

 0 stars  0 forks

 Star

 Notifications

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

 master ▾

[Go to file](#)



Feqzz ...

32 minutes ago



[View code](#)

☰ README.md

# Using HLS with Zybo Zynq-7000 development board

## 🔗 Table of Contents

- [Introduction](#)
  - [Prerequisites](#)
1. [High Level Synthesis in Vitis HLS](#)
    - i. [Create New Project](#)
    - ii. [Create Source Files](#)
    - iii. [Get Source Code from Github](#)
    - iv. [Simulation and Synthesis](#)
    - v. [Export RTL as IP](#)
  2. [Hardware Setup in Vivado](#)
    - i. [Create New Project](#)
    - ii. [Create Block Design](#)
    - iii. [Create HDL Wrapper and Generate Bitstream](#)
    - iv. [Export Hardware Platform](#)
  3. [Application Setup in Vitis](#)
    - i. [Create Platform Project](#)

- ii. [Create Application Project](#)
- iii. [Get Source Code from Github](#)
- iv. [Build and Run Project](#)

## Introduction

---

This is a complete guide for utilising a Vitis HLS design with the Zybo Zynq-7000 development board. We will use a simple FIR (Finite Impulse Response) moving average filter as an example.

The guide has been made for the USN course CS4110 by Kent Odde and Stian Onarheim upon request of Professor Jose Ferreira.

## Prerequisites

---

- Vitis 2021.X
- Vivado 2021.X
- Vitis HLS 2021.X
- Zybo Zynq-7000 development board

# 1. High Level Synthesis in Vitis HLS

---

## 1.1 Create New Project

Open up Vitis HLS and choose `Create Project`.

# VITIS HLS

## PROJECT

Create Project

Open Project

Clone Examples

## RESOURCES

Tutorials

User Guide

Release Notes

Give the project a suitable name and location.

## New Vitis HLS Project



### Project Configuration

Create Vitis HLS project of selected type



Project name:

Location:  Browse...

< BackNext >CancelFinish

For now, skip adding design and testbench files. We will do this at a later stage. Click `Next`.

## New Vitis HLS Project



### Add/Remove Design Files

Add/remove C-based source files (design specification)



Top Function:

[Browse...](#)

### Design Files

Name	CFLAGS	CSIMFLAGS	
			<a href="#">Add Files...</a> <a href="#">New File...</a> <a href="#">Edit CFLAGS...</a> <a href="#">Edit CSIMFLAGS...</a> <a href="#">Remove</a>

[< Back](#)

[Next >](#)

[Cancel](#)

[Finish](#)

Clik the "Three vertical dots" button under Part Selection .

## New Vitis HLS Project



### Solution Configuration

Create Vitis HLS solution for selected technology



Solution Name:

Clock

Period:

Uncertainty:

Part Selection

Part: ***xcvu11p-f1ga2577-1-e***



Flow Target

Vivado IP Flow Target



Configure [several options](#) for the selected flow target

< Back

Cancel

Finish

In the Device Selection Dialog , press Boards and search for Zybo .

Device Selection Dialog X

Select:  Parts  Boards

Filter

Vendor: All ▼

Display Name: All ▼

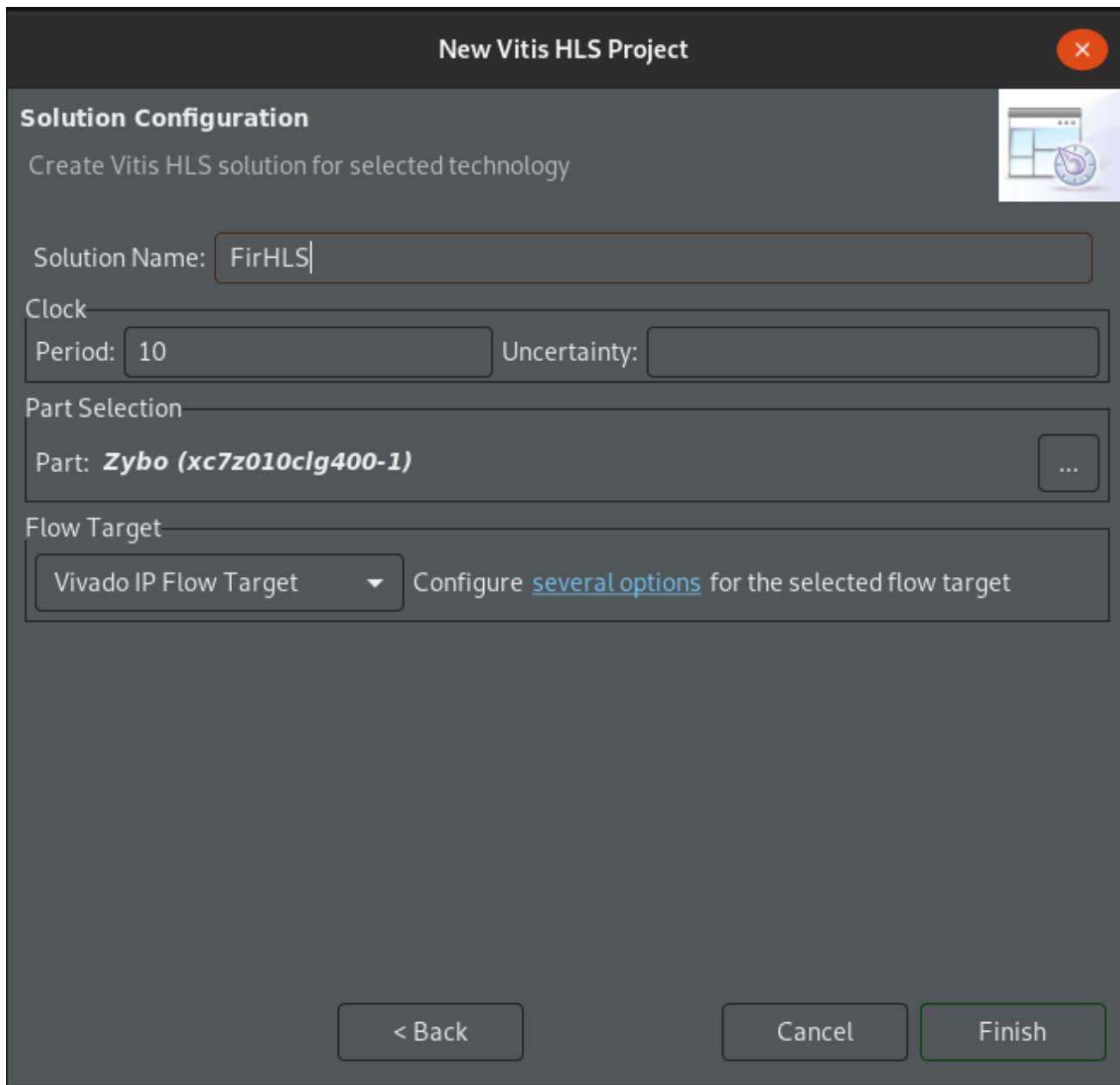
Reset All Filters

Search: zybo (2 matches)

Display Name	Part	Family	Vendor
Zybo Z7-10	xc7z010clg400-1	zynq	dililentinc.com
Zybo	xc7z010clg400-1	zynq	dililentinc.com

Cancel OK

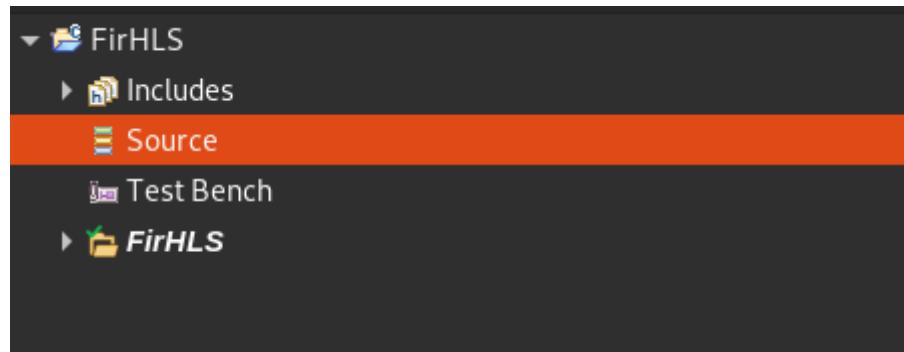
Give the solution a suitable name and press **Finish**.



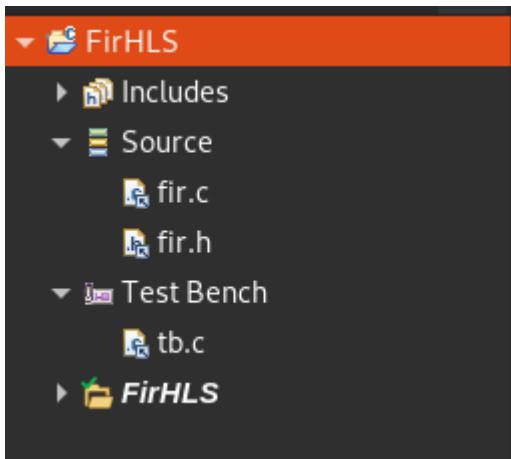
## 1.2 Create Source Files

On the left side in the Explorer window, right-click Sources and click New file. Add the files fir.c and fir.h.

Right-click on Test Bench, click New file and add the file tb.c.



The file tree should now look like this:



## 1.3 Get Source Code from Github

The source code can be found here:

- [tb.c](#)
- [fir.c](#)
- [fir.h](#)

The files should now look like this:

`fir.h`

```
1 #ifndef FIR_H
2 #define FIR_H
3
4
5 #include <stdint.h>
6
7 void fir( const uint8_t input, uint8_t* output );
8
9 #endif
10
```

`fir.c`

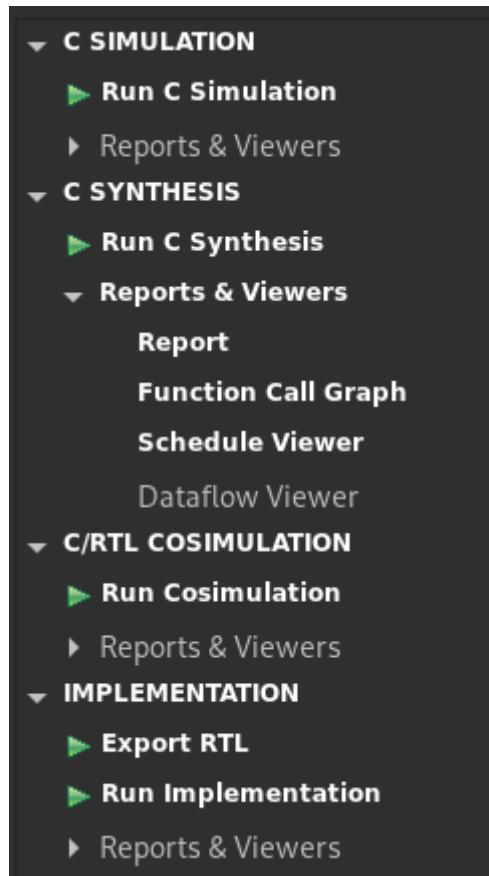
```
.c fir.c ✘
1
2
3 #include "fir.h"
4
5 #define N 3
6
7 //Modified example by Prof. Young-kyu Choi
8 void fir( const uint8_t input, uint8_t* output )
9 {
10 #pragma HLS INTERFACE mode=s_axilite port=fir
11 #pragma HLS INTERFACE mode=s_axilite port=input
12 #pragma HLS INTERFACE mode=s_axilite port=output
13
14     const float coef = 1.0f/3.0f;
15     static float shift_reg[N];
16     float acc = 0;
17
18     for (int8_t i = N - 1; i > 0; i--)
19     {
20         shift_reg[i] = shift_reg[i - 1];
21         acc += shift_reg[i] * coef;
22     }
23     acc += input * coef;
24     shift_reg[0] = (float)input;
25     *output = (uint8_t)(acc+0.5f);
26 }
27
```

tb.c

```
.c tb.c ✘
1
2
3 #include "fir.h"
4 #include <stdio.h>
5
6
7 int main()
8 {
9     uint8_t in = 'a';
10    uint8_t out;
11    printf("\nOutput: ");
12    for(int i = 0; i < 20; i++)
13    {
14        uint8_t out;
15        fir(in++, &out);
16        if(in == 'd') in = 'a';
17        printf("%c", (char)out);
18    }
19    printf("\n");
20    return 0;
21 }
```

## 1.4 Simulation and Synthesis

In the Flow navigator window as seen below:



Run the C simulation and ensure that the output looks something like this:

```
utput: Abbbbbbbbbbbbbbbb
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] **** CSIM finish ****
INFO: [HLS 200-111] Finished Command csim_design CPU user time: 23.45 seconds. CPU
system time: 1.31 seconds. Elapsed time: 24.48 seconds; current allocated memory:
251.756 MB.
Finished C simulation.
```

Run the C Synthesis and ensure that the output looks something like this:

Synthesis Summary(FirHLS) 3

### Synthesis Summary Report of 'fir'

- General Information**

Date: Sun Nov 14 14:59:47 2021  
 Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:07 MDT 2021)  
 Project: FirHLS

Solution: FirHLS (Vivado IP Flow Target)  
 Product family: zynq  
 Target device: xc7z010-clg400-1
- Timing Estimate**

Target	Estimated	Uncertainty
10.00 ns	7.256 ns	2.70 ns
- Performance & Resource Estimates**

Modules & Loops

Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fir	II Violation	-	38	380.000	-	39	-	no	0	5	720	1437	0	
VITIS_LOOP_18_1	II Violation Memory Dependency	1	-	15	150.000	11	5	2	yes	-	-	-	-	
- HW Interfaces**

S\_AXILITE

Interface	Data Width	Address Width	Offset	Register
s_axi_control	32	5	16	0

TOP LEVEL CONTROL

Interface	Type	Ports
ap_clk	clock	ap_clk
ap_rst_n	reset	ap_rst_n
interrupt	interrupt	interrupt
ap_ctrl	ap_ctrl_hs	

The code is not optimised and has timing violations. Since this is only an example, this is fine.

## [Optional] Improve the latency

Swap out the content of `fir.c` with [fir\\_improved.c](#) and run C Synthesis again.

### Synthesis Summary Report of 'fir'

- General Information**

Date: Tue Nov 16 13:46:08 2021  
 Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:07 MDT 2021)  
 Project: FirHLS

Solution: FirHLS (Vivado IP Flow Target)  
 Product family: zynq  
 Target device: xc7z010-clg400-1
- Timing Estimate**

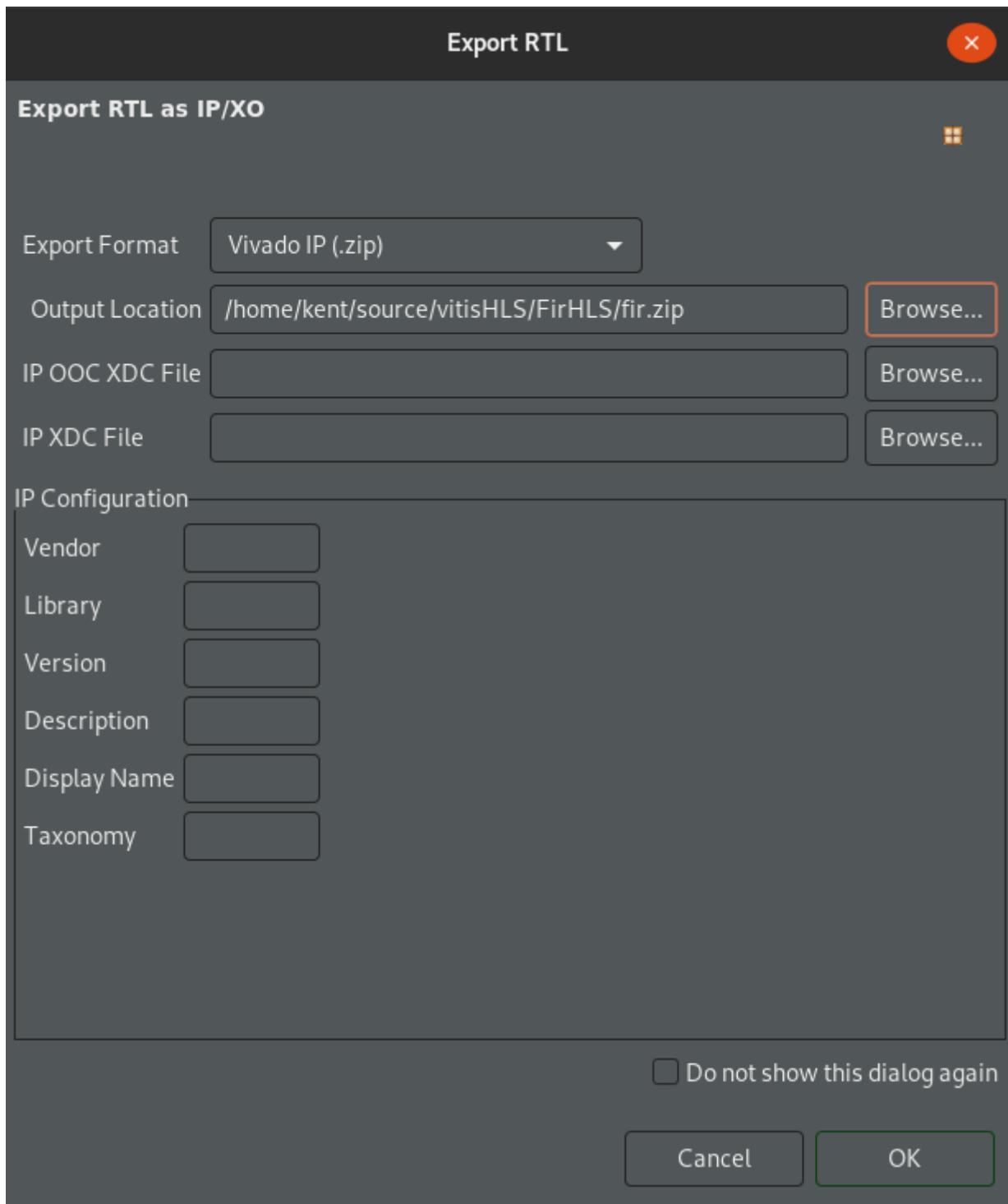
Target	Estimated	Uncertainty
10.00 ns	6.912 ns	2.70 ns
- Performance & Resource Estimates**

Modules & Loops

Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
fir	II Violation	-	7	70.000	-	8	-	no	0	1	300	240	0	
VITIS_LOOP_31_1	II Violation Memory Dependency	-	2	20.000	1	1	2	yes	-	-	-	-		

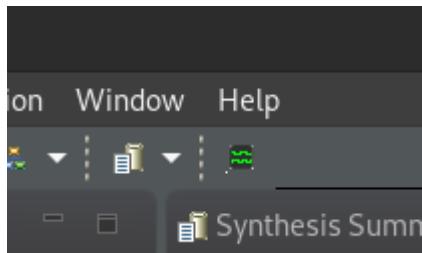
## 1.5 Export RTL as IP

In the Flow navigator window, press the `Export RTL` button and save the ZIP file in a suitable location.

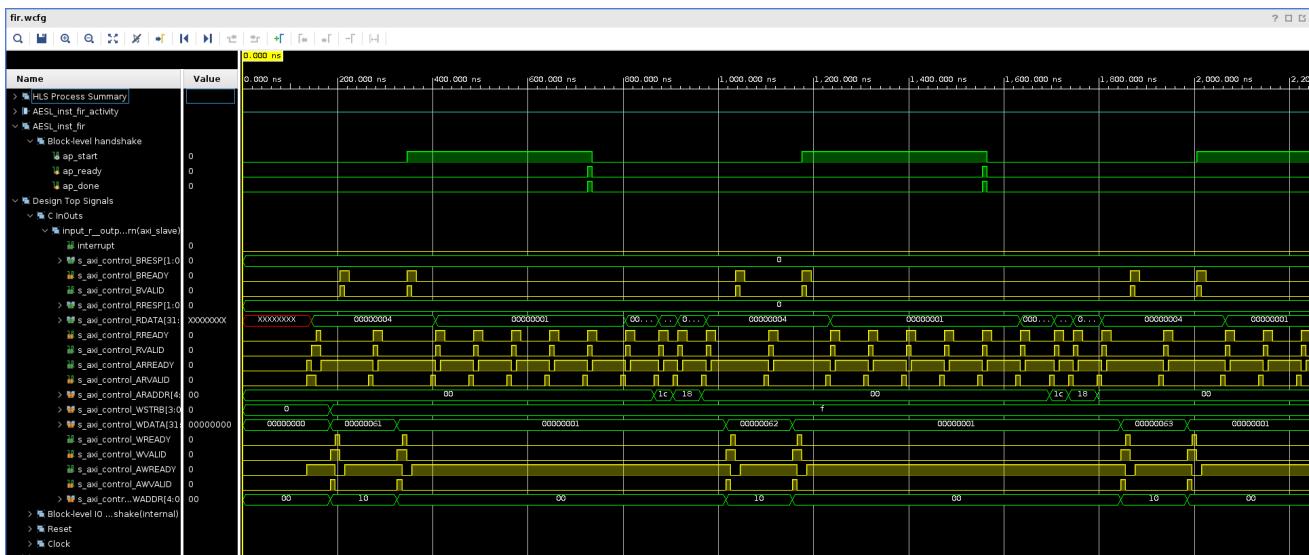


### [Optional] View the waveform

Run Cosimulation and press the `Waveform` button in the top menu and wait for the Vivado program to open.



It should look something like this:



## 2. Hardware Setup in Vivado

### 2.1 Create New Project

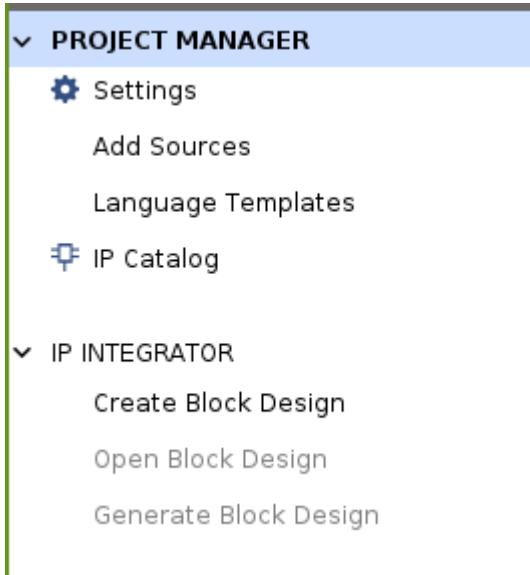
As we assume some familiarity with the Vivado Design Suite, we will not go over how to create a new project. Create a new Vivado project and do not create any files.

Remember to select the correct board file. You can find it by searching for `zybo`. (You might have to click the 'Refresh' button). Ensure that the `Board Rev` is `B.3`.

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board Rev	Available IOBs	LUT Elements	FlipFlops
Zybo			digilentinc.com	1.0	xc7z010clg400-1	400	B.3	100	17600	35200
Zybo Z7-10			digilentinc.com	1.0	xc7z010clg400-1	400	B.2	100	17600	35200
Zybo Z7-20			digilentinc.com	1.0						

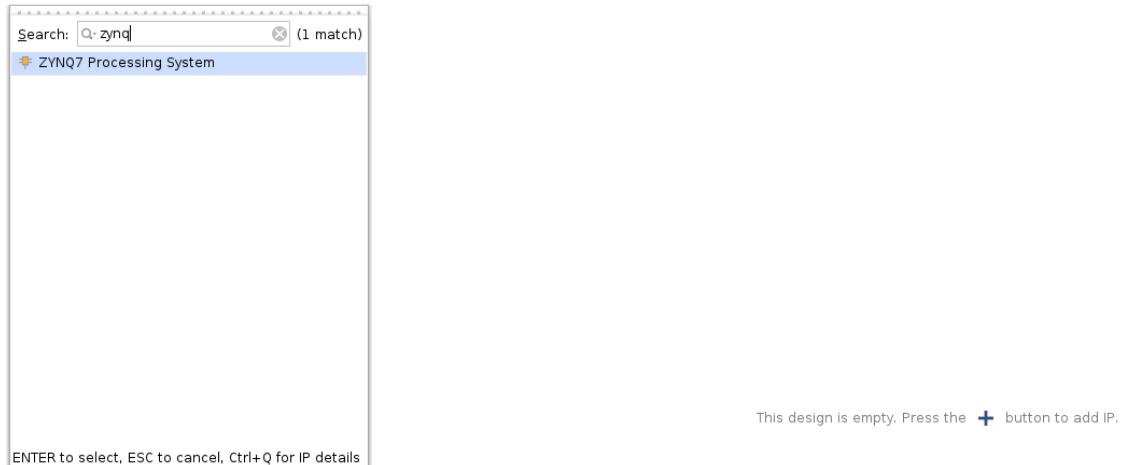
### 2.2 Create Block Design

In the Flow Navigator window, under IP INTEGRATOR click Create Block Design.

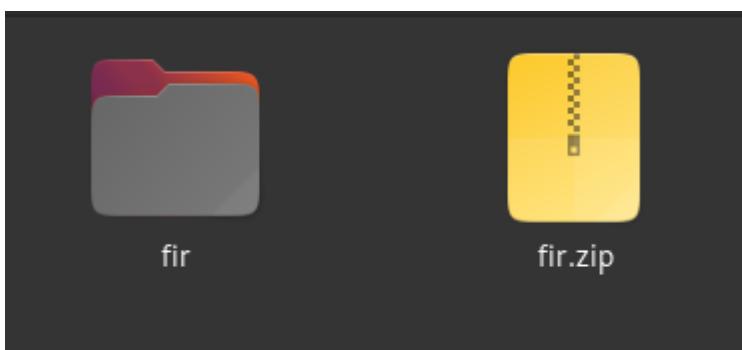


In the `Diagram` window, press the `+` button or right-click and press `Add IP`.

Search for `zynq` in the pop-up window and double-click `ZYNQ7 Processing System`.

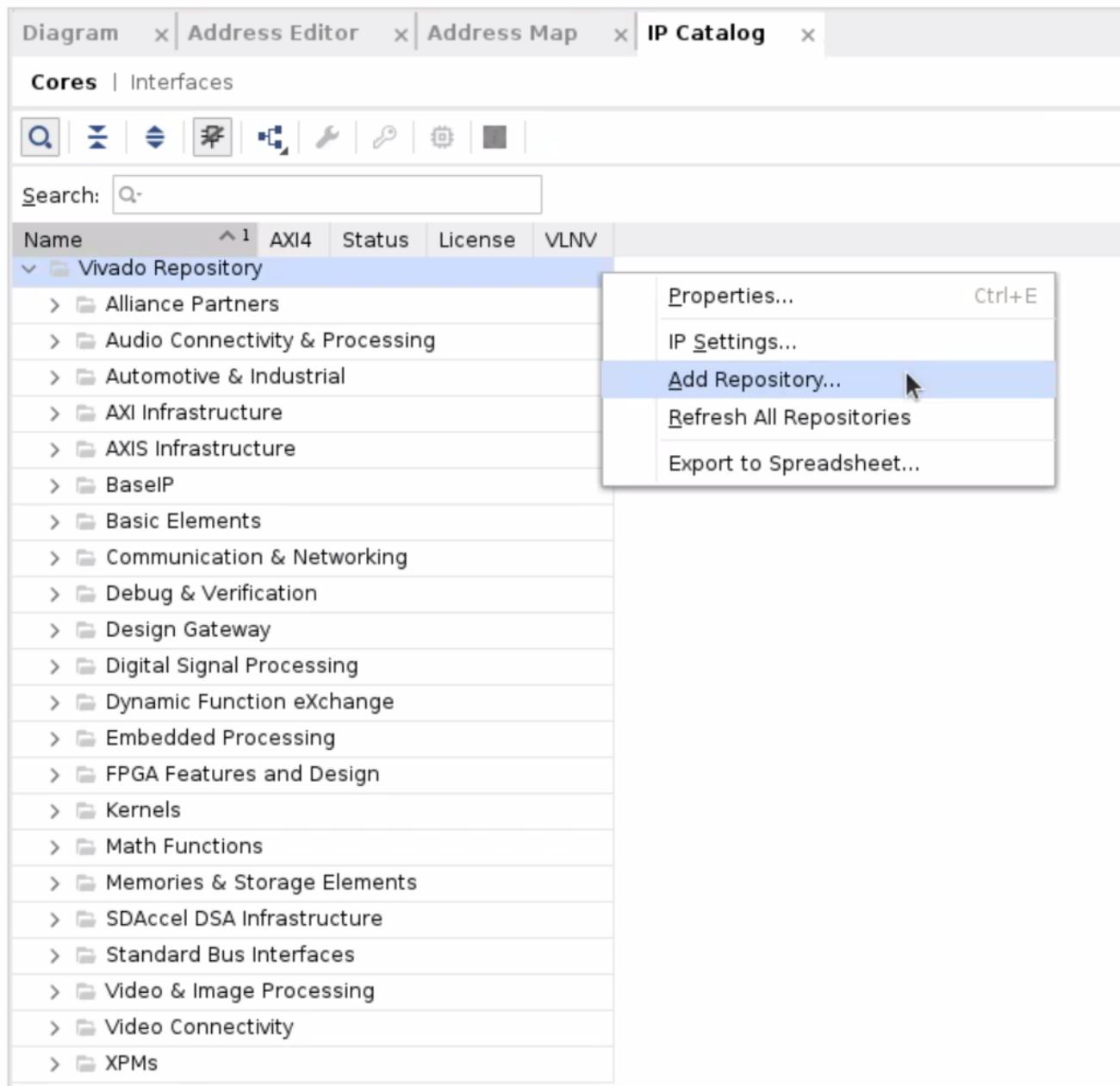


Locate the ZIP file that we exported in step [1.5](#), and unzip the file.



Go back to the `Flow Navigator` window in Vivado and click `IP Catalog` under `PROJECT MANAGER`.

Right-click anywhere in the IP Catalog tab and click Add Repository and select the folder that we unzipped.



Expand the User Repository folder and double-click Fir .

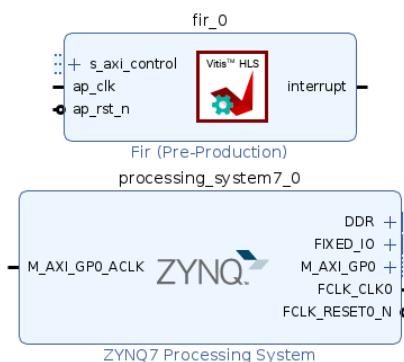
IP Catalog (2)					
Cores   Interfaces					
Name	AXI4	Status	License	VLAN	
User Repository (/home/kent/source/vitisHLS/ip)					
VITIS HLS IP					
Fir	AXI4	Pre-Production	Included	xilinx.com:hls:fir:1.0	
Inc	AXI4	Pre-Production	Included	xilinx.com:hls:inc:1.0	
Vivado Repository					
Alliance Partners					
Audio Connectivity & Processing					
Automotive & Industrial					
AXI Infrastructure					
AXIS Infrastructure					
BaselP					
Basic Elements					
Communication & Networking					

Click the Add IP to Block Design button.

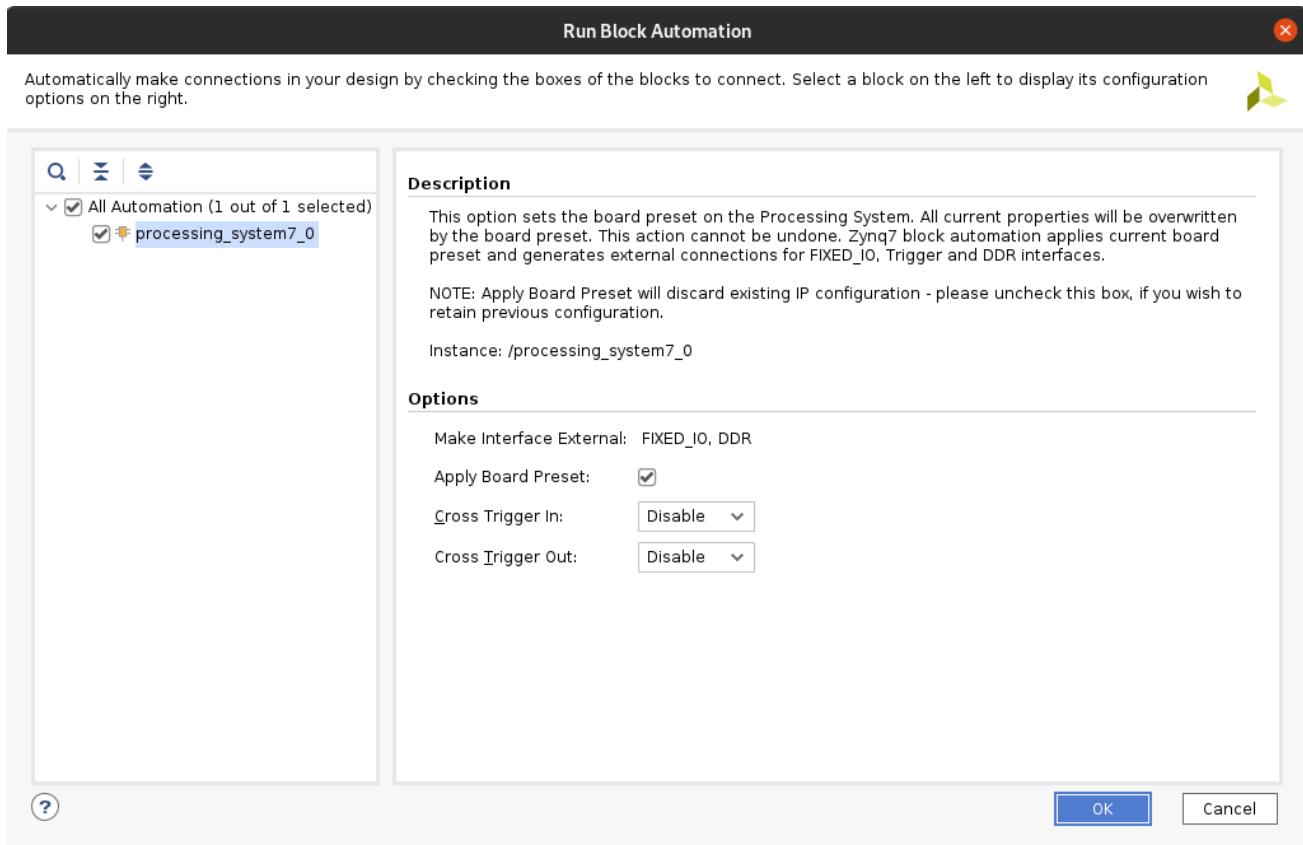


Return to the Diagram window. It should now look something like this:

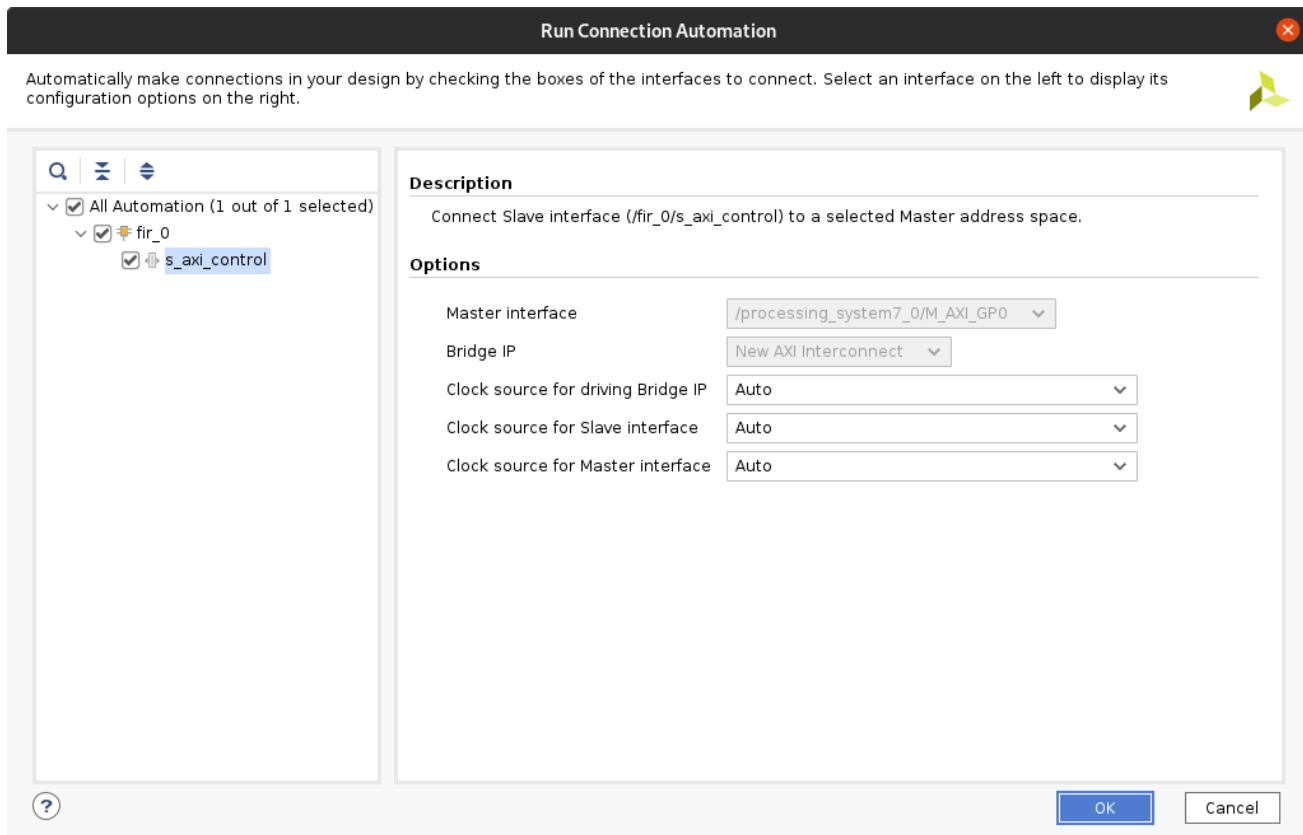
Designer Assistance available. Run Block Automation Run Connection Automation



Click Run Block Automation . The presets should look like the image below. Click the ok button.



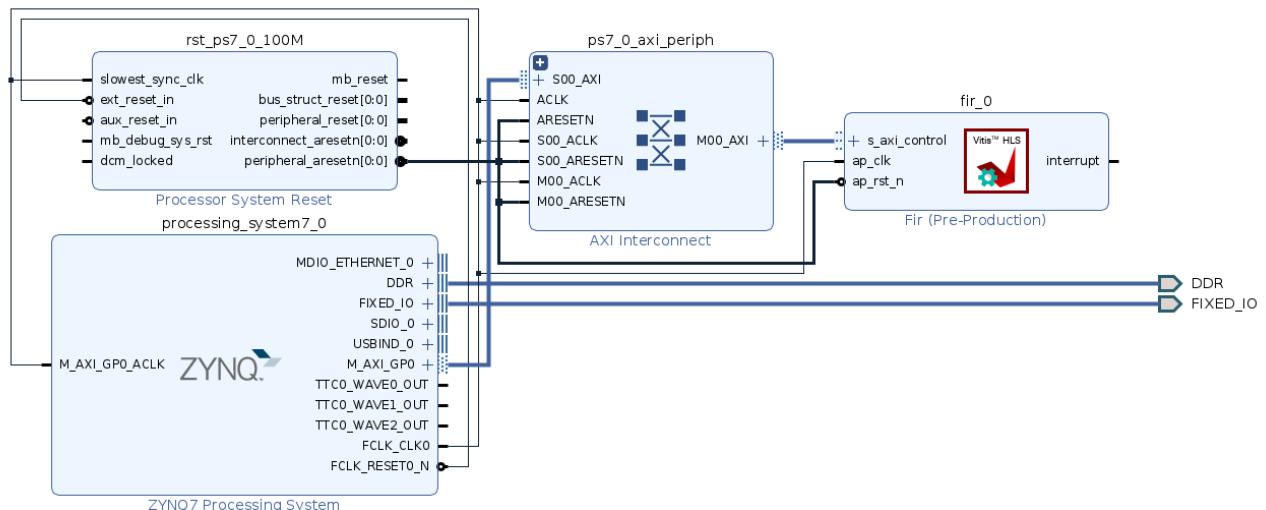
Click Run Connection Automation . The presets should look like the image below. Click the ok button.



Click the Regenerate Layout button located in the Diagram window's toolbar.



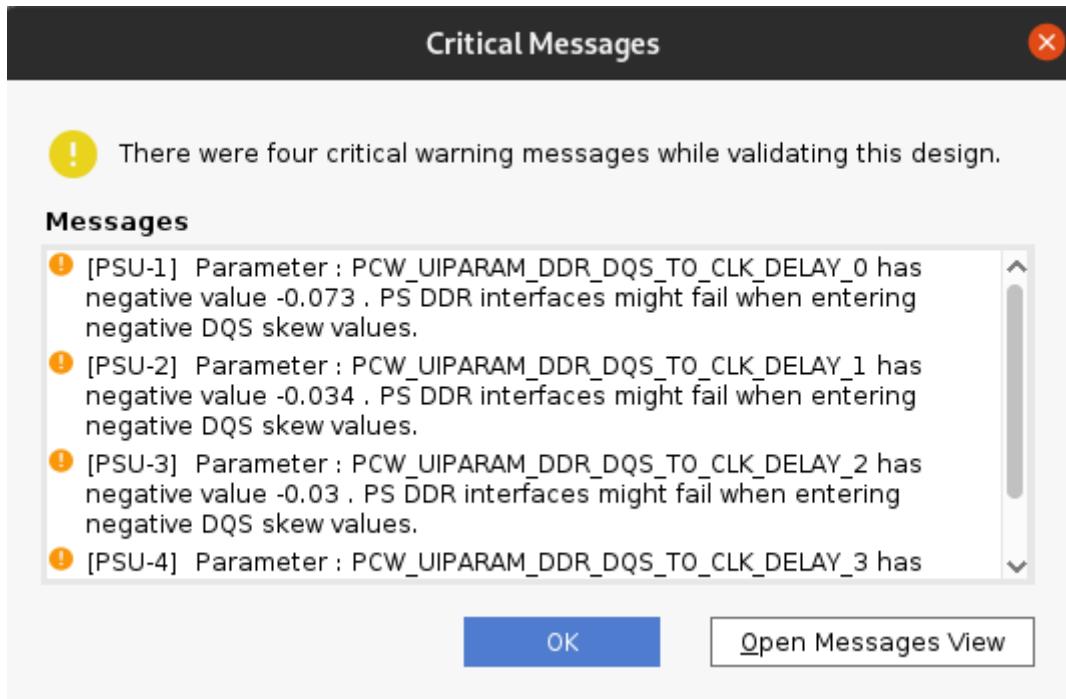
If everything is done correctly, the diagram should now look like this:



Click the **Validate Design** button located in the **Diagram** window's toolbar.

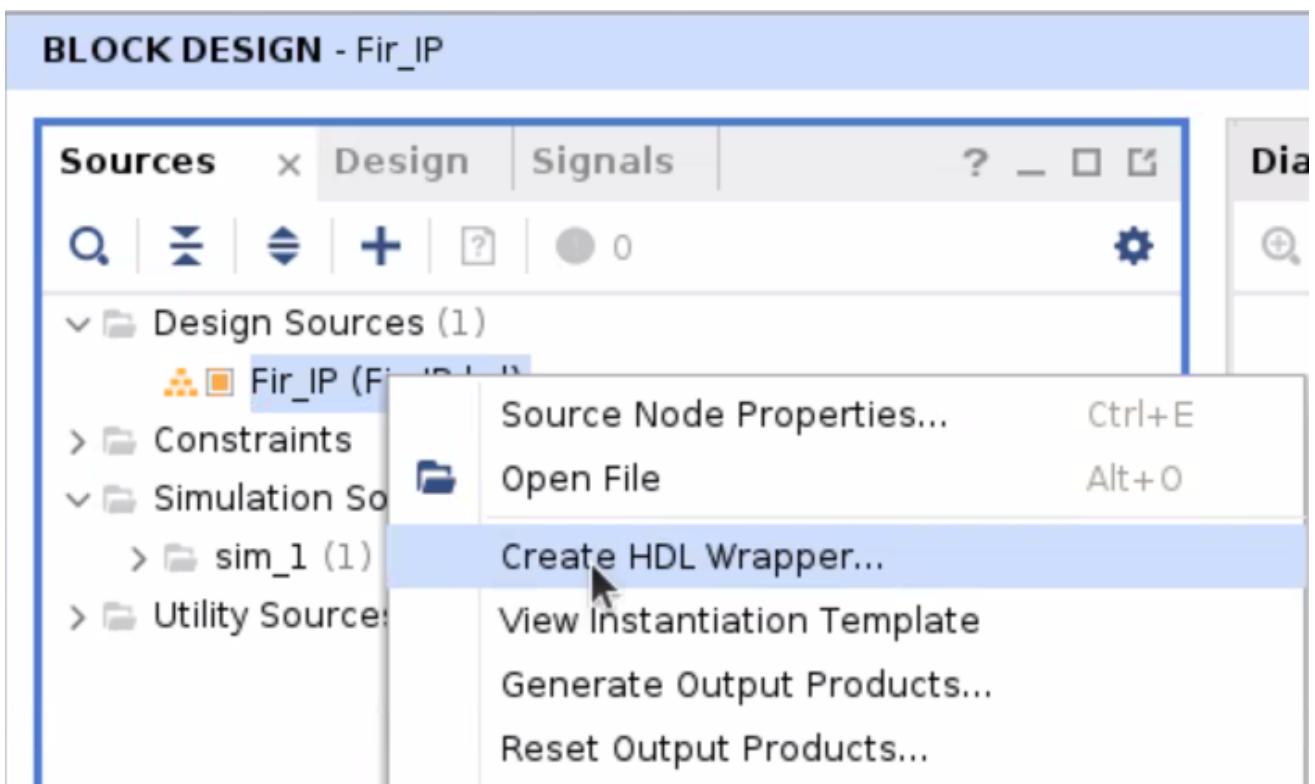


Ignore the four critical warning messages. Click the **OK** button.

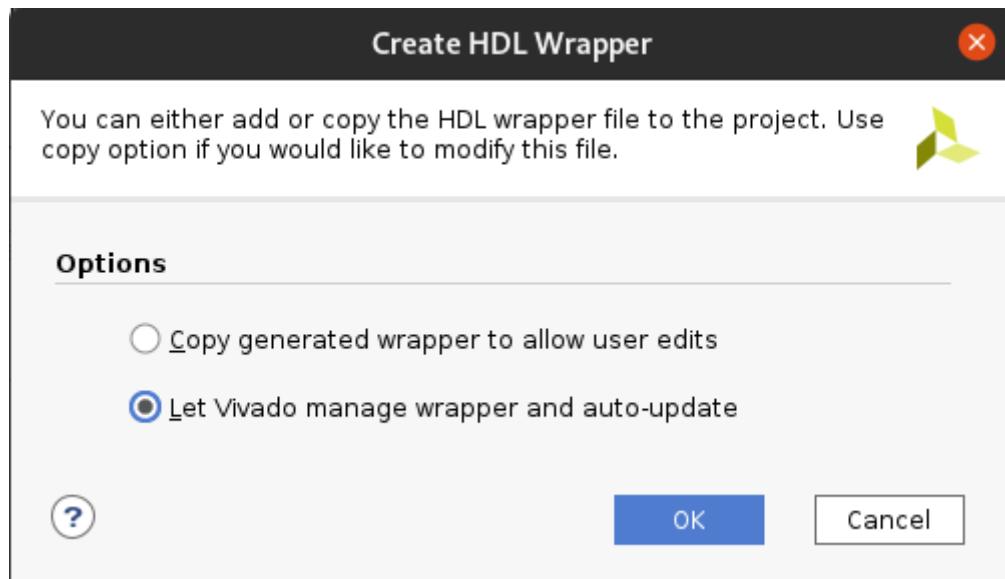


## 2.3 Create HDL Wrapper and Generate Bitstream

Right-click the design file under **Sources** and click **Create HDL Wrapper**.



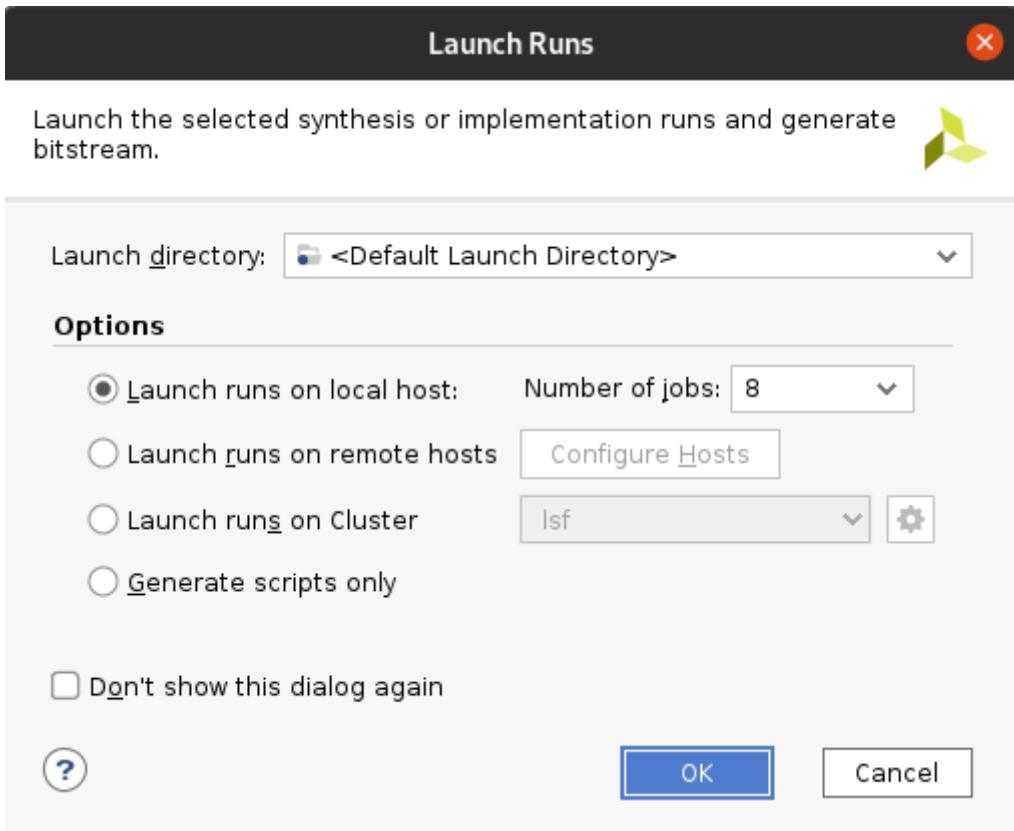
Select the `Let Vivado manage wrapper` and `auto-update` option and click the `ok` button.



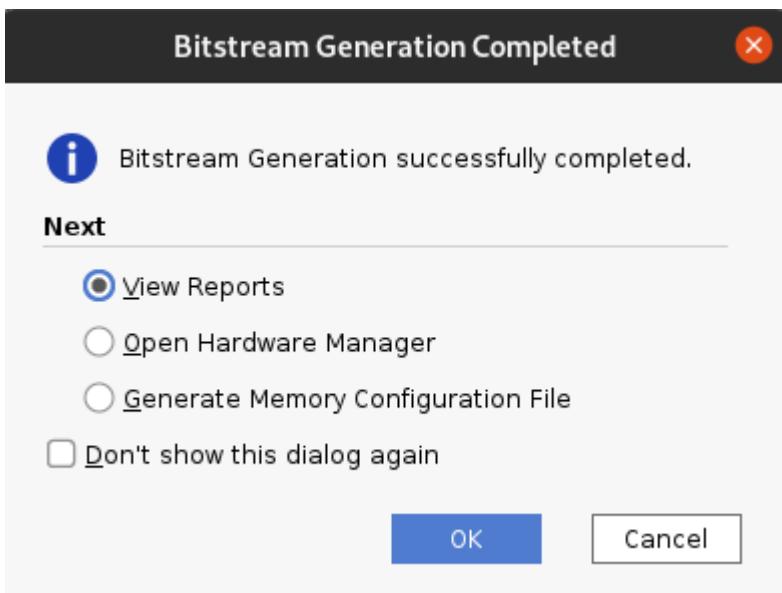
In the Flow Navigator window, click the `Generate Bitstream` button located under `PROGRAM AND DEBUG`.

- ✓ PROGRAM AND DEBUG
  - ⬇ Generate Bitstream
  - > Open Hardware Manager

Click the `ok` button.

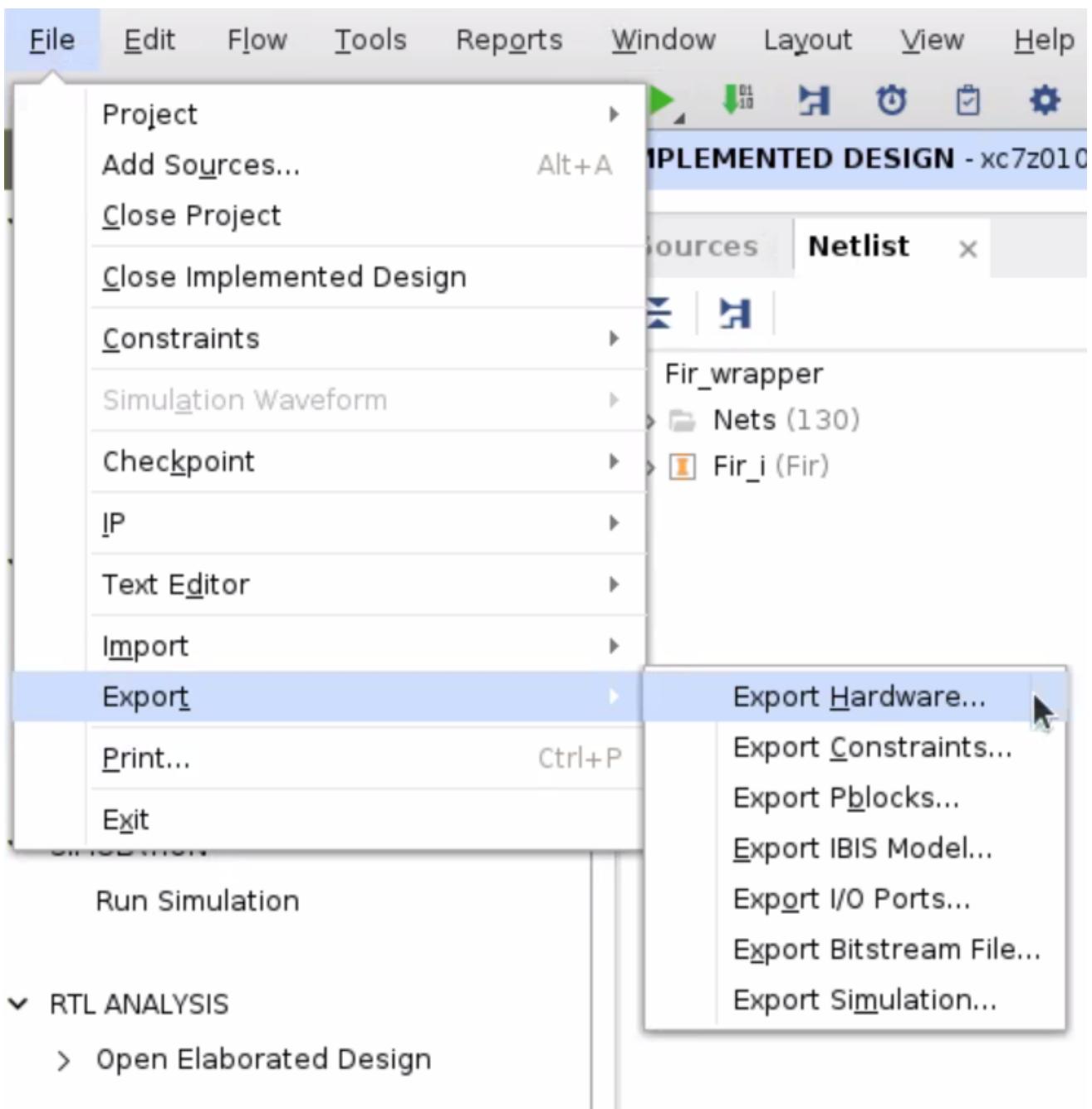


This will take some time. Eventually you will be met with a window like this:



## 2.4 Export Hardware Platform

Click File --> Export --> Export Hardware .



Choose the `Include bitstream` option and click the `Next` button.

**Output**

Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.

 Pre-synthesis

This platform includes a hardware specification for downstream software tools.

 Include bitstream

This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for software tools.

[< Back](#)[Next >](#)[Finish](#)[Cancel](#)

Choose a suitable XSA file name and location. In the final window, click the `Finish` button.

**Files**

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.



XSA file name:  ✖

Export to:  ✖ ...

The XSA will be written to: /home/kent/source/vhdl/Fir\_IP/Fir\_wrapper.xsa

< Back

Next >

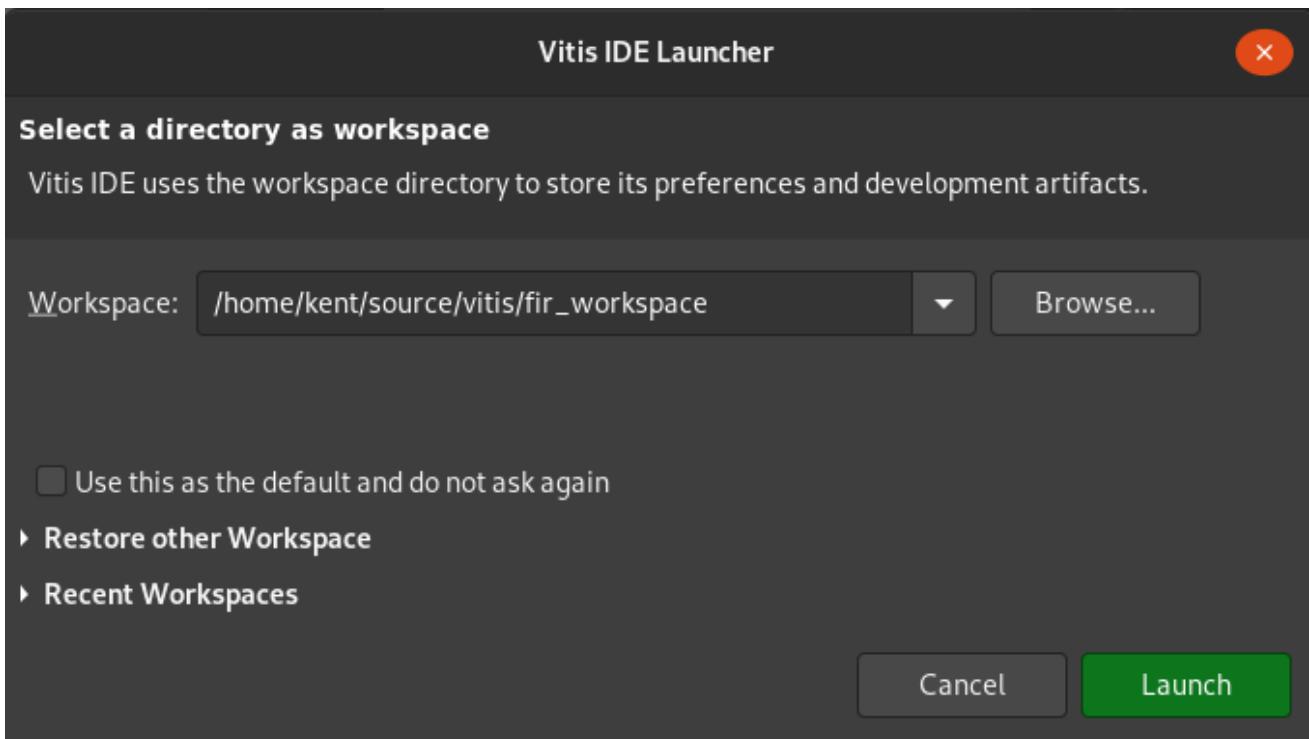
Finish

Cancel

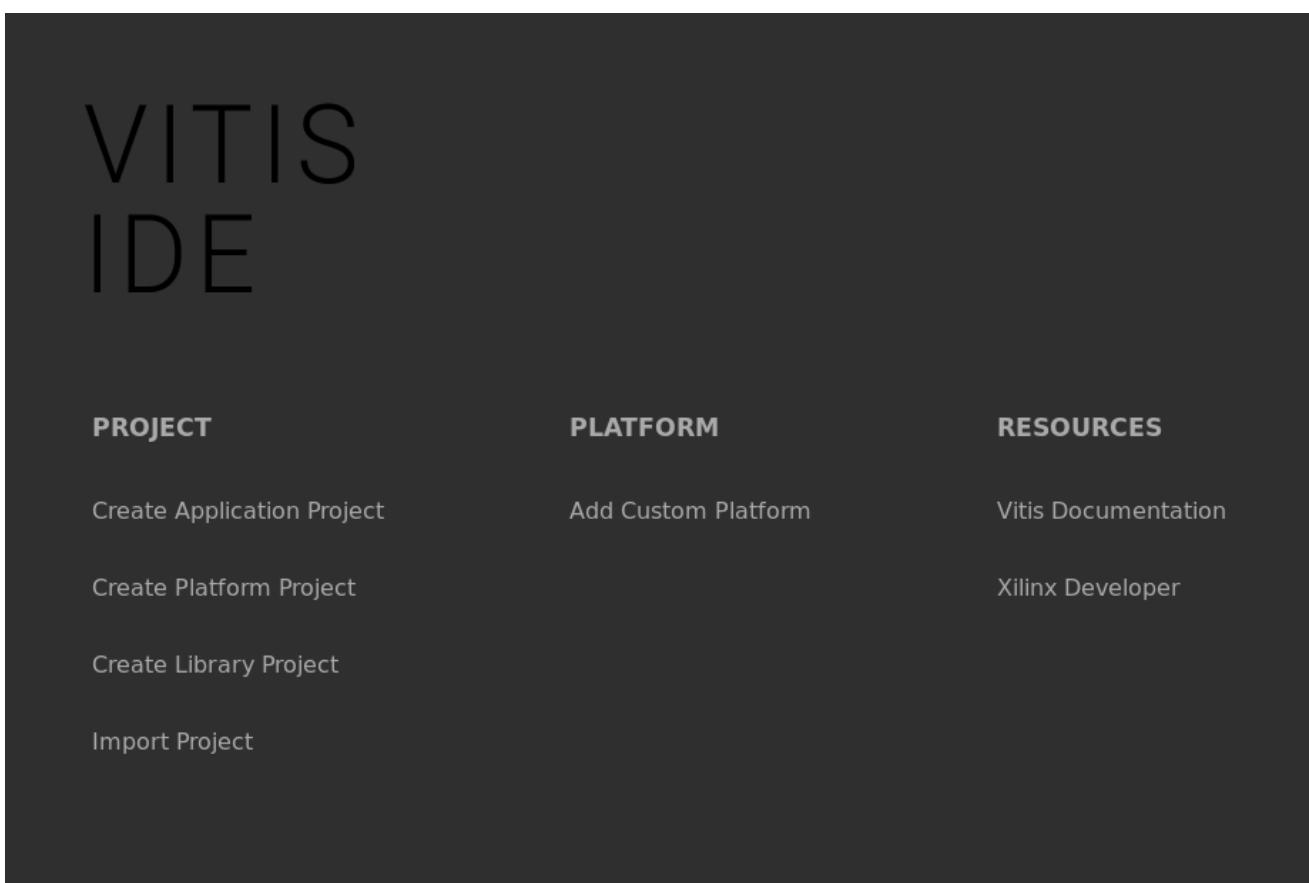
### 3. Application Setup in Vitis

#### 3.1 Create Platform Project

Start Vitis IDE and choose a suitable location for your workspace.



When met by the welcome screen, choose `Create Platform Project`.



Choose a suitable platform project name and click `Next`.

New Platform Project

**Create new platform project**

Enter a name for your platform project

This wizard will guide you through creation of a platform project from the output of Vivado [Xilinx Shell Archive (XSA)] or from an existing platform. A platform will enable you to specify options for the kernels, BSPs, as well as settings required for creating new applications. Platforms are currently supported for embedded software developers.

Platform project name: Fir\_Platform

The diagram illustrates the components of a Platform Project. On the left, a 'Processor' is shown. To its right is a 'Domain' box, which contains an 'XSA' file. Further to the right is a 'System Project' box, which contains an 'App'. Red dashed lines connect the Processor to the Domain, and the Domain to the System Project.

- A platform provides hardware information and software environment settings.
- A system project contains one or more applications that run at the same time.
- A domain provides runtime for applications, such as operating system or BSP.
- A workspace can contain unlimited platforms and unlimited system projects.

A new platform project can be created from one of the two inputs:

**From hardware specification (XSA)**  
Create a new platform project from a hardware specification file. You can specify the OS and processor to start with. The platform can be customized later from the platform project editor.

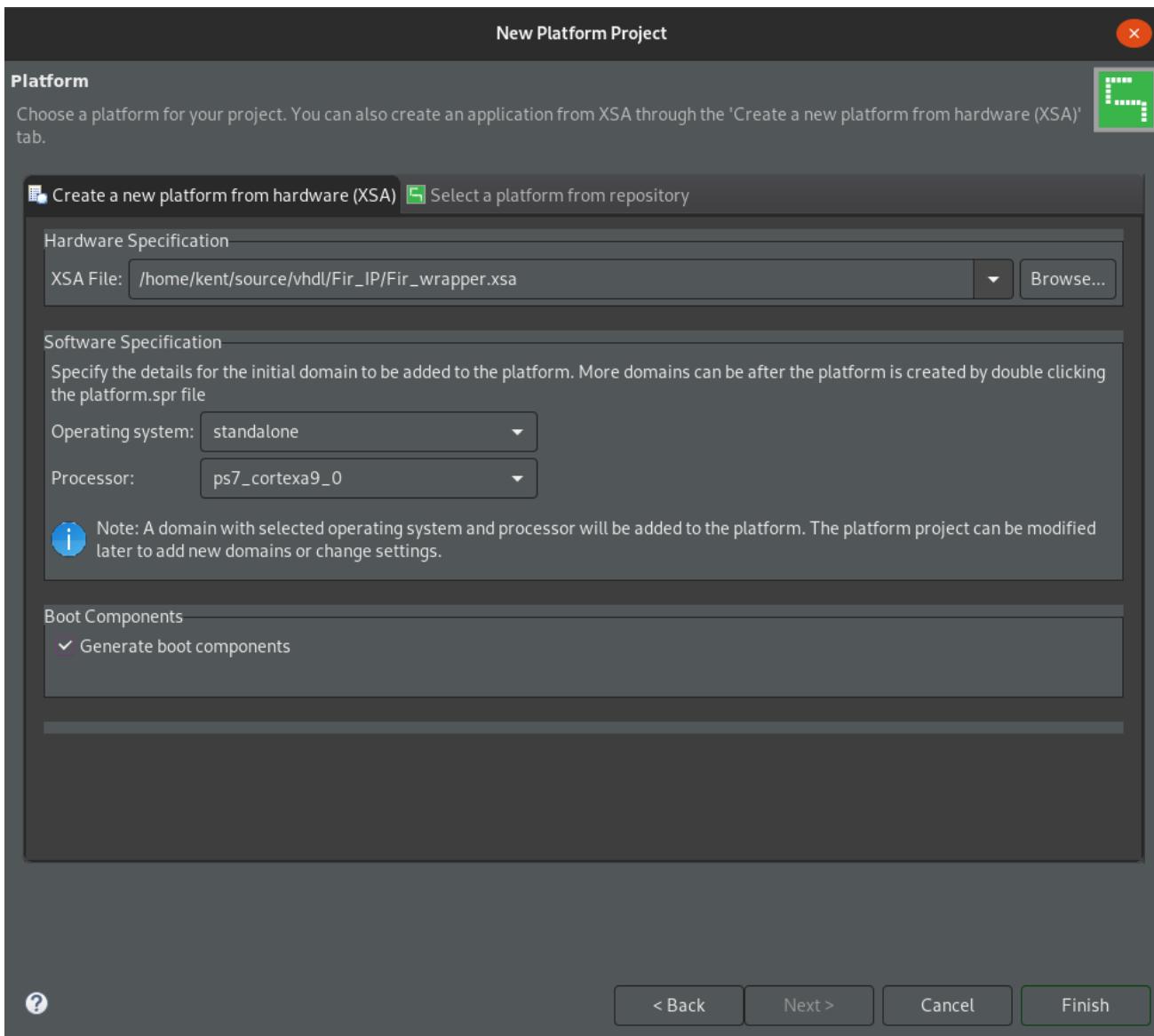
**From existing platform**  
Load the platform definition from an existing platform. You can choose any platform from the platform repository as a base for your platform project.

?

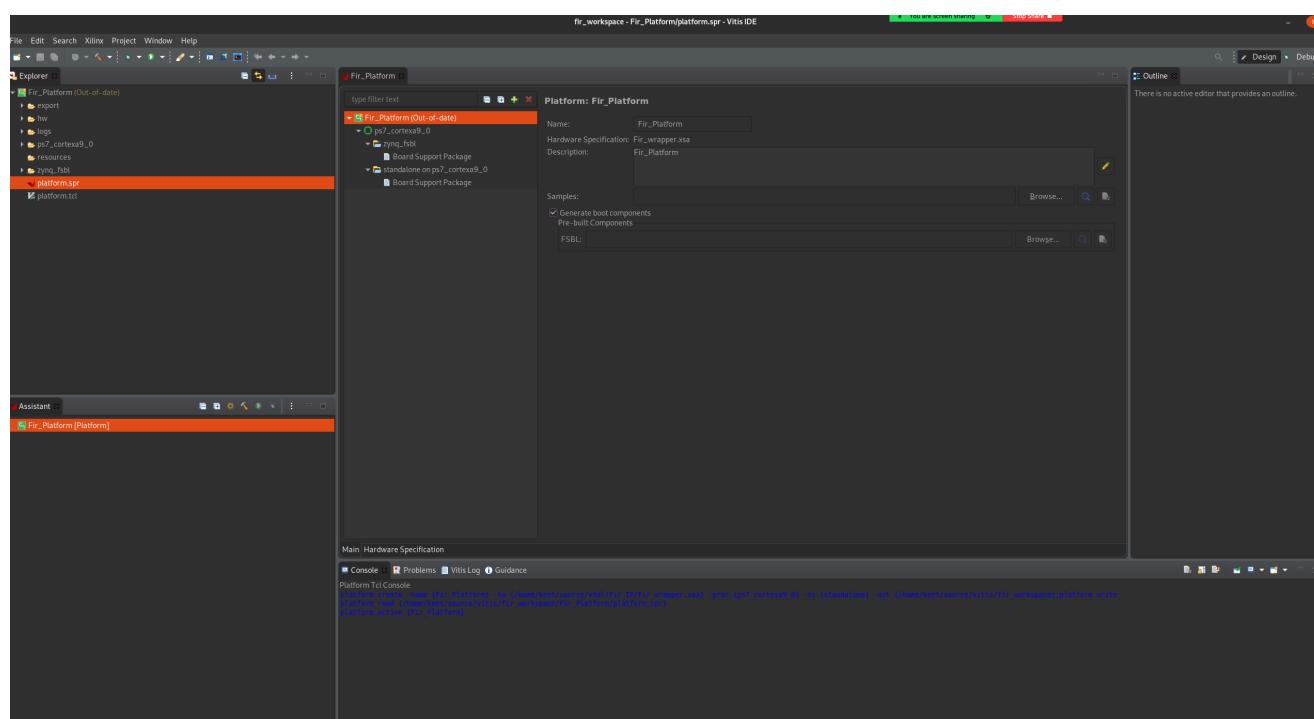
< Back    Next >    Cancel    Finish

Under Hardware Specification click Browse and select the XSA file that we created in step [2.4](#).

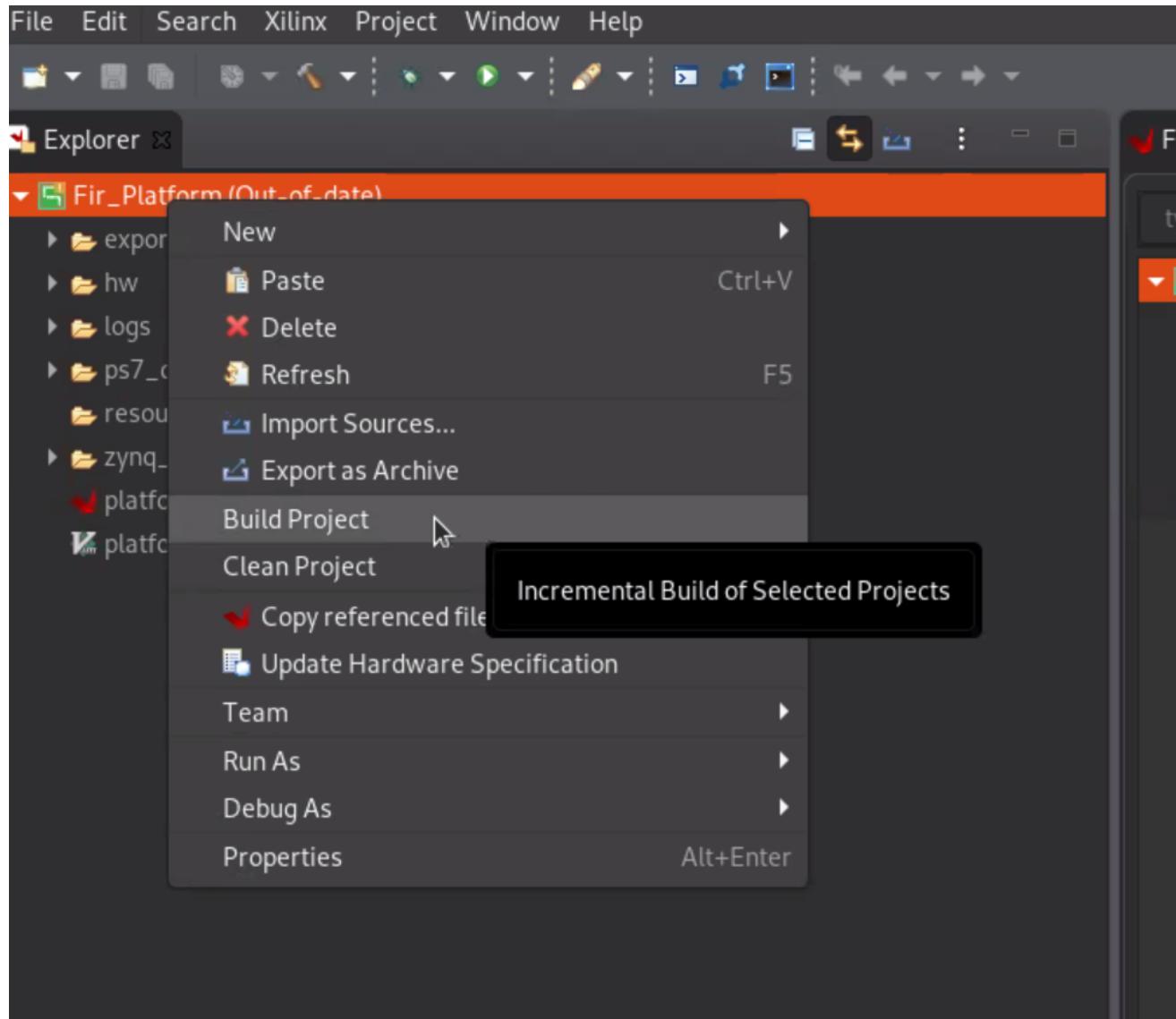
Ensure that the Software Specification section matches the image below, and click Finish .



After the project has been generated, you will be met with this window.

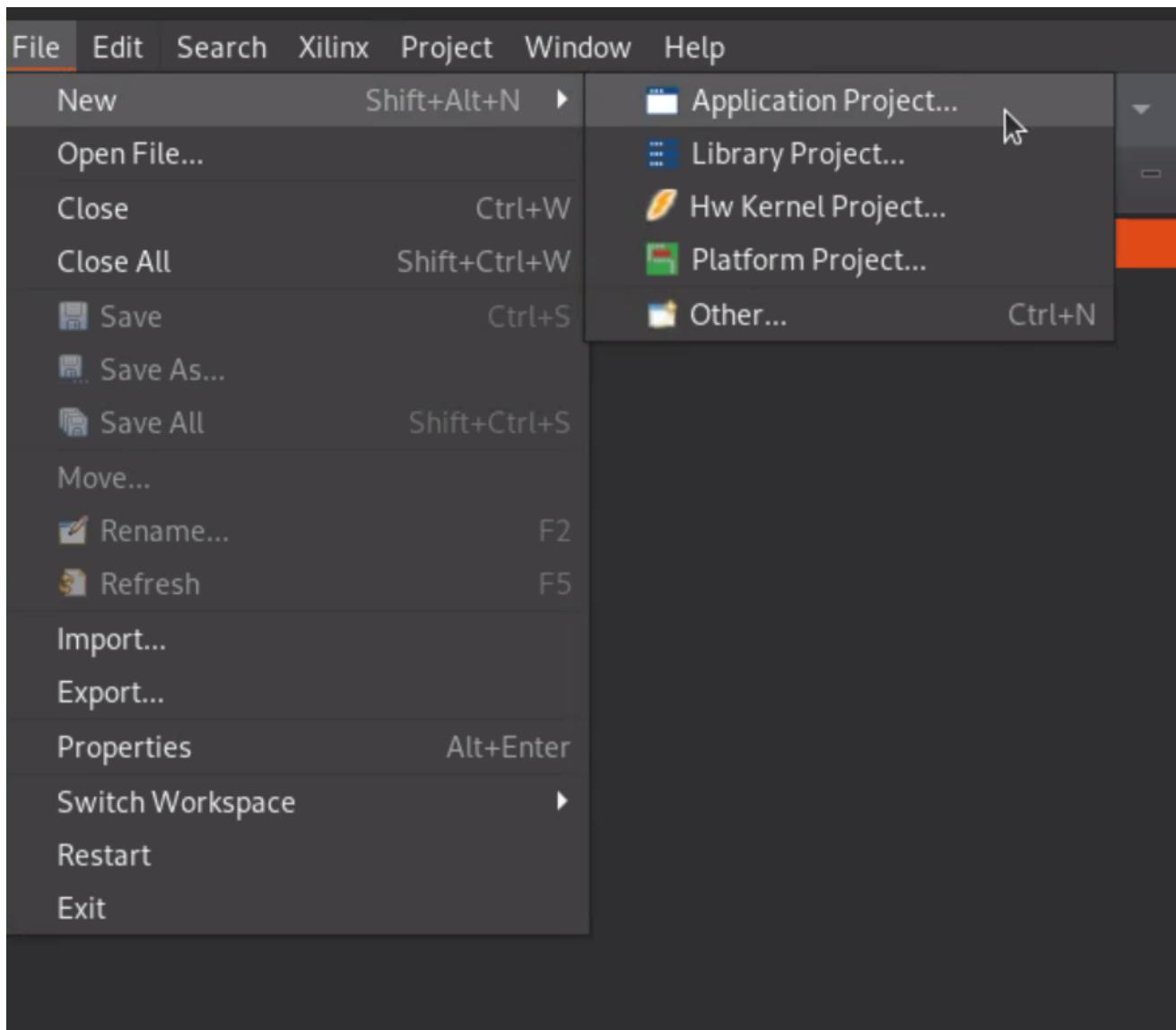


In the Explorer window on the left, right-click the platform project, and click Build Project .



### 3.2 Create Application Project

Create the Application Project. Click File --> New --> Application Project .



Select the platform you created previously and click **Next**.

New Application Project

**Platform**

Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

Select a platform from repository Create a new platform from hardware (XSA)

Add Manage

Name	Board	Flow	Vendor	Path
Fir_Platform [custom]	zybo	Embedded SW Dev	xilinx	/home/kent/source/vitis/fir_workspace/Fir_Platform

Platform Info

General Info

Name:	Fir_Platform
Part:	xc7z010clg400-1
Family:	zyng
Description:	Fir_Platform

Acceleration Resources

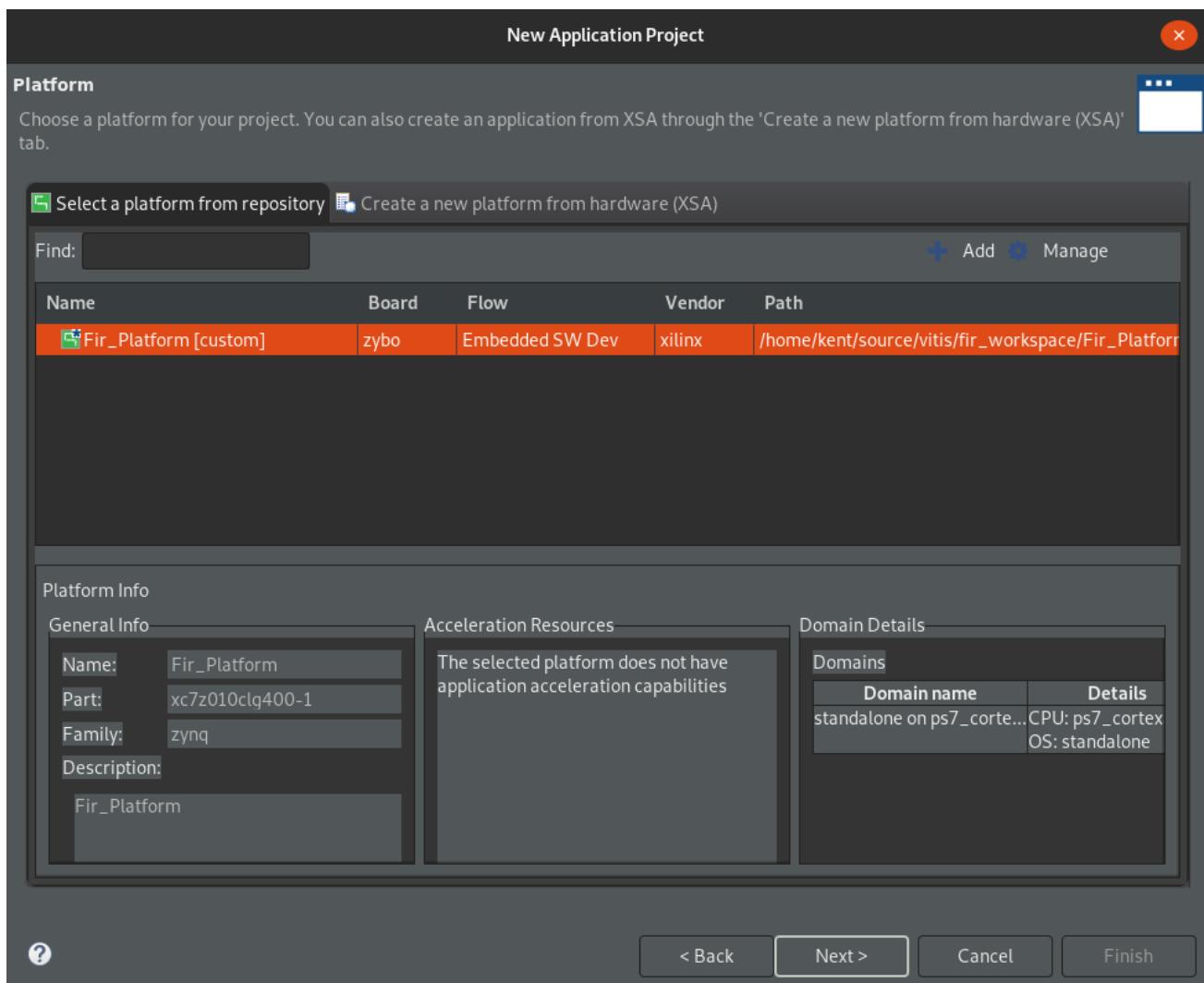
The selected platform does not have application acceleration capabilities

Domain Details

Domains	Domain name	Details
standalone on ps7_corte...	CPU: ps7_cortex	OS: standalone

?

< Back Next > Cancel Finish



Give the Application Project a suitable name and click **Next**.

## New Application Project



### Application Project Details

Specify the application project name and its system project properties



Application project name:

#### System Project

Create a new system project for the application or select an existing one from the workspace

Select a system project

Create new...

System project details

System project name:

Target processor

Select target processor for the Application project.

Processor	Associated applications
ps7_cortexa9_0	FirApplication

Show all processors in the hardware specification



< Back

Next >

Cancel

Finish

Click Next .

## New Application Project



### Domain

Select a domain for your project or create a new domain



Select the domain that the application would link to or create a new domain

Note: New domain created by this wizard will have all the requirements of the application template selected in the next step

#### Select a domain

standalone on ps7\_cortexa9\_0

+ Create new...

#### Domain details

Name:	standalone_domain
Display Name:	standalone on ps7_cortexa9_0
Operating System:	standalone
Processor:	ps7_cortexa9_0



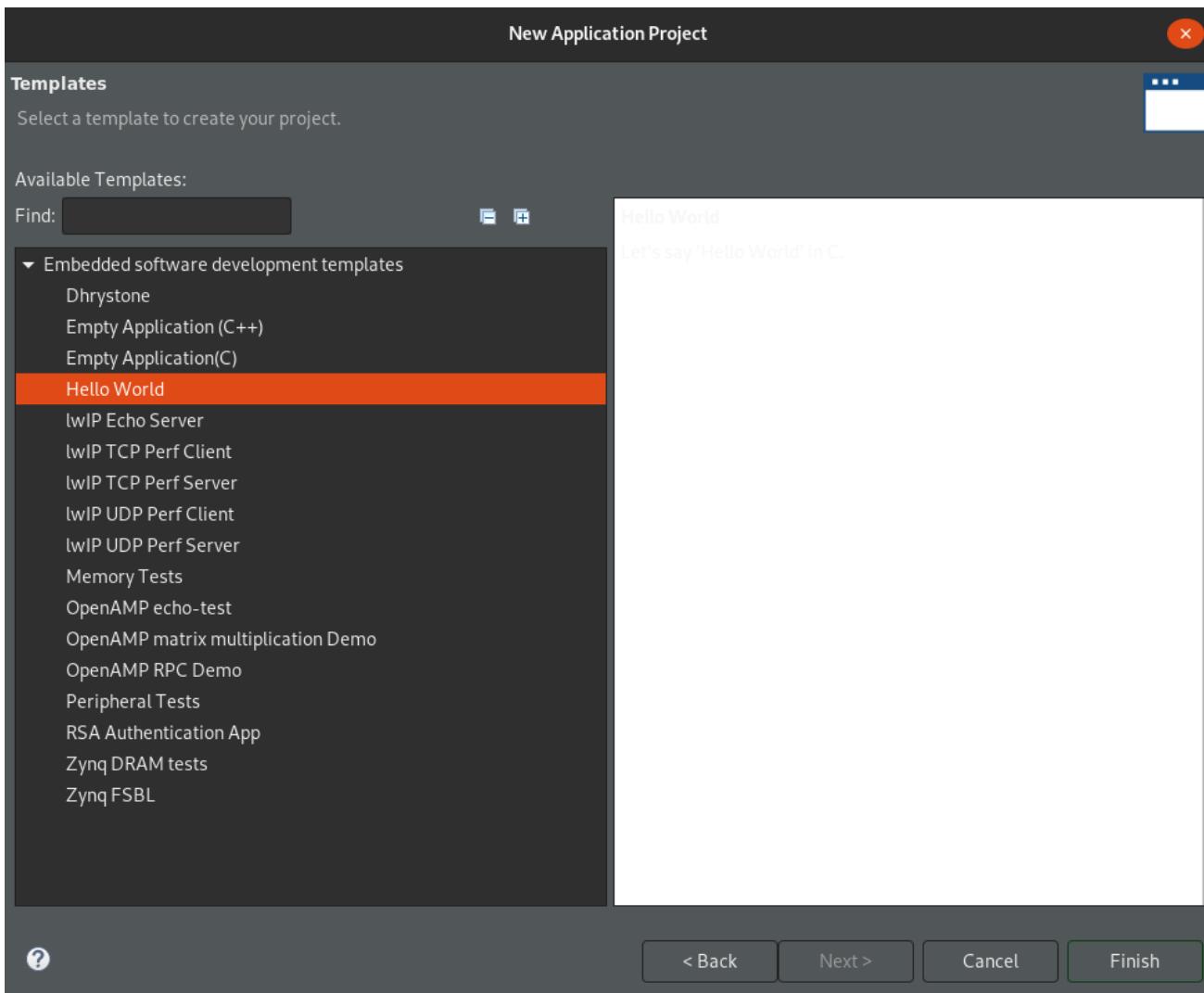
< Back

Next >

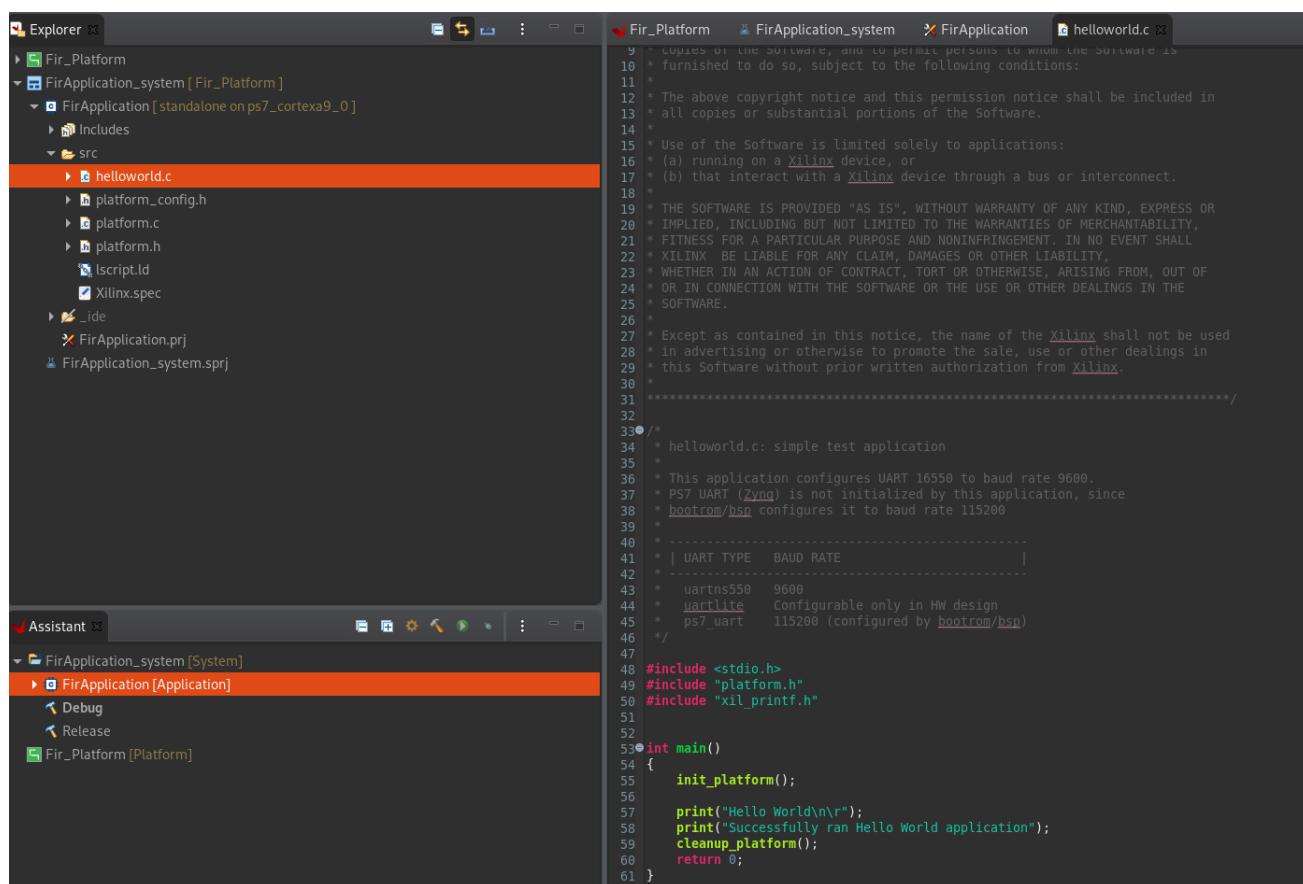
Cancel

Finish

Select the `Hello World` template and click `Finish`.



You will now be met by the generated Application Project. It should look something like this:



### 3.3 Get Source Code from Github

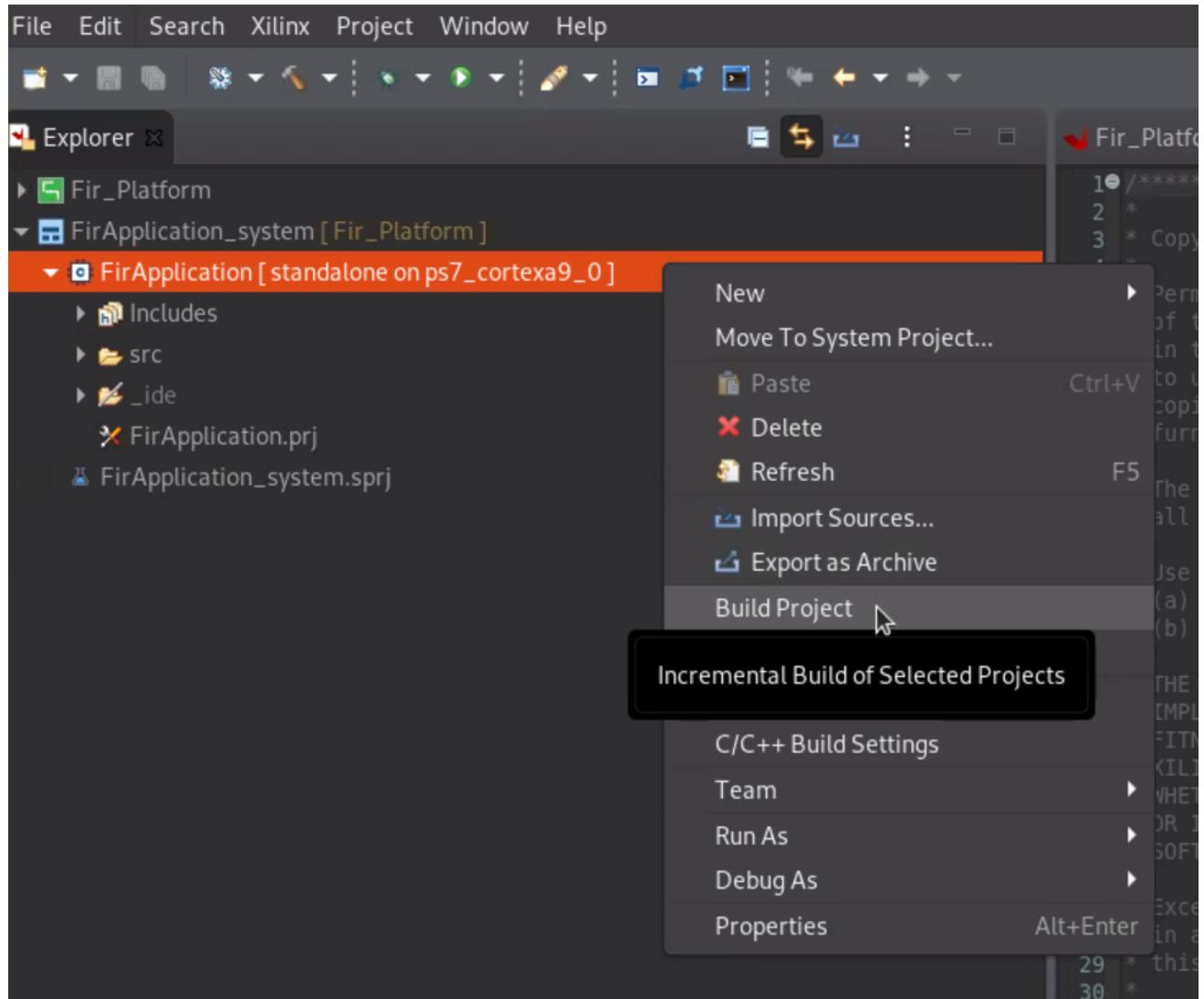
The source code can be found here:

- [main.c](#)

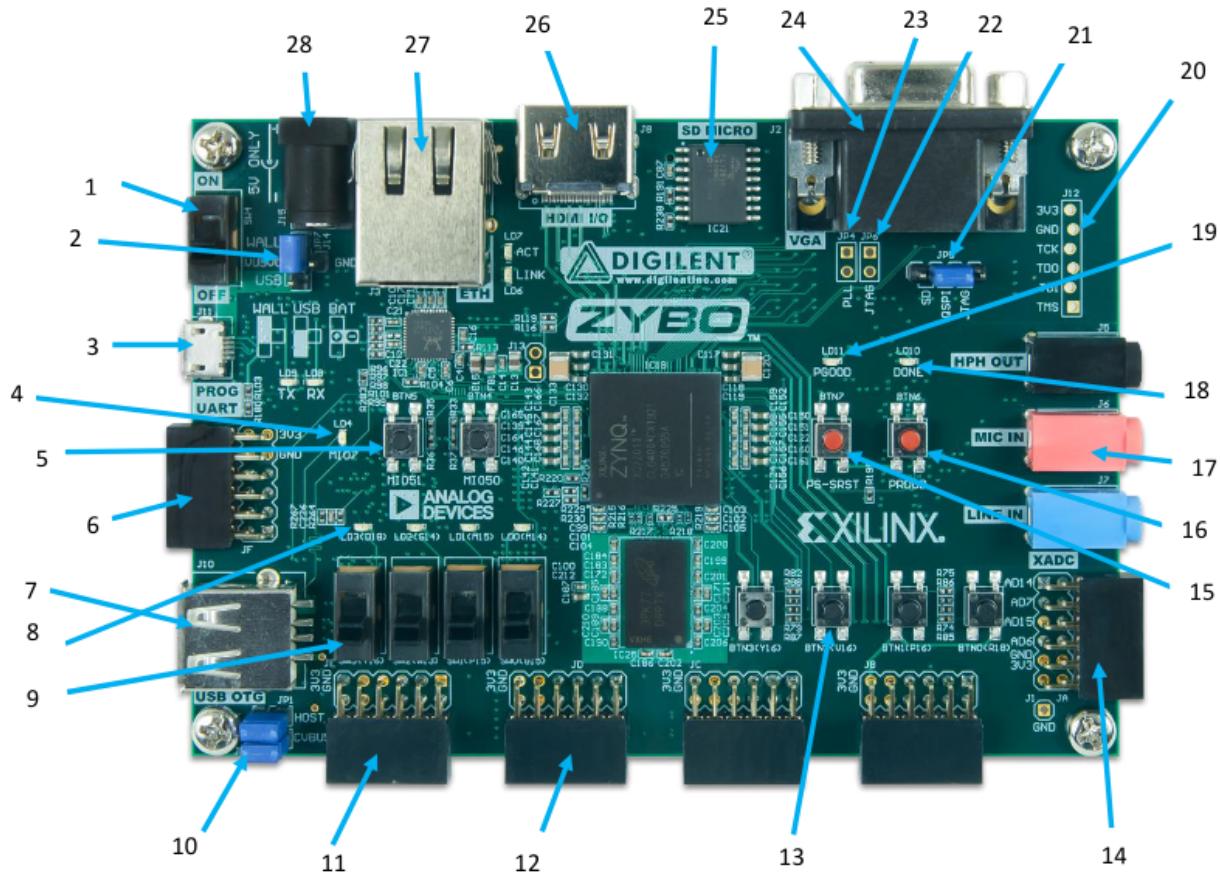
Swap the file content of `helloworld.c` with `src/main.c` from the Git repository.

### 3.4 Build and Run Project

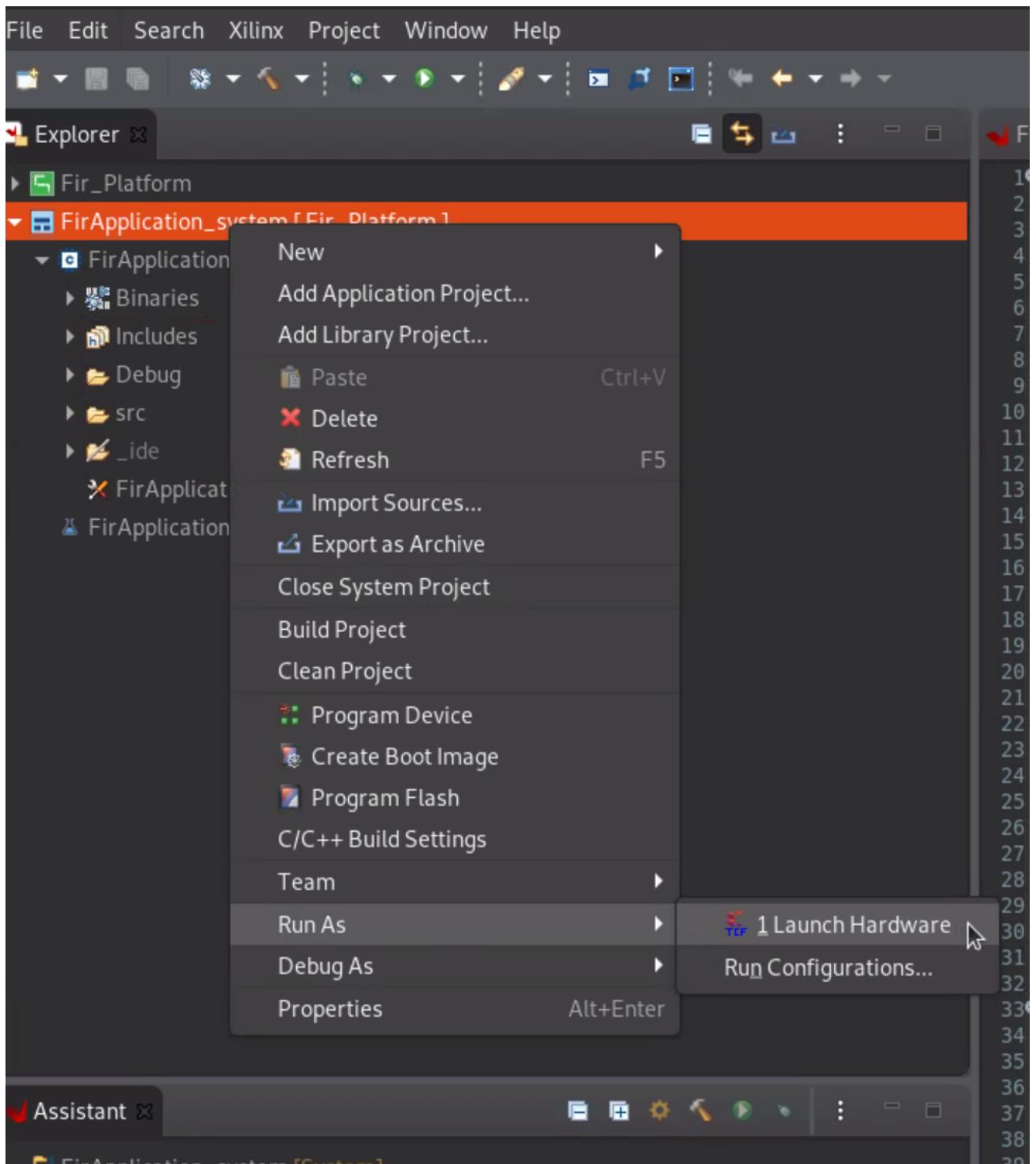
In the Explorer window, right-click your application and choose `Build Project`.



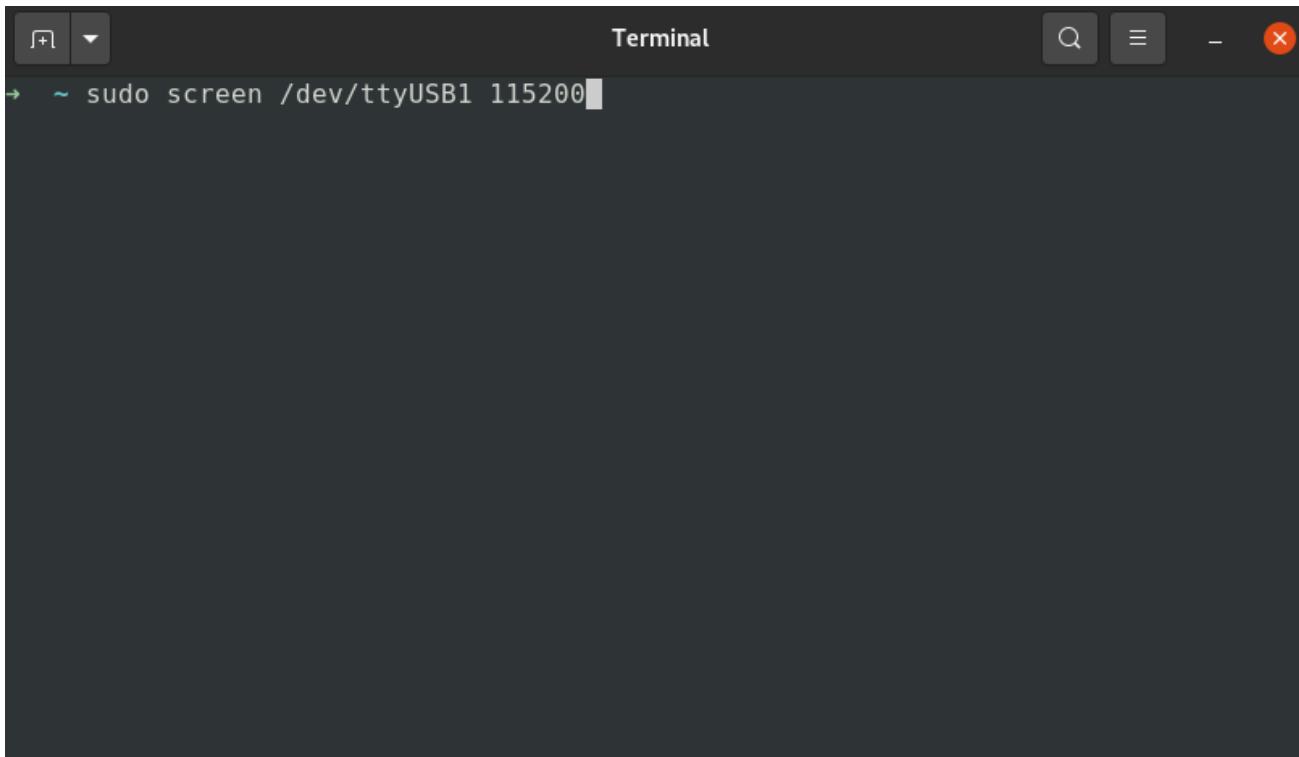
Remember to set the boot mode jumper (21 in the Figure) to JTAG mode (to the far right).



Then right-click your application again and choose Run As --> Launch Hardware .



Use a serial communication tool (screen, PuTTY) to interact with your application running on the Zybo.

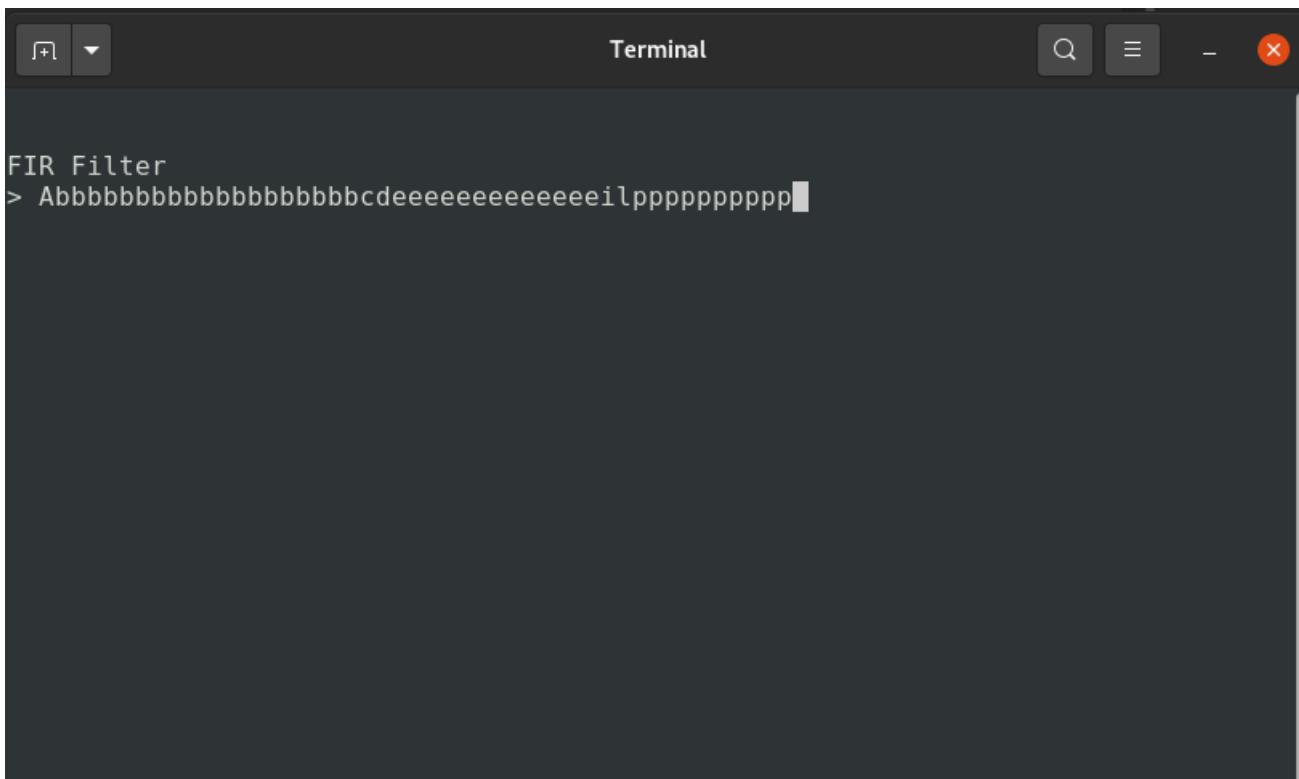


```
~ sudo screen /dev/ttyUSB1 115200
```

It should look something like this:

At all times it should print out the moving average of the three last entered characters.

For instance, entering the sequence `a b c` repeatedly, should result in only `b`'s as output as this will be the current moving average.



```
FIR Filter
> Abbbbbbbbbbccccdeeeeeeeeilppppppppp
```

---

## Releases

No releases published

---

## Packages

No packages published

---

## Contributors 2

 **oddek** Kent Odde

 **Feqzz** Stian Onarheim

---

## Languages

- **C** 100.0%