

simpleCounter Documentation

Release 1.0.0

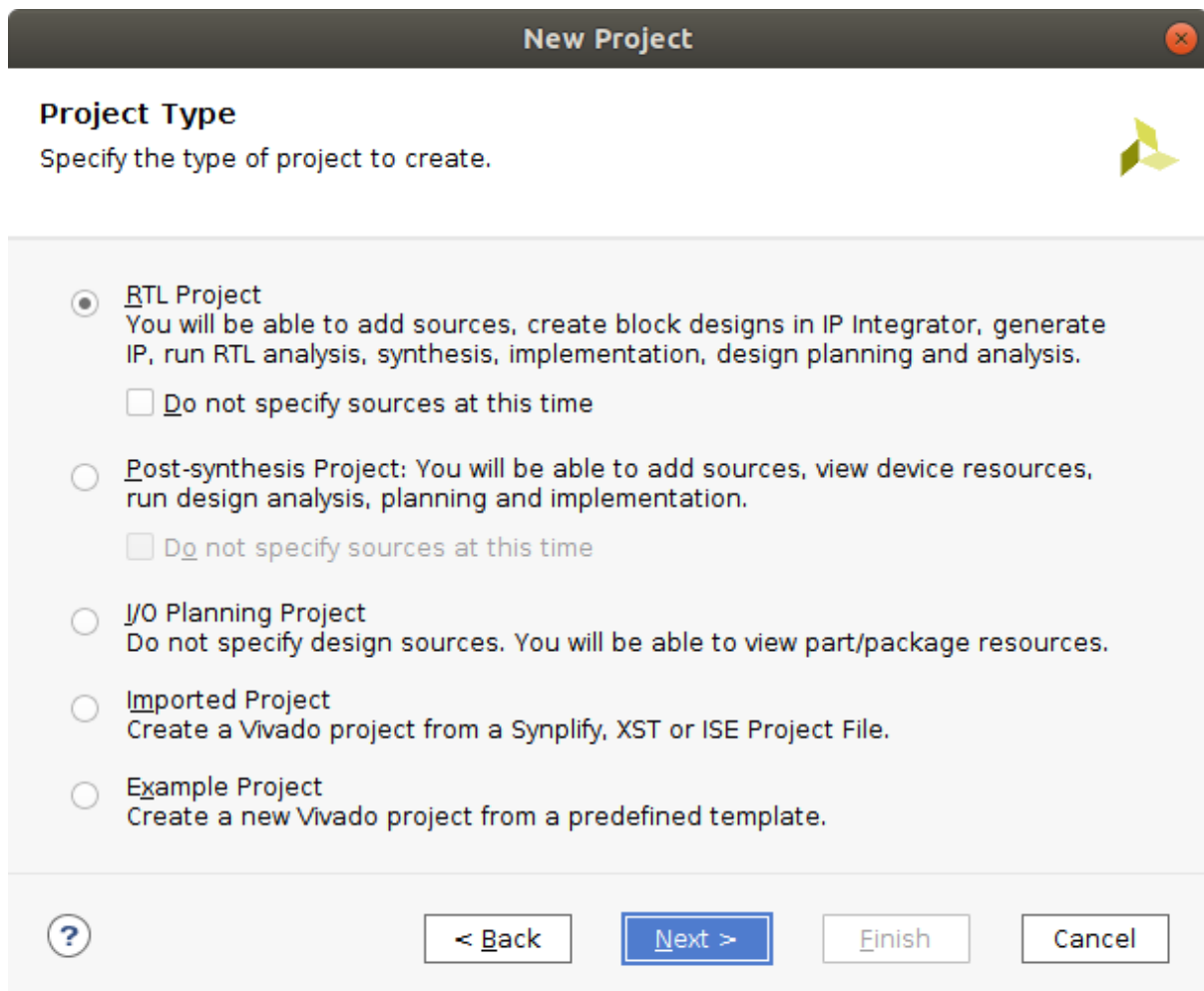
©

August 12, 2022

1	Vivado/Vivado	1
1.1	Uploading the Verilog HDL into Vivado	1
1.2	Getting the Netlist from Vivado	2
1.3	Exporting the Netlist.	4
1.4	Triplicating the design - SpyDrNet TMR	4
1.5	SpyDrNet TMR to Vivado.	5
1.6	Vivado to Bitstream	6
1.7	Verifying if the Design Works.	8
2	Yosys/F4PGA	9
2.1	Setting up Yosys and F4PGA	9
2.2	Running Yosys, SpyDrNet TMR, and F4PGA	9
2.3	Verifying if the Design Works	10

1.1 Uploading the Verilog HDL into Vivado

The first step is to create 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
- ☐ **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.

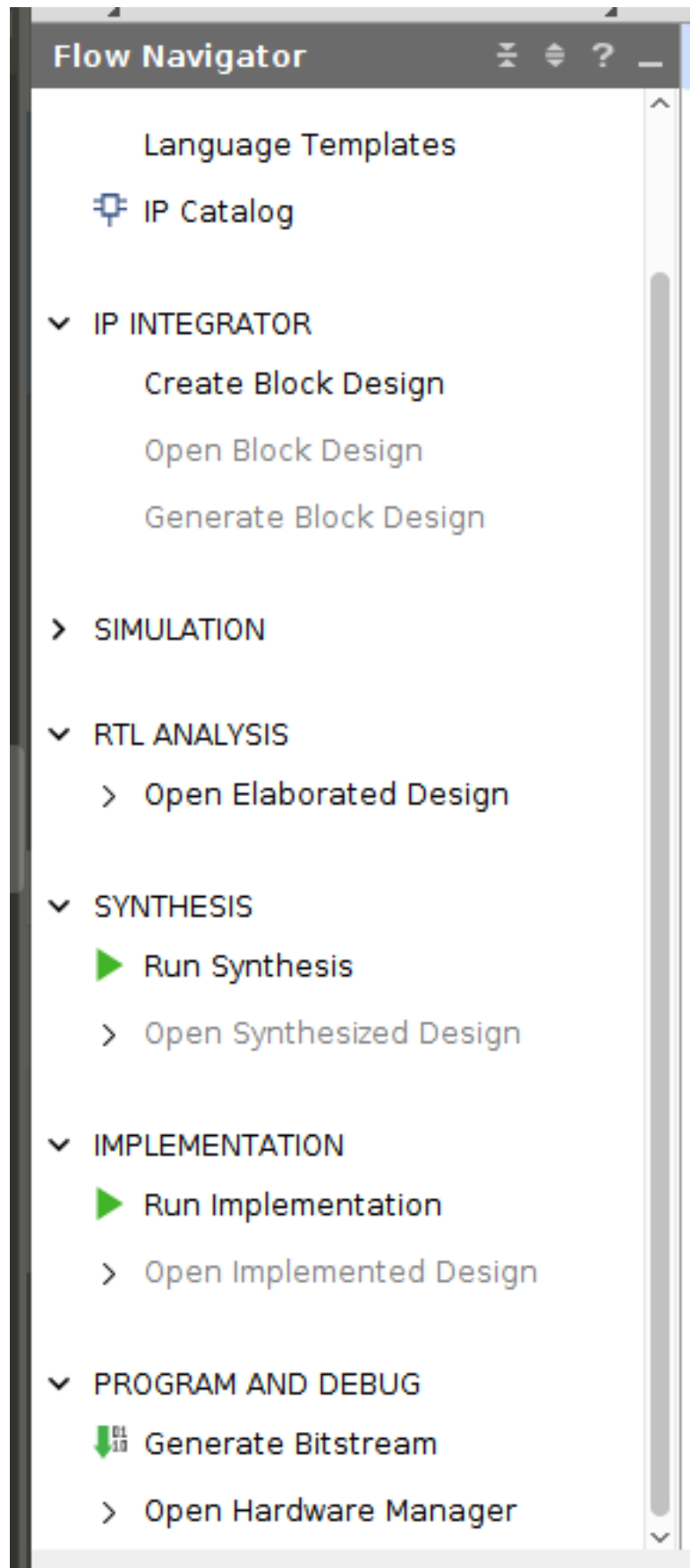
? < Back Next > Finish Cancel

Add **simpleCounter.sv** to the project.

Download: `simpleCounter.sv`

1.2 Getting the Netlist from Vivado

After adding the simpleCounter.sv file to your project, go to the “Flow Navigator” window on the left hand side of the screen, and click on “Run Synthesis.”

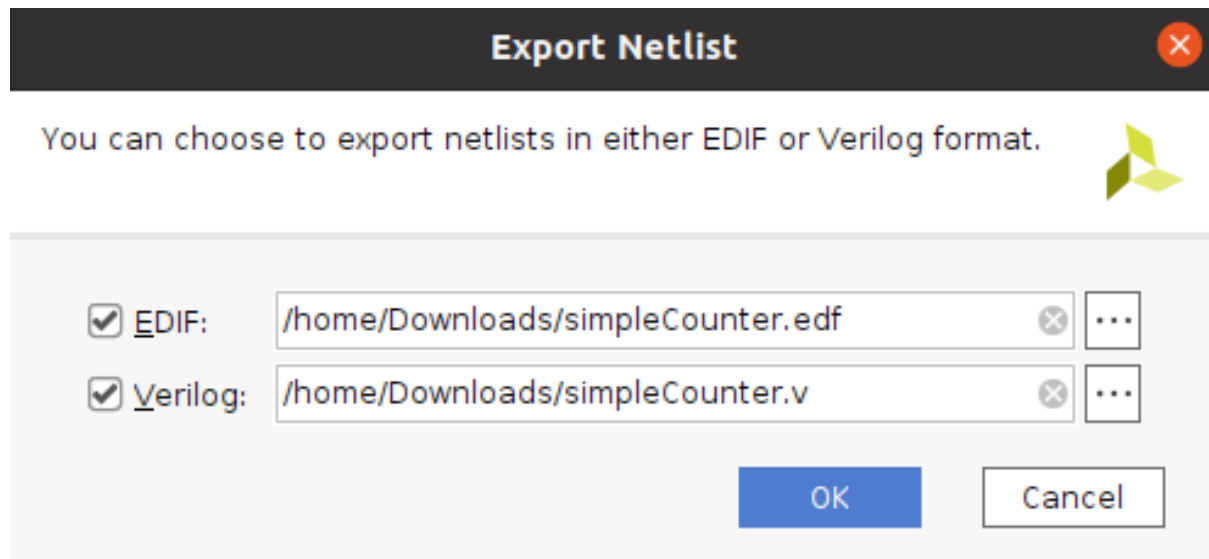


1.3 Exporting the Netlist

Once synthesis has been run expand the “Open Synthesized Design” tab under the “Synthesis” section, and click on the “Schematic” option.

To Export the Netlist click on file in the upper left, go down to Export, then click on Export Netlist.

A window pops up with the option to export EDIF and Verilog Netlists. Select the desired file format and file destination.



- If downloading an EDIF file change the file type from .edn to .edf

1.3.1 Using the tcl command line

- To export the netlist using the tcl command line in Vivado type **write_edif simpleCounter.edf** or **write_verilog simpleCounter.v**

1.4 Triplicating the design - SpyDrNet TMR

If using the .edf file, the following code runs the netlist through SpyDrNet TMR.

```
import spydrnet as sdn
from spydrnet.uniquify import uniquify
from spydrnet_tmr.support_files.vendor_names import XILINX
from spydrnet_tmr.apply_tmr_to_netlist import apply_tmr_to_netlist

# Parse in the downloaded .edf netlist
netlist = sdn.parse("simpleCounter.edf")

# Makes all instances unique in the netlist
uniquify(netlist)

# Gets all of the hinstances in the design but leaves out VCC GND and IBUF as
those should not be triplicated
hinstances_to_replicate = list(
    netlist.get_hinstances(
        recursive=True, filter=lambda x: x.item.reference.is_leaf() is True
```



```

x.name.lower()
x.item.reference.name ))

# Gets all of the OUT hports in the design
hports_to_replicate = list(netlist.get_hports(filter=lambda x: x.item.direction is
sdn.OUT))

valid_voter_point_dict = dict()
valid_voter_point_dict["after_ff"] = [
    *hinstances_to_replicate,
    *hports_to_replicate,
]

# Triplicates the design and inserts the voters
apply_tmr_to_netlist(
    netlist,
    XILINX,
    hinstances_and_hports_to_replicate=[
        *hinstances_to_replicate,
        *hports_to_replicate,
    ],
    valid_voter_point_dict=valid_voter_point_dict,
)

# Compose the triplicated netlist
netlist.compose("simpleCounter_tmr.edf")

```

Download: [edf_tmr_script.py](#)

If using the .v file the following code can be downloaded to triplicate the design

Note: To be able to run the design through Spydrnet TMR any instances, ports, or wires that have a as the first character and a space as the last character need to have them removed before replication and then replaced after triplication, but before composing.

Download: [verilog_tmr_script.py](#)

1.5 SpyDrNet TMR to Vivado

Create new project in Vivado and choose Post-synthesis 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 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 Project File.

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

If working with a .edf file, upload **simpleCounter_tmr.edf** that was just created and **simpleCounter_tmr.xdc** to the project.

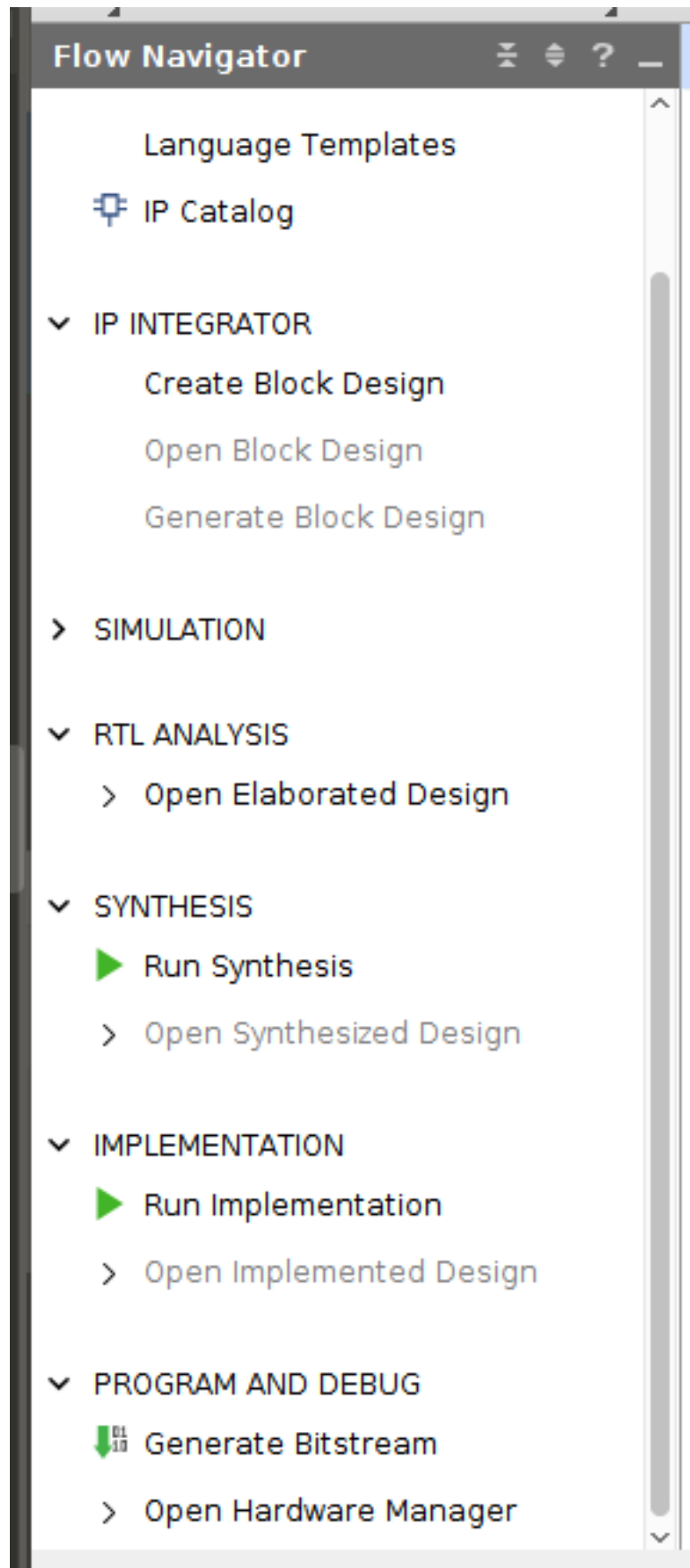
If working with a .v file, upload **simpleCounter_tmr.v** and **simpleCounter_tmr.xdc** to the project.

Download: `simpleCounter_tmr.xdc`

1.6 Vivado to Bitstream

The verilog netlist asks for a top module to be specified. *Click on auto find*

After adding the source files for your project, go to the “Flow Navigator” window on the left hand side of the screen, and click on “Generate Bitstream” under “Program and Debug.”



Once the Bitstream has been generated click on “Open Hardware Manager” under the tab “Program

and Debug”, click on “Open Target” then click on “Auto Connect”, next click on “Program Device” then click on the box that pops up. A “Program Device” window should open up, click on program to download the bitstream to your device.

1.7 Verifying if the Design Works

The counter in this design increments when btnc is pressed. Once it reaches a count of 15 (4 LEDs are on) it rolls over to 0 and starts counting up again. If at any point btnc is pressed the count is reset to 0. Since this is a TMR design there should be 3 sets of 4 leds in total.

- Set 1 (LEDS 0-3), Set 2 (LEDS 5-8), Set 3 (LEDS 10-13)

Note

1. This example was designed using a BASYS 3 board. If a different FPGA is being used the simpleCounter_tmr.xdc file will need to be modified accordingly.
2. The Verilog Netlist portion of this example is not functioning properly. The design composes with no problems, but in Vivado the design is not getting the desired output.

Files:

```
simpleCounter.sv
simpleCounter.edf
simpleCounter_tmr.edf
simpleCounter_tmr.xdc
edf_tmr_script.py
simpleCounter.v
simpleCounter_tmr.v
verilog_tmr_script.py
```

Yosys/F4PGA

2.1 Setting up Yosys and F4PGA

Install F4PGA and follow all of the steps [here](#).

1. Using the command line type the following

```
>>> cd ~/opt/f4pga/xc7/share/f4pga/scripts/xc7
```

2. Open synth.tcl

- Add the “-nocarry” option to all of the synth_xilinx commands.

3. Open conv.tcl

- Add “hierarchy -purge_lib” as the first command
- Add option “-blackbox” to both write_blif commands

2.1.1 Download all of the following files

```
simpleCounter.sv
simpleCounter.xdc
new_constraints.txt
tmr_script.py
common.mk
Makefile
```

2.2 Running Yosys, SpyDrNet TMR, and F4PGA

Once all of the files have been downloaded type navigate to their location and type in the command line

```
>>> make -C .
```

or

```
>>> make download -C .
```

Note: This will run Yosys and F4PGA then download the bitstream to the board

2.2.1 Programming Device

To download the bitstream to the board using the command line

```
>>> make download
```

2.3 Verifying if the Design Works

The counter in this design increments when btnc is pressed. Once it reaches a count of 15 (4 LEDs are on) it rolls over to 0 and starts counting up again. If at any point btnc is pressed the count is reset to 0. Since this is a TMR design there should be 3 sets of 4 leds in total.

- Set 1 (LEDS 0-3), Set 2 (LEDS 5-8), Set 3 (LEDS 10-13)

Note

1. This example was designed using a BASYS 3 board. If a different FPGA is being used the simple-Counter.xdc and the new_constraints.txt file will need to be modified accordingly.
2. Make sure all of the files are in the same location. If not the path to the file will need to be specified in the Makefile
3. If getting errors with common.mk make sure that all of the indents are tabs and not spaces.

-
- [genindex](#)
 - [modindex](#)
 - [search](#)

