PSG COLLEGE OF TECHNOLOGY

Department of Electronics and Communication Engineering

**Smart Home Automation using Zybo Z7-10**

Main tutorial source: <https://www.youtube.com/watch?v=z613J8RVpog&t=13s> (materials in the description)

I completed upto 3rd step. The final step is simple but I dint have the time to do it

Contact : samuelpriyadarshan@gmail.com

Software Environment:

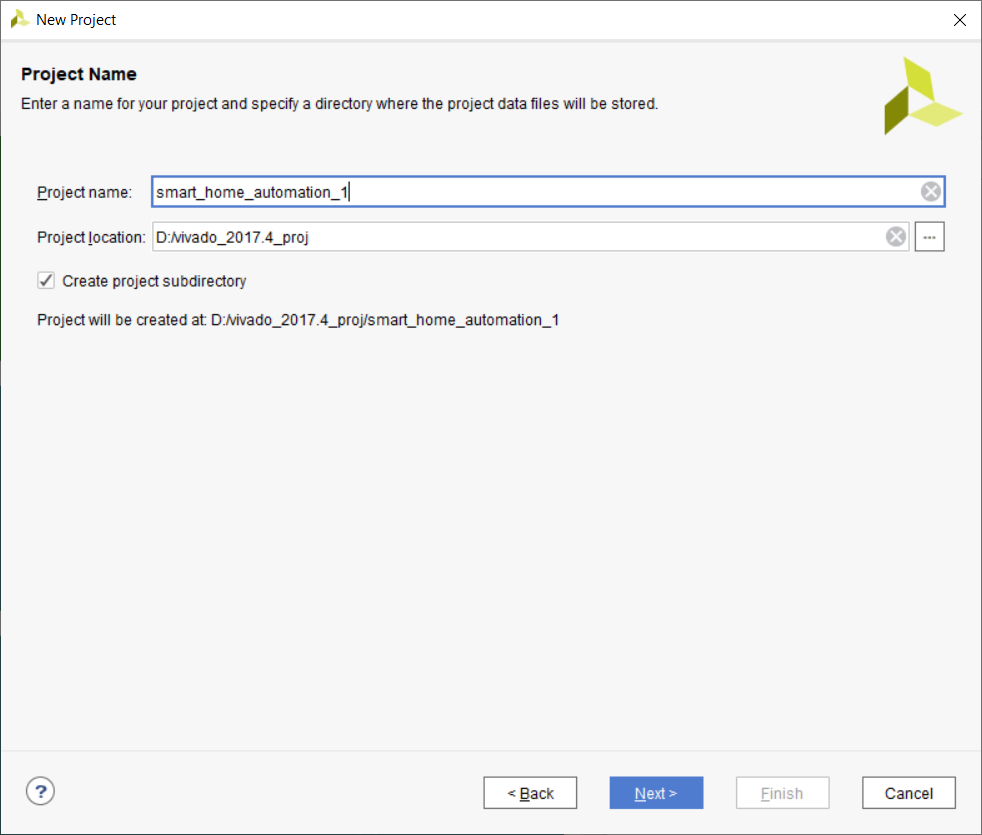
* Xilinx Vivado 2017.4
* Xilinx SDK 2017.4
* Xilinx PetaLinux 2017.4
* Ubuntu 16.04 (64-bit)

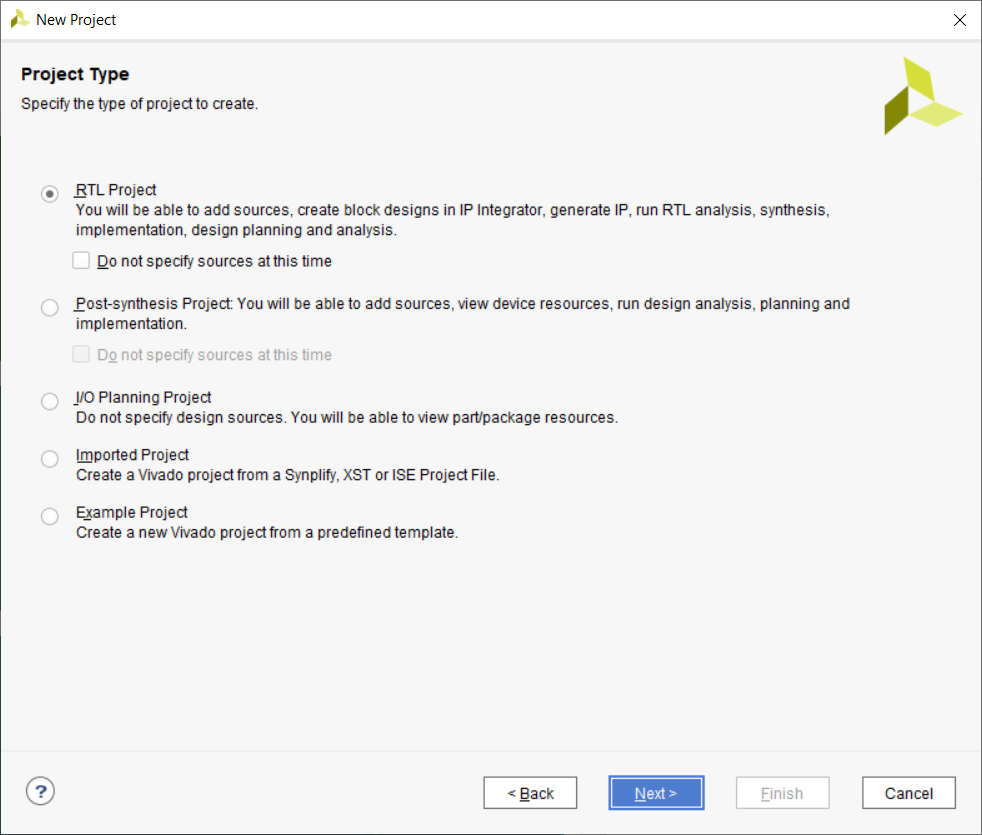
\*I tried a lot of versions but the environment worked perfectly only with these exact same versions

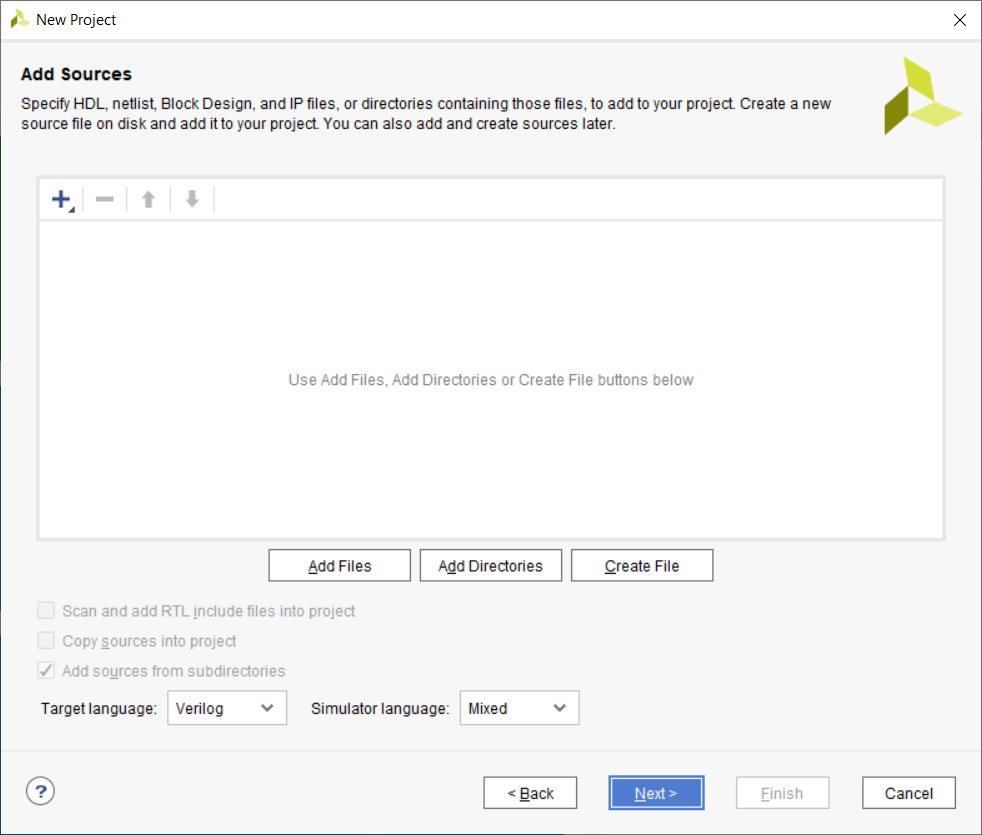
Step 1: Creating project in Xilinx Vivado 2017.4

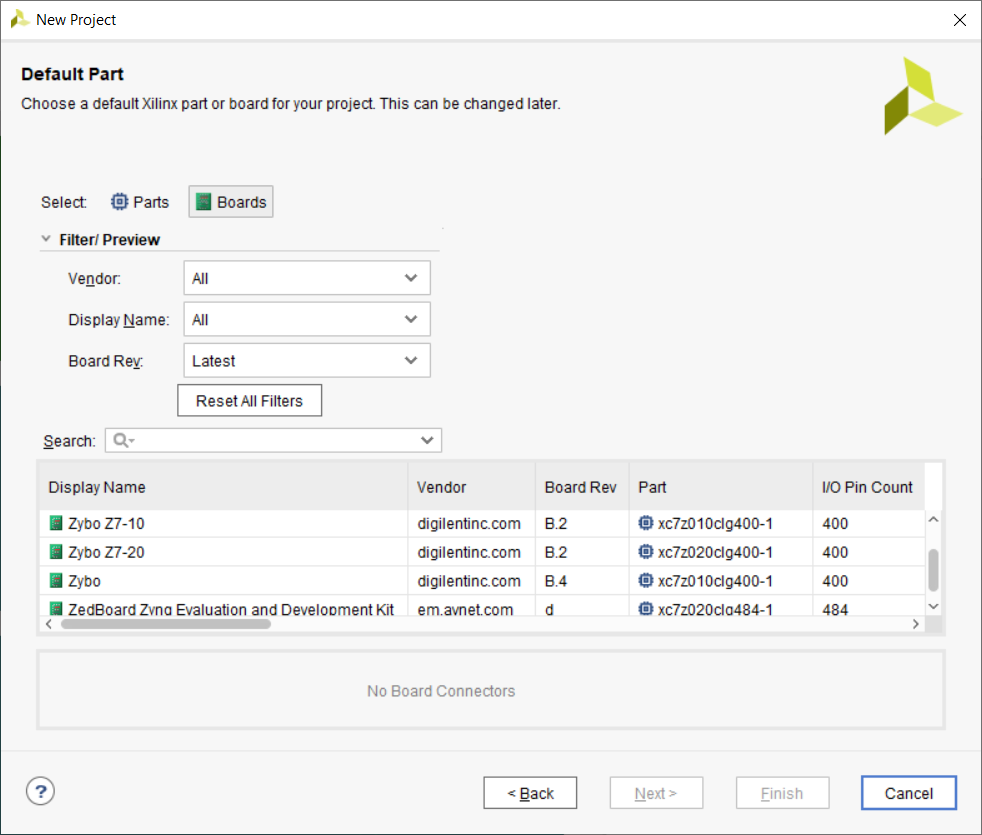
Open Xilinx Vivado 2017.4 > Create Project > Enter project name > RTL project > Target Language : Verilog, Simulator Language : Mixed > Under default parts choose boards tab and select Zybo (board must be installed manually)\* > Finish

\*https://reference.digilentinc.com/reference/software/vivado/board-files









Step 2: Block design

Under IP INTEGRATOR click Create Block Design

Enter design name

Click on the  symbol or (Ctrl + I)

ZYNQ7 Processing System (Double Click)

Make a connection between FCLK\_CLK0 and M\_AXI\_GP0\_ACLK

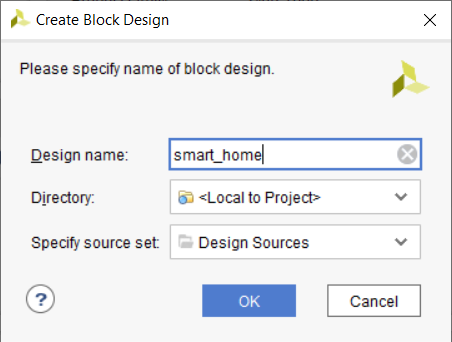
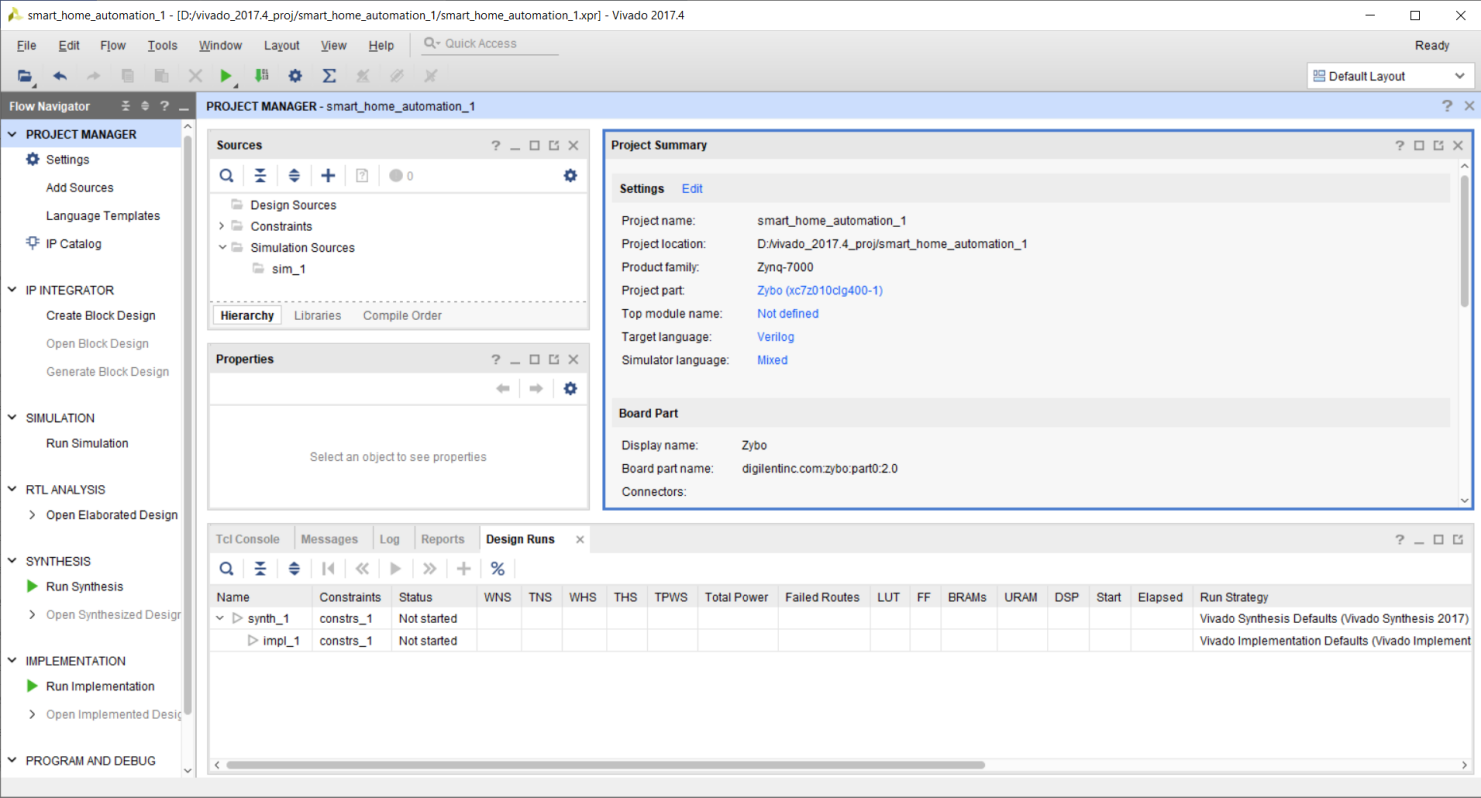
Click Run Block Automation

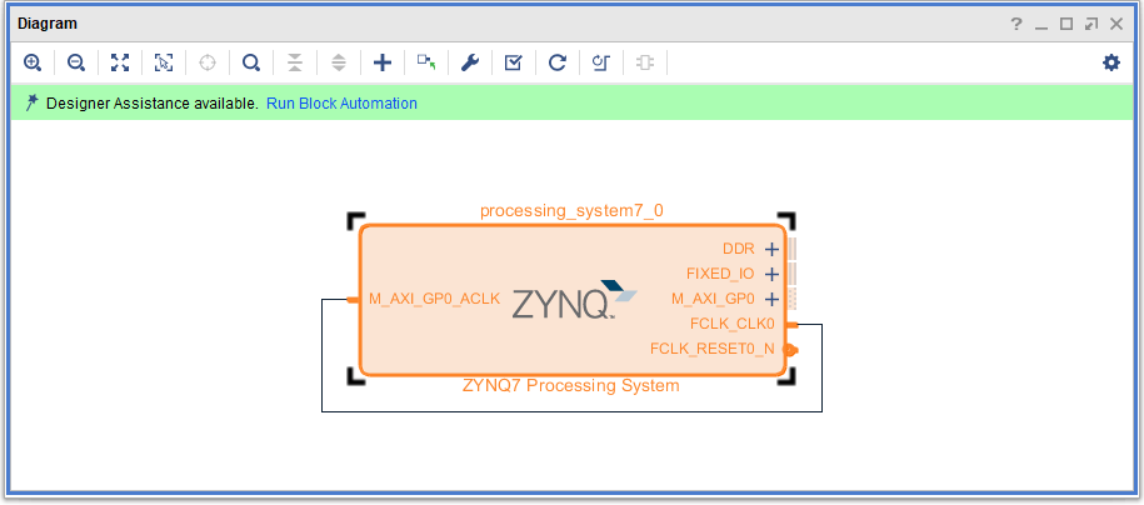
(Double Click the IP)

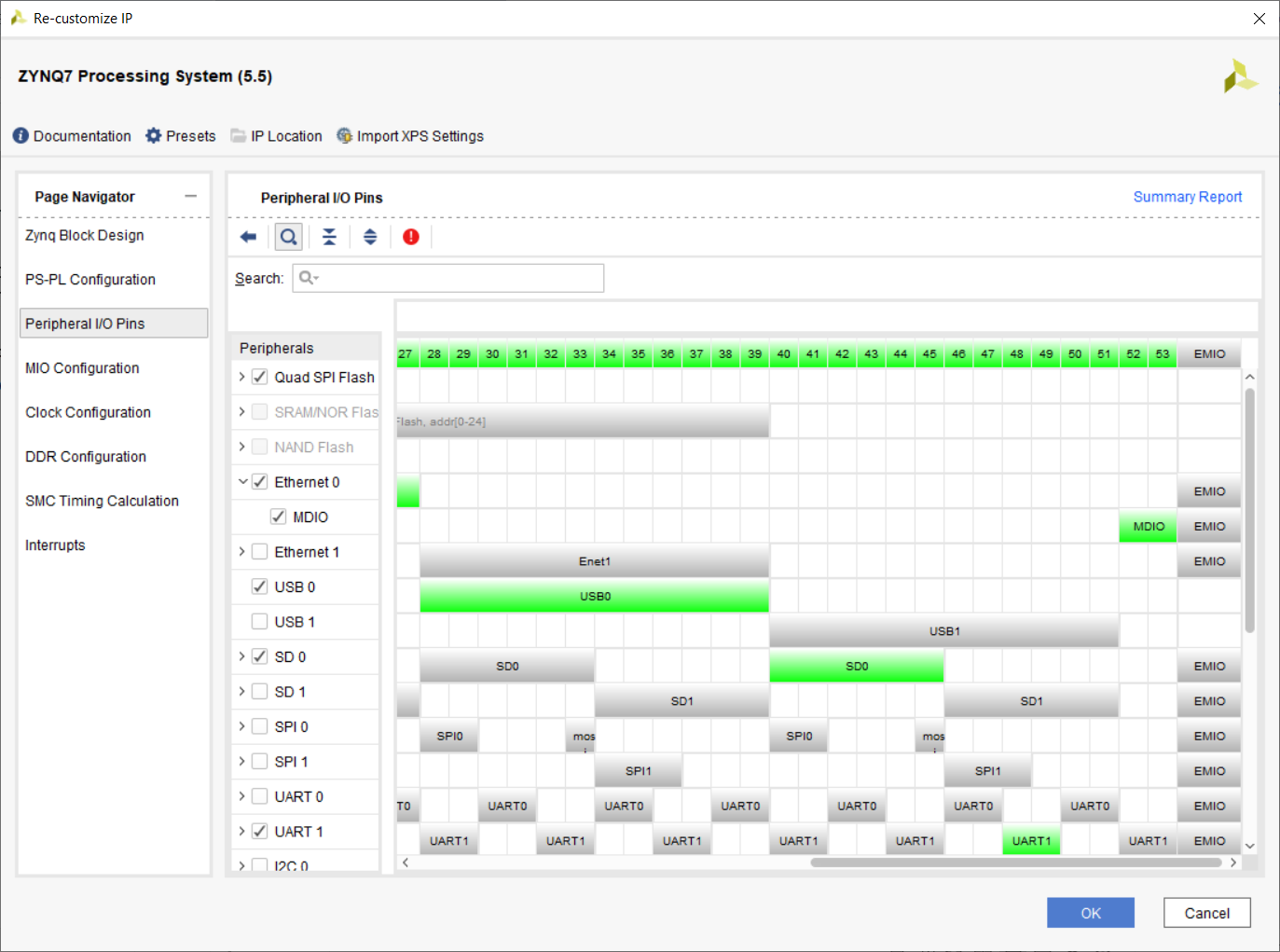
Select Peripheral I/O Pins > Ethernet0 (Drop down arrow) > Enable MDIO

Ensure all the options in processor\_1,2,3.jpg are selected (attached with the project documentation)

\*However only Ethernet, SD and GPIO only will be necessary, but if you have time try only with these

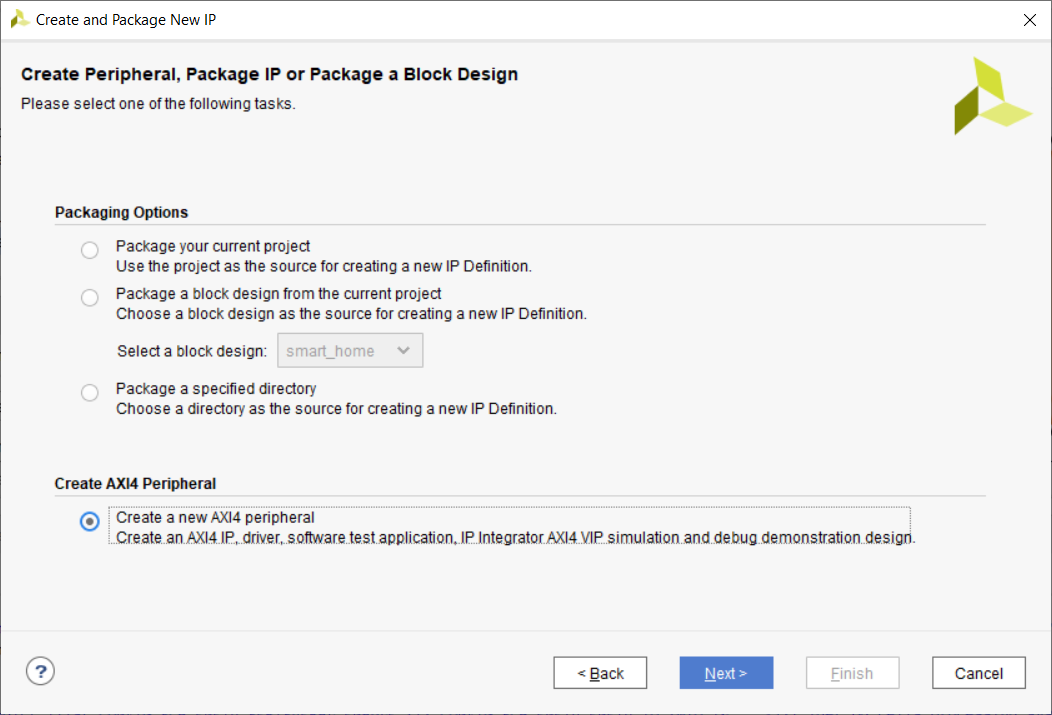
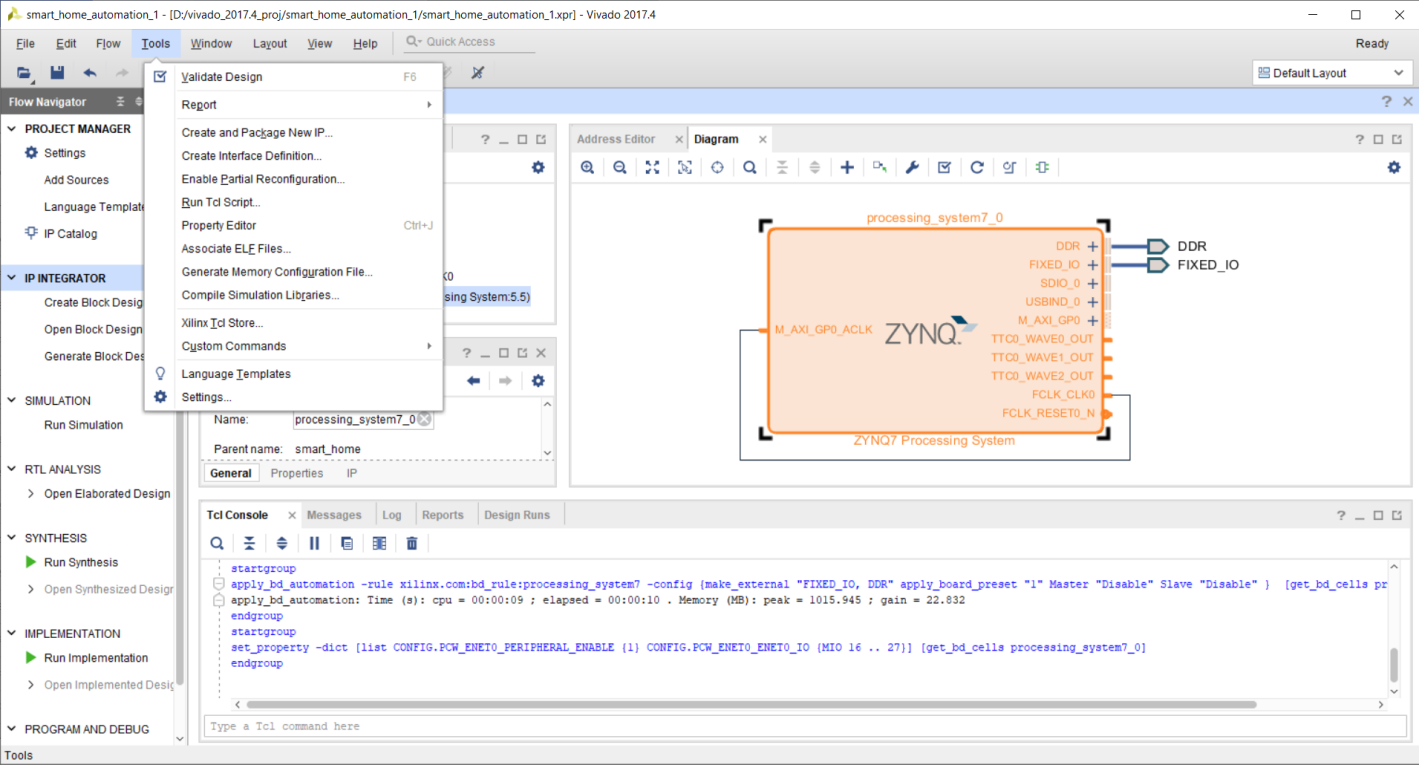


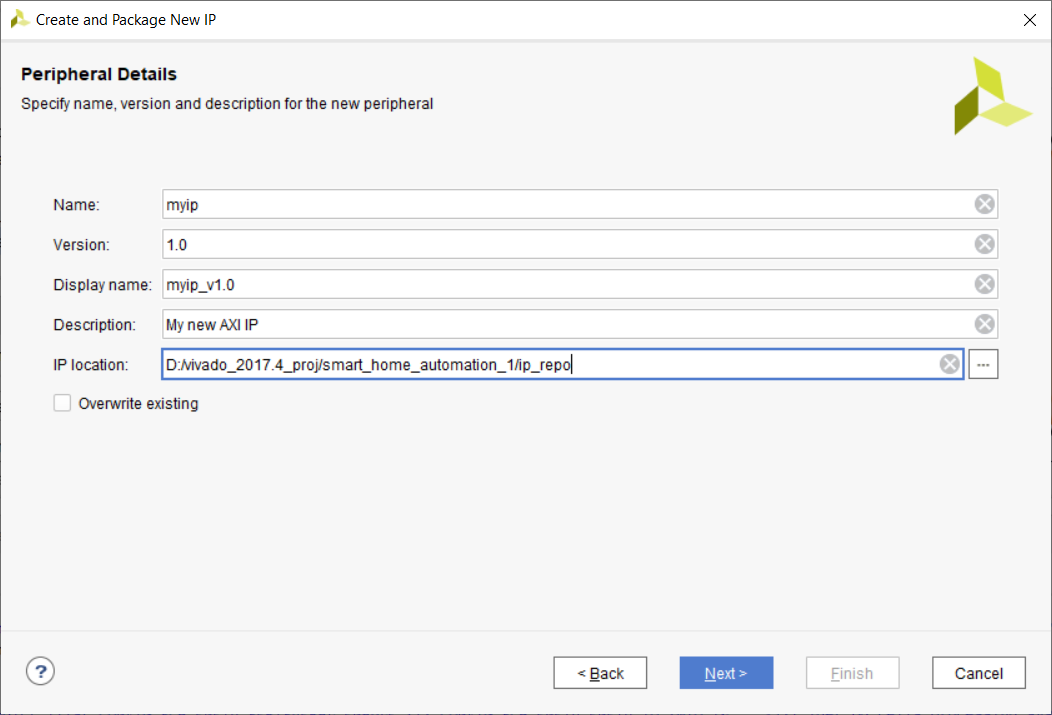


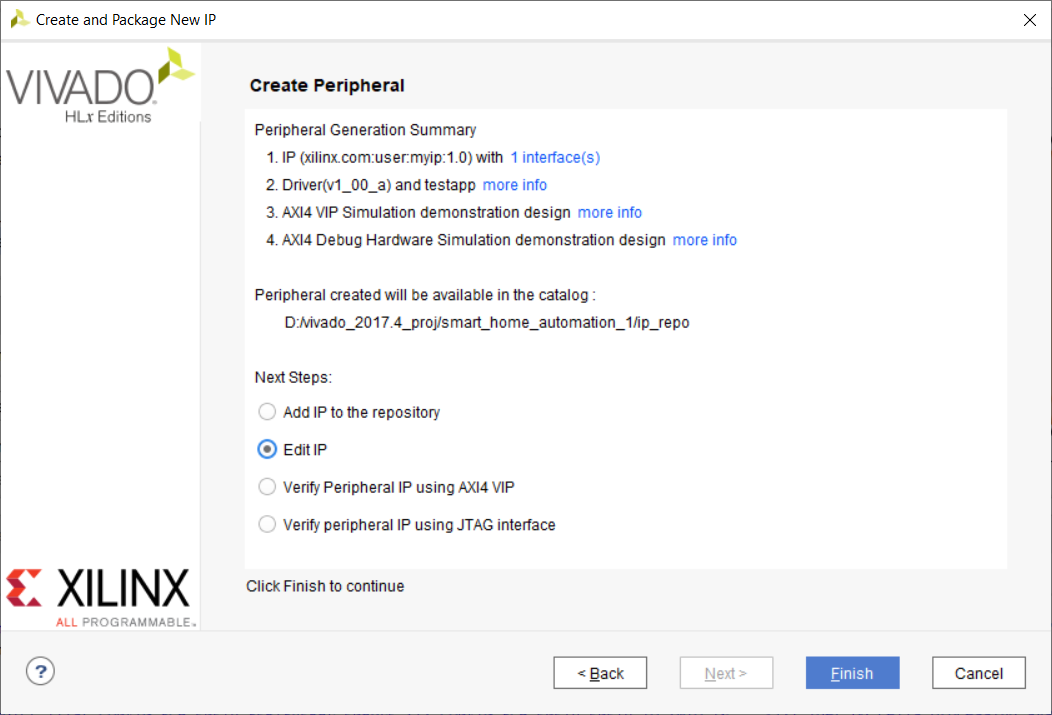
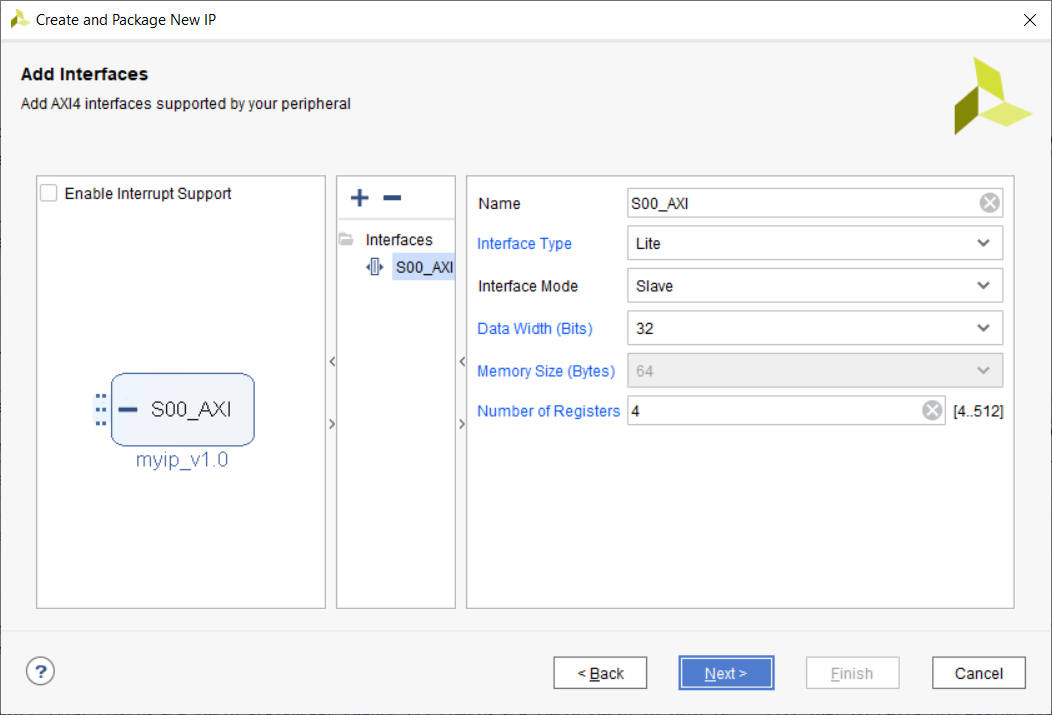


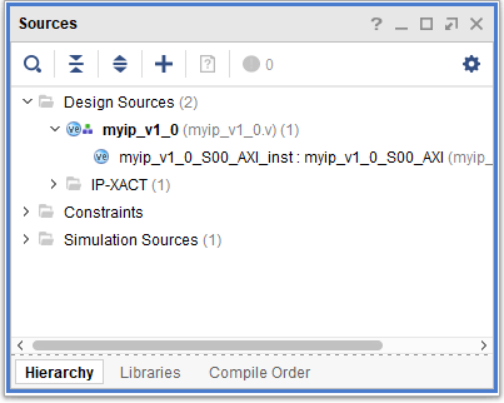
Step 3: Custom IP

Tools > Create and Package new IP > Select “Create a new AXI4 peripheral” > Enter name\* > No.of Registers:4 > Select “Edit IP”.







In sources window edit the myip\_v1\_0.v & myip\_v1\_0\_S00\_AXI.v

**myip\_v1\_0.v file:**

Line 18: (below “// Users to add ports here”)

output wire [3:0] LEDs,

input wire [3:0] JB\_Port,

input wire [3:0] Switch,

After pasting

Line 53: (below ”myip\_v1\_0\_S00\_AXI\_inst ( ”)

.LEDs(LEDs),

.JB\_Port(JB\_Port),

.Switch(Switch),

**myip\_v1\_0\_S00\_AXI.v file:**

Line 18: (below “// Users to add ports here”)

output wire [3:0] LEDs,

input wire [3:0] JB\_Port,

input wire [3:0] Switch,

After pasting

Line 403:

always @( posedge S\_AXI\_ACLK )

begin

if ( S\_AXI\_ARESETN == 1'b0 )

begin

slv\_reg1 <= 0;

slv\_reg2 <= 0;

end

else

begin

slv\_reg1[3:0] <= JB\_Port;

slv\_reg2[3:0] <= Switch;

end

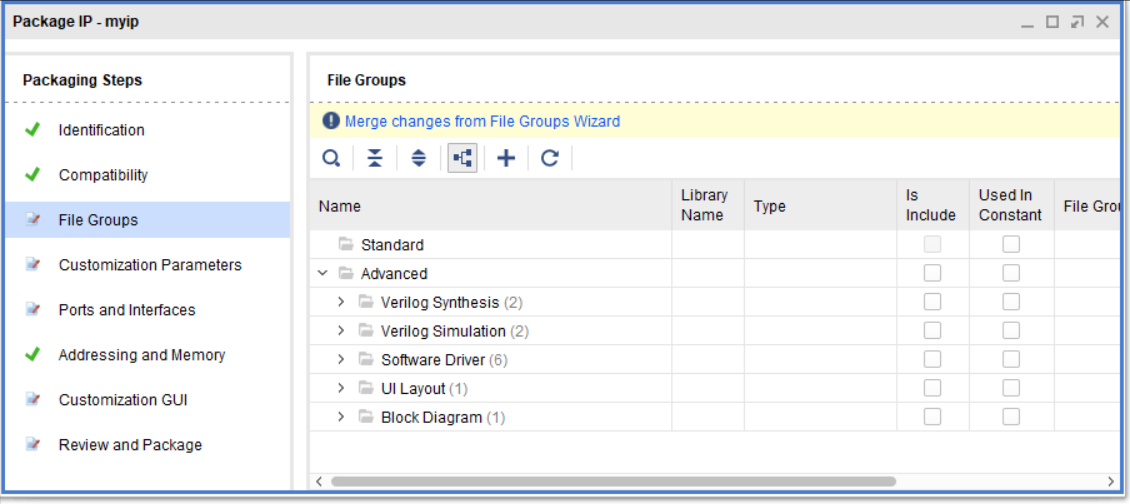
end

assign LEDs = slv\_reg0[3:0];

After Pasting

**DISABLE**(type // at the starting)

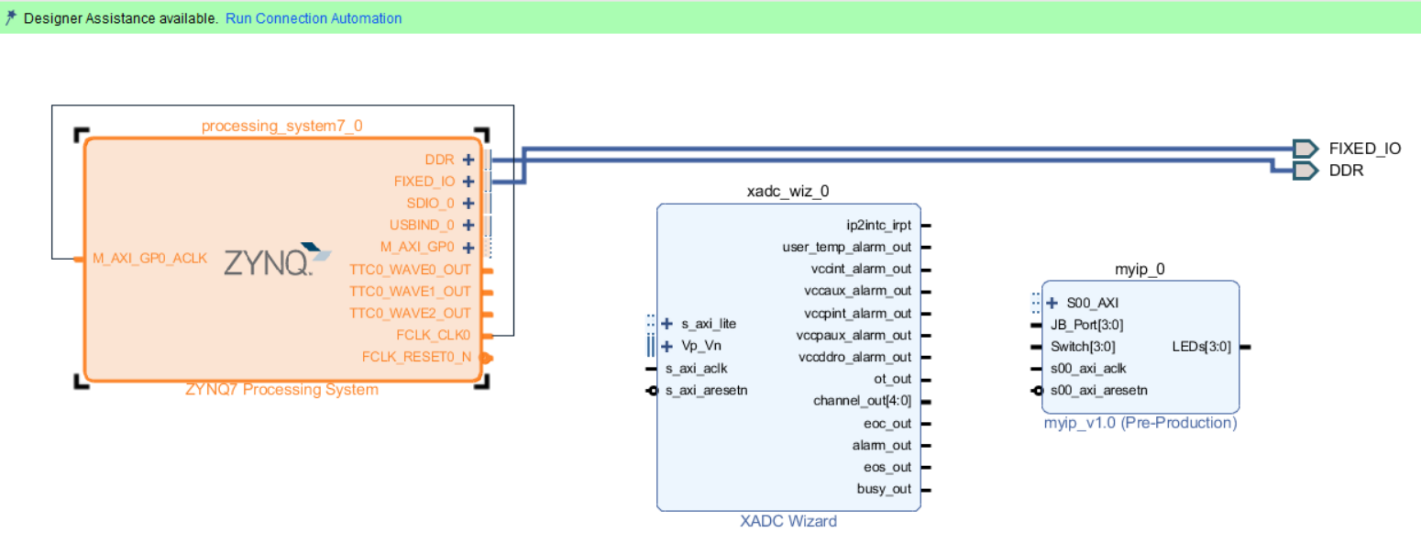
Lines 226, 227, 246, 253, 264, 265.

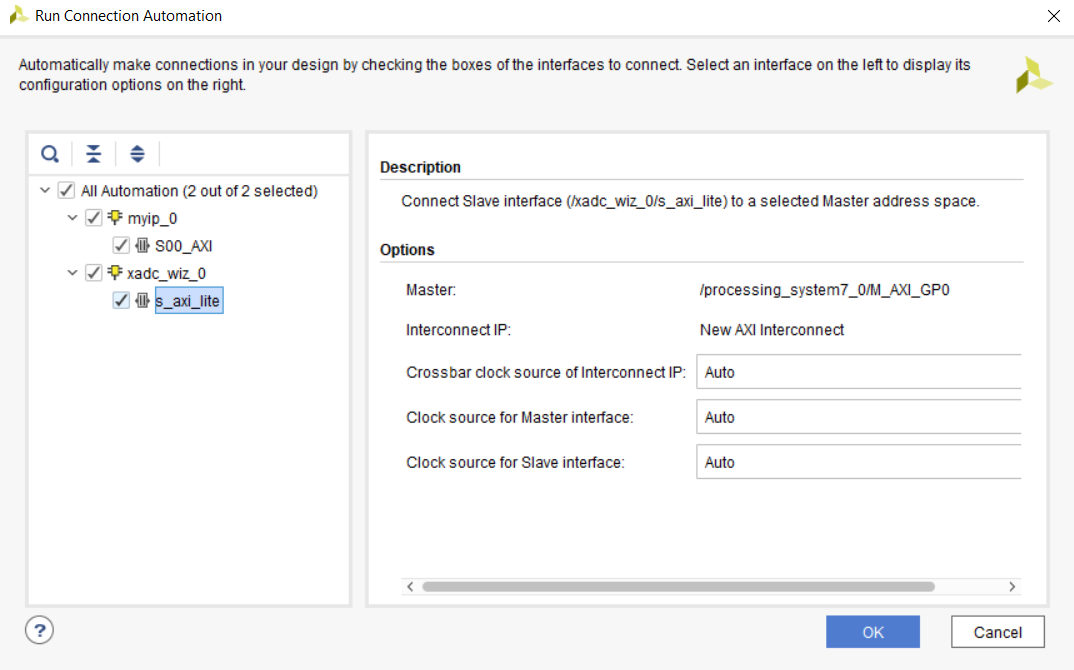
Come back to Package IP > click on the non ticked packaging steps and then click on “merge changes…” at the top

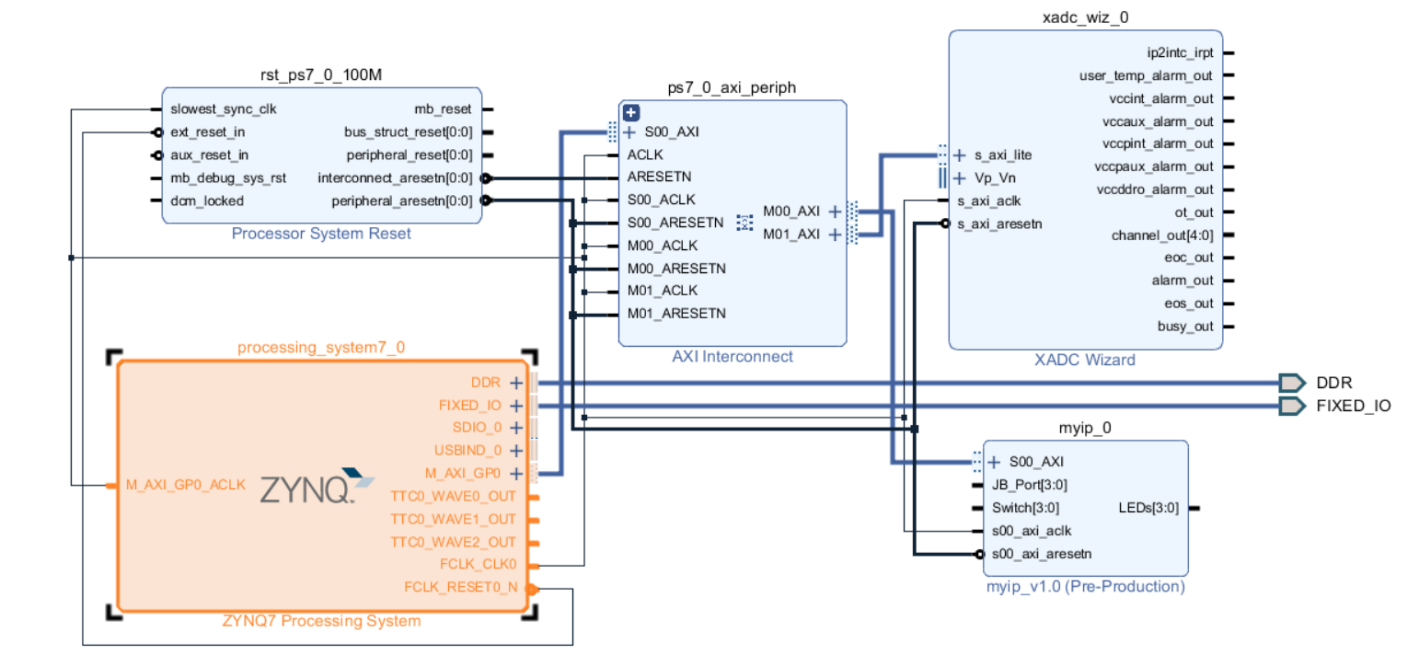
Under Review and Package > Click “IP has been modified” > Click Re-Package IP > Click yes

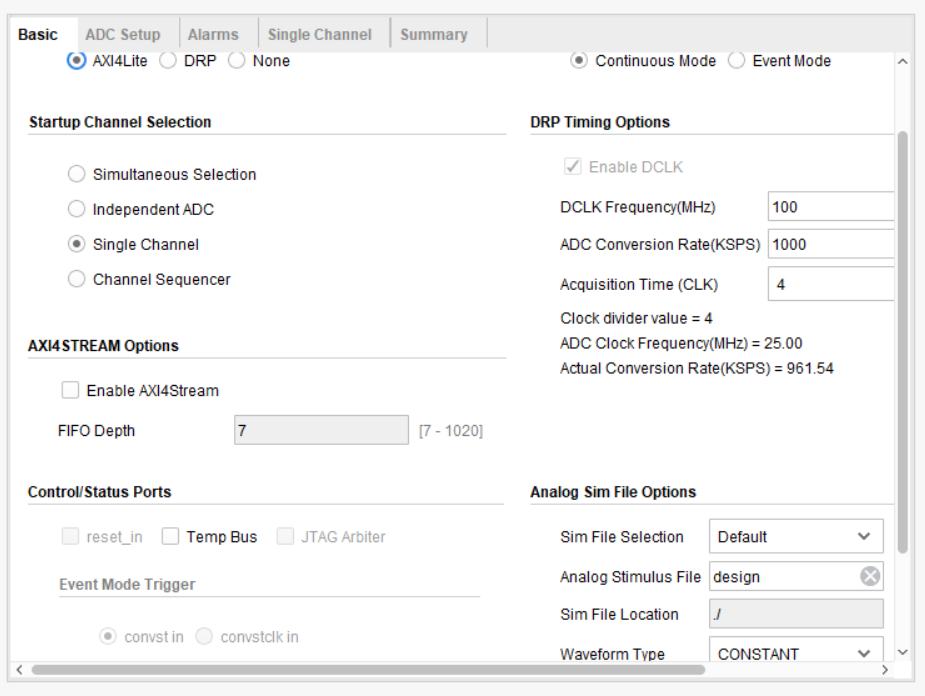
Step 4: Block Design (Contd.)

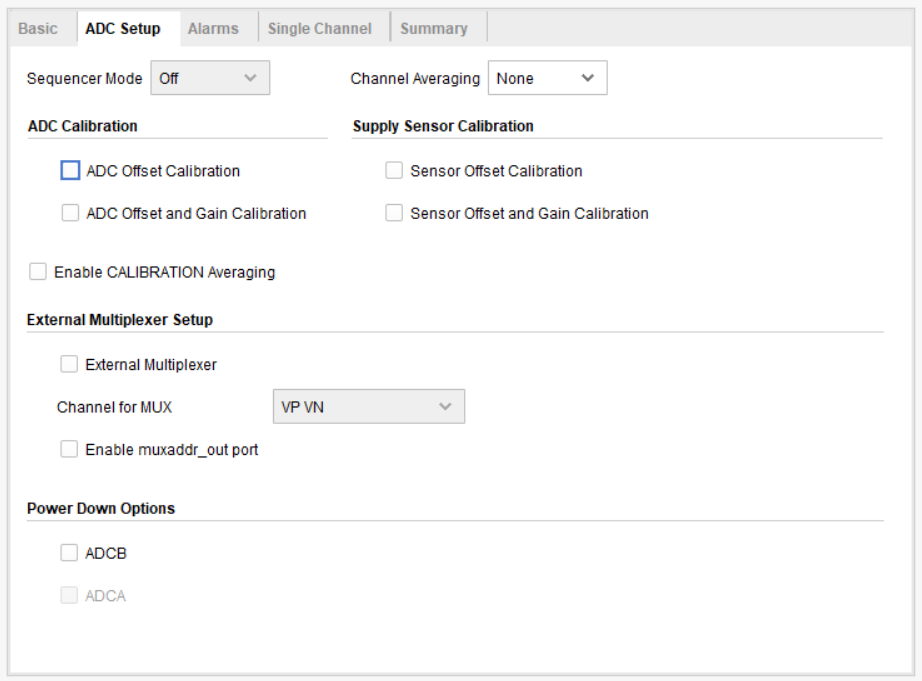
Add myip\_0 and XADC Wizard (Ctrl + I) > Click “Run Connection Automation” > Enable S00\_AXI & s\_axi\_lite > Ensure everything is set to Auto > Make External to the 3 ports in myip\_0 (Select the ports and press Ctrl + T) > Double Click xadc\_wiz\_0 > Check images for configuration\* > Click Ok > Bring the cursor near Vaux7 > Once it changes to down arrow, click it > Make External Connection for Vauxp7 and Vauxn7 > Save the design (Ctrl + S)

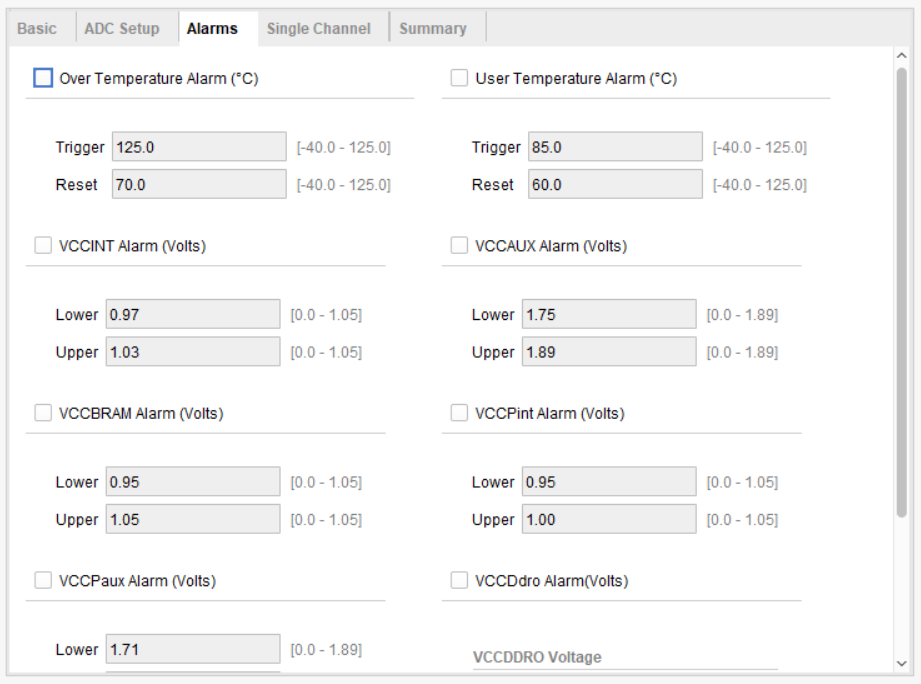


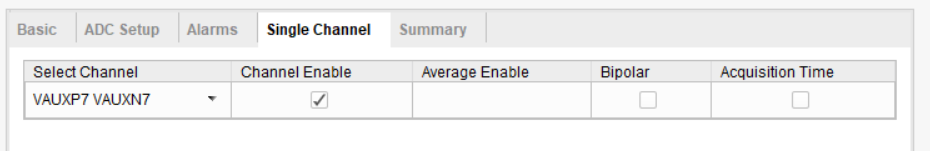


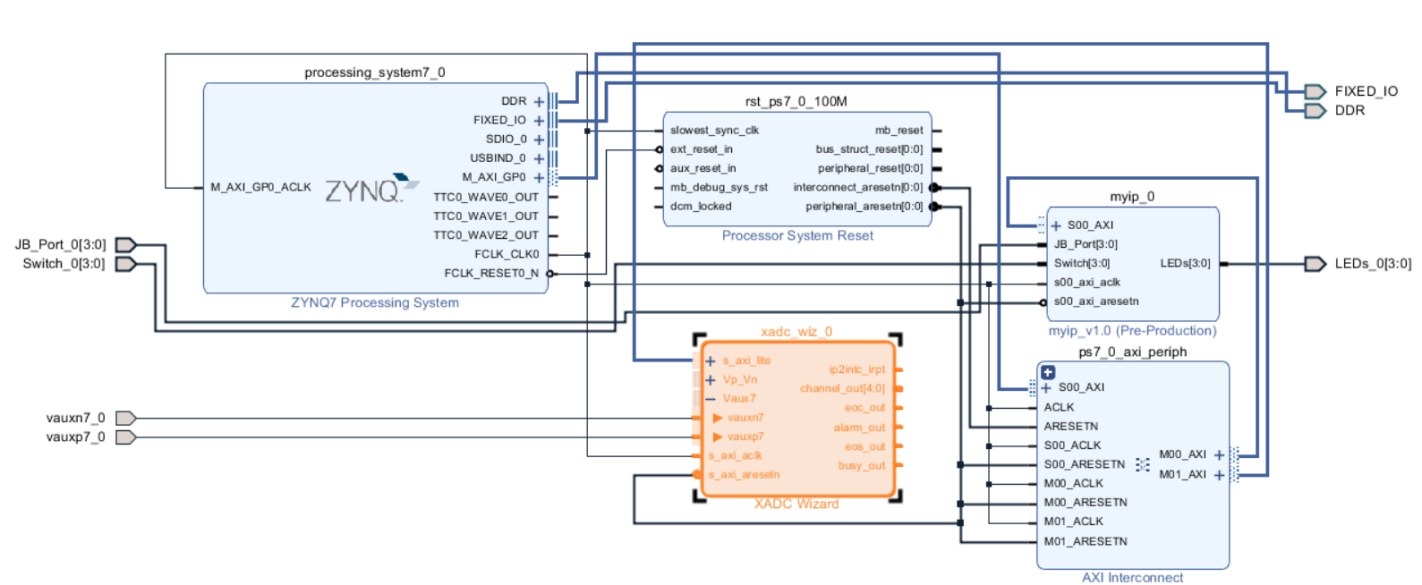












Step 5: Generate Bitstream

Under Source tab >Click the + Symbol > Select Add or Create Constraints > Click Create File > Copy the Constrint file from folder and save > Under Design Sources > Right Click smart\_home.bd > Click Create HDL Wrapper > Ignore the message > Under flow navigator - PROGRAM AND DEBUG - Click Generate Bitstream

Once bitstream is successfully generated, Click on File > Export > Export hardware > Check Include bitstream > Click OK

The hardware file is saved in “.hdf” format in smart\_home\_automation\_1.sdk folder (subject to change if u go for a different project name) under the project folder. Copy this file and paste it in the shared folder of Ubuntu.

Step 6: Generating Linux Kernels

OS Platform: Ubuntu 16.04

Open Terminal and type the following commands

cd ~/opt/pkg/petalinux/source settings.sh

mkdir project

cd project

mkdir smart\_home\_automation\_1

(you can make the file directory according to your convenience. When you are working on multiple designs, its better to name them accordingly)

cd smart\_home\_automation\_1

petalinux-create --type project --template zynq --name PjIot

(the --name can be changed according to your convenience. I named it as PjIot)

Copy the .hdf file from the shared folder and paste it into project/smart\_home\_automation\_1/PjIot

Type the following commands in terminal

petalinux-config --get-hw-description

(Shows a new menu, but you need not change and cofiguration for this project)

In files go to project/smart\_home\_automation\_1/PjIot/project-spec/meta-user/recipes-bsp/device-tree/file

In this location you’ll find a system-user.dtsi file. Either replace this file with the files files provided or type it out manually\*. The file sets the custom IP as a UIO device and sets the Vaux7 pin.

\*changing indentation can give you an error.

petalinux-config -c rootfs (Can take a long time to run)

(opens a new window)

Filesystem Packages → devel → python → python (Select all the files in it using ‘Y’)

Filesystem Packages → misc → gcc-runtime-> libstdc ++

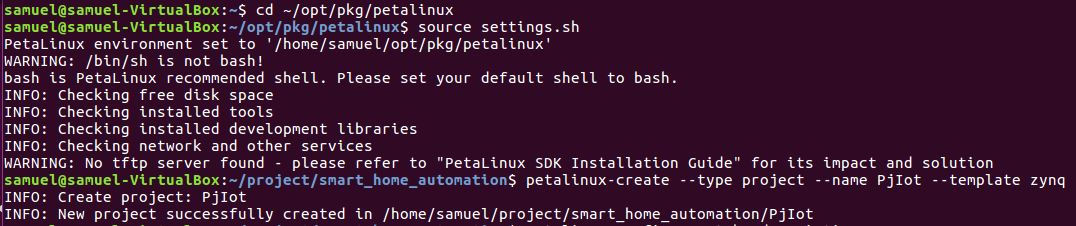
(save on exit)

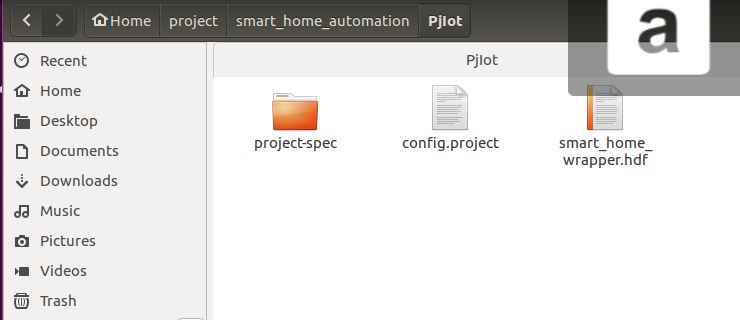
petalinux-build (Can take a long time to run)

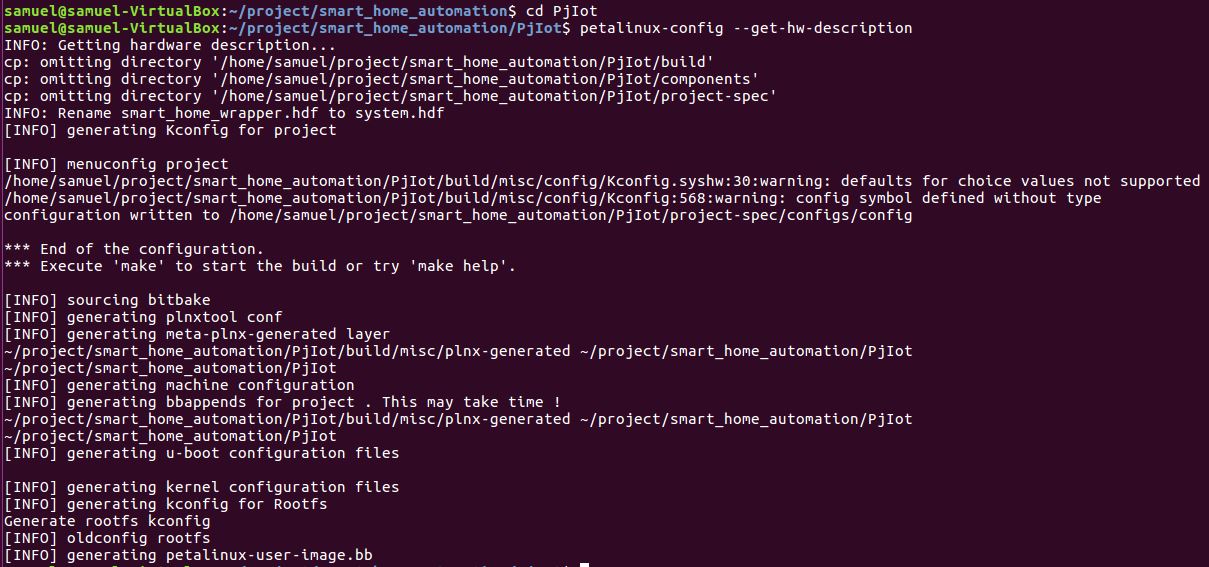
petalinux-package --boot --force --fsbl images/linux/zynq\_fsbl.elf --fpga images/linux/smart\_home\_wrapper.bit --u-boot

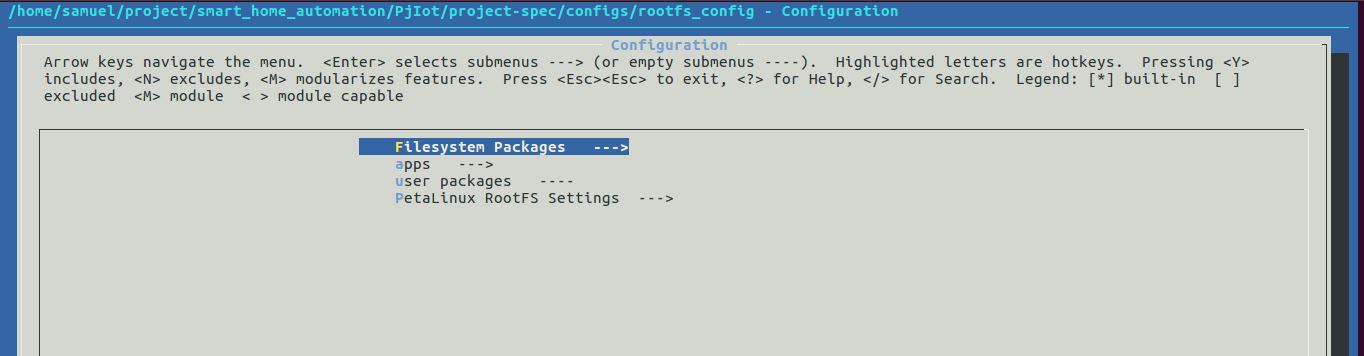
Go to the project/smart\_home\_automation\_1/PjIot/images/linux

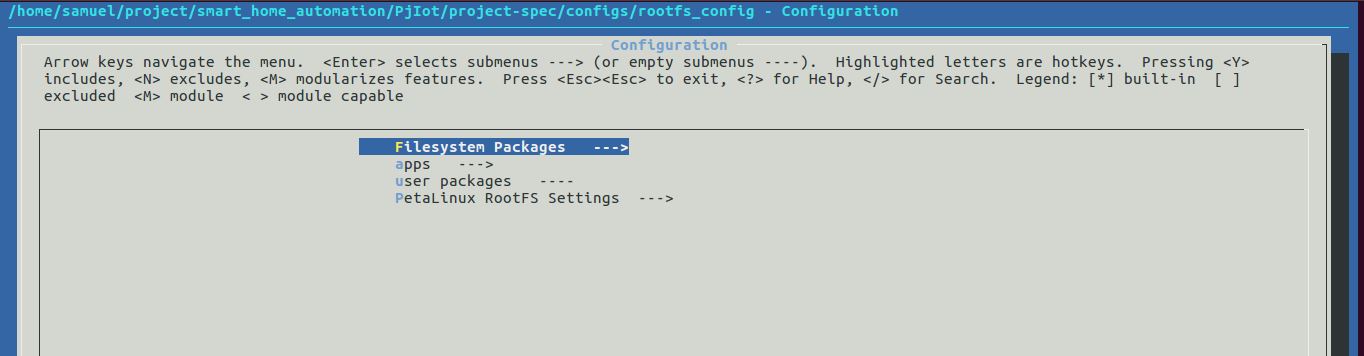
Copy and paste BOOT.BIN and image.ub file into the shared folder



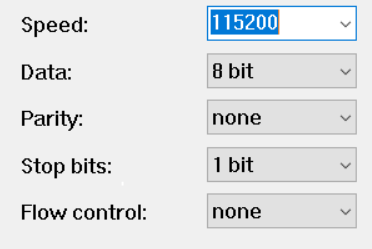


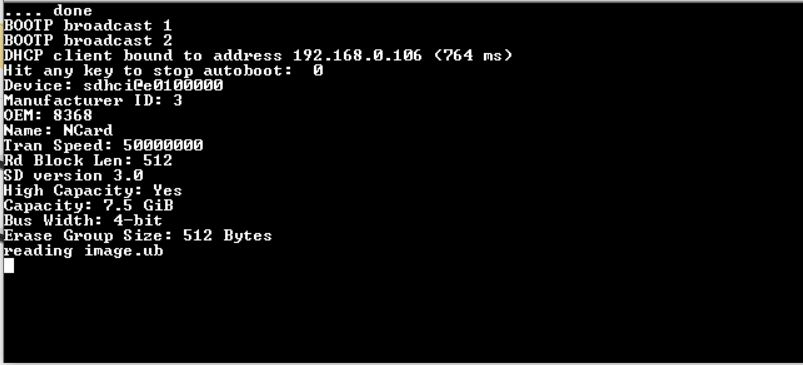


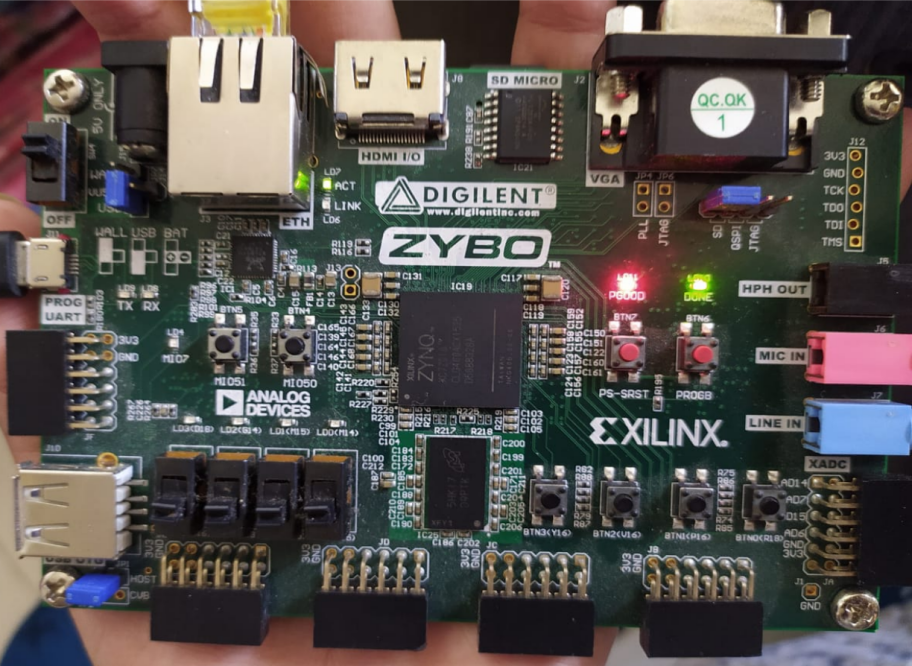




STEP 7: SD Card and booting using Serial Port monitor’

Copy the BOOT.BIN and image.ub file into a formatted SD Card. Note that the SD Card is FAT32, which is mostly the format of any SD Card from 4GB memory or above. Once it is copied, connect the ethernet cable into the Ethernet port of Zybo Board and turn the board ON (Do not insert SD Card now). Make sure that the Modem/Router is DHCP enabled. Also, ensure that the jumper near VGA port is set to SD mode . Open Serial Port Monitor in PC (I used tera term). You can use any kind of serial port monitor like tera term, PuTty etc. On opening Tera Term, it should show a window with TCF Agent and Port option. If your zybo board is connected, it will show the respective COM port and make sure to enable the Serial Option. You must be able to find the COM port of Zybo in Device manager of windows. Next, click on Setup -> Serial Port from the task bar. A new window appears. Ensure that the following parameters are set accordingly. Once it is done, turn OFF the board and then insert SD Card and now turn the board ON again. You should see the Board booting successfully.





SD Mode

Booting

Step 8: Xilinx SDK



Once the SD card has successfully booted, enter login and password. In this case both of them are root. Once you have come inside the root folder, create two fifo files with 666 mode as it helps in communicating between python and CPP.

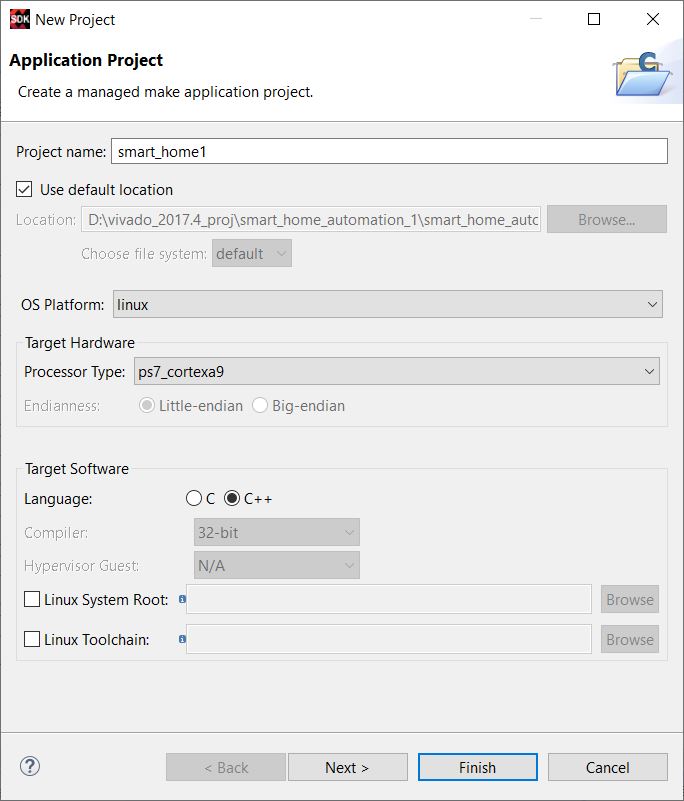
For this, type the following commands in the serial monitor

Once these commands are typed they pave the way for making a connection between Python and CPP. Also type “ifconfig” to get the IP address of the zybo board as it will be necessary in later stages.

Next Open the project in xilinx > click on File > Launch SDK. This procedure will direct you to the workspace in Xilinx SDK. If you wish to do this directly by opening Xilinx SDK, mention the workspace and the hdf file and you’ll be good to go.

Once you’re inside the work space, click on file > new > Application project (Alt+Shift+N)

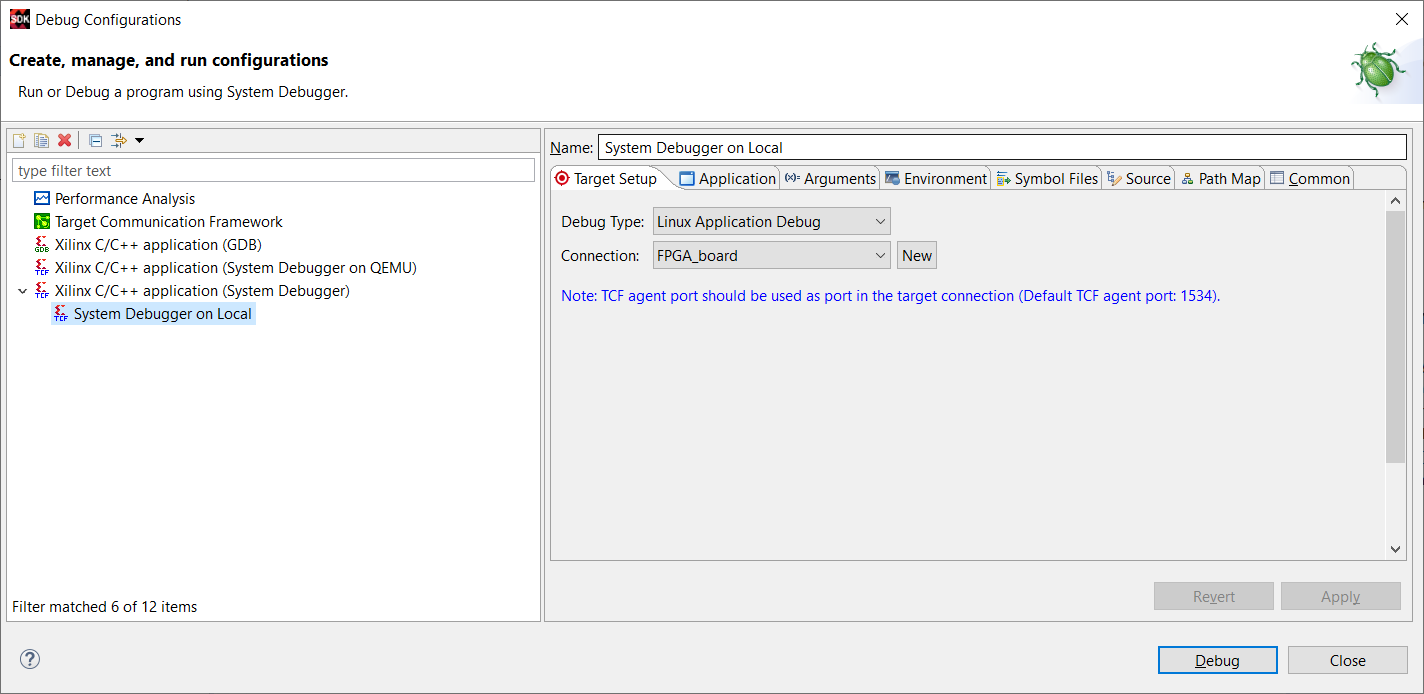
Give a project name of your choice (for this project ‘smart\_home’)> OS platform select linux > Language : C++ > Click Next > Choose Empty Application and click on Finish

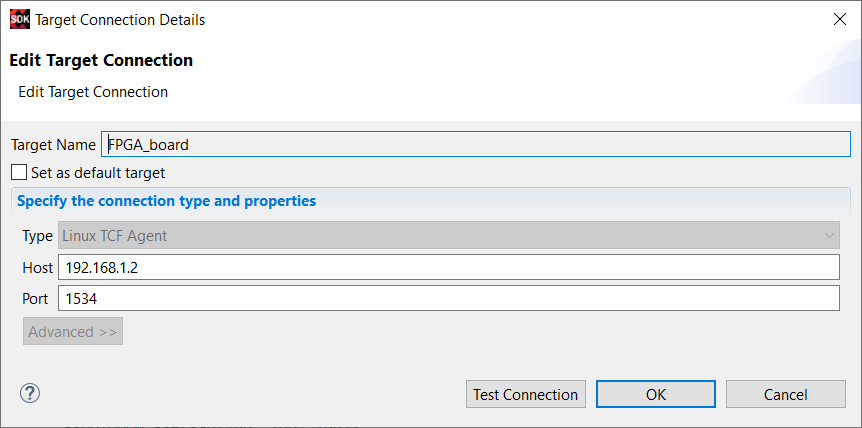


Click the dropdown next to ‘smart\_home’ > src > replace this folder by src folder attached with this documentation

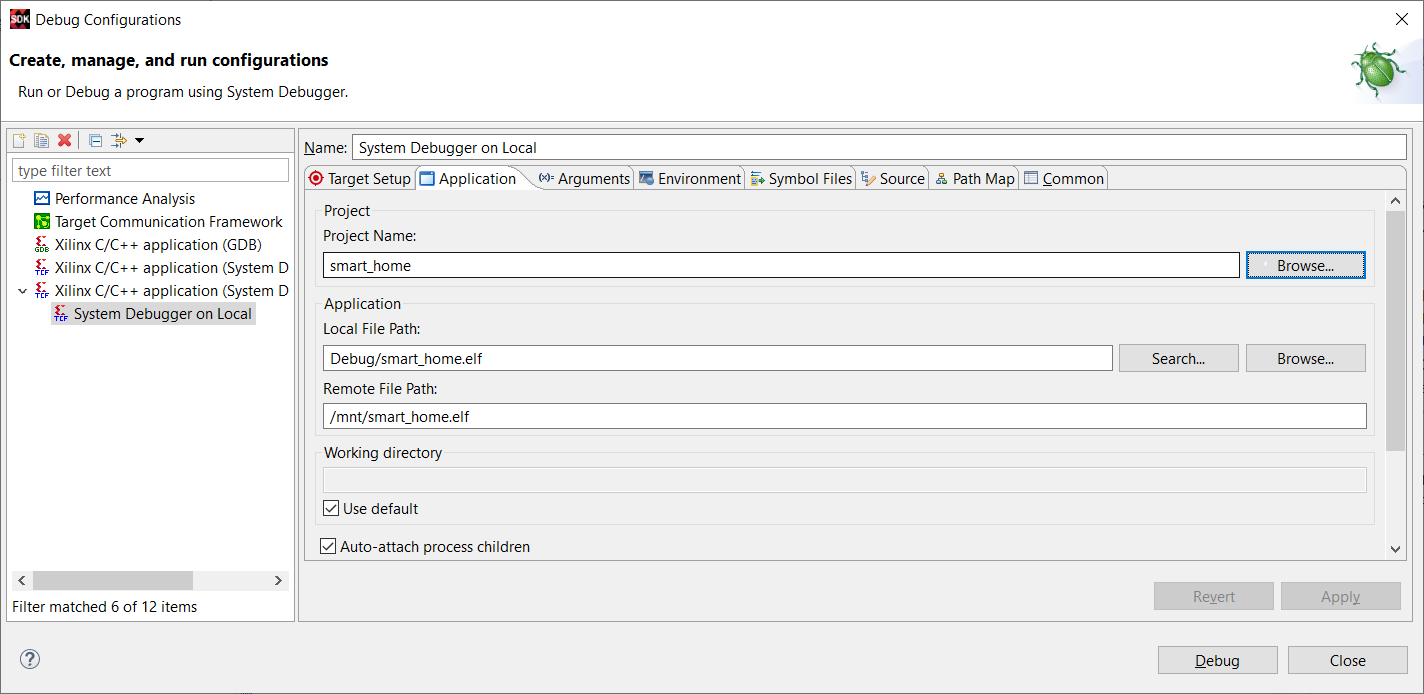
Click on Project in the toolbar > Clean > Clean all projects > Click OK (This builds the project. Most probably it wouldnt give any error)

Click on Run in the Toolbar > debug configuration > Xilinx C/C++ application (System Debugger) > Under target setup tab > Debug Type: Linux Application Debug > Click on New next to connection (may differ from screenshot) > Enter a Target name > Host: The IP address of your board > Click test connection > It should give a success message (Make sure your laptop or PC is connected to the same network as that of your Zybo board. I used an old wireless router so that it can be accessed wirelessly. As the modem is DHCP enabled, the IP address is subject to change most of the time)





Next, under Application tab next to target setup > Click on browse under project name > Select smart\_home(your project name) > click apply > click debug



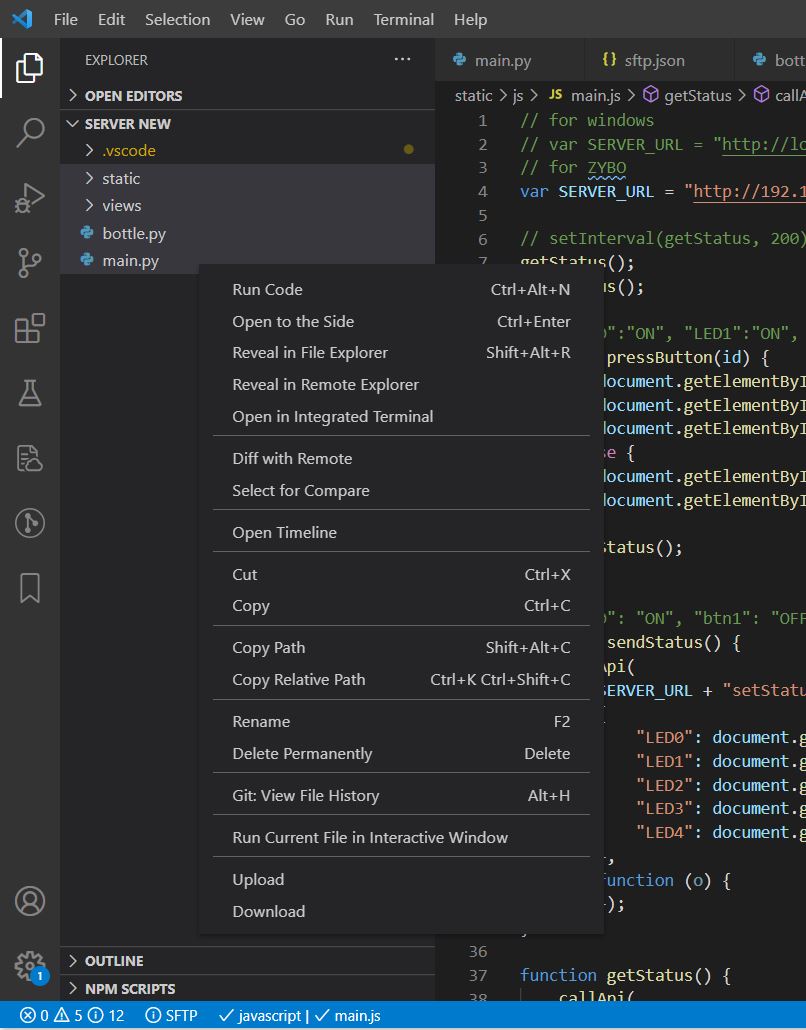
This will take you to the debug window. The debugger will start automatically. If it doesnt start, right click on the system debugger and select relaunch.

Step 9 : Python application development using Visual Studio

Once the debug starts running, open VS Code > File > Open folder > Smart home server (attached with the project documantation)

Open command pallete (Ctrl + Shift + P) > type SFTP: Config (enable SFTP in Extension menu) > This will open a sftp.json file in the project folder > Copy the contents from .vscode/sftp.json file or just replace it entirely (make sure you enter your IP address in python, sftp.json and views/main.js files)> Select all the static, views, main.py and bottle.py file > right-click and select upload

Check if the files are uploaded to your board by typing ‘ls’ in the serial monitor



Now in the serial port monitor > type python main.py (this will start the server)

Open browser > type 192.168.1.2:8080 (Enter your IP address and add 8080) > the web page should work