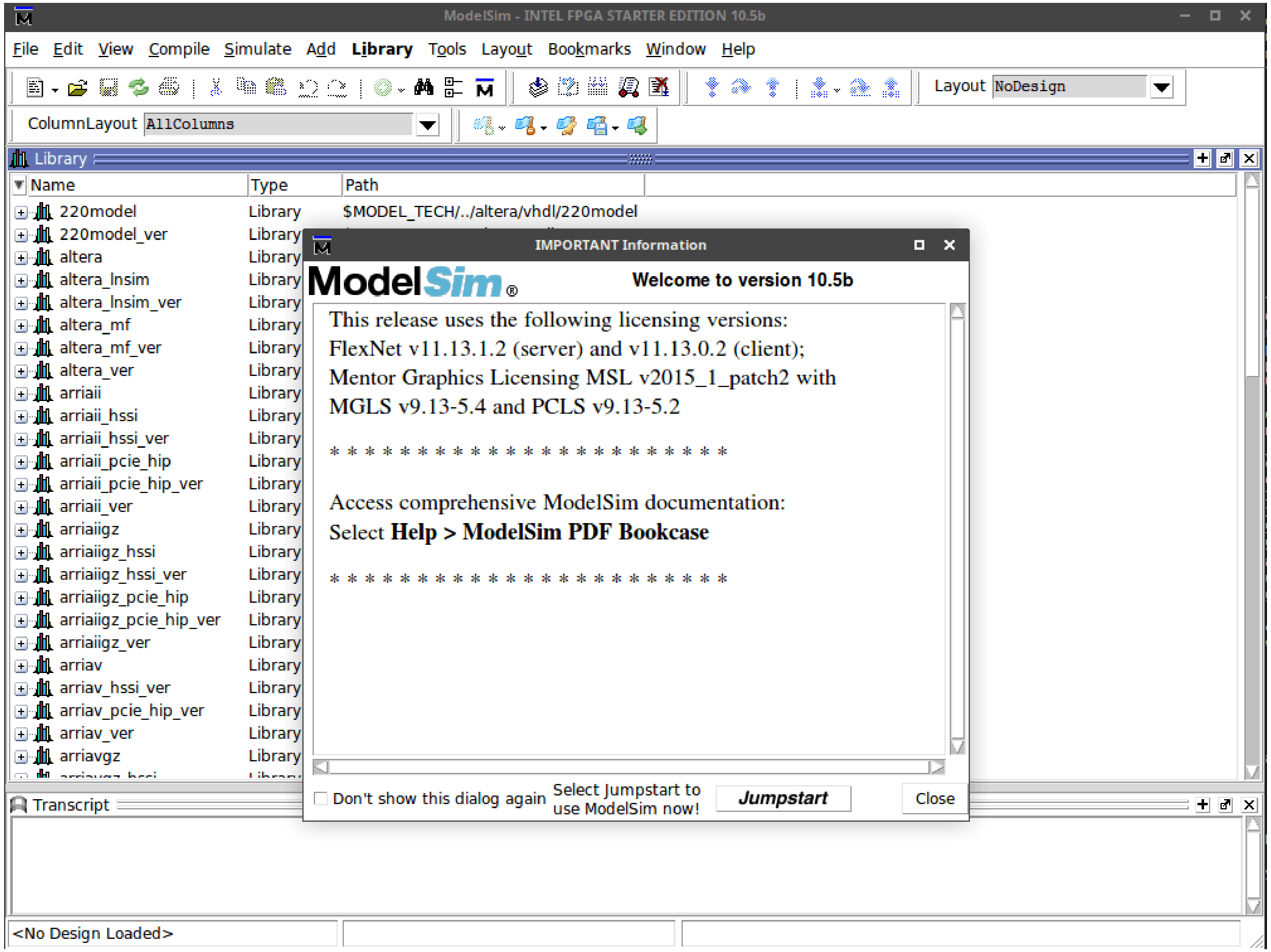
```
$ sudo dpkg -i libpng12-0_1.2.54-1ubuntu1_i386.deb
```

9) There's also some kind of incompatibility with the latest freetype library, so we need to install an older version. Version 2.5.0.1 seems to be working. You can build it from source, but thankfully there's an already compiled 32-bit version for Ubuntu at https://launchpad.net/ubuntu/+source/freetype/2.5.0.1-0ubuntu2/+build/5222114/+files/libfreetype6_2.5.0.1-0ubuntu2_i386.deb

Click on the link above to download it. Go to where you downloaded the file and enter

```
$ mkdir libfreetype
$ dpkg -x libfreetype6_2.5.0.1-0ubuntu2_i386.deb libfreetype
$ sudo mkdir /opt/intelFPGA/16.1/modelsim_ase/lib32
$ sudo cp libfreetype/usr/lib/i386-linux-gnu/libfreetype.so.6* /opt/intelFPGA/16.1/modelsim_ase/lib32
```

Now run `vsim &` and hopefully that's what you'll see


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Transcript
IMPORTANT Information
ModelSim
Welcome to version 10.5b
Don't show this dialog again
Select Jumpstart to use ModelSim now!
Jumpstart
Close
<No Design Loaded>

Name	Type	Path
220model	Library	$MODEL_TECH/../altera/vhdl/220model
220model_ver	Library	
altera	Library	
altera_lnsim	Library	
altera_lnsim_ver	Library	
altera_mf	Library	
altera_mf_ver	Library	
altera_ver	Library	
arriaii	Library	
arriaii_hssi	Library	
arriaii_hssi_ver	Library	
arriaii_pcie_hip	Library	
arriaii_pcie_hip_ver	Library	
arriaii_ver	Library	
arriaiigz	Library	
arriaiigz_hssi	Library	
arriaiigz_hssi_ver	Library	
arriaiigz_pcie_hip	Library	
arriaiigz_pcie_hip_ver	Library	
arriaiigz_ver	Library	
arriav	Library	
arriav_hssi_ver	Library	
arriav_pcie_hip_ver	Library	
arriav_ver	Library	
arriavgz	Library	

This release uses the following licensing versions:
FlexNet v11.13.1.2 (server) and v11.13.0.2 (client);
Mentor Graphics Licensing MSL v2015_1_patch2 with
MGLS v9.13-5.4 and PCLS v9.13-5.2

* *

Access comprehensive ModelSim documentation:
Select **Help > ModelSim PDF Bookcase**

* *

Install Xilinx libraries for Modelsim

If you want to use Modelsim to simulate Xilinx IP cores you need to compile the Xilinx libraries for Modelsim. Officially the Vivado compiler does not support the Intel Modelsim Starter Edition, but there is a workaround.

1) Create a symbolic link named **linuxpe** that points to the Modelsim **linux** directory

```
$ sudo ln -s /opt/intelFPGA/16.1/modelsim_ase/linux /opt/intelFPGA/16.1/modelsim_ase/linuxpe
```

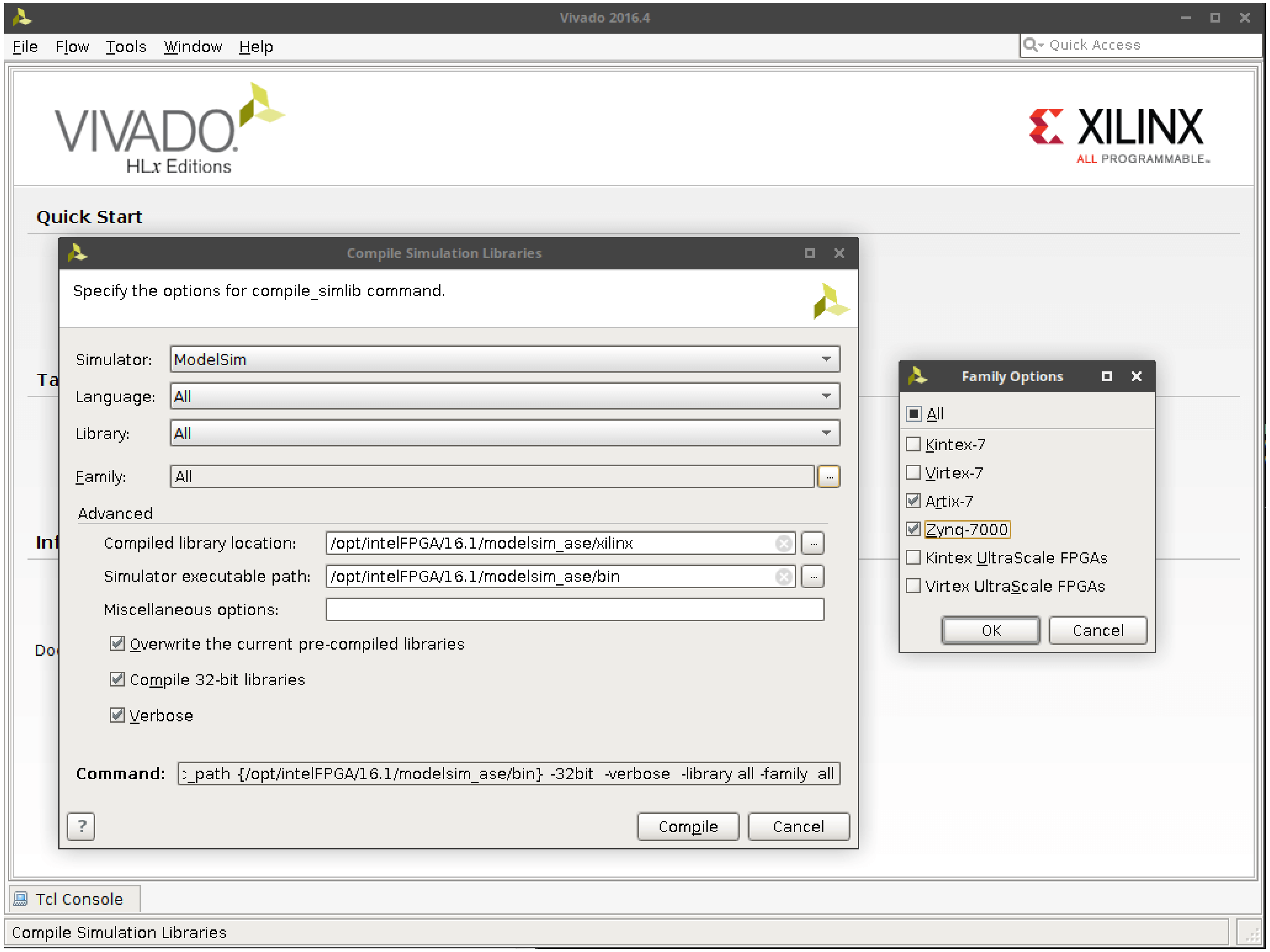
2) Create a folder named **xilinx** in the **modelsim_ase** folder

```
$ sudo mkdir /opt/intelFPGA/16.1/modelsim_ase/xilinx
```

3) Open Vivado with root privileges (if you installed Modelsim in /opt). We need to specify the complete path because vivado is not in the PATH for the root user

```
$ sudo /opt/Xilinx/Vivado/2016.4/bin/vivado &
```

4) From the *Tools* menu select *Compile Simulation Libraries*. In Family Options select the devices you need. Set the rest of the options as shown and click Compile



Specify the options for compile_simlib command.

Simulator: ModelSim
Language: All
Library: All
Family: All

Advanced
Compiled library location: /opt/intelFPGA/16.1/modelsim_ase/xilinx
Simulator executable path: /opt/intelFPGA/16.1/modelsim_ase/bin
Miscellaneous options:

☑ Overwrite the current pre-compiled libraries
☑ Compile 32-bit libraries
☑ Verbose

Command: :_path {/opt/intelFPGA/16.1/modelsim_ase/bin} -32bit -verbose -library all -family all

Family Options
▣ All
☐ Kintex-7
☐ Virtex-7
☑ Artix-7
☑ Zynq-7000
☐ Kintex UltraScale FPGAs
☐ Virtex UltraScale FPGAs

5) The compiler created a **modelsim.ini** file in your home folder which contains the paths for every xilinx library.

These are a lot of libraries and probably you won't need most of them. I always leave the major Xilinx libraries and comment out the unnecessary/rarely used ones

By major libraries I mean the following:
secureip, unisim, unimacro, unifast (for VHDL)

unisims_ver, unimacro_ver, unifast_ver, simprims_ver (for Verilog)

The rest are just IP libraries which you will only need if you plan to simulate those IPs (as part of a zynq/microblaze simulation for example). If you ever need them you can either uncomment them here or add them in you project **modelsim.ini** file (Modelsim always creates a **modelsim.ini** file in your project directory)

Anyway, here's how my xilinx **modelsim.ini** [Library] section looks like:

```
; Copyright 1991-2009 Mentor Graphics Corporation
;
; All Rights Reserved.
;
; THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION WHICH IS THE PROPERTY OF
; MENTOR GRAPHICS CORPORATION OR ITS LICENSORS AND IS SUBJECT TO LICENSE TERMS.
;

[Library]
others = $MODEL_TECH/../modelsim.ini

; Altera Primitive libraries
;
; VHDL Section
;
;
; Verilog Section
;

secureip = /opt/intelFPGA/16.1/modelsim_ase/xilinx/secureip
unisim = /opt/intelFPGA/16.1/modelsim_ase/xilinx/unisim
unimacro = /opt/intelFPGA/16.1/modelsim_ase/xilinx/unimacro
unifast = /opt/intelFPGA/16.1/modelsim_ase/xilinx/unifast
unisims_ver = /opt/intelFPGA/16.1/modelsim_ase/xilinx/unisims_ver
unimacro_ver = /opt/intelFPGA/16.1/modelsim_ase/xilinx/unimacro_ver
unifast_ver = /opt/intelFPGA/16.1/modelsim_ase/xilinx/unifast_ver
simprims_ver = /opt/intelFPGA/16.1/modelsim_ase/xilinx/simprims_ver
;xpm = /opt/intelFPGA/16.1/modelsim_ase/xilinx/xpm
;ecc_v2_0_12 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/ecc_v2_0_12
;mipi_dphy_v3_0_1 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/mipi_dphy_v3_0_1
;emc_common_v3_0_5 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/emc_common_v3_0_5
;v_vcresampler_v1_0_5 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/v_vcresampler_v1_0_5
;axi_infrastructure_v1_1_0 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/axi_infrastructure_v1_1_0
;bsip_v1_1_0 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/bsip_v1_1_0
;fifo_generator_v13_0_5 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/fifo_generator_v13_0_5
;xaui_v12_2_7 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/xaui_v12_2_7
;sem_v4_1_7 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/sem_v4_1_7
;lmb_bram_if_cntlr_v4_0_10 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/lmb_bram_if_cntlr_v4_0_10
;jesd204_v7_1_1 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/jesd204_v7_1_1
;l_ethernet_v2_0_1 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/l_ethernet_v2_0_1
;hdcp22_cipher_v1_0_0 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/hdcp22_cipher_v1_0_0
;axi_amm_bridge_v1_0_1 = /opt/intelFPGA/16.1/modelsim_ase/xilinx/axi_amm_bridge_v1_0_1
;etc, etc
```

NOTE: if you run the compiler with `sudo` the file owner is root so you may need to change the owner (or alter the file using sudo)

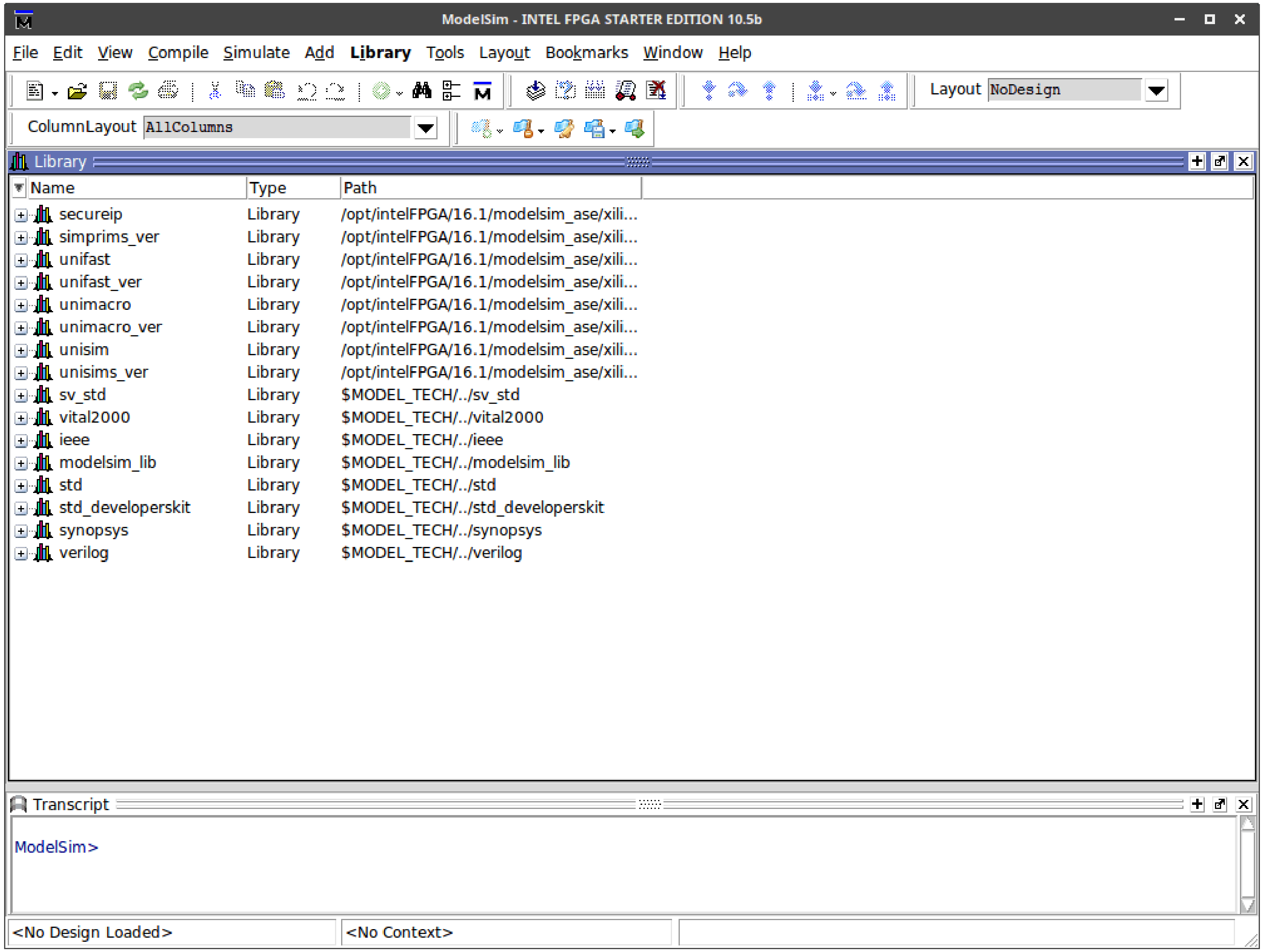
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

library UNISIM;
use UNISIM.VComponents.all;

entity dvi_tx_clkgen is
  port(
    clk_i       : in  std_logic; -- 125 MHz reference clock
    arst_i      : in  std_logic; -- asynchronous reset (from board pin)
    locked_o    : out std_logic; -- synchronous to reference clock
    pixel_clk_o : out std_logic; -- pixel clock
    sclk_o      : out std_logic; -- serdes clock (framing clock)
    sclk_x5_o   : out std_logic  -- serdes clock x5 (bit clock)
  );
end entity dvi_tx_clkgen;

architecture rtl of dvi_tx_clkgen is

  signal clkfb_x            : std_logic;
  signal refrst_x           : std_logic;
  signal mmcm_locked_x      : std_logic;
  signal mmcm_locked_sync_x : std_logic;
  signal mmcm_rst_r         : std_logic;
  signal bufr_rst_r         : std_logic;
  signal pixel_clk_x        : std_logic;
  signal sclk_x5_x          : std_logic;

  type fsm_mmcm_rst_t is (WAIT_LOCK, LOCKED);
  signal state_mmcm_rst : fsm_mmcm_rst_t := WAIT_LOCK;
begin

  -- The reset bridge will make sure we can use the async rst
  -- safely in the reference clock domain
  refrst_inst : entity work.rst_bridge
    port map(
      arst_in  => arst_i,
      sclk_in  => clk_i,
      srst_out => refrst_x
    );

  -- sync MMCM lock signal to the reference clock domain
  sync_mmcm_locked_inst : entity work.sync_dff
    port map(
      async_in => mmcm_locked_x,
      sclk_in  => clk_i,
      sync_out => mmcm_locked_sync_x
    );

  -- Need to generate an MMCM reset pulse >= 5 ns (Xilinx DS191).
  -- We can use the reference clock to create the pulse. The fsm
  -- below will only work is the reference clk frequency is < 200MHz.
  -- The BUFR needs to be reset any time the MMCM acquires lock.
  fsm_mmcm_rst_p : process(refrst_x, clk_i)
  begin
    if (refrst_x='1') then
      state_mmcm_rst <= WAIT_LOCK;
      mmcm_rst_r <= '1';
      bufr_rst_r <= '0';
    elsif rising_edge(clk_i) then
      mmcm_rst_r <= '0';
      bufr_rst_r <= '0';
      case state_mmcm_rst is
        when WAIT_LOCK =>
          if (mmcm_locked_sync_x='1') then
            bufr_rst_r     <= '1';
            state_mmcm_rst <= LOCKED;
          end if;

        when LOCKED =>
          if (mmcm_locked_sync_x='0') then
            mmcm_rst_r     <= '1';
          end if;
      end case;

    end if;
  end process;


  mmcme2_adv_inst : MMCME2_ADV
    generic map (
      BANDWIDTH             => "OPTIMIZED",
      CLKFBOUT_MULT_F       => 12.0,
      CLKFBOUT_PHASE        => 0.0,
      CLKIN1_PERIOD         => 8.0,  -- 125 MHz
      CLKOUT0_DIVIDE_F      => 2.0,  -- 1.0 for 1080p
      CLKOUT1_DIVIDE        => 10,   -- 5 for 1080p
      COMPENSATION          => "ZHOLD",
      DIVCLK_DIVIDE         => 2,
      REF_JITTER1           => 0.0
    )
    port map (
      CLKOUT0               => sclk_x5_x,
      CLKOUT0B              => open,
      CLKOUT1               => pixel_clk_x,
      CLKOUT1B              => open,
      CLKOUT2               => open,
      CLKOUT2B              => open,
      CLKOUT3               => open,
      CLKOUT3B              => open,
      CLKOUT4               => open,
      CLKOUT5               => open,
      CLKOUT6               => open,
      CLKFBOUT              => clkfb_x,
      CLKFBOUTB             => open,
      CLKIN1                  => clk_i,
      CLKIN2                  => '0',
      CLKFBIN                 => clkfb_x,
      CLKINSEL                => '1',

      DCLK                    => '0',
      DEN                     => '0',
      DWE                     => '0',
      DADDR                   => (others=>'0'),
      DI                      => (others=>'0'),
      DO                      => open,
      DRDY                    => open,

      PSCLK                   => '0',
      PSEN                    => '0',
      PSINCDEC                => '0',
      PSDONE                  => open,

      LOCKED                  => mmcm_locked_x,
      PWRDWN                  => '0',
      RST                     => mmcm_rst_r,
      CLKFBSTOPPED            => open,
      CLKINSTOPPED            => open
    );

  bufio_inst : BUFIO
    port map (
      O => sclk_x5_o,
      I => sclk_x5_x
    );

  -- If the clock to the BUFR is stopped, then a reset (CLR)
  -- must be applied after the clock returns (see Xilinx UG472)
  bufr_inst : BUFR
    generic map (
      BUFR_DIVIDE => "5",
      SIM_DEVICE => "7SERIES"
    )
    port map (
      O   => sclk_o,
      CE  => '1',
      CLR => bufr_rst_r,
      I   => sclk_x5_x
    );


  -- The tools will issue a warning that pixel clock is not
  -- phase aligned to sclk_x, sclk_x5_x. We can safely
  -- ignore it as we don't care about the phase relationship
  -- of the pixel clock to the sampling clocks.
  bufg_inst : BUFG
    port map (
      O => pixel_clk_o,
      I => pixel_clk_x
    );
  locked_p : process(mmcm_locked_x, clk_i)
  begin
    if (mmcm_locked_x='0') then
      locked_o <= '0';
    elsif rising_edge(clk_i) then
      -- Raise locked only after BUFR has been reset
      if (bufr_rst_r='1') then
        locked_o <= '1';
      end if;
    end if;
  end process;

end architecture;
```

Modelsim by default also includes Altera libraries. If you don't own any Altera devices you can comment them out in the main **modelsim.ini** file located at **/opt/intelFPGA/16.1/modelsim_ase/modelsim.ini**

Now at the terminal run

```
$ vsim &
```

You should see the Xilinx libraries like in the picture below:


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Transcript

Name	Type	Path
secureip	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
simprims_ver	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
unifast	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
unifast_ver	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
unimacro	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
unimacro_ver	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
unisim	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
unisims_ver	Library	/opt/intelFPGA/16.1/modelsim_ase/xili...
sv_std	Library	$MODEL_TECH/../sv_std
vital2000	Library	$MODEL_TECH/../vital2000
ieee	Library	$MODEL_TECH/../ieee
modelsim_lib	Library	$MODEL_TECH/../modelsim_lib
std	Library	$MODEL_TECH/../std
std_developerskit	Library	$MODEL_TECH/../std_developerskit
synopsys	Library	$MODEL_TECH/../synopsys
verilog	Library	$MODEL_TECH/../verilog

```
ModelSim>
```

<No Design Loaded> <No Context>

And you're done. Easy, right? :)

6 Comments **lvoudouris** 1 **Login**

Recommend 4 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Ben Xu • a year ago
Thank you! So helpful.
Reply • Share ›

Vincent • a year ago
Thanks! Worked instantly on the first walk through - no problems thanks to your guide.
Reply • Share ›

Robert • a year ago
Hi, does this procedure solves the issue in Modelsim regarding DPI-C in Ubuntu? Can you run DPI-C now?
Reply • Share ›

> **Lymperis Voudouris** Mod → Robert • a year ago
> Sorry I don't know, I've never used dpi
> Reply • Share ›

cornfalke • a year ago
Thank you so much for this blog post! Worked fine with ModelSim 17.0 and Ubuntu 16.04.
Reply • Share ›

> **Lymperis Voudouris** Mod → cornfalke • a year ago
> Glad it worked
> Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy