# User guide for co-emulation test bench:

## Before to start:

### Documents must to read:

1. **ZedBoard Getting Started Guide**

<http://zedboard.org/sites/default/files/documentations/GS-AES-Z7EV-7Z020-G-V7-1.pdf>

1. **Understand Zynq hardware and software development**

**Access this link to download Lab files: (you need to register an account first and fill a questionnaire)**

**Hardware:**

<http://zedboard.org/course/developing-zynq-hardware-vivado-20171-and-20174>

**Software:**

<http://zedboard.org/course/developing-zynq-software-vivado-20171-and-20174>

**Zedboard offers the video lab and lab instruction file in Download Load Files.**

### Pre-condition:

1. **Software tools: Vivado, Xilinx SDK, recommend version (2016.4) for both and Tera Term.**

Usually you will install the Xilinx SDK while you install the Vivado by default setting.

Tera Term is a terminal emulator to display the test result, you can download it with the below link:

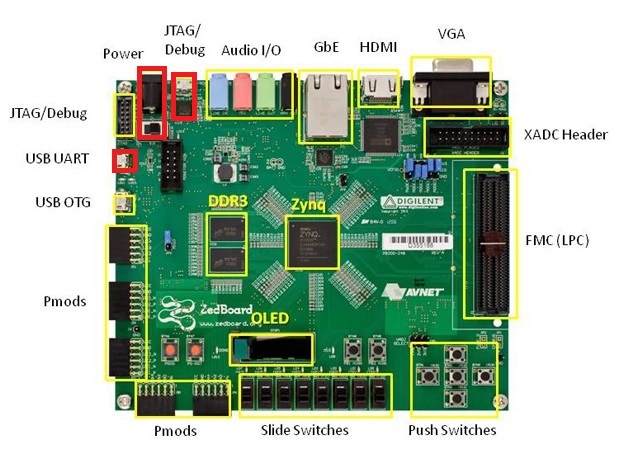
<https://ttssh2.osdn.jp/index.html.en>

1. **Hardware development board:**

Zedboard: **Before the experiment please make sure the connection of Power, JTAG/Debug, and USB-UART connection and Zedboard is power on.**

**For more detail information please check ZedBoard Getting Started Guide (section: ZedBoard Basic Setup and Operation)**

<http://zedboard.org/sites/default/files/documentations/GS-AES-Z7EV-7Z020-G-V7-1.pdf>



1. **Download the source code from the follow link**

Unzip the co\_emulation\_testbench.part01.rar -- co\_emulation\_testbench.part12.rar in one folder under your system disk.

<https://github.com/DinoCaoGit/a_serial_co_emulation_test_bench.git>

For example:

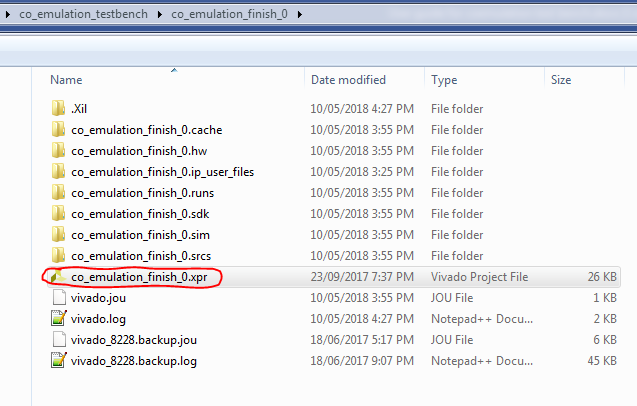


**Attention:**

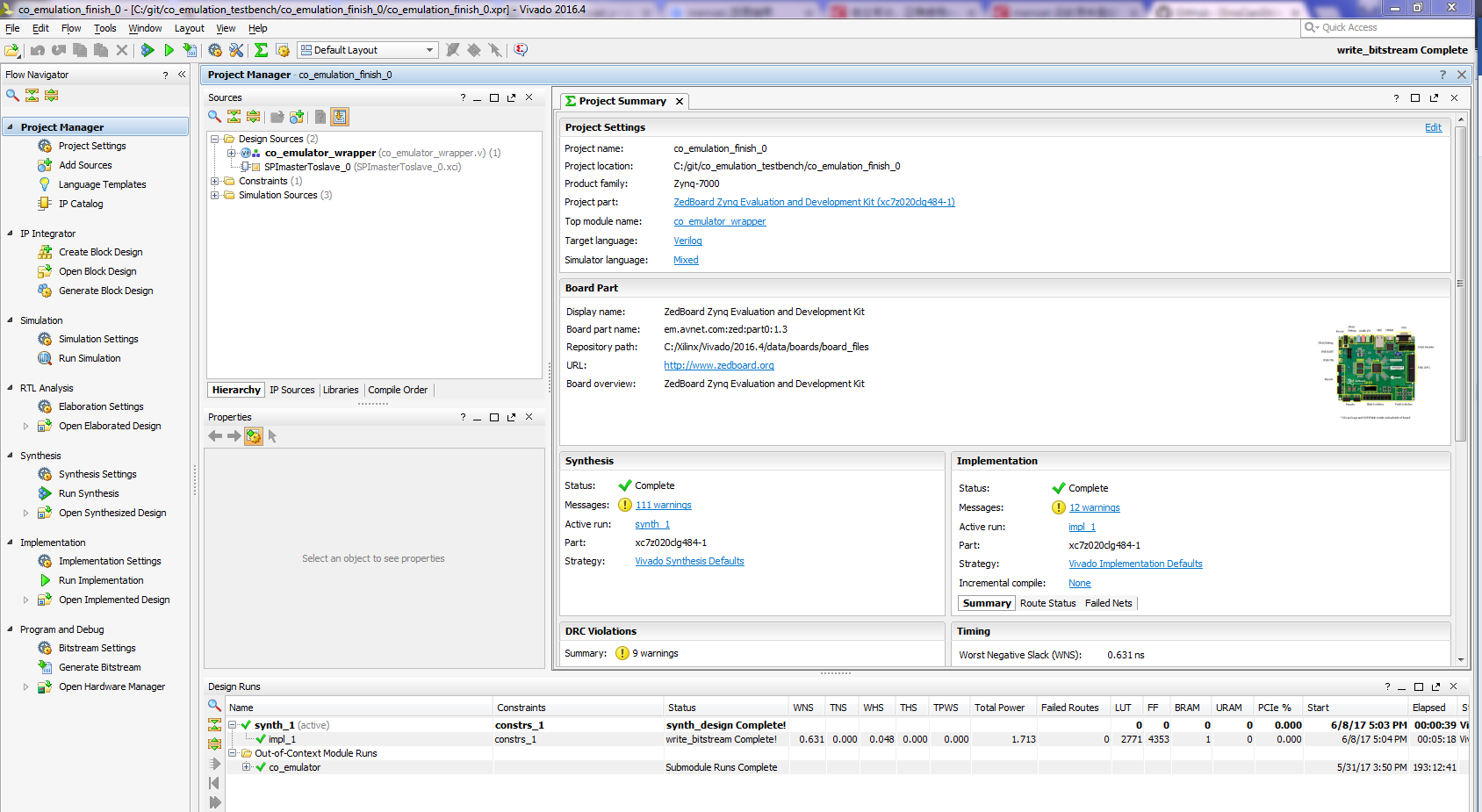
1. **The folder you create must be under the same logic disk when you install the Vivado (Recommend disk is your system disk)**
2. **The folder name can not include blank or other special characters**

## Open the project

1. Find the project file “**co\_emulation\_finish\_0.xpr**” in **..\co\_emulation\_testbench\co\_emulation\_finish\_0** folder



1. Double click it, the Vivado will open the project. Maybe need couple of minutes, be patient to wait!
2. The project is shown as follow:

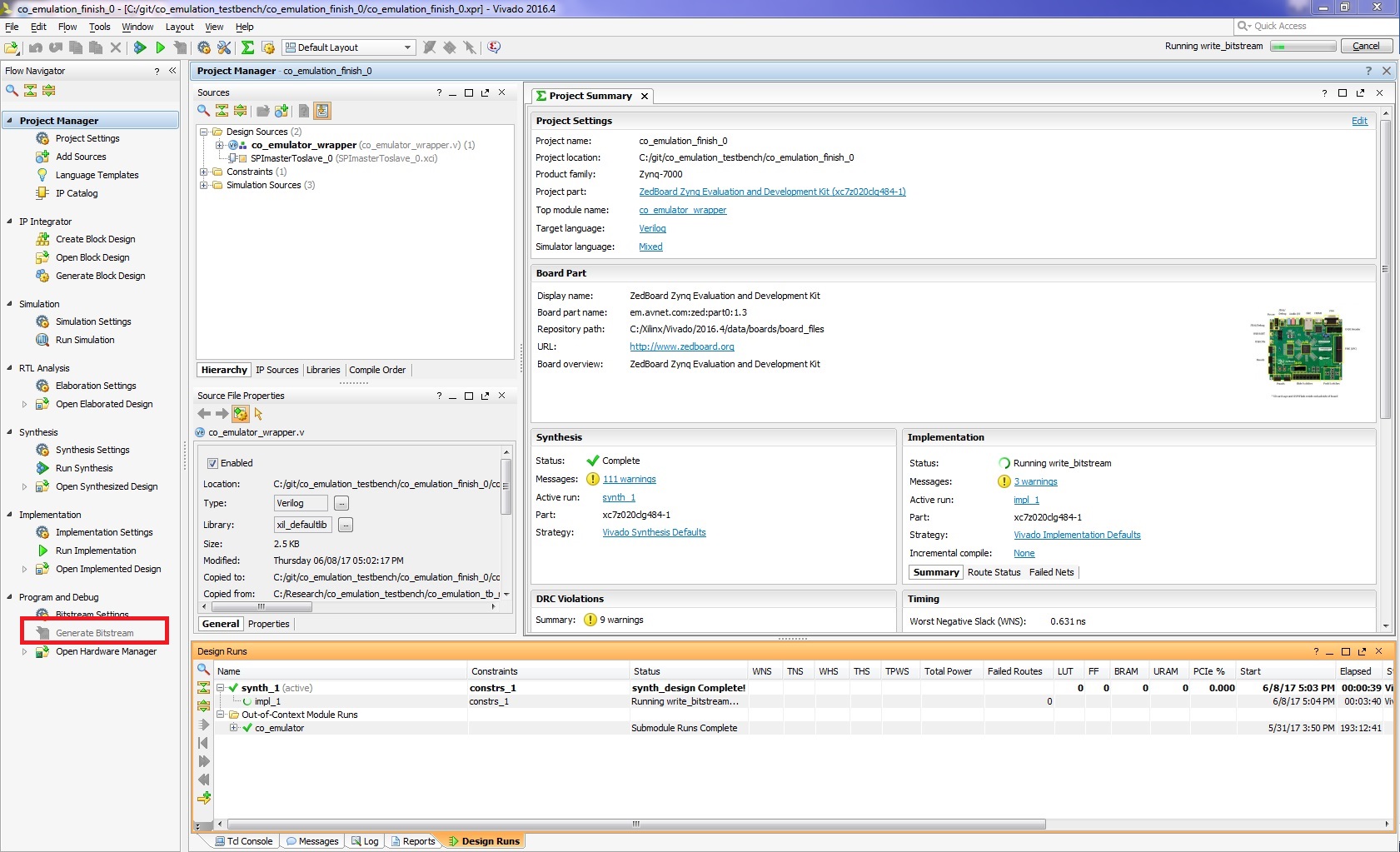


## Run the program

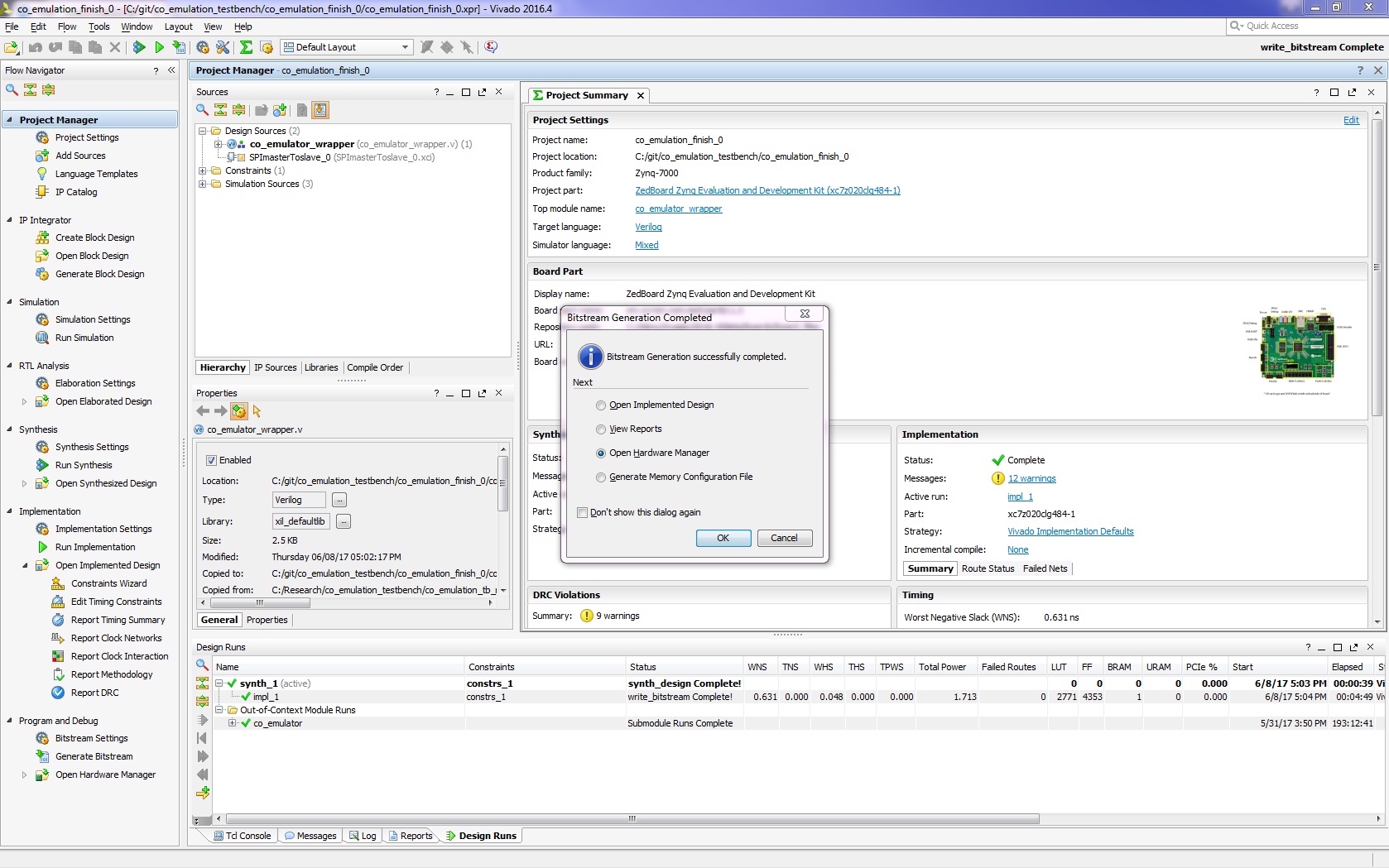
Before run the program, you should make sure your Vivado’s licence is valid.

### Generate the bit stream

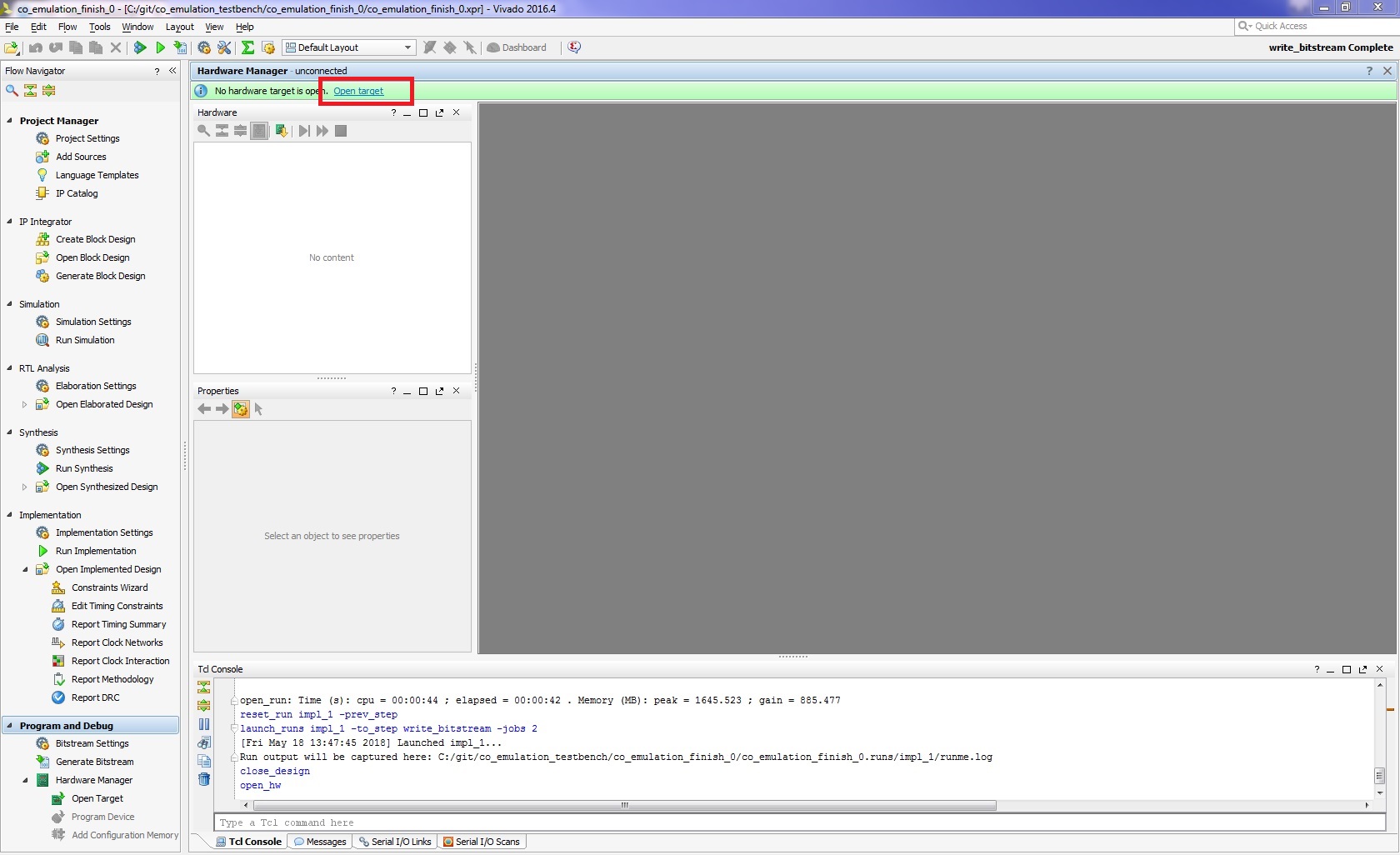
Double click “generate bitstream” to generate the bitstream file for FPGA.



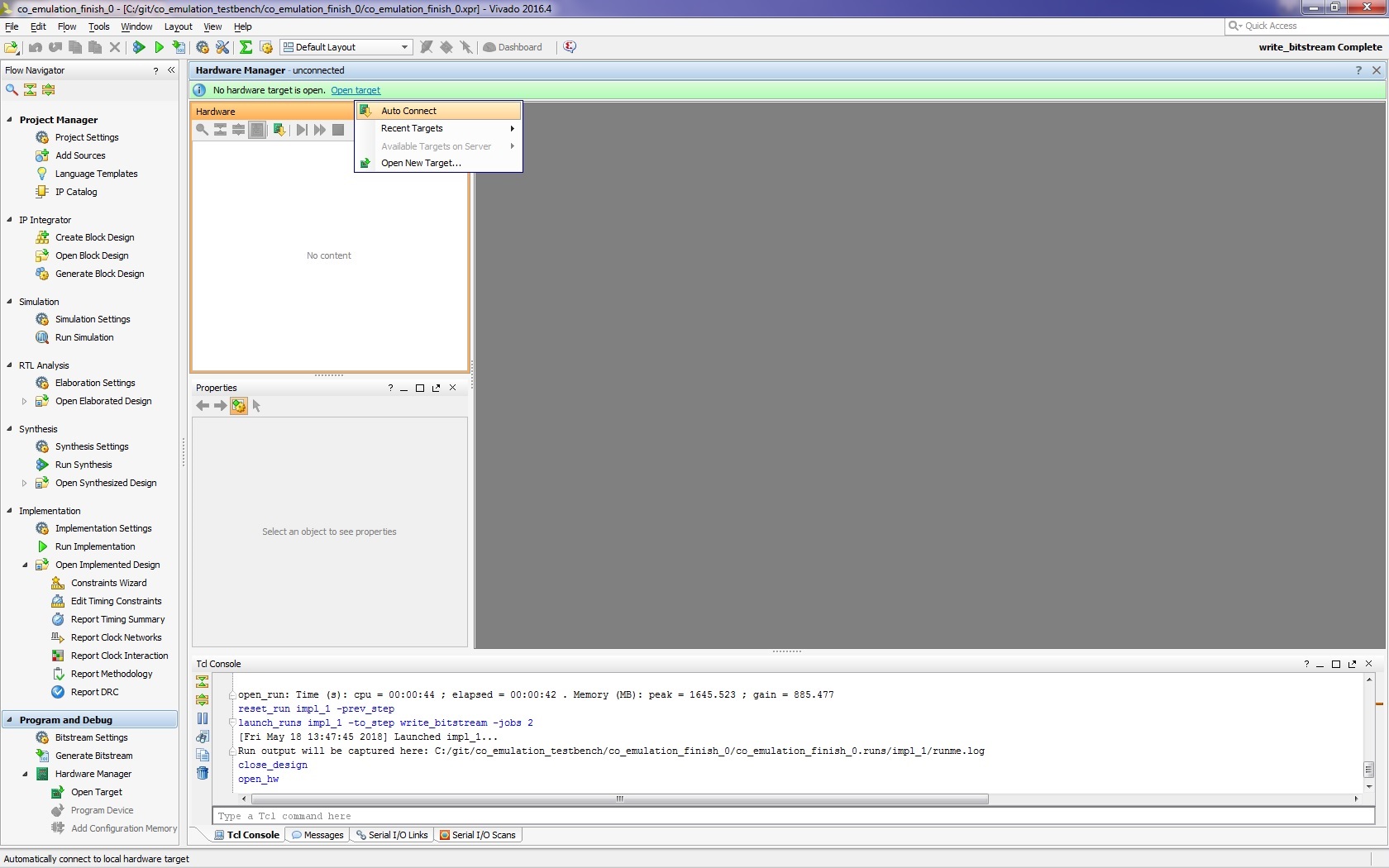
Choose “Open Implemented Design” and click “OK”



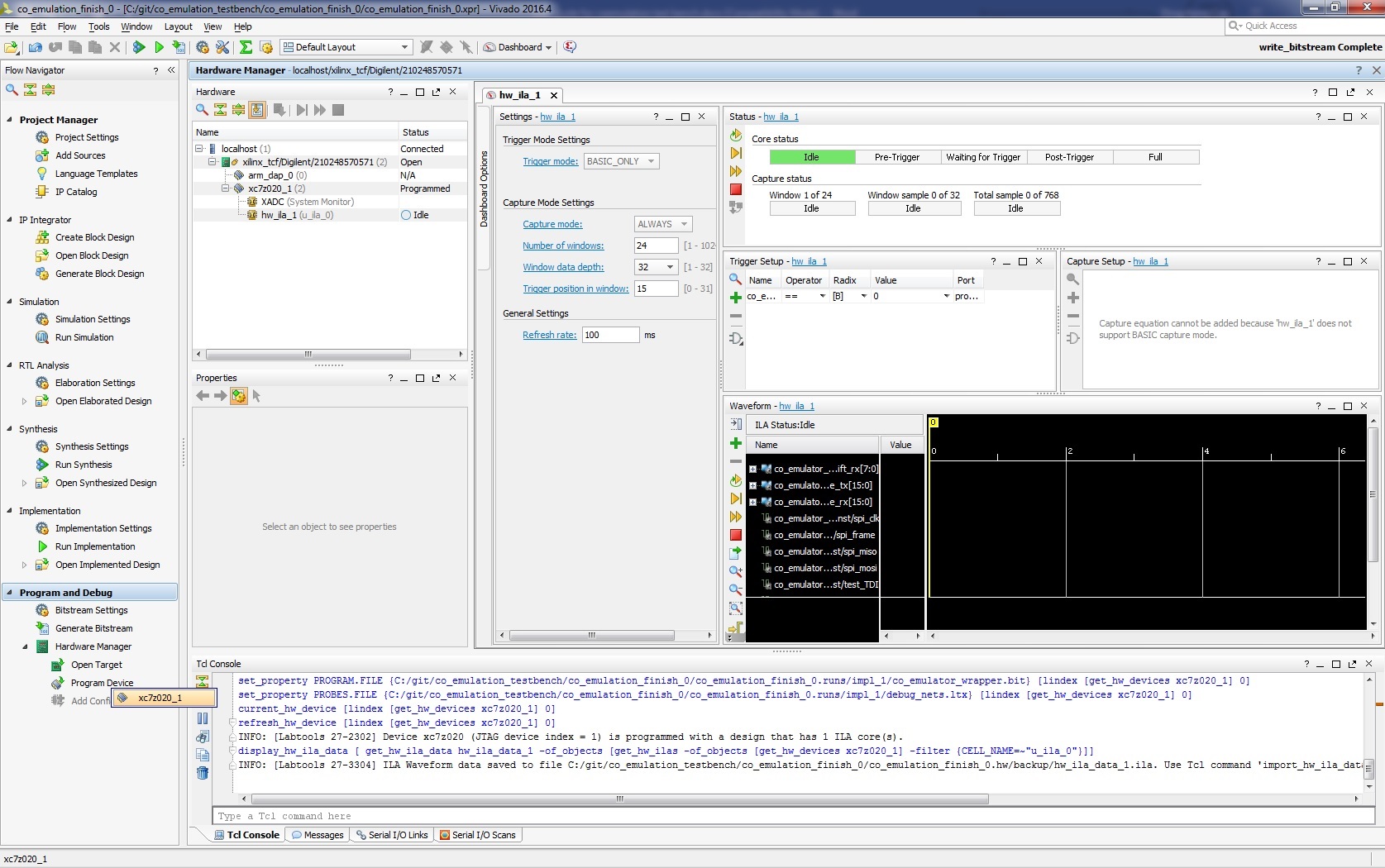
Open target (make sure the board connection is correct and green led is light)



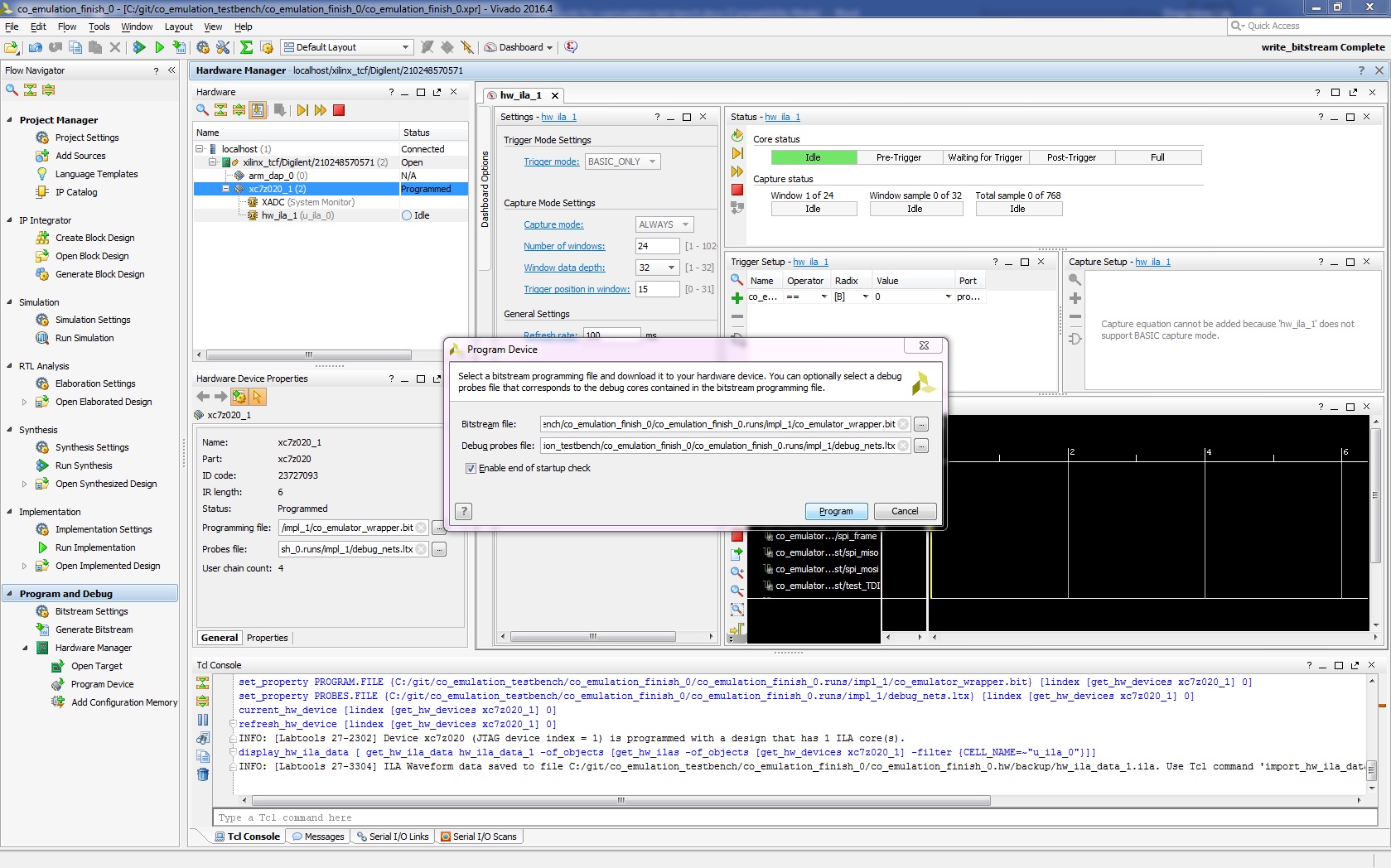
Click Auto Connect



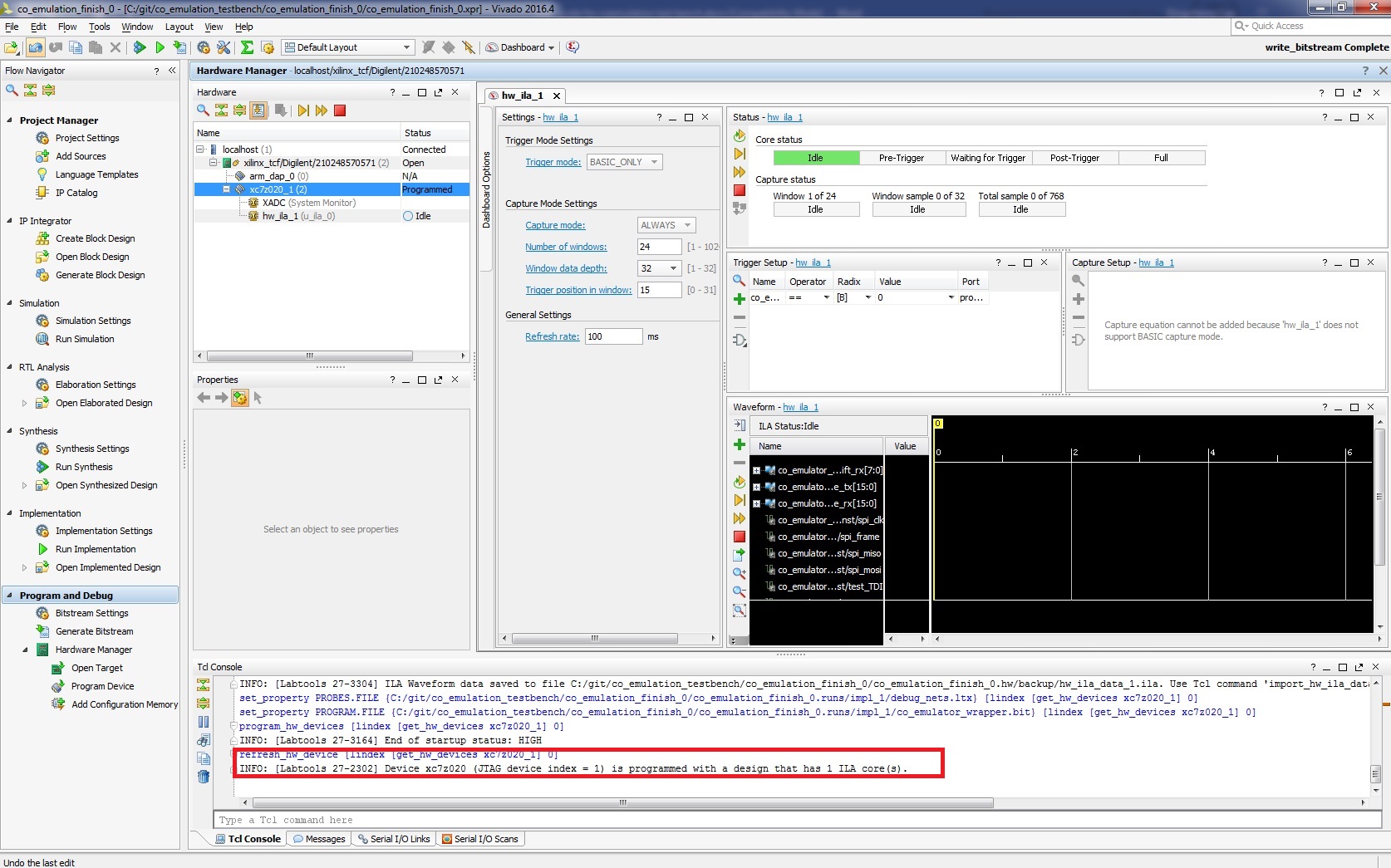
Program Device, click xc7z020\_1



Click program and make sure the blue led is light on.

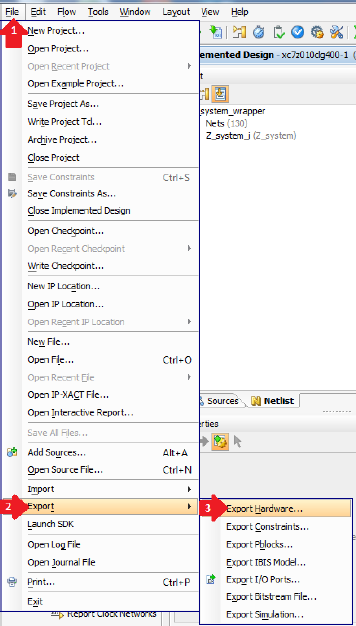


Check the program successful information

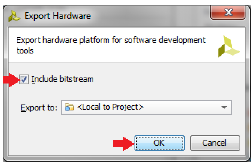


### Execute the software test bench

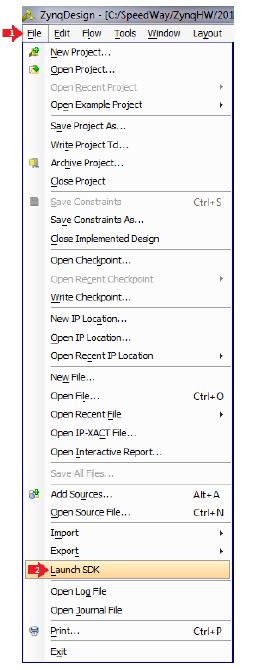
Export hardware



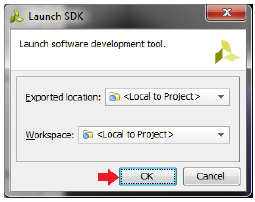
Include bitstream



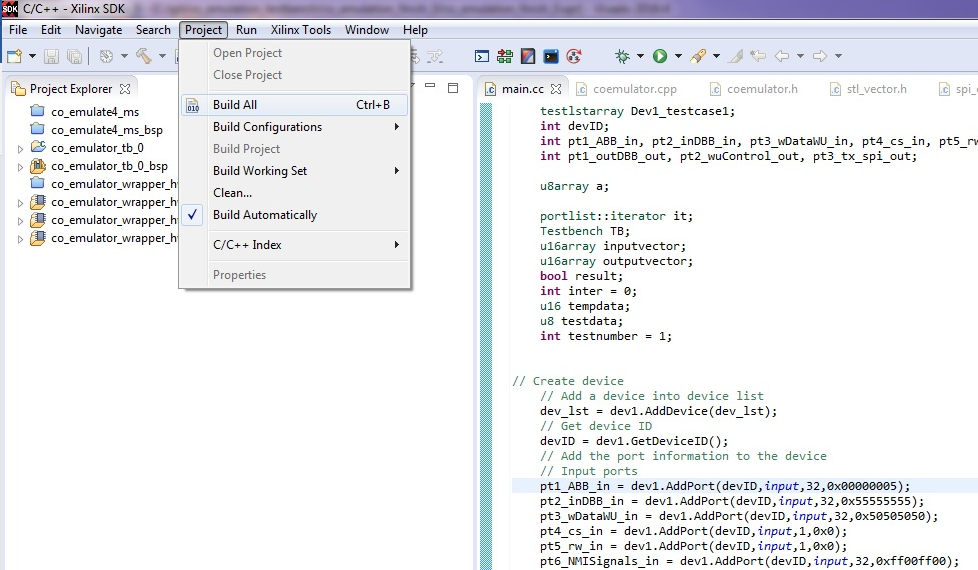
Launch SDK



Accept default Exported location and workspace

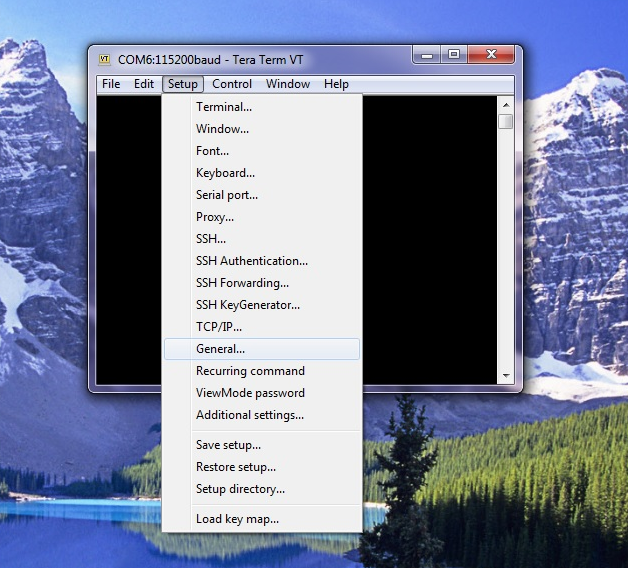


Build the project



Open Tera Term and setup UART port

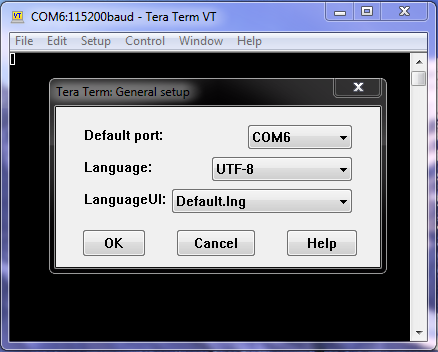
Setup->General…



Open device manger to check the USB serial Port (COM6)



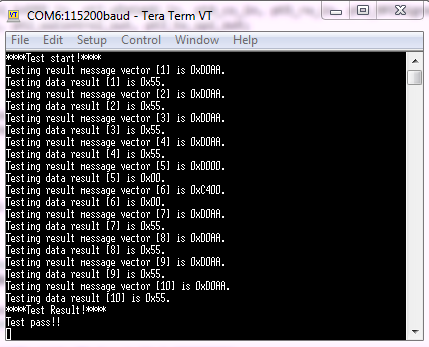
Set default port



Run the program



The result



## Software test bench

This part introduces the software test bench design and you can design your software test bench according to the introduction.

The source code of software test bench is stored in the project folder “co\_emulator\_tb\_0”



The software test bench includes spi\_commands.cpp, spi\_commands.h, coemulator.cpp, coemulator.h and main.cc files

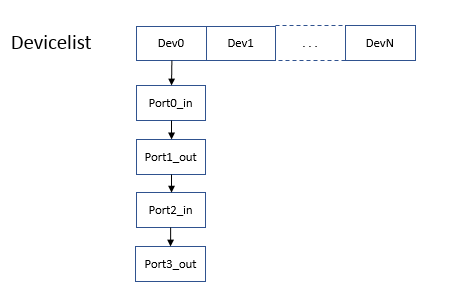
They are located in **C:git\co\_emulation\_testbench\co\_emulation\_finish\_0\co\_emulation\_finish\_0.sdk\co\_emulator\_tb\_0\src**

**The spi\_commands.cpp and spi\_commands.h include the functions to realize the spi communication on the software test bench.**

**The coemulator.cpp and coemulator.h include the classes used in the software test bench.**

### The design of software test bench

#### Device

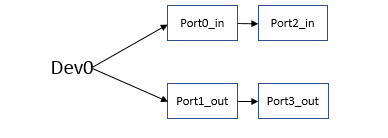


The devicelist: it’s a vector to store the devices to be tested.

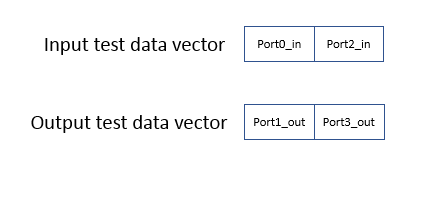
Every device is a class which includes a port list of the device.

#### Test case

Firstly, Test case separates the port list of the device into input port list and output port list.

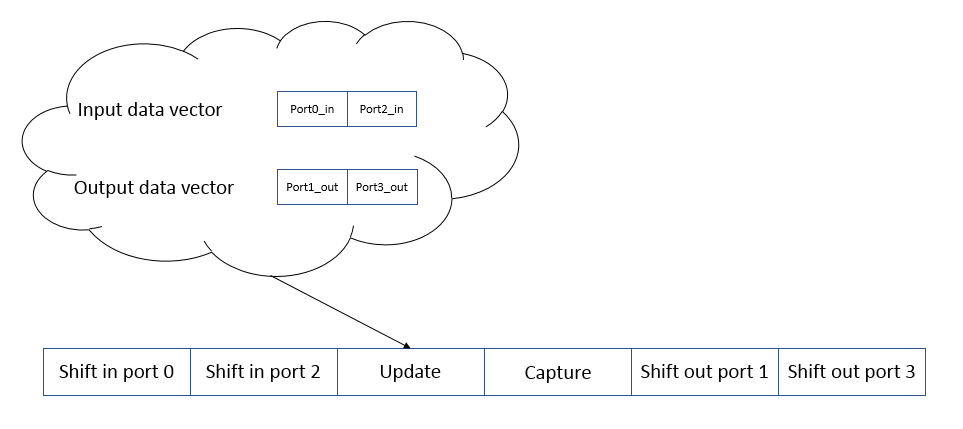


Secondly, Test case transfers the input port list and the output list into input test data vector and output test data vector.



#### Test Bench

The test bench generates the message vector for the testing

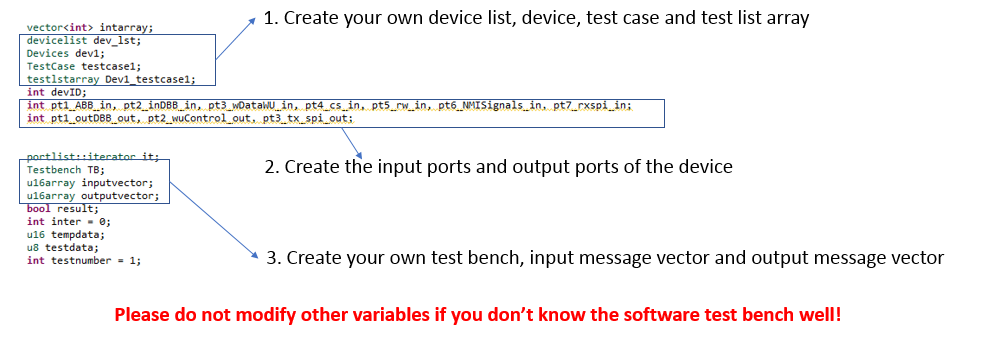


### Create a software test bench

The main.cc file includes three major parts of the software test bench: variables declaration, create DUT, and Execute test.

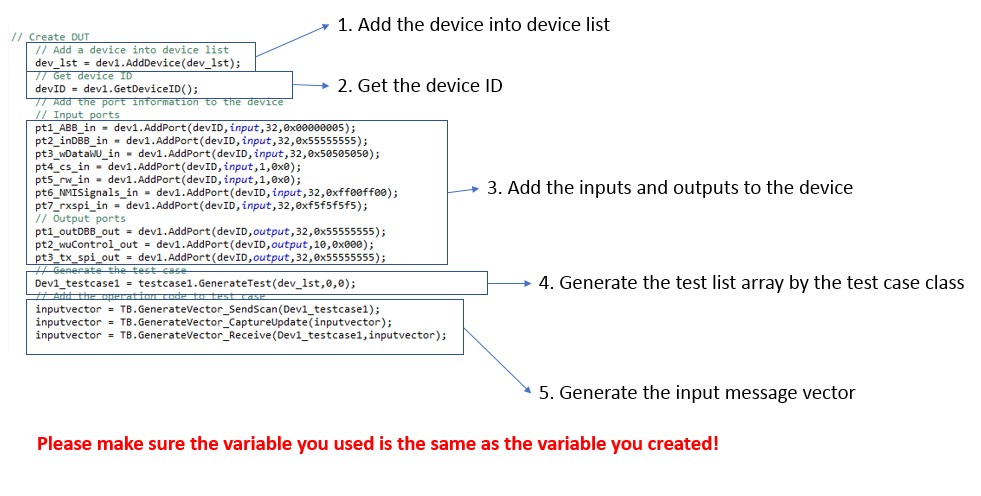
#### Variables declaration

Creates the variable for the software test bench in the following three steps:

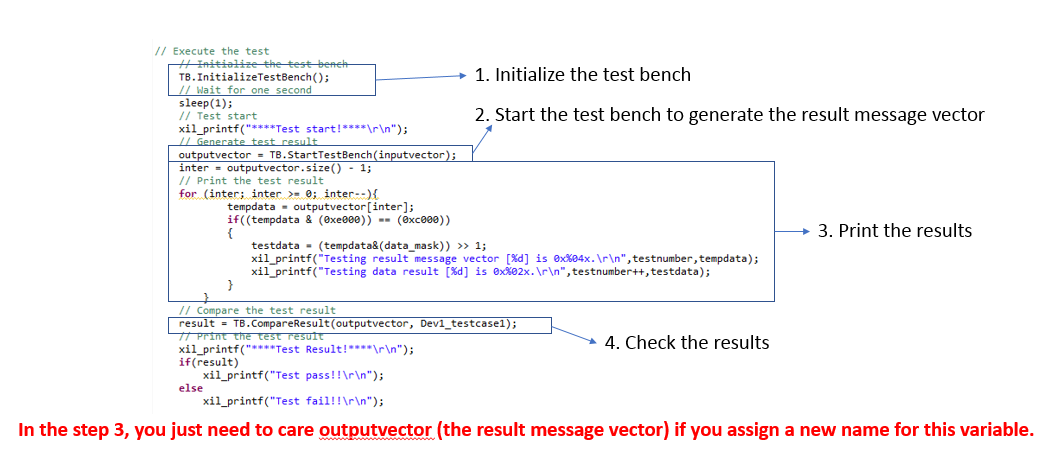


#### Create DUT

Add your own DUT into the software test bench with the five following steps.



#### Execute test



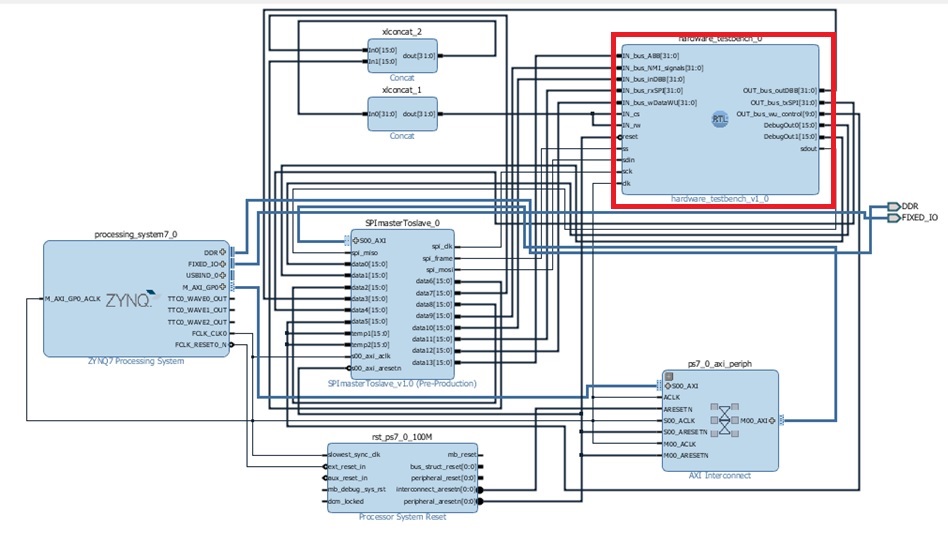
## Hardware test bench

The hardware test bench is designed as an IP core.

Please go through Zedboard **Hardware lab:**

<http://zedboard.org/course/developing-zynq-hardware-vivado-20171-and-20174>

The diagram of the hardware test bench



This manual just introduces the hardware test bench related code, the design of SPImasterToslave\_0 is descripted in the follow link:

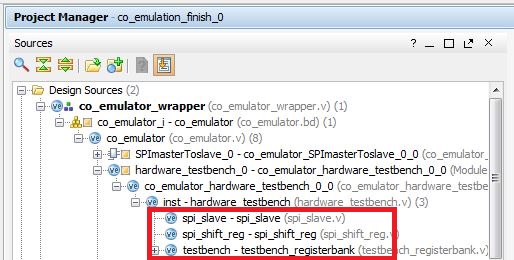
<https://www.beyond-circuits.com/wordpress/tutorial/tutorial17/>

Highly recommend going through the FPGA tutorial web set as the follow to understand the how to design a FPGA device on the Zedboard:

<https://www.beyond-circuits.com/wordpress/tutorial/>

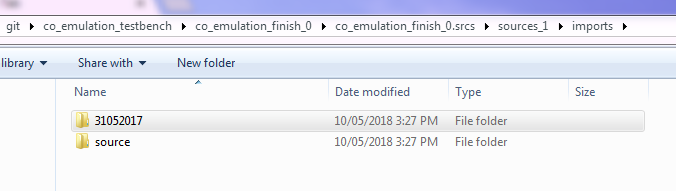
### Hardware test bench source code:

The source code is in the design source as following:



All the hardware test bench related file is stored in the folder:

..\git\co\_emulation\_testbench\co\_emulation\_finish\_0\co\_emulation\_finish\_0.srcs\sources\_1\imports\



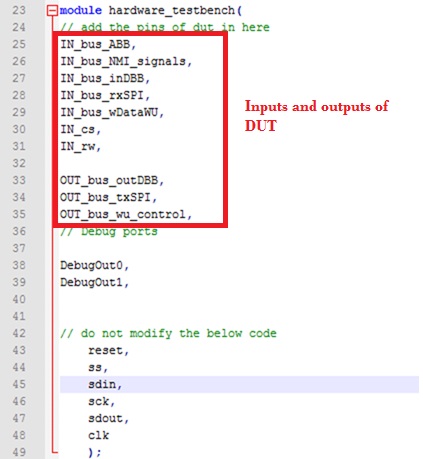
The folder 31052017 saves the spi slave code and the source folder saves the hardware test bench codes the other codes of the hardware test bench.

To add your DUT into the hardware test bench you should modify the file “hardware\_testbench.v” and “testbench\_registerbank.v”.

#### The “hardware\_testbench.v” source code

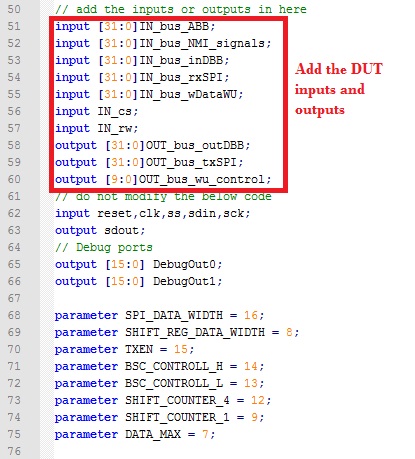
1. Module declaration

Adding the inputs and outputs of DUT in line 25 – 36.



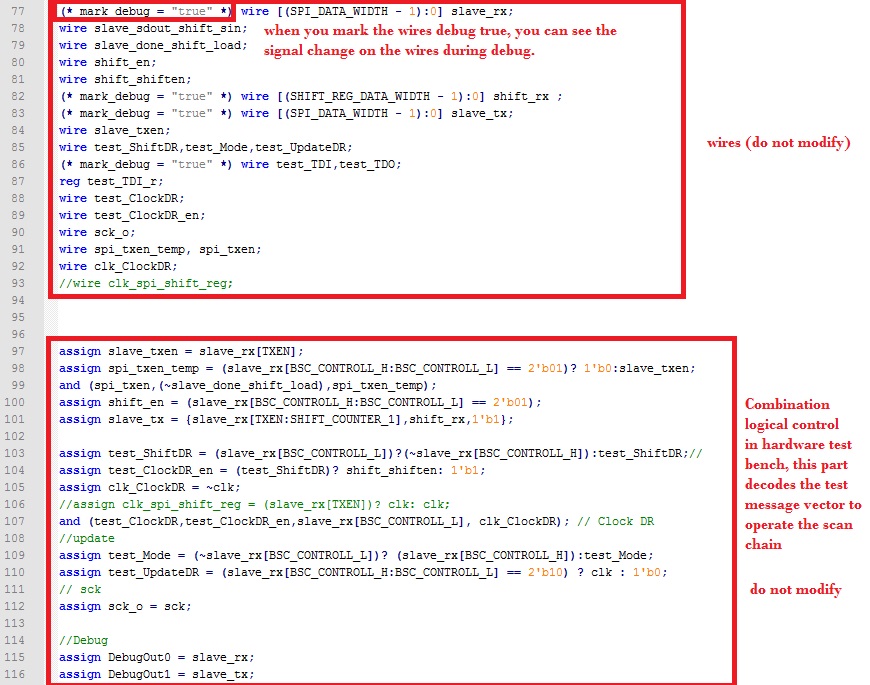
1. Inputs, outputs and parameters

Adding the DUT inputs and outputs variables in line 51 - 60

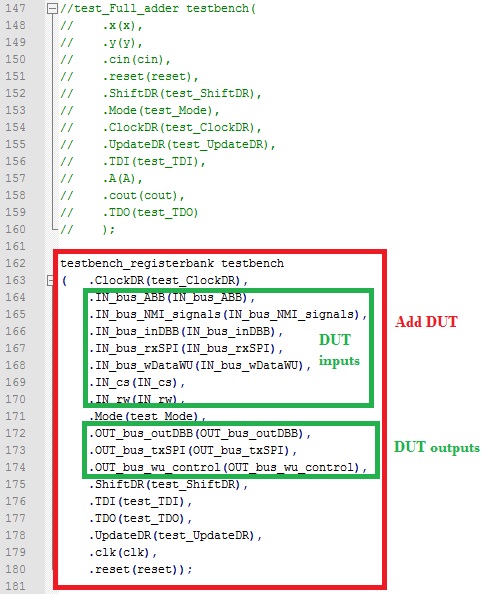


1. Combinational logic control in the hardware test bench and wires used in the hardware test bench

You do not need to change anything in this part



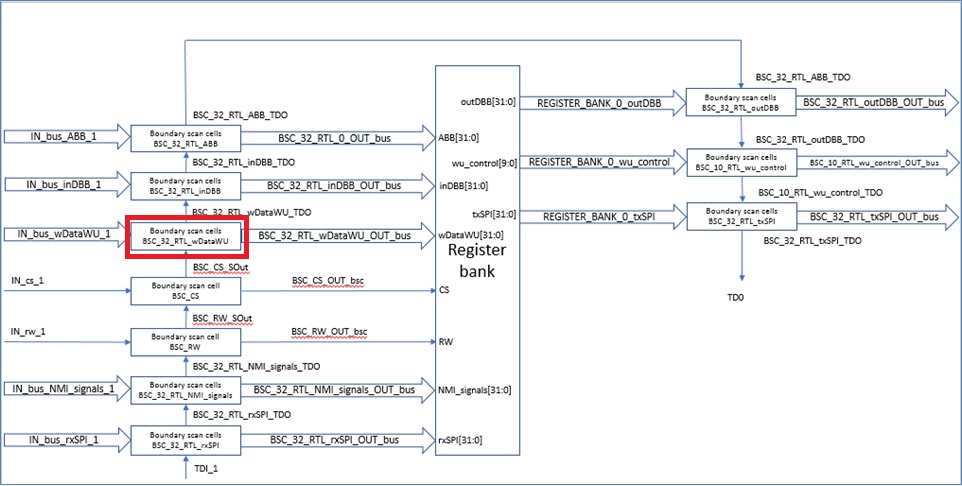
1. Adding DUT into the test bench



#### The “testbench\_registerbank.v” source code

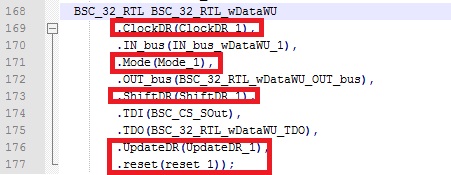
This file describes the connection of the scan chain and DUT

The connection between scan chain and DUT is presented as follow: this figure doesn’t show the ClockDR, Mode, ShiftDR, UpdateDR and reset connection.

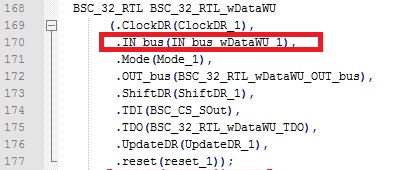


Here is an example how to connect boundary scan cells with DUT

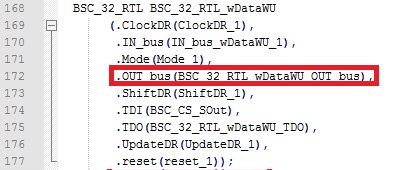
1. Connect ClockDR, Mode, ShiftDR, UpdateDR and reset ports of boundary scan cells to ClockDR\_1, Mode\_1, ShiftDR\_1, UpdateDR\_1 and reset\_1 wires which are connected to the ClockDR, Mode, ShiftDR, UpdateDR and reset ports of testbench\_registerbank.



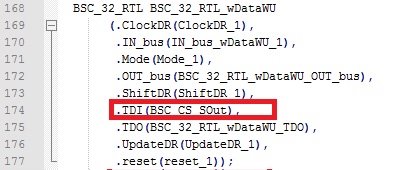
1. Connect IN\_bus of boundary scan cells to IN\_bus\_wDataWU\_1 wire which is connected IN\_bus\_wDataWU port of testbench\_registerbank.



1. Connect OUT\_bus of boundary scan cells to BSC\_32\_RTL\_wDataWU\_OUT\_bus wire which is connected wDataWU port of DUT.



1. Connect TDI of boundary scan cells to the output of boundary scan cell BSC\_CS\_Sout.



1. Connect TDO of boundary scan cells to wire of BSC\_32\_RTL\_wDataWU\_TDO which is connected to the input of boundary scan cells BSC\_32\_RTL\_inDBB.

