Xilinx Vivado IDE and the Arty A7 FPGA

Introduction

This guide should get you up and running with Xilinx Vivado IDE (version 2018.2 at the time of writing). The target board used for this tutorial is the Arty A7-35T FPGA but you can still use this guide for other FPGA boards with little modifications.

Installation

1. Download the Vivado webPack from Xilinx's website (https://www.xilinx.com/support/download.html). You will need to create an account to download Vivado and also for generating a license key later on this tutorial.
2.

Creating a new project

1. Launch the Vivado IDE. From the start page, select "Create Project".
2. A dialog box (see Fig. 1) will open describing the steps you will be taken to create a project. Click "Next" to continue.

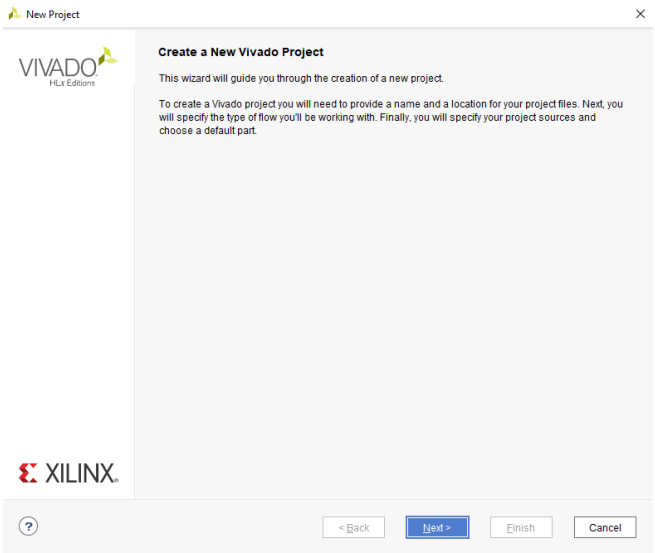

Figure 1: New Project Window

3. The next step is to provide a name for the new project. Vivado will use this name when generating the project's folder structure. For this tutorial, set the name of the project as "updown_counter" (see Fig. 2). After setting the project's name, click "Next" to continue.

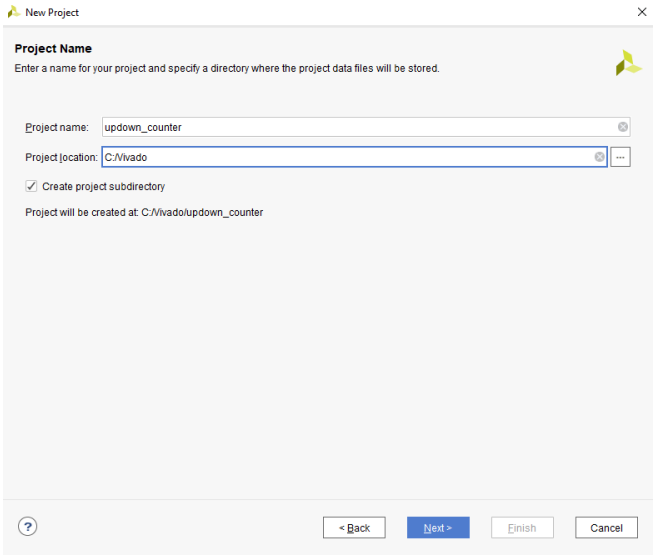

Figure 2: Setting project name

4. Now that the project has a name and a place to save its files, the type of project needs to be selected. For this tutorial, we will create our digital designs from scratch. Select "RTL Project" and check "Do not specify sources at this time" (see Fig. 3). We will create the source files later on this tutorial. Click "Next" to continue.

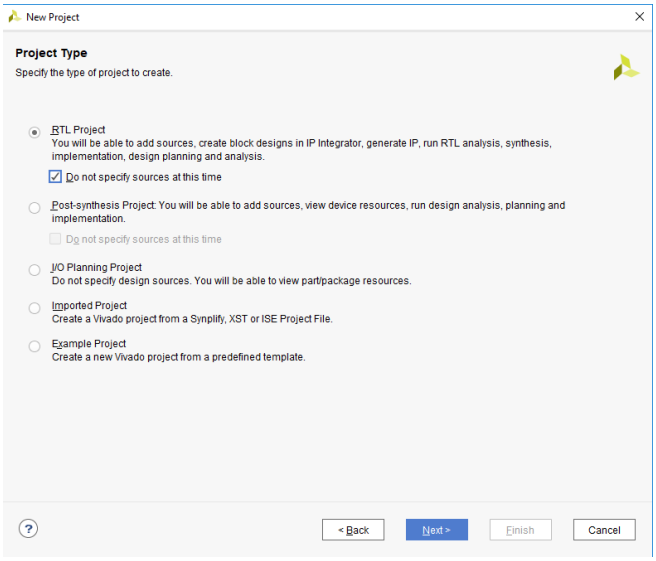

Figure 3: Setting project type

5. Now, we will set the target device for our project (see Fig. 4). Click the "Boards" tab at the top of the dialog box. Under "vendor", select "digilentinc.com" while under "Name", select "Arty A7-35". Select the

device from the search results and click "Next".

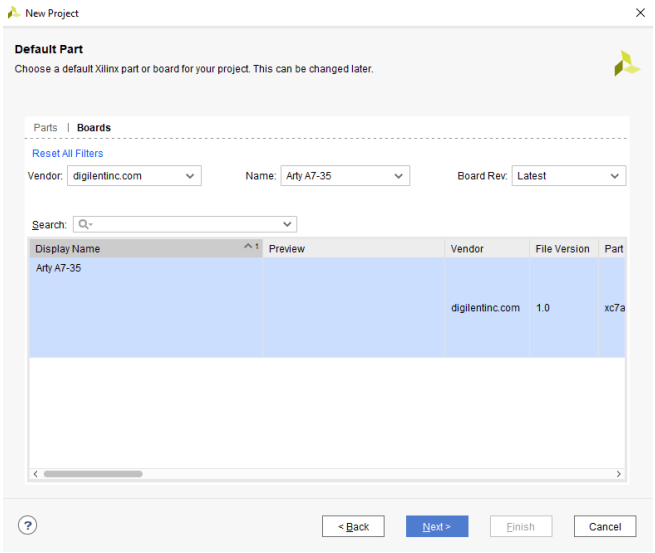

Figure 4: Setting target device

6. The next window will show the summary of the project to be created. Verify that the project settings are correct (see Fig. 5). Click "Finish" to create the project.

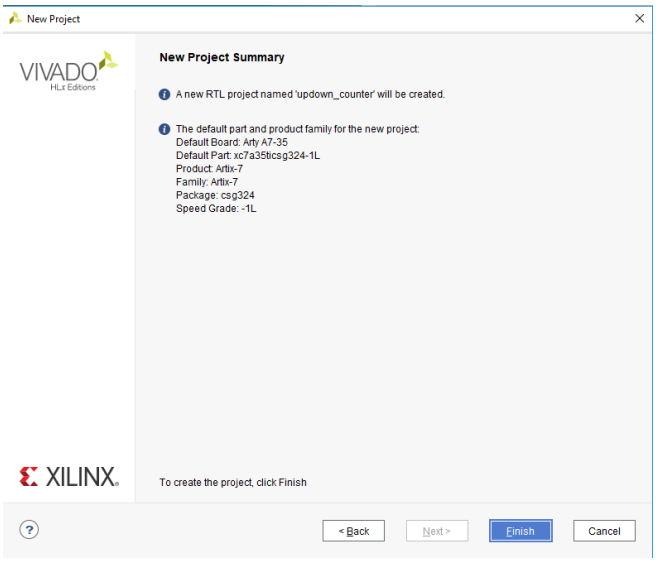

Figure 5: Project Summary

7. After creating the project, the "Project Manager" window will open (see Fig. 6).

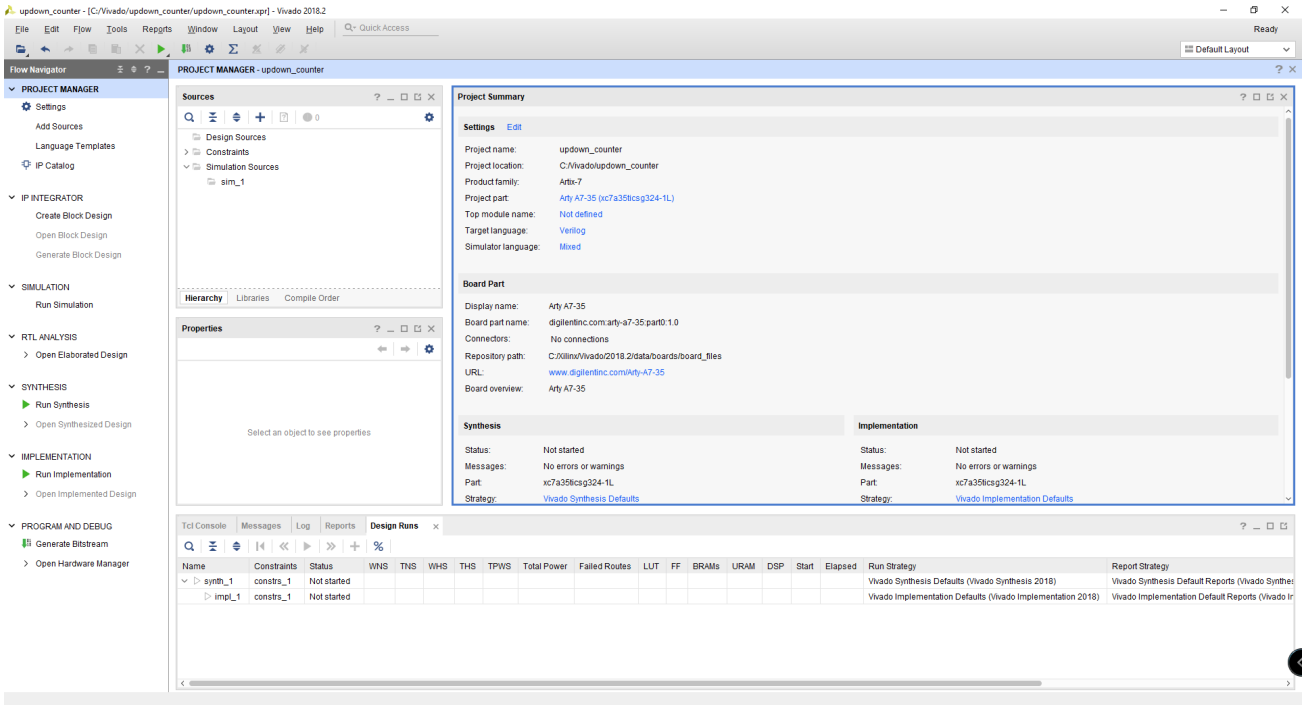

Figure 6: Project Manager window

Adding a constraint file

Now that we have created our project, we need to add a constraint file which will allow us to connect our HDL code with the physical pins of the FPGA. For this tutorial, we will use the default Xilinx Design Constraint (XDC) file provided by Digilent which can be found on this repository (https://github.com/Digilent/digilent-xdc). The repository contains a list of XDC files targeted for each of the FPGA boards produced by Digilent. For this tutorial, we will use the "Arty-A7-35-Master.xdc".

1. Under the "Flow Navigator" pane, click "Add Sources". This will open the "Add Sources" dialog box shown in Fig. 7. In the dialog box, select "Add or create constraints" and click "Next".

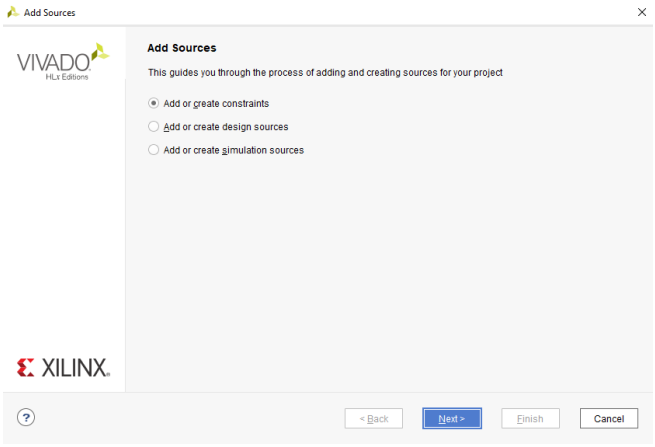

Figure 7: Add Sources Dialog Box

2. In the "Add or Create Constraints" dialog box (see Fig. 8), select "Add Files".

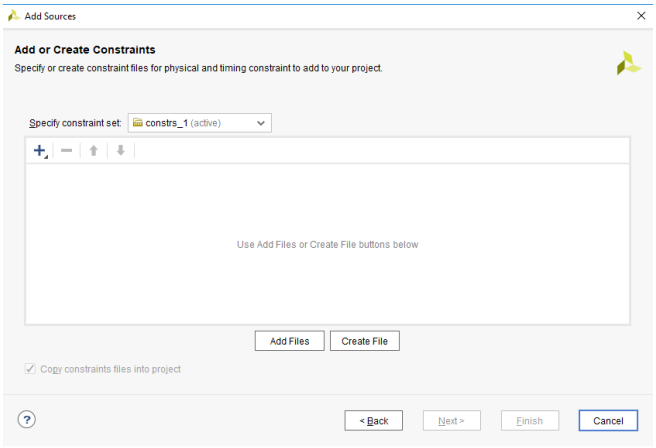

Figure 8: Add Constraint File

3. The next step is to locate the XDC file to be added. Make sure to select "Arty-A7-35-Master.xdc" (see Fig. 9. After selecting the XDC file, click "OK".

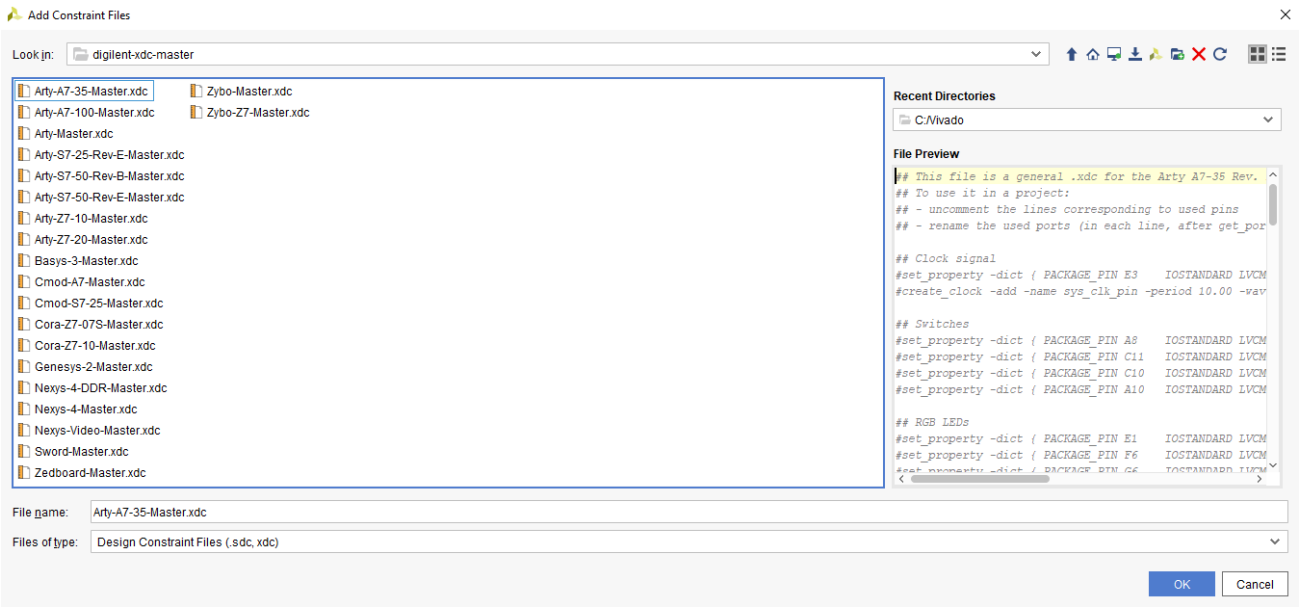

Figure 9: Select Constraint File

4. You will be prompted back to the "Add or Create Constraints" dialog box but now with the XDC file added (see Fig 10). Click "Finish" to continue.

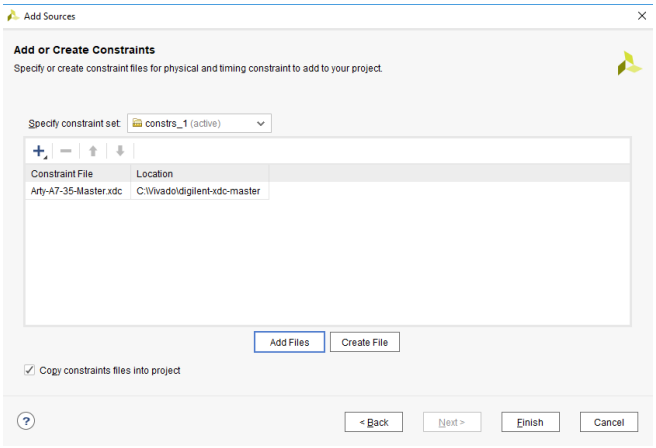

Figure 10: XDC file added

5. After adding the constraint file, you will be prompted back to the "Project Manager" window. Notice that the added constraint file is listed under "Constraints - constr_1" in the "Project Manager" window (see Fig. 11).

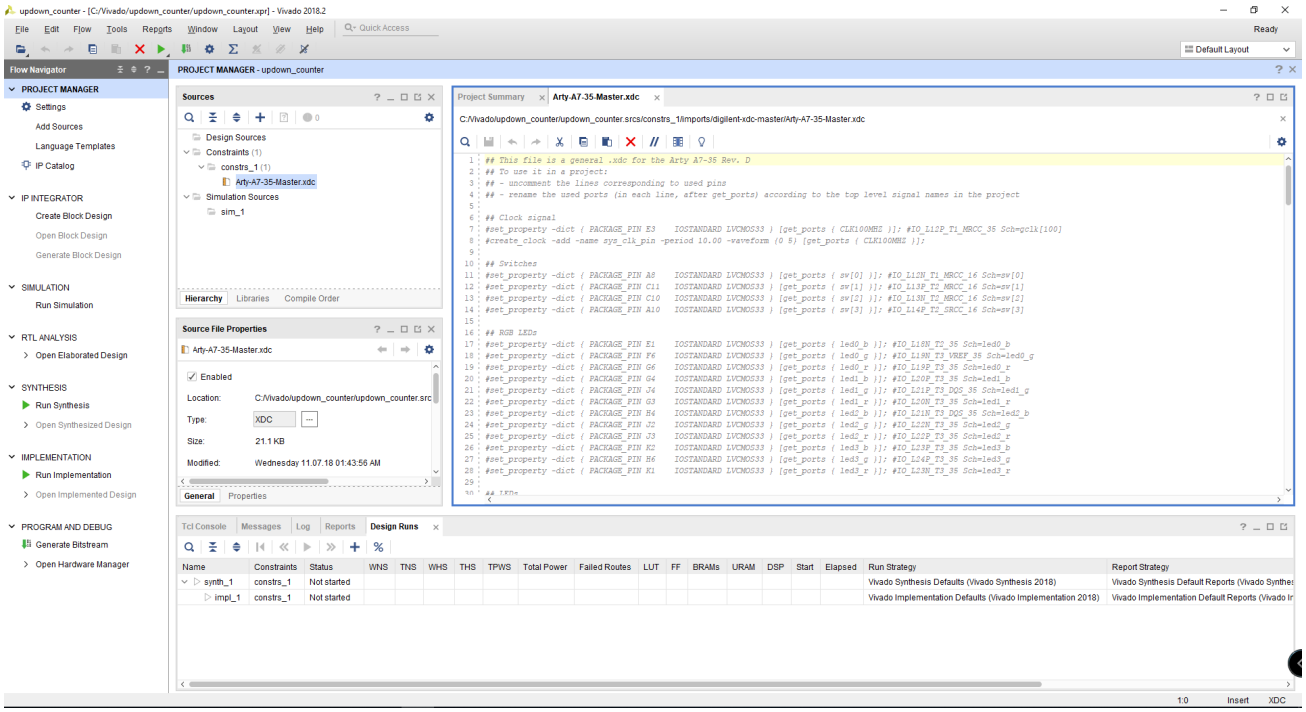

Figure 11: XDC file added

RTL Logic Design & Behavioral Simulation

For this part of the tutorial, we will create our HDL source files for our intended digital design.

1. To create a source file, click the "Add Sources" under the "Flow Navigator" pane. This will open the "Add Sources" dialog box (see Fig. 7). Select "Add or create design sources" and click "Next".

2. The "Add or Create Design Sources" will appear. You can add existing source files or create new ones. For this tutorial, we will be creating the HDL source files. In the dialog box, select "Create File" (see Fig. 12).

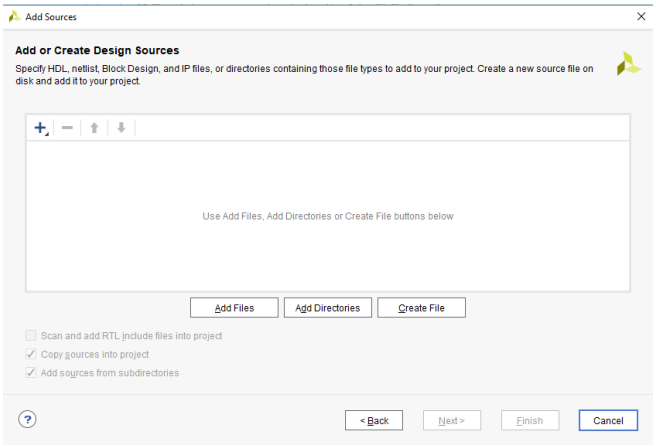

Figure 12: Add source file

3. Now, you will be asked for the name and type of the new source file. Select "Verilog" for the file type and set the file name to "updown.v" (see Fig. 13). After that, click "OK" to continue.

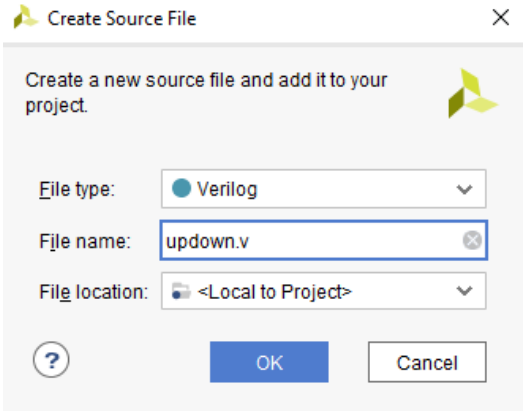

Figure 13: Set file name and type of the source file

4. You will be prompted back to the "Add or Create Design Sources" with the new source file added (see Fig. 14. Click "Finish" to continue.

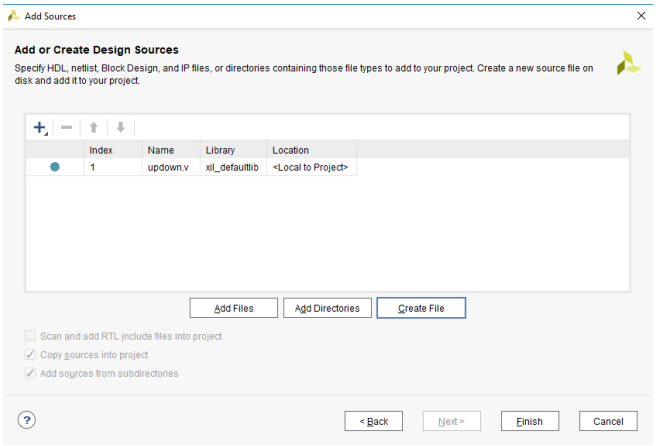

Figure 14: Source file added

5. The next step is to define the input and output ports of our module. Add single bit ports clk, nrst, and dir as inputs. For the output port, add a port named count with the bus option enabled, and set the most significant bit (MSB) to 3 and the least significant bit (LSB) to 0. Fig. 15 shows the port declaration for the updown module. Note that you can still modify the module's ports after creating the source file. This step just allows Vivado to auto-generate the port declaration part of the source code. Click "OK" to continue.

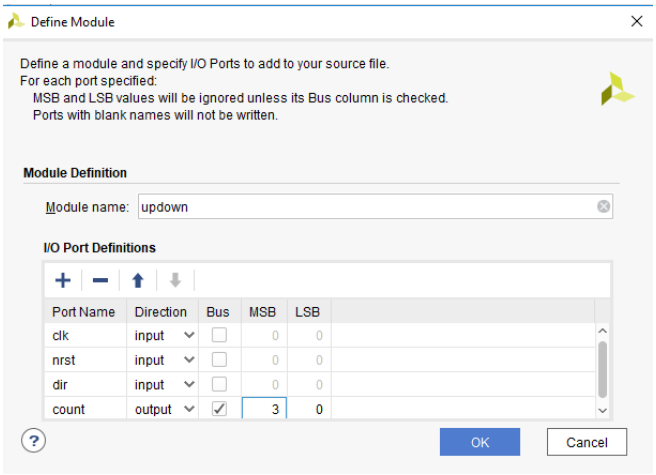

Figure 15: Port declaration for the updown module

6. You will be prompted back to the "Project Manager" window. Notice that the created source file is listed under the "Design Sources" in the "Project Manager" window (see Fig. 16).

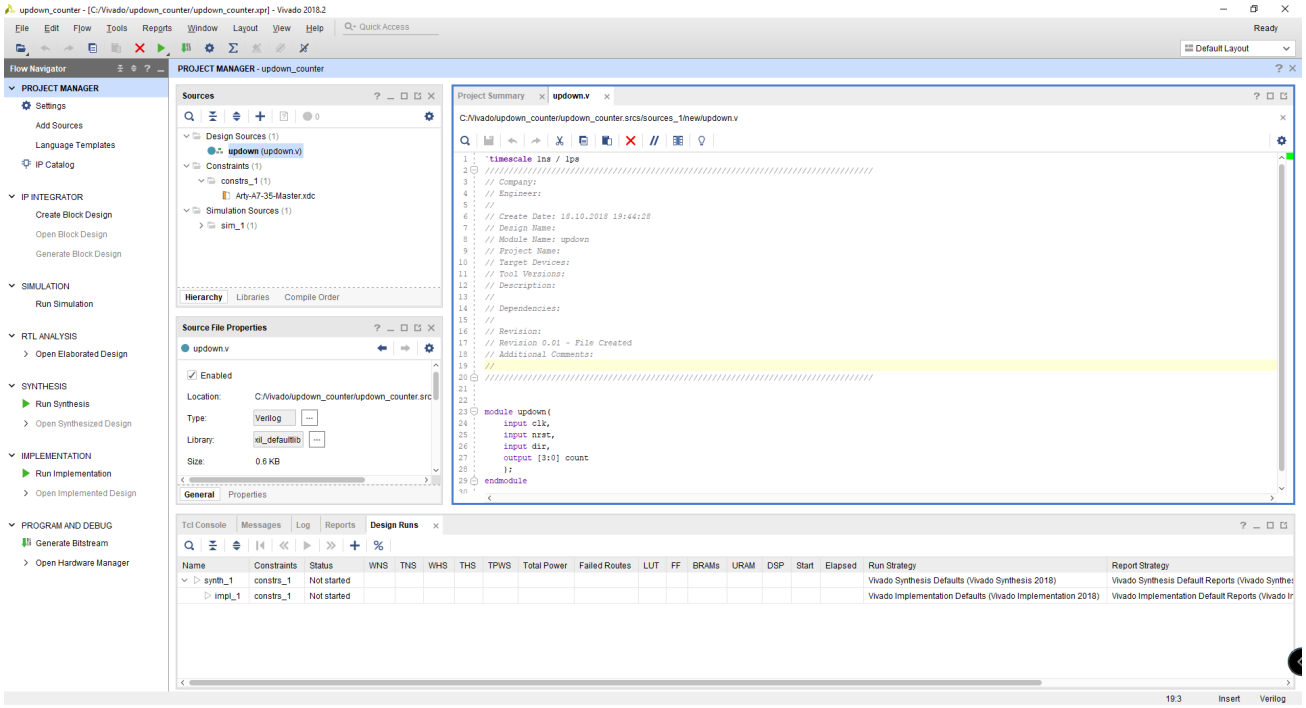

Figure 16: Source file added

7. In the source pane, double-click the "updown.v" source file for editing. The contents of the source file will be shown in the workspace pane. Insert the following code shown in Code 1 on your source file.

```
module updown(
  input clk,
  input nrst,
  input dir,
  output reg [3:0] count
  );

  always @(posedge clk or negedge nrst) begin
    if (!nrst) begin
        count <= 0;
    end
    else begin
      case (dir)
        1'b1:       count <= count + 1;
        default:    count <= count - 1;
      endcase
    end
  end
endmodule
```