Ivica Matic
November 2021

Amazon EC2 Services - Kria KV260 Vivado Tutorial + LynSyn Lite

1. Introduction

In this tutorial, we are going to recreate a project which enables us to control the cooling fan on the Kria KV260 Development board using the Vivado tool from Xilinx.

The purpose of this tutorial is to provide a guideline on how to use Amazon AWS EC2 Virtual Machine to create a hardware project using the Vivado tool and to provide a guide on how to upload the bitstream to our development board using LynSyn Lite tool from Sundance utilizing the SSH tunnel between our local machine and EC2 VM Instance.

The tutorial we are going to follow for our hardware part was created by our colleague, Jack Bonell, and is available at:

https://www.hackster.io/jack-bonnell2/xilinx-kv260-jtag-fan-toggle-using-vio-e9479d

What is LynSyn?

Lynsyn is a power measurement utility board, designed to measure the power usage of a system and correlate power values with the source code of the program running on the system.

In addition to being a power profiler board, lite version of LynSyn, LynSyn Lite can also serve as a generic JTAG pod as it is a replacement for the Xilinx Platform Cable USB-II and can, therefore, also be used as a generic JTAG programming device with the Xilinx Vivado tool suite and a remotely controlled current/voltage meter over USB.

In this tutorial, we are going to use it as a JTAG programming device.

2. Local developer machine requirements

Compiling tools for LynSyn board

As a first step, we need to install the dependencies needed to compile the LynSyn tools. We can do so by running:

Ubuntu 18.04 and before:

```
sudo apt install build-essential qt5-default libqt5sql5-sqlite libusb-1.0 git
```

Ubuntu 20.04:

```
sudo apt install build-essential qt5-default libqt5sql5-sqlite libusb-1.0-0-dev git
```

Next, clone the LynSyn tools repository from GitHub to your local machine:

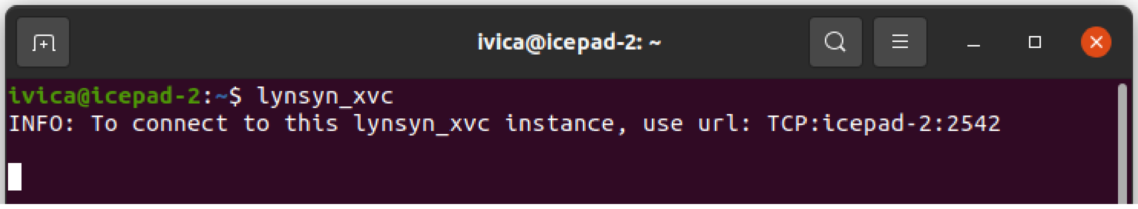
```
git clone https://github.com/EECS-NTNU/lynsyn-host-software.git
```

Navigate to the cloned folder and run:

```
cd lynsyn-host-software
make
make install
```

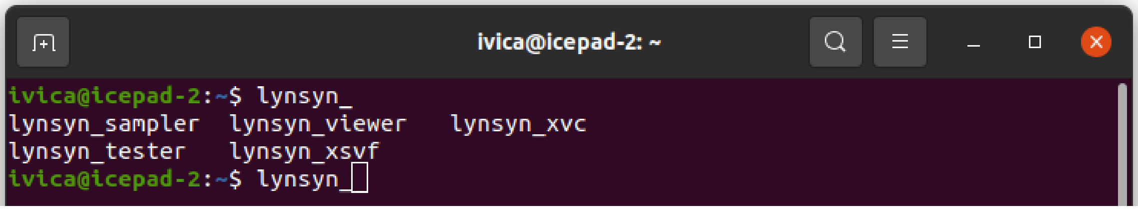
At this point, you should be able to run LynSyn virtual cable driver by running:

```
lynsyn_xvc
```



```
ivica@icepad-2:~$ lynsyn_xvc
INFO: To connect to this lynsyn_xvc instance, use url: TCP:icepad-2:2542
```

You can also explore other LynSyn options such as power profiler, tester and more.



```
ivica@icepad-2:~$ lynsyn_
lynsyn_sampler  lynsyn_viewer   lynsyn_xvc
lynsyn_tester   lynsyn_xsvf
ivica@icepad-2:~$ lynsyn_
```

3. AWS VM Requirements

On your AWS VM machine, you need to have Xilinx Tools installed with the version you need to create your hardware design. As we are using the LynSyn Virtual Cable server our local machine does need to have Xilinx Tools installed.

4. Creating a project

Connecting to your EC2 VM Instance

As we are going to work on our EC2 remote machine we need to connect to it via ssh command. We are going to forward the ports required for remote VNC desktop connection, together with the port for our LynSyn Virtual Cable Driver. Connect the JTAG tool to your PC via a micro-USB cable.

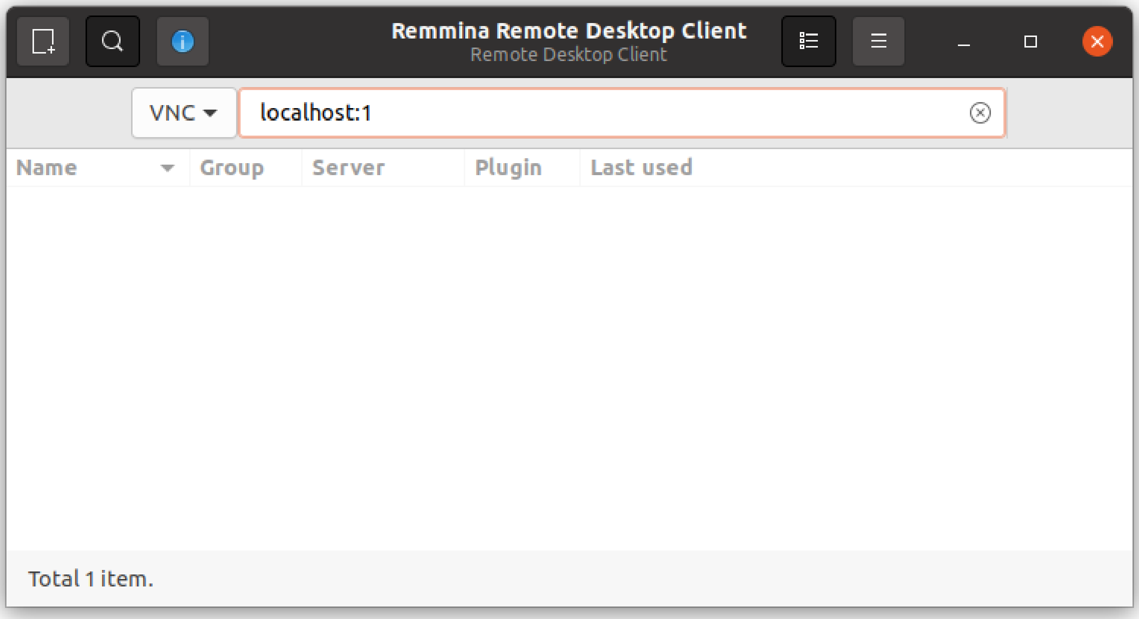
First, start LynSyn virtual cable driver by running:

```
lynsyn_xvc
```

To connect, forwarding required ports mentioned above, run:

```
ssh -i keyforyourvm.pem -L 5901:localhost:5901 -R 2542:localhost:2542
ubuntu@vmipaddressssh
```

Afterwards, connect to a GUI instance using the Remmina tool



localhost:1

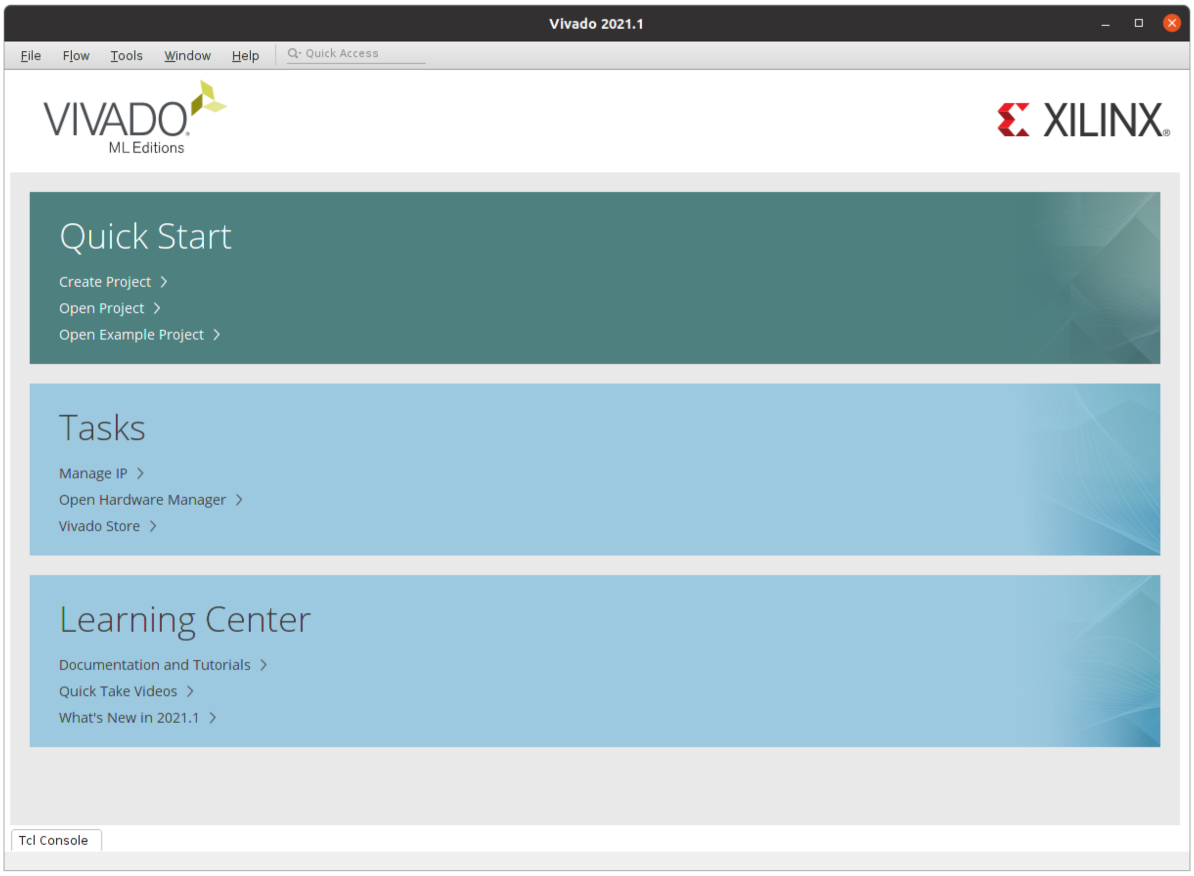
Total 1 item.

Vivado flow

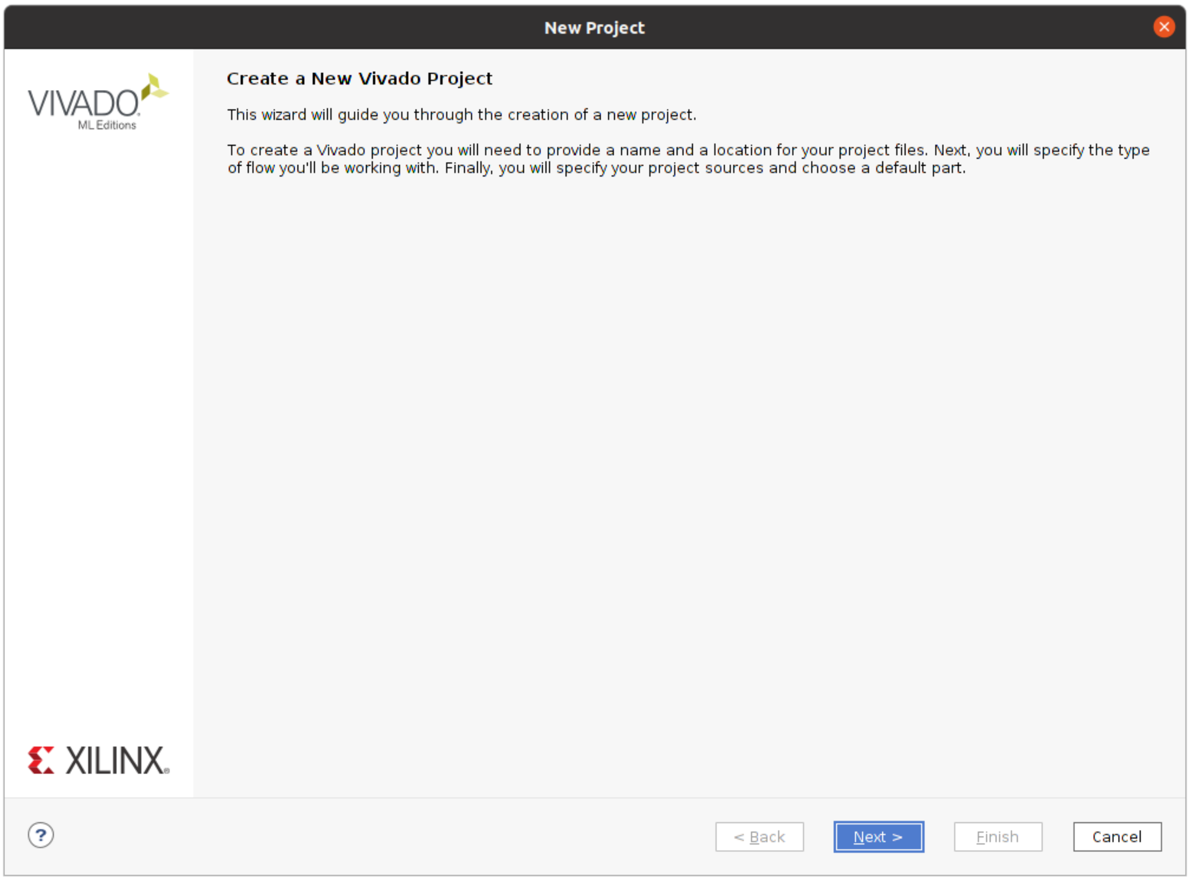
Open the Vivado tool by sourcing your Vivado installation (usually in /tools/Xilinx) and run the Vivado command.

```
source /tools/Xilinx/Vivado/2021.1/settings64.sh &&
vivado
```

Select the Create Project option.



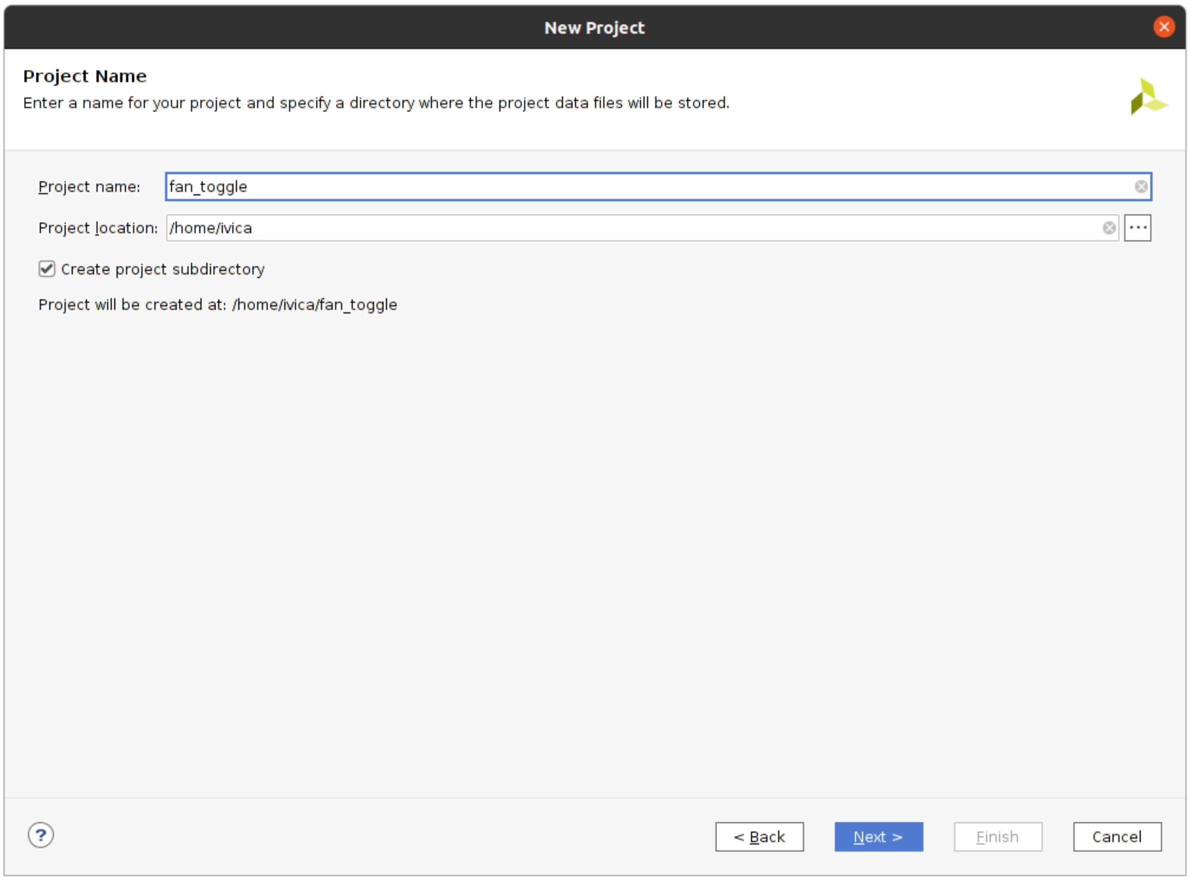
Follow the Create Project prompt, naming your project (for this tutorial we are going to use "fan_toggle" saved in our home directory.


New Project

Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

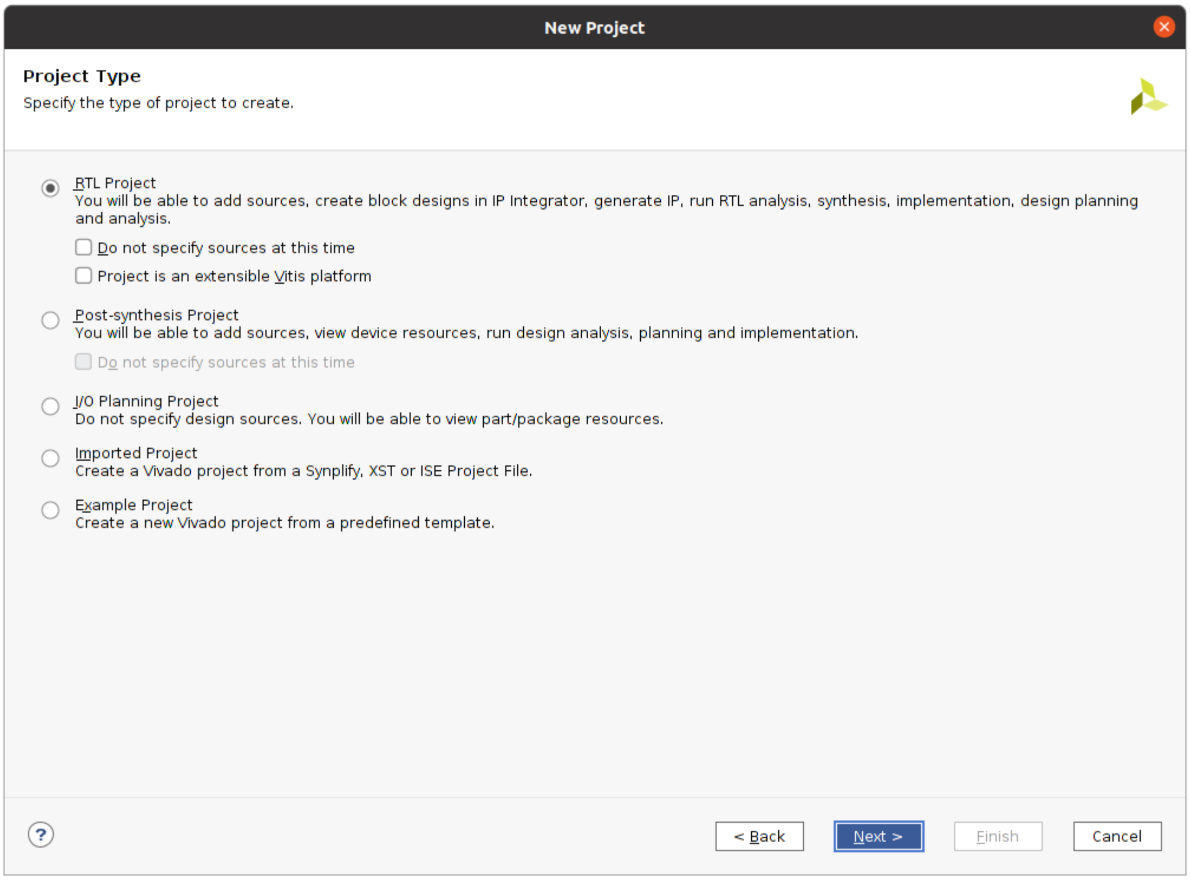
Project name: fan_toggle

Project location: /home/ivica

☑ Create project subdirectory

Project will be created at: /home/ivica/fan_toggle

Specify the project type as an RTL Project



New Project

Project Type
Specify the type of project to create.

[✓ RTL Project]
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time
☐ Project is an extensible Vitis platform

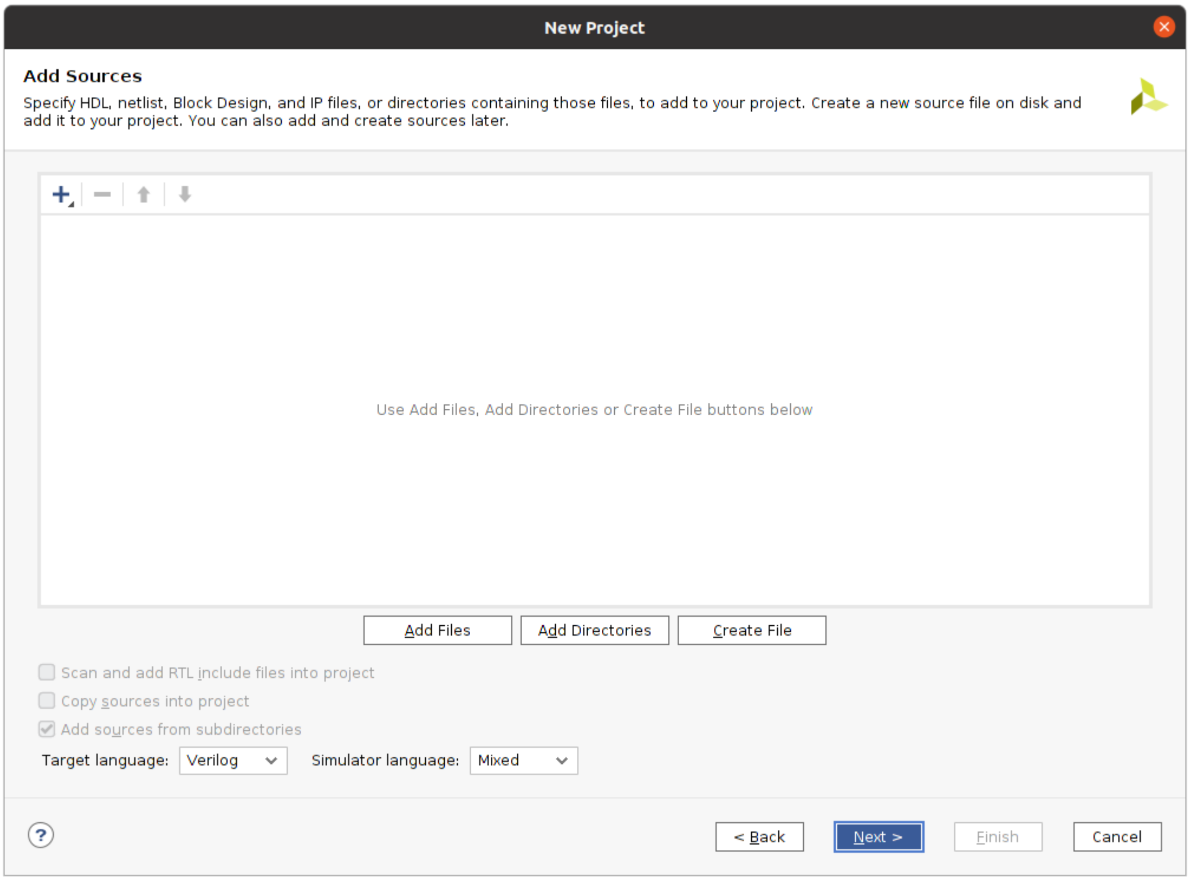
Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project
Create a new Vivado project from a predefined template.

And proceed without adding sources and constraints at this point by clicking next.


New Project

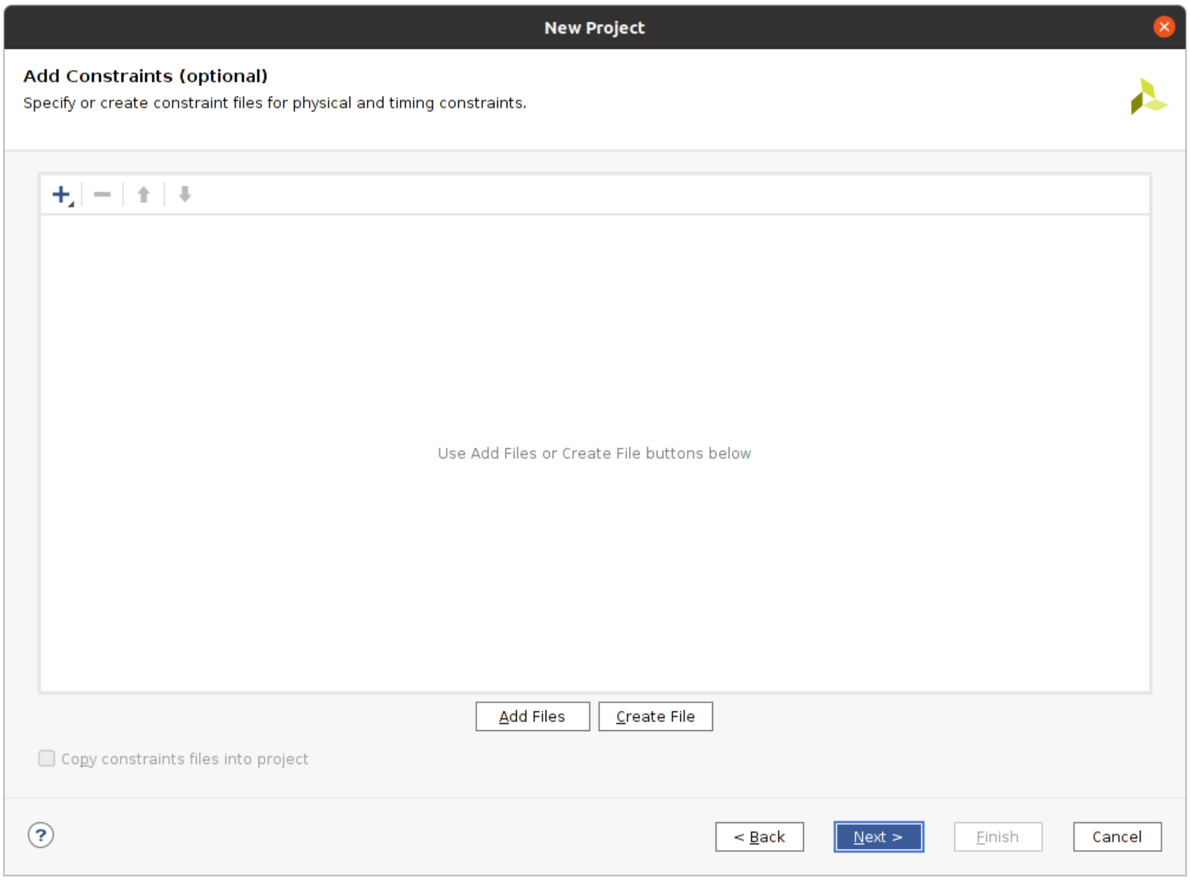
Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Use Add Files, Add Directories or Create File buttons below

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [✓] Add sources from subdirectories

Target language: Verilog   Simulator language: Mixed


New Project

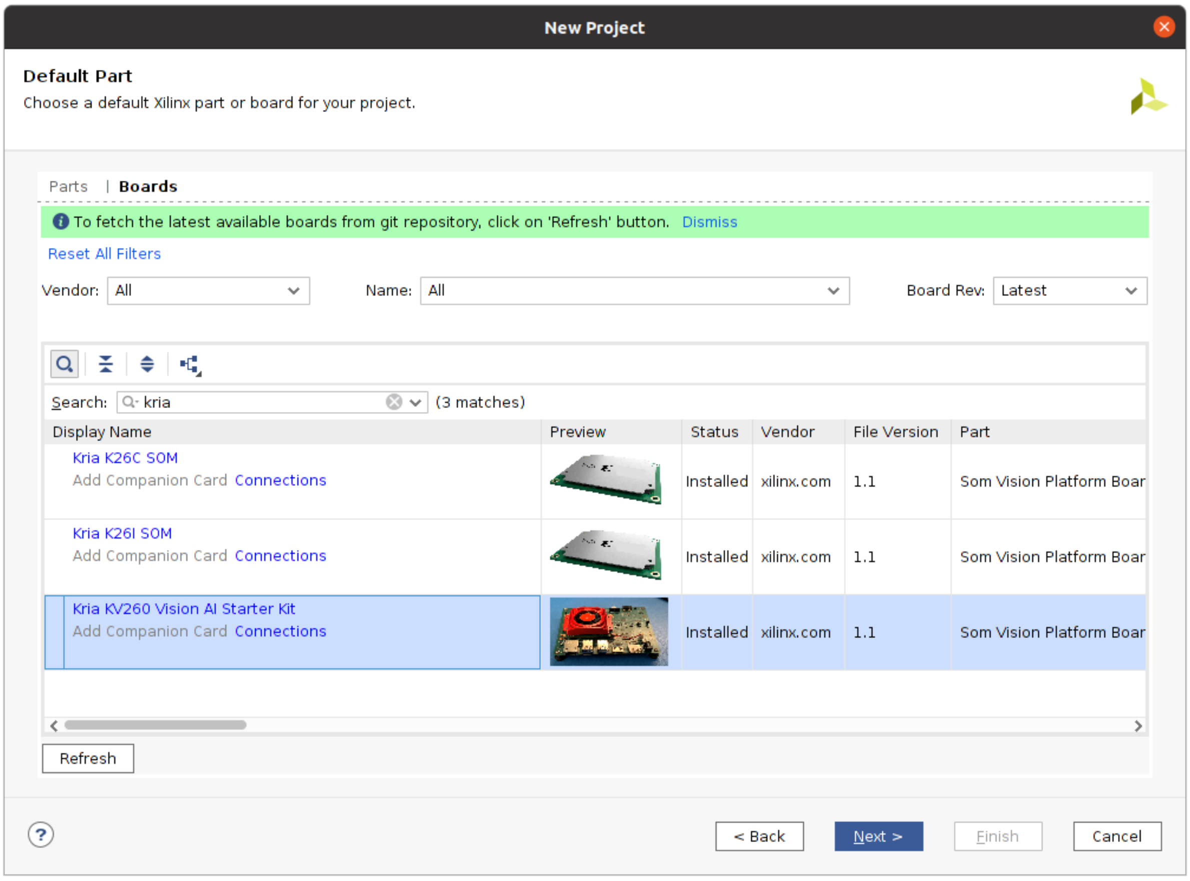
Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Use Add Files or Create File buttons below

- [ ] Copy constraints files into project

Select the Kria KV260 from the Boards tab as our project board and click Next.



New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | **Boards**

To fetch the latest available boards from git repository, click on 'Refresh' button. Dismiss

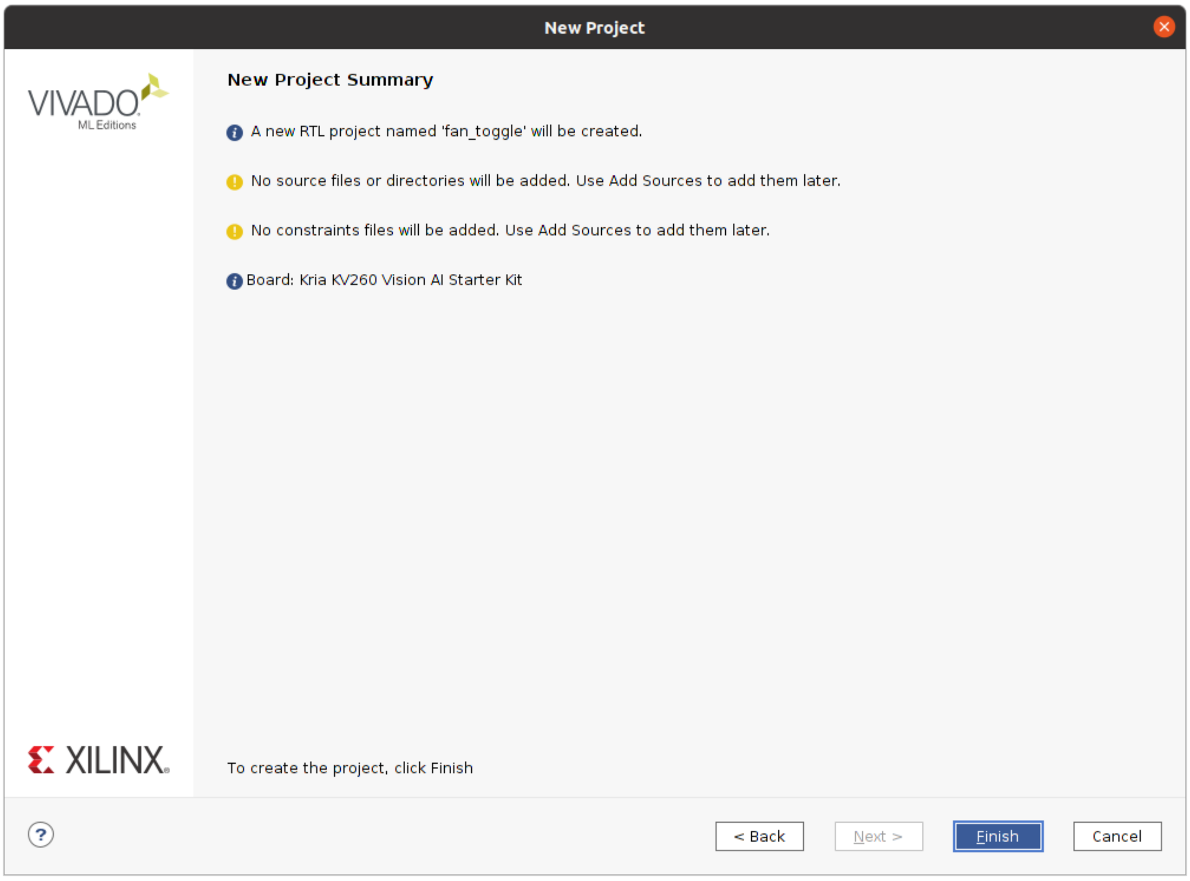
Reset All Filters

Vendor: All   Name: All   Board Rev: Latest

Search: kria (3 matches)

Display Name	Preview	Status	Vendor	File Version	Part
Kria K26C SOM Add Companion Card Connections		Installed	xilinx.com	1.1	Som Vision Platform Boar
Kria K26I SOM Add Companion Card Connections		Installed	xilinx.com	1.1	Som Vision Platform Boar
Kria KV260 Vision AI Starter Kit Add Companion Card Connections		Installed	xilinx.com	1.1	Som Vision Platform Boar

Finally, click finish and your Vivado project should be successfully created.



New Project Summary

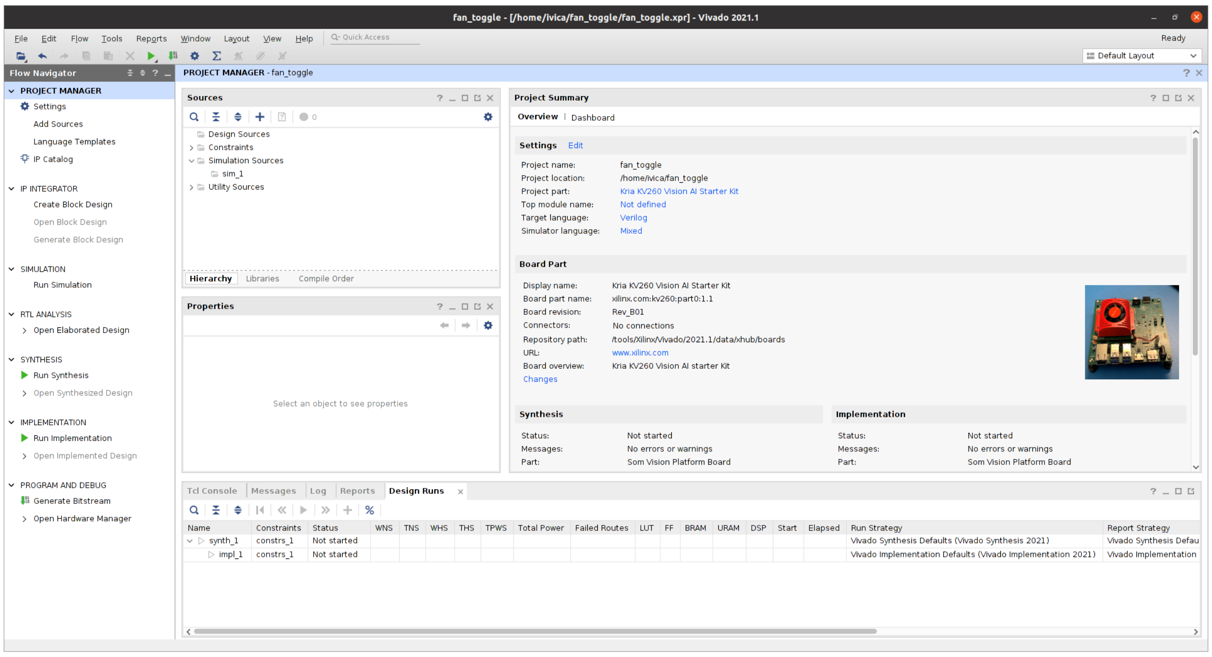
A new RTL project named 'fan_toggle' will be created.

No source files or directories will be added. Use Add Sources to add them later.

No constraints files will be added. Use Add Sources to add them later.

Board: Kria KV260 Vision AI Starter Kit

To create the project, click Finish



Settings Edit

Project name:	fan_toggle
Project location:	/home/ivica/fan_toggle
Project part:	Kria KV260 Vision AI Starter Kit
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

Board Part

Display name:	Kria KV260 Vision AI Starter Kit
Board part name:	xilinx.com:kv260:part0:1.1
Board revision:	Rev_B01
Connectors:	No connections
Repository path:	/tools/Xilinx/Vivado/2021.1/data/xhub/boards
URL:	www.xilinx.com
Board overview:	Kria KV260 Vision AI starter Kit

Changes

Synthesis

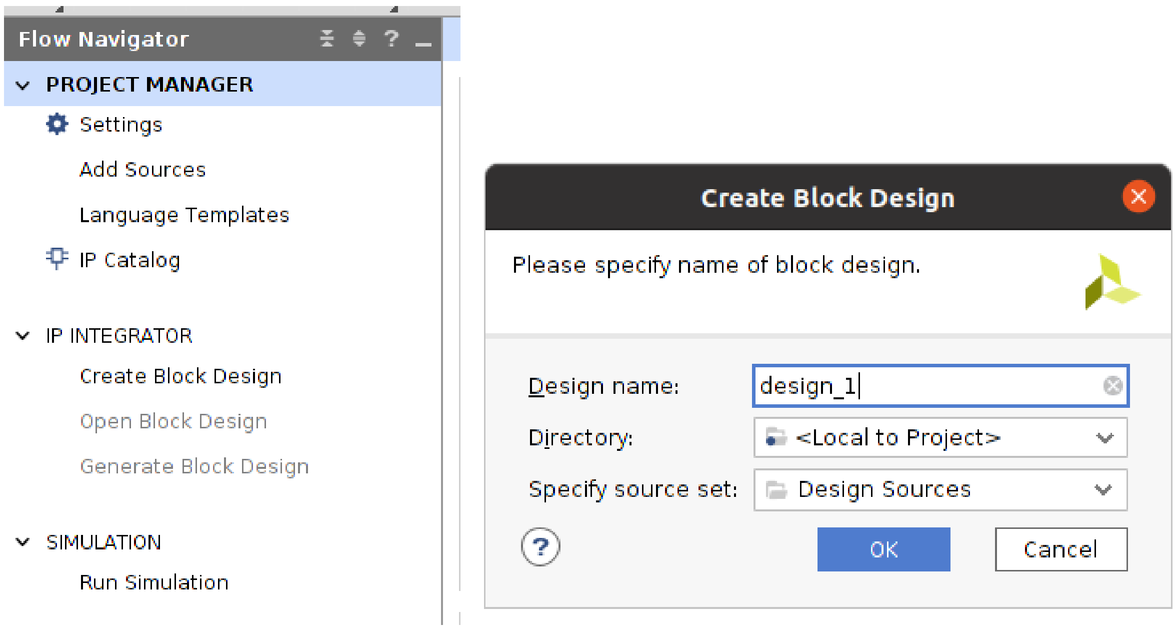
Status:	Not started
Messages:	No errors or warnings
Part:	Som Vision Platform Board

Implementation

Status:	Not started
Messages:	No errors or warnings
Part:	Som Vision Platform Board

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2021)	Vivado Synthesis Defau
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2021)	Vivado Implementation

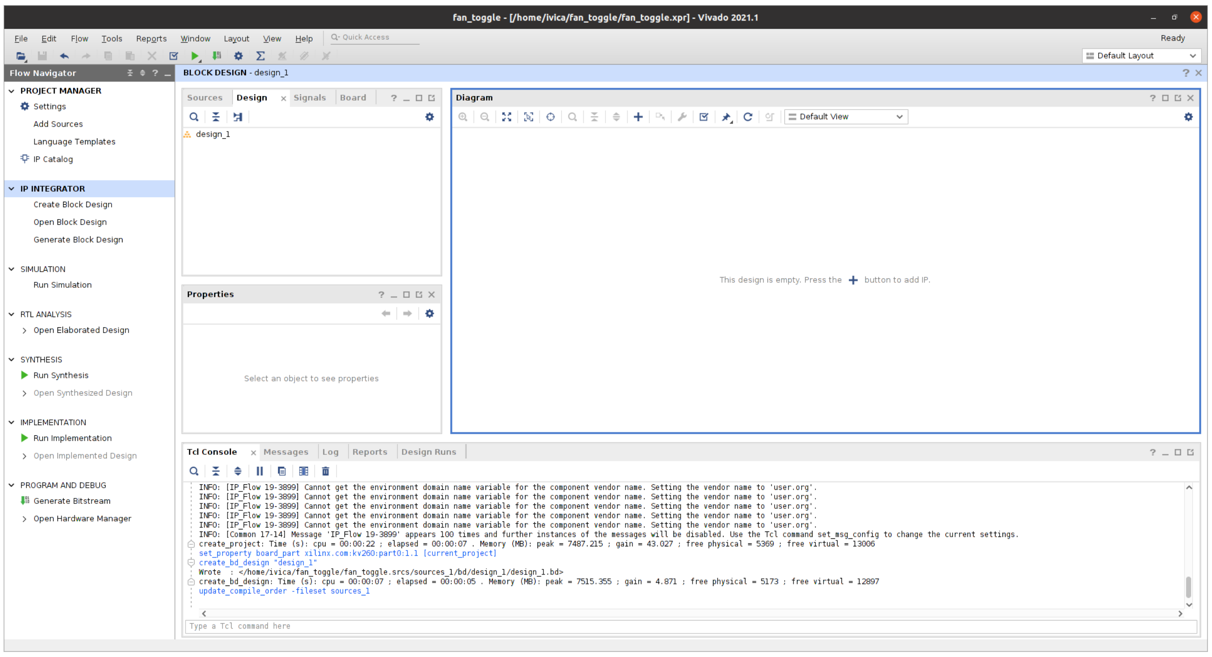
On the left hand side of Vivado, under "IP INTEGRATION" select "Create Block Design". Change the name of your block design if you want and click Ok.



Please specify name of block design.

Design name: design_1
Directory: <Local to Project>
Specify source set: Design Sources

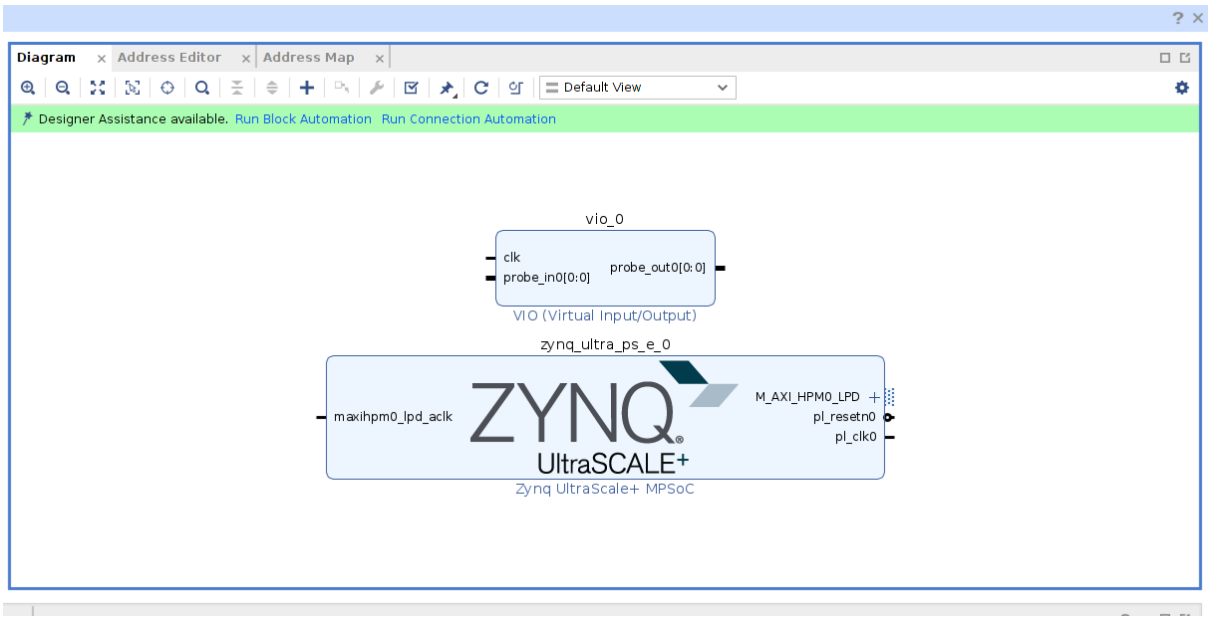
Your block design diagram should now open.



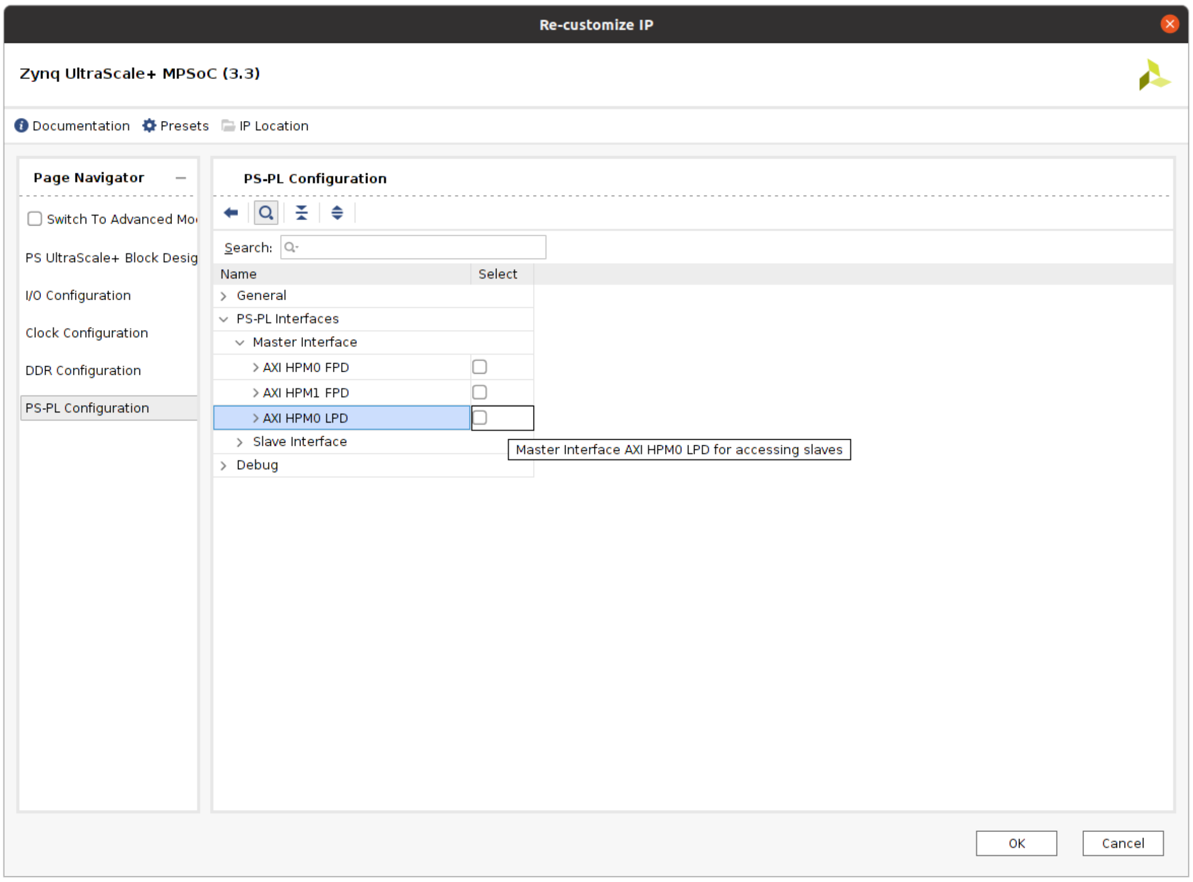
```
INFO: [IP_Flow 19-3899] Cannot get the environment domain name variable for the component vendor name. Setting the vendor name to 'user.org'.
INFO: [IP_Flow 19-3899] Cannot get the environment domain name variable for the component vendor name. Setting the vendor name to 'user.org'.
INFO: [IP_Flow 19-3899] Cannot get the environment domain name variable for the component vendor name. Setting the vendor name to 'user.org'.
INFO: [IP_Flow 19-3899] Cannot get the environment domain name variable for the component vendor name. Setting the vendor name to 'user.org'.
INFO: [IP_Flow 19-3899] Cannot get the environment domain name variable for the component vendor name. Setting the vendor name to 'user.org'.
INFO: [Common 17-14] Message 'IP_Flow 19-3899' appears 100 times and further instances of the messages will be disabled. Use the Tcl command set_msg_config to change the current settings.
create_project: Time (s): cpu = 00:00:22 ; elapsed = 00:00:07 . Memory (MB): peak = 7487.215 ; gain = 43.027 ; free physical = 5369 ; free virtual = 13006
set_property board_part xilinx.com:kv260:part0:1.1 [current_project]
create_bd_design "design_1"
Wrote  : </home/ivica/fan_toggle/fan_toggle.srcs/sources_1/bd/design_1/design_1.bd>
create_bd_design: Time (s): cpu = 00:00:07 ; elapsed = 00:00:05 . Memory (MB): peak = 7515.355 ; gain = 4.871 ; free physical = 5173 ; free virtual = 12897
update_compile_order -fileset sources_1
```

Once here, click the + button to add IP and add the following (you can also press Ctrl + I)

- VIO (Virtual input/Output)
- Zynq Ultrascale+ MPSoc



Double click on the Zynq block and navigate to PS-PL configuration > PS-PL Interfaces and disable AXI HPM0 LPD. Click Ok.

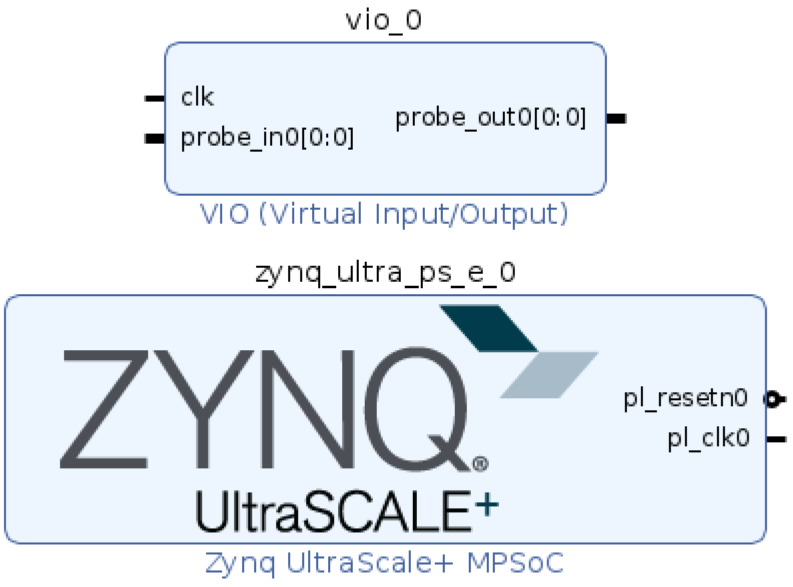


PS-PL Configuration

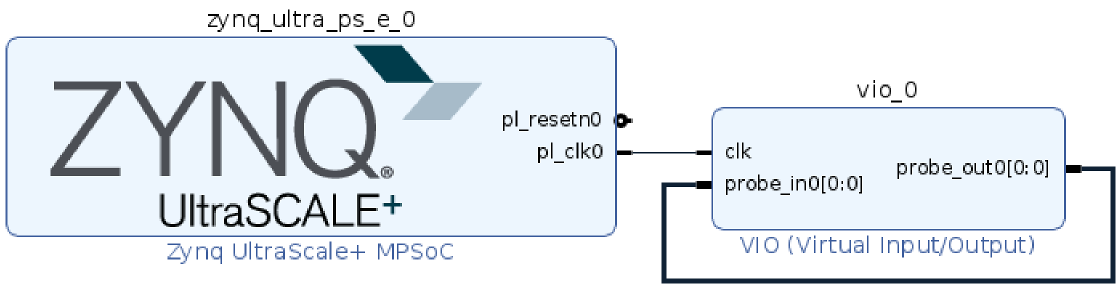
Name	Select
> General	
∨ PS-PL Interfaces	
∨ Master Interface	
> AXI HPM0 FPD	☐
> AXI HPM1 FPD	☐
> AXI HPM0 LPD	☐
> Slave Interface	
> Debug	

Master Interface AXI HPM0 LPD for accessing slaves

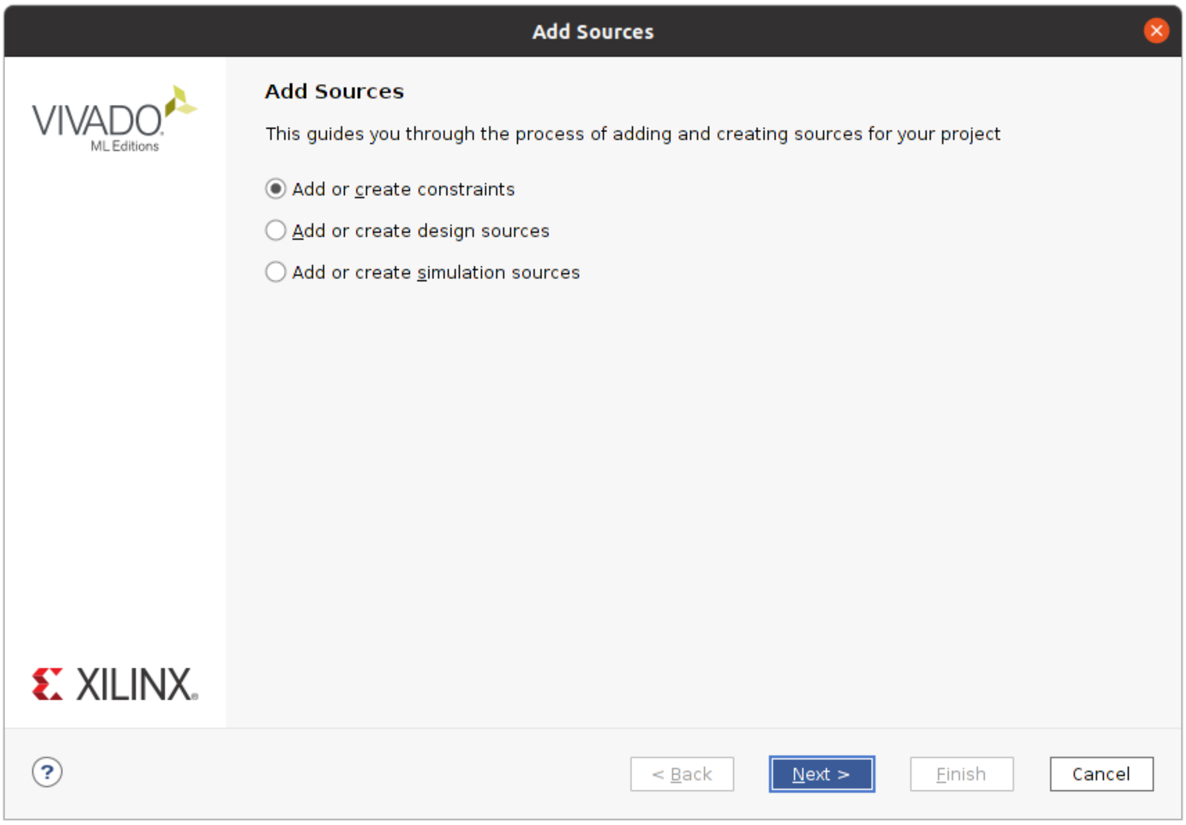
Your block design now should look like this:



Connect the blocks in the following way (Do not perform and block automation). Hint: you can reorganise your layout by doing right click > Regenerate Layout.



We are still missing a connection to our fan in our block diagram. To do so, we need to add a constraint file with an entry for a fan pin. To do so, click on the Sources tab left from the Block Diagram, right-click on the Constraints entry and click add sources.



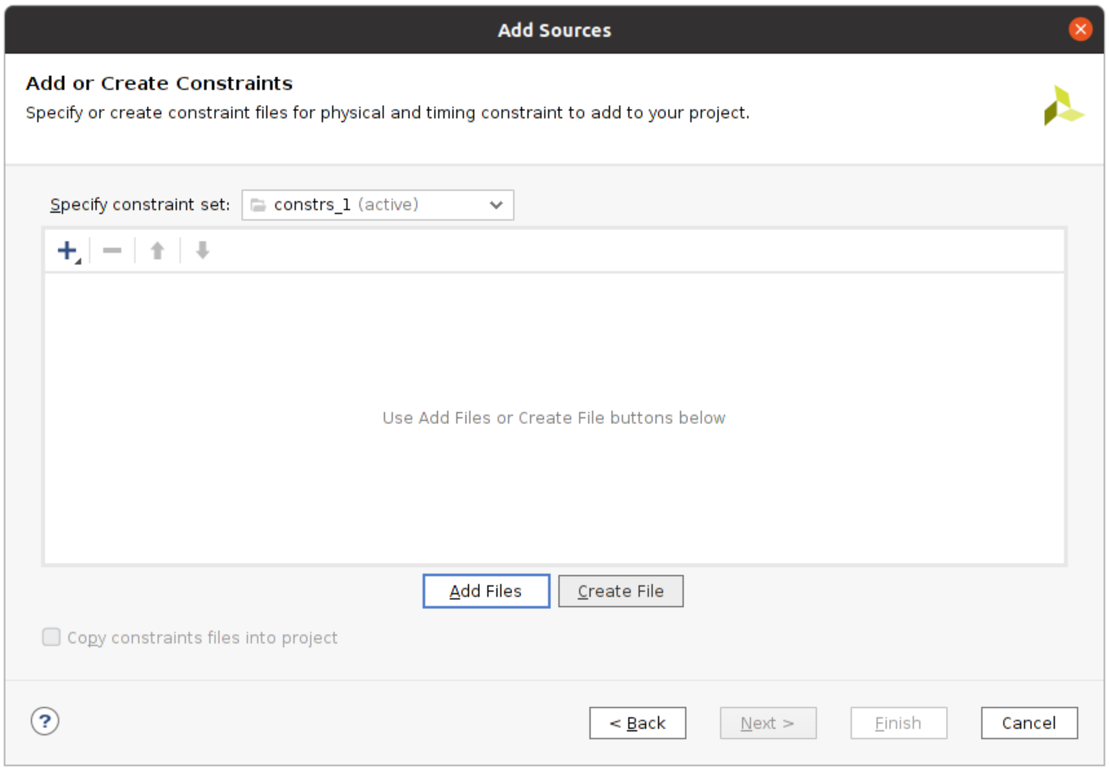
Add Sources

Add Sources

This guides you through the process of adding and creating sources for your project

- [✓ Add or create constraints]
- Add or create design sources
- Add or create simulation sources

Proceed by clicking next. Click on Create File button and name your constraints file. Click Ok and exit the wizard by clicking finish.



Add Sources

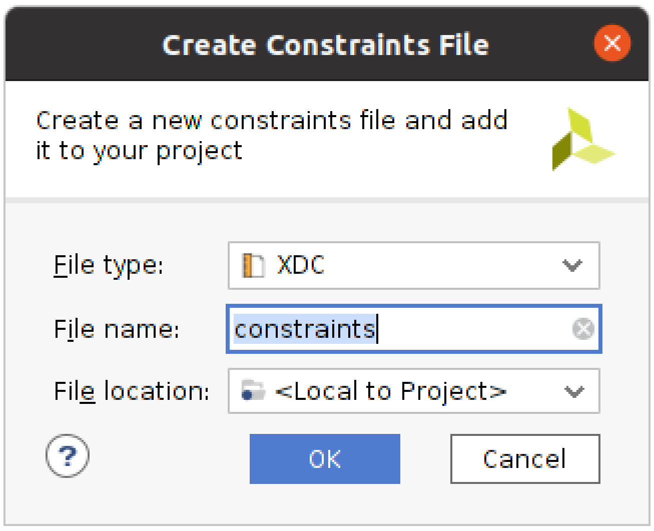
Add or Create Constraints

Specify or create constraint files for physical and timing constraint to add to your project.

Specify constraint set: constrs_1 (active)

Use Add Files or Create File buttons below

Copy constraints files into project

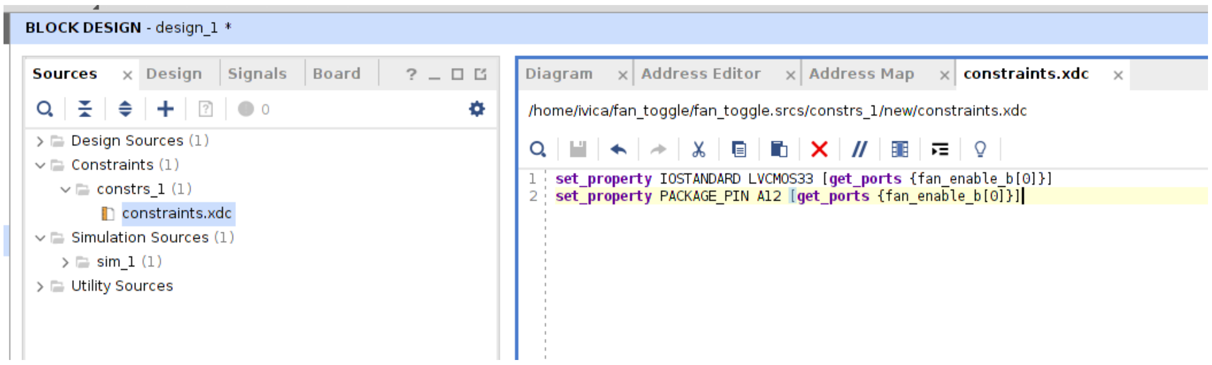


File type: XDC
File name: constraints
File location: <Local to Project>

Open your constraints file from the left side pane under the constraints tab and paste the following:

```
set_property IOSTANDARD LVCMOS33 [get_ports {fan_enable_b[0]}]
set_property PACKAGE_PIN A12 [get_ports {fan_enable_b[0]}]
```

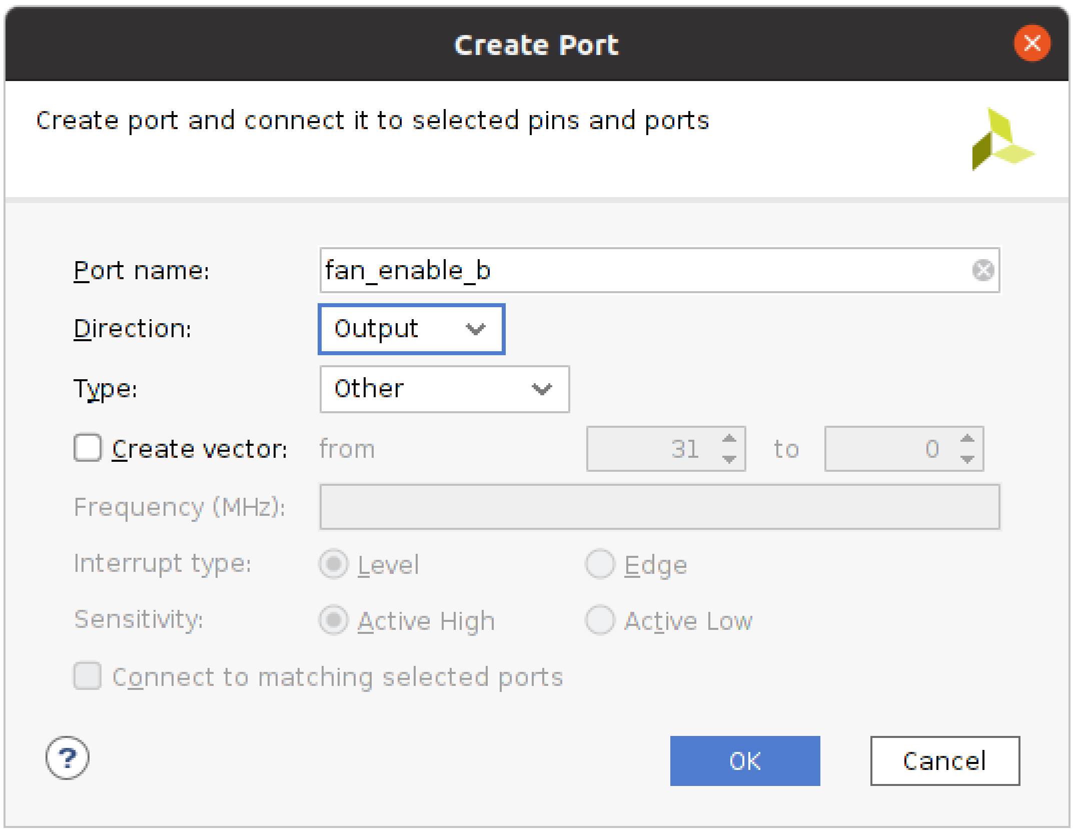
Save the file using Ctrl + S



/home/ivica/fan_toggle/fan_toggle.srcs/constrs_1/new/constraints.xdc

```
set_property IOSTANDARD LVCMOS33 [get_ports {fan_enable_b[0]}]
set_property PACKAGE_PIN A12 [get_ports {fan_enable_b[0]}]
```

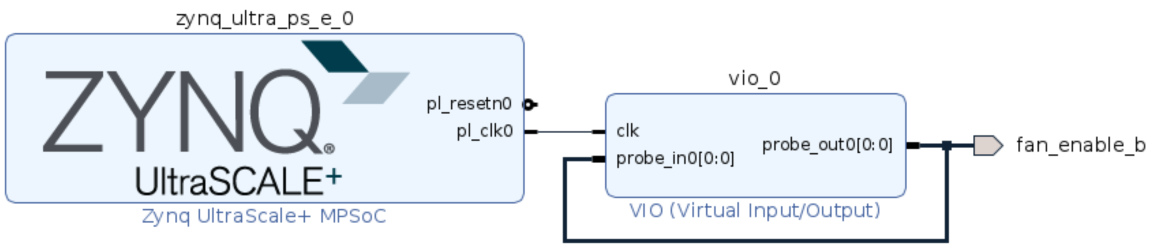
Now navigate back to your block design and right-click on the wire going out from the probe_out and click Create Port. Set your port to be output and name it fan_enable_b[0:0].


Create Port

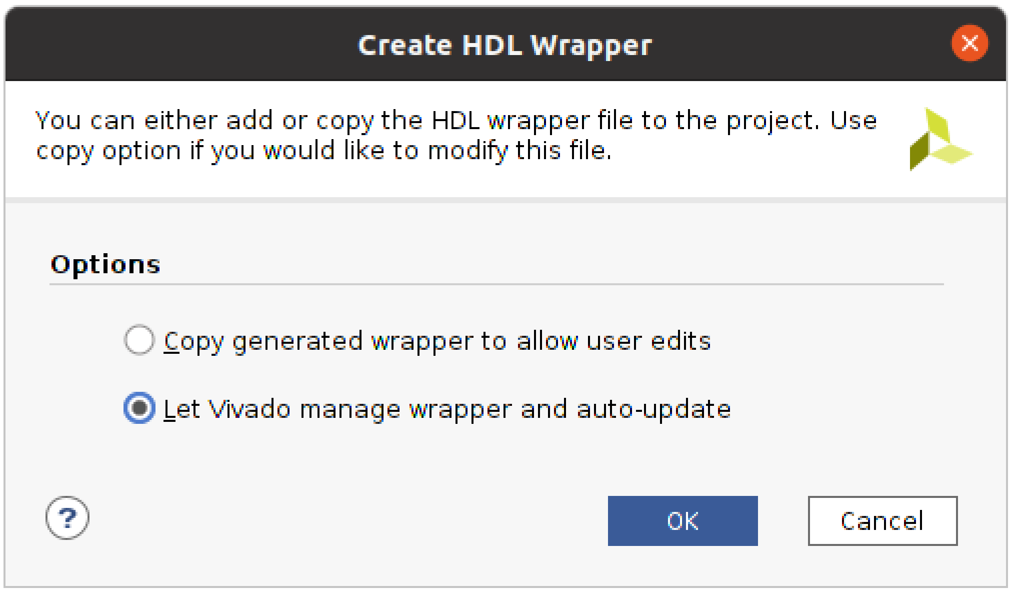
Create port and connect it to selected pins and ports

Port name: fan_enable_b
Direction: Output
Type: Other
☐ Create vector: from 31 to 0
Frequency (MHz):
Interrupt type: ● Level ○ Edge
Sensitivity: ● Active High ○ Active Low
☐ Connect to matching selected ports

If Vivado doesn't automatically connect your port to probe_out do it manually, Your diagram should now look like this.



You are now ready to create an HDL Wrapper around your block design. To do so, right-click on your block design under Design Sources on your left side pane and click Create HDL Wrapper. Select "Let Vivado manage wrapper and auto-update".



Create HDL Wrapper

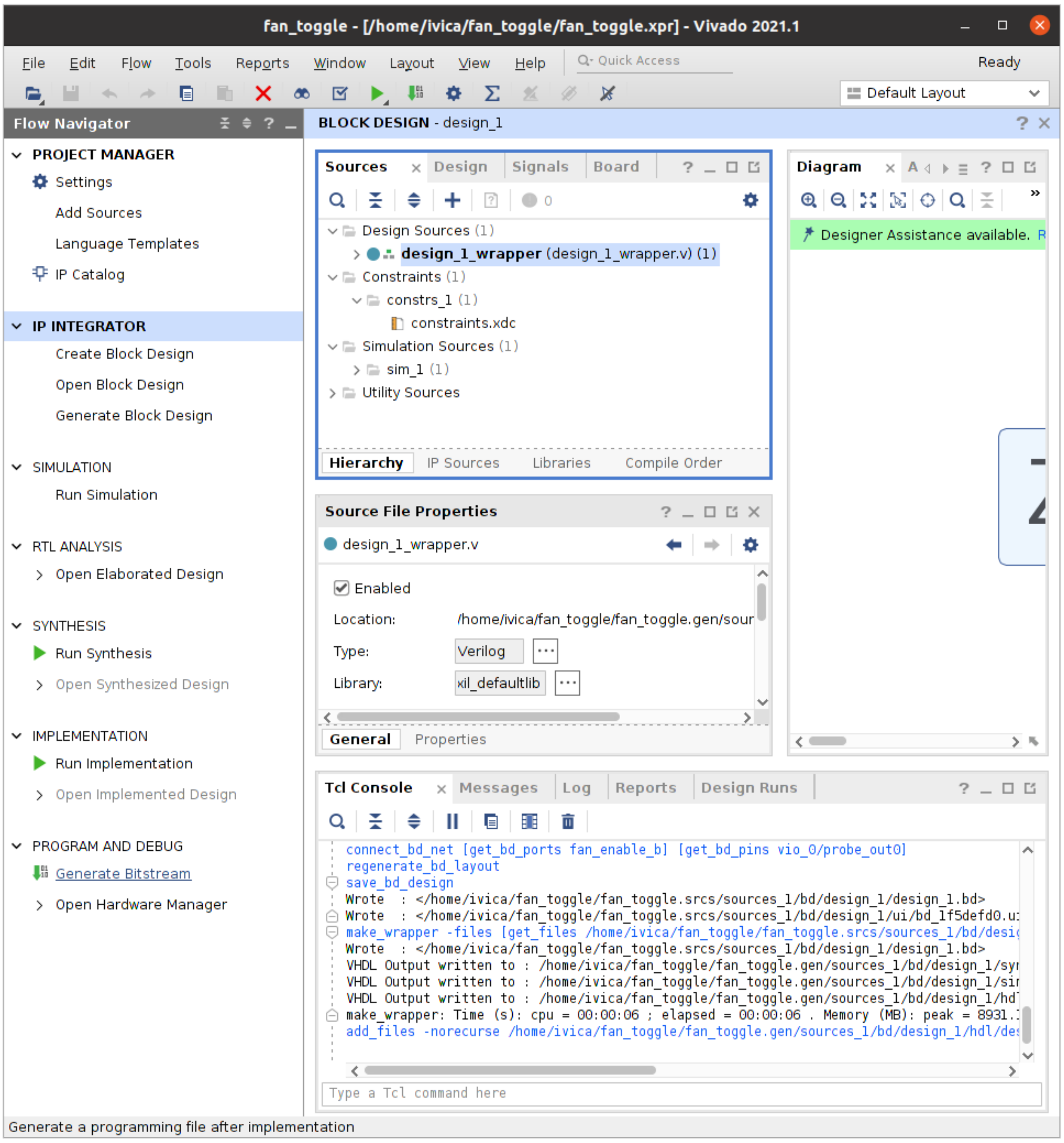
You can either add or copy the HDL wrapper file to the project. Use copy option if you would like to modify this file.

Options

( ) Copy generated wrapper to allow user edits

(●) Let Vivado manage wrapper and auto-update

We can now click Generate Bitstream on the left-hand side of our Vivado tool to generate our bitstream which we are going to use to program our Kria board.



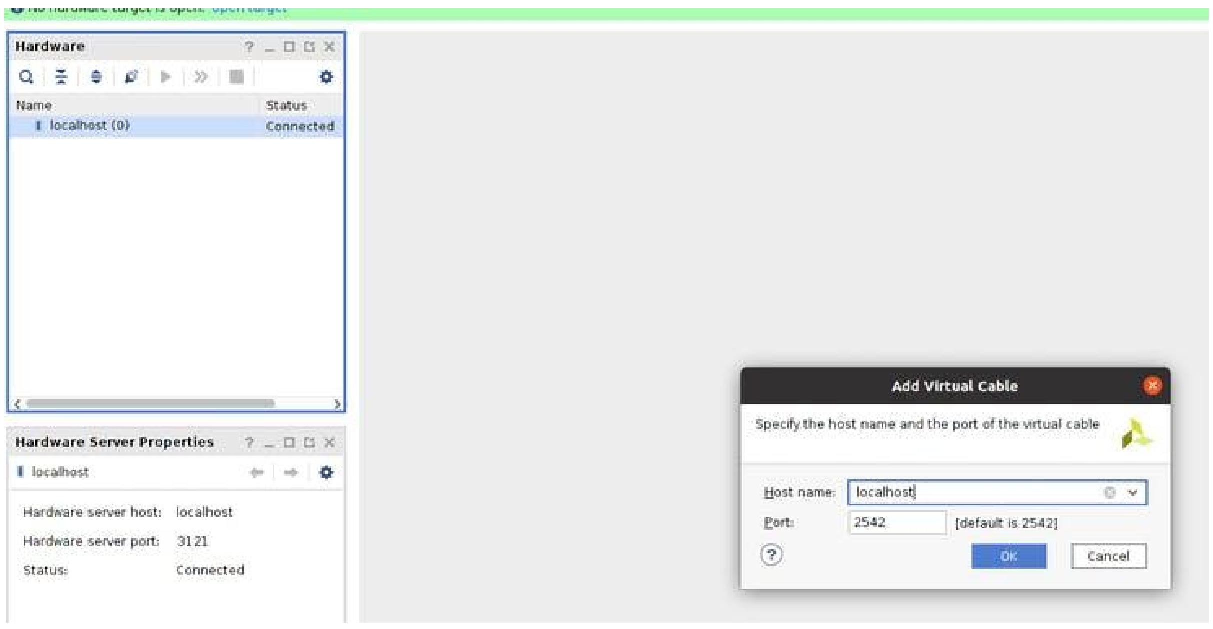
Enabled
Location: /home/ivica/fan_toggle/fan_toggle.gen/sour
Type: Verilog
Library: xil_defaultlib

```
connect_bd_net [get_bd_ports fan_enable_b] [get_bd_pins vio_0/probe_out0]
regenerate_bd_layout
save_bd_design
Wrote  : </home/ivica/fan_toggle/fan_toggle.srcs/sources_1/bd/design_1/design_1.bd>
Wrote  : </home/ivica/fan_toggle/fan_toggle.srcs/sources_1/bd/design_1/ui/bd_1f5defd0.u
make_wrapper -files [get_files /home/ivica/fan_toggle/fan_toggle.srcs/sources_1/bd/desi
Wrote  : </home/ivica/fan_toggle/fan_toggle.srcs/sources_1/bd/design_1/design_1.bd>
VHDL Output written to : /home/ivica/fan_toggle/fan_toggle.gen/sources_1/bd/design_1/sy
VHDL Output written to : /home/ivica/fan_toggle/fan_toggle.gen/sources_1/bd/design_1/si
VHDL Output written to : /home/ivica/fan_toggle/fan_toggle.gen/sources_1/bd/design_1/hd
make_wrapper: Time (s): cpu = 00:00:06 ; elapsed = 00:00:06 . Memory (MB): peak = 8931.
add_files -norecurse /home/ivica/fan_toggle/fan_toggle.gen/sources_1/bd/design_1/hdl/de
```

Programing Kria

Once the bitstream is successfully created, click on Open Hardware Manager. Select Open target and select Auto Connect

Right-click "localhost (0)" and select Add Xilinx Virtual Cable (XVC). For the hostname enter localhost and click Ok.


Hardware

Name	Status
localhost (0)	Connected

Hardware Server Properties

localhost

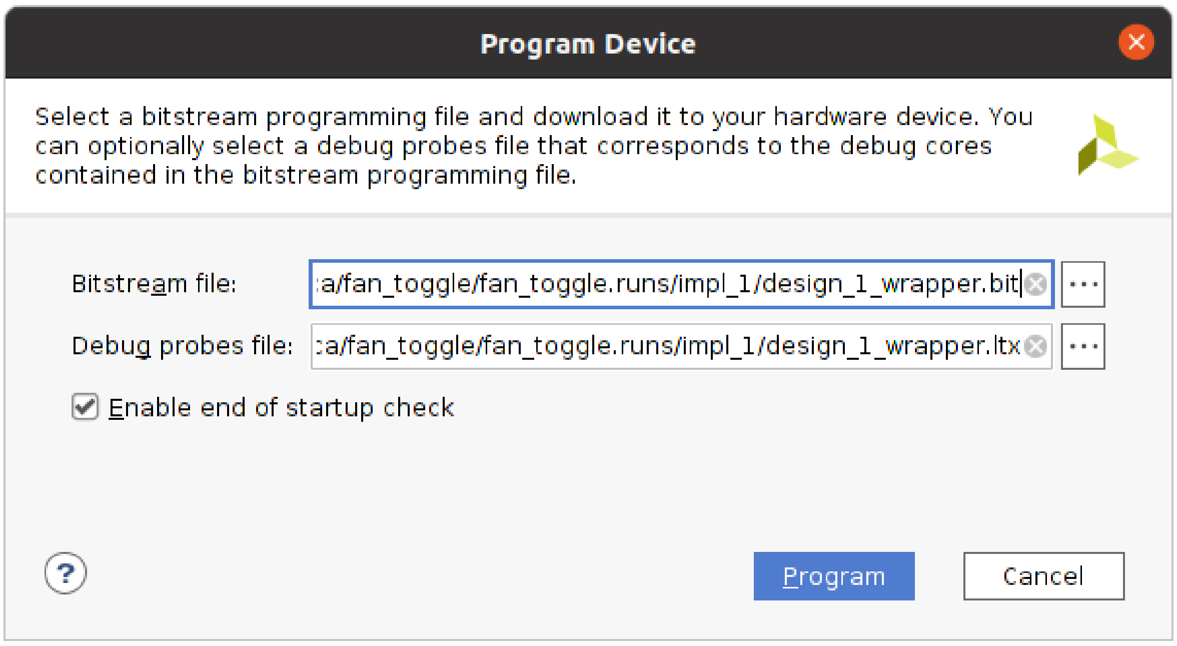
Hardware server host: localhost
Hardware server port: 3121
Status: Connected

Add Virtual Cable

Specify the host name and the port of the virtual cable

Host name: localhost
Port: 2542 [default is 2542]

Click on Program Device. Path to your bitstream file and debug file should be automatically configured.



Program Device

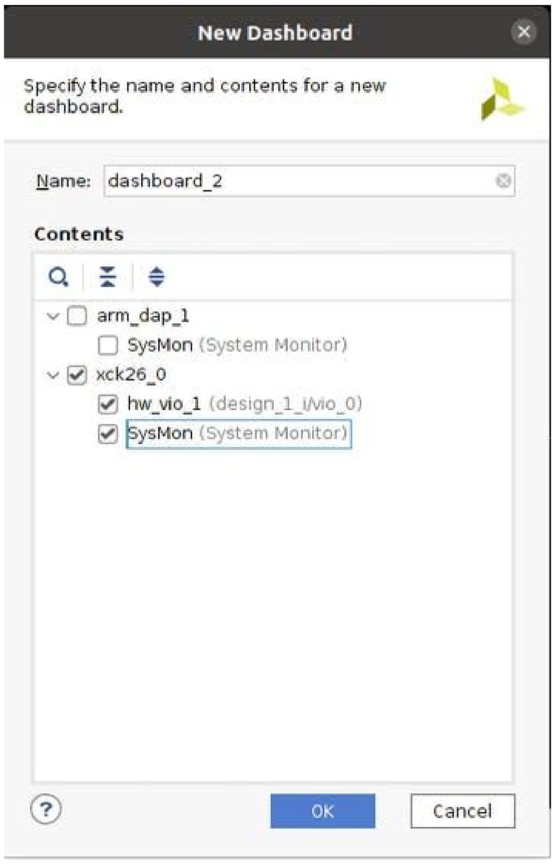
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: :a/fan_toggle/fan_toggle.runs/impl_1/design_1_wrapper.bit

Debug probes file: :a/fan_toggle/fan_toggle.runs/impl_1/design_1_wrapper.ltx

☑ Enable end of startup check

After flashing the Kria, on the left hand side double click on xck26_0 and select both vio_o and System Monitor. Click OK.



New Dashboard

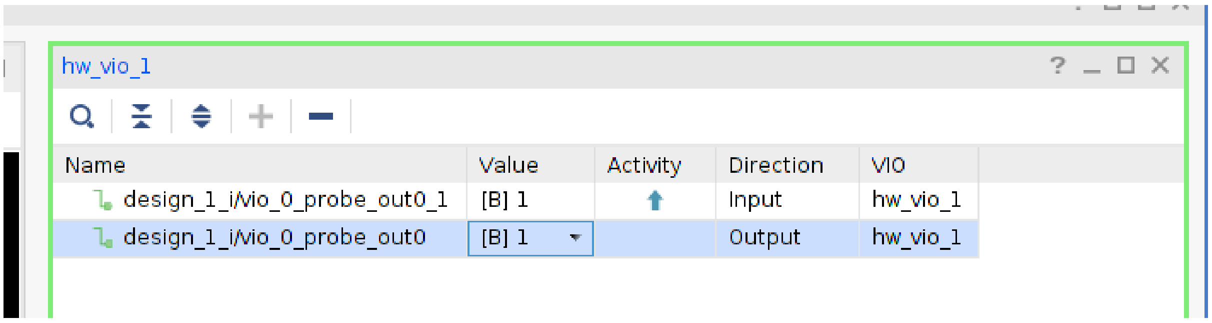
Specify the name and contents for a new dashboard.

Name: dashboard_2

Contents

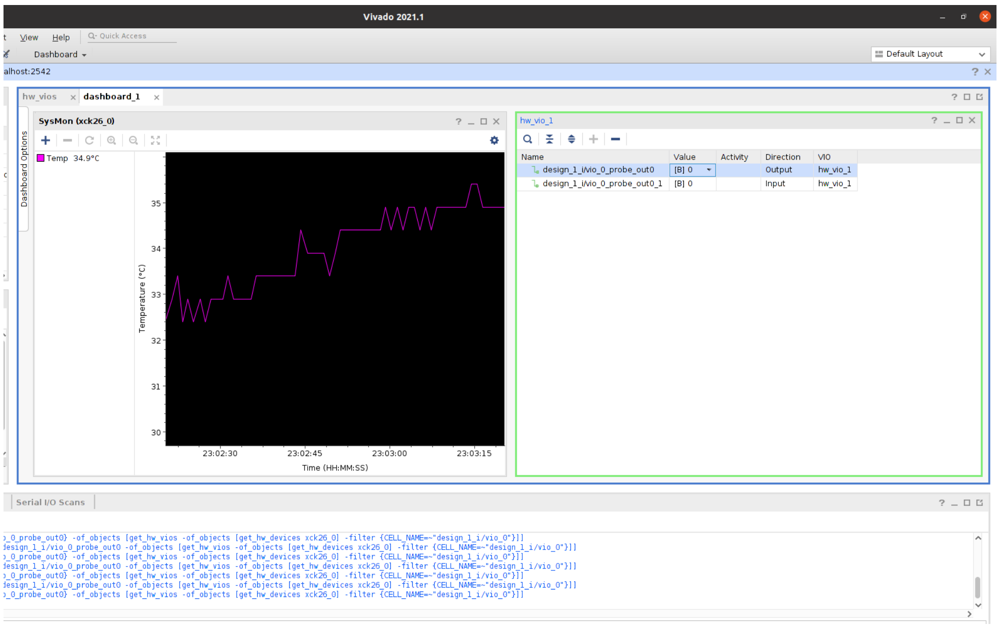
- ☐ arm_dap_1
  - ☐ SysMon (System Monitor)
- ☑ xck26_0
  - ☑ hw_vio_1 (design_1_i/vio_0)
  - ☑ SysMon (System Monitor)

On your newly open hw_vio pane add all available probes using the + button. Now you can change your fan state by setting the output on the probe_out port.


hw_vio_1

Name	Value	Activity	Direction	VIO
design_1_i/vio_0_probe_out0_1	[B] 1	⬆	Input	hw_vio_1
design_1_i/vio_0_probe_out0	[B] 1		Output	hw_vio_1

We can see that turning off the fan results in an increase in temperature.



hw_vio_1

Name	Value	Activity	Direction	VIO
design_1_i/vio_0_probe_out0	[B] 0		Output	hw_vio_1
design_1_i/vio_0_probe_out0_1	[B] 0		Input	hw_vio_1

```
o_0_probe_out0} -of_objects [get_hw_vios -of_objects [get_hw_devices xck26_0] -filter {CELL_NAME=~"design_1_i/vio_0"}]]
design_1_i/vio_0_probe_out0 -of_objects [get_hw_vios -of_objects [get_hw_devices xck26_0] -filter {CELL_NAME=~"design_1_i/vio_0"}]]
o_0_probe_out0} -of_objects [get_hw_vios -of_objects [get_hw_devices xck26_0] -filter {CELL_NAME=~"design_1_i/vio_0"}]]
design_1_i/vio_0_probe_out0 -of_objects [get_hw_vios -of_objects [get_hw_devices xck26_0] -filter {CELL_NAME=~"design_1_i/vio_0"}]]
o_0_probe_out0} -of_objects [get_hw_vios -of_objects [get_hw_devices xck26_0] -filter {CELL_NAME=~"design_1_i/vio_0"}]]
design_1_i/vio_0_probe_out0 -of_objects [get_hw_vios -of_objects [get_hw_devices xck26_0] -filter {CELL_NAME=~"design_1_i/vio_0"}]]
o_0_probe_out0} -of_objects [get_hw_vios -of_objects [get_hw_devices xck26_0] -filter {CELL_NAME=~"design_1_i/vio_0"}]]
```

5. Notes

For some unknown reason, for VIO to work properly you need to have an SD card inserted in your Kria Dev Board and reset the board manually using the onboard reset button after turning it on, before flashing the firmware using Hardware Management.