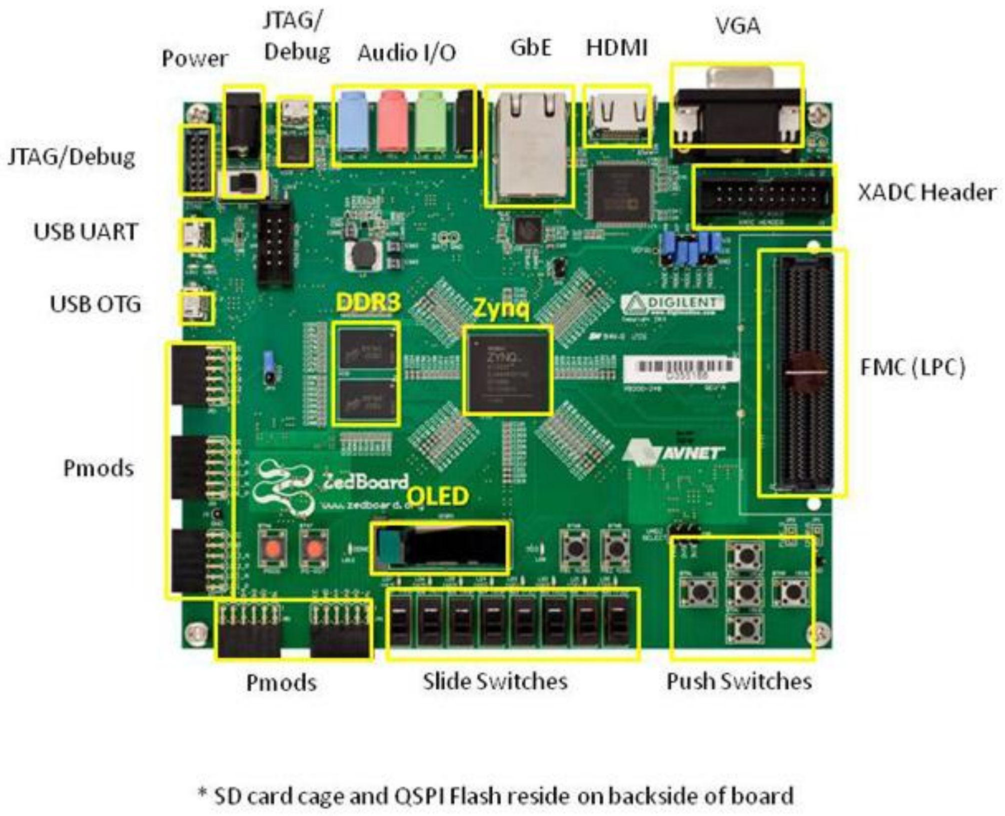
Wireless Interface for FPGA BRAM System User Manual

1. Requirements:

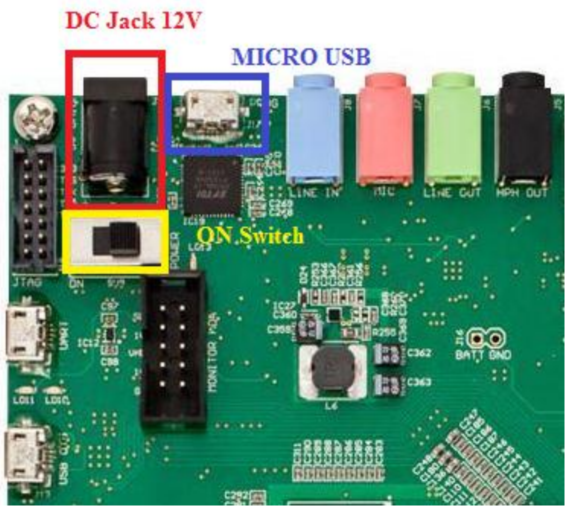
1. A Computer with Arduino IDE and Xilinx Vivado installed.
2. A Computer running Linux
3. Digilent™ ZedBoard
4. NodeMCU
5. NodeMCU PMOD interface PCB
6. 2 MicroUSB cables
7. Router
8. Stable Internet connectivity

2. Hardware Connections

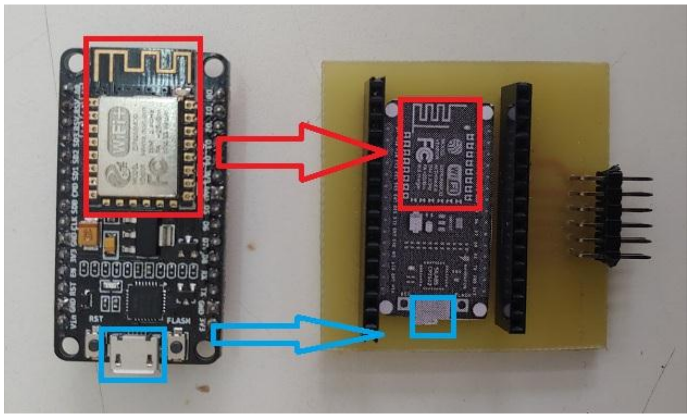
1. Place the Zedboard in the orientation as shown in the figure below.



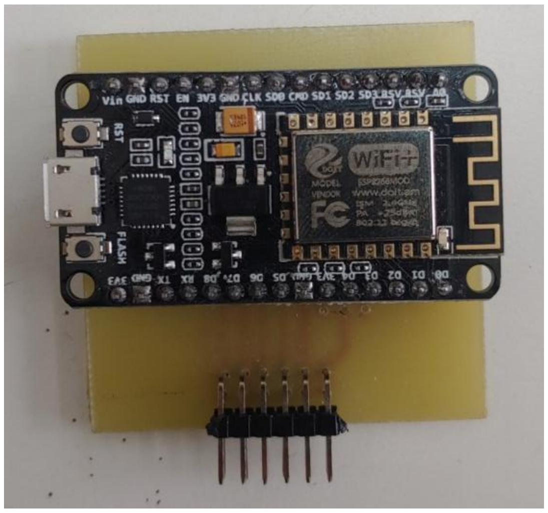
2. At the TOP LEFT of the board, connect the DC connector of 12V adapter into the DC Power jack. Plug the adapter into AC mains and turn it ON
3. Connect a microUSB cable to the PROG usb port as indicated by blue box in the figure below. Connect the USB cable to your PC.
4. Move the ON switch to the LEFT position to Turn ON the ZedBoard. This will be indicated by the Green Power LED turning ON



5. Now connect the NodeMCU to the NodeMCU interface PCB as shown in the figure below. Make sure to connect the board in the right orientation as shown on the picture on the PCB.

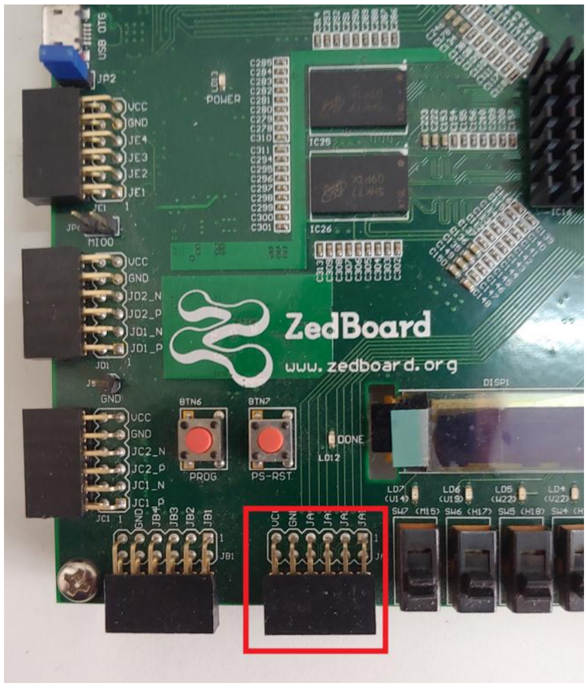


6. Your connected unit should look like this:

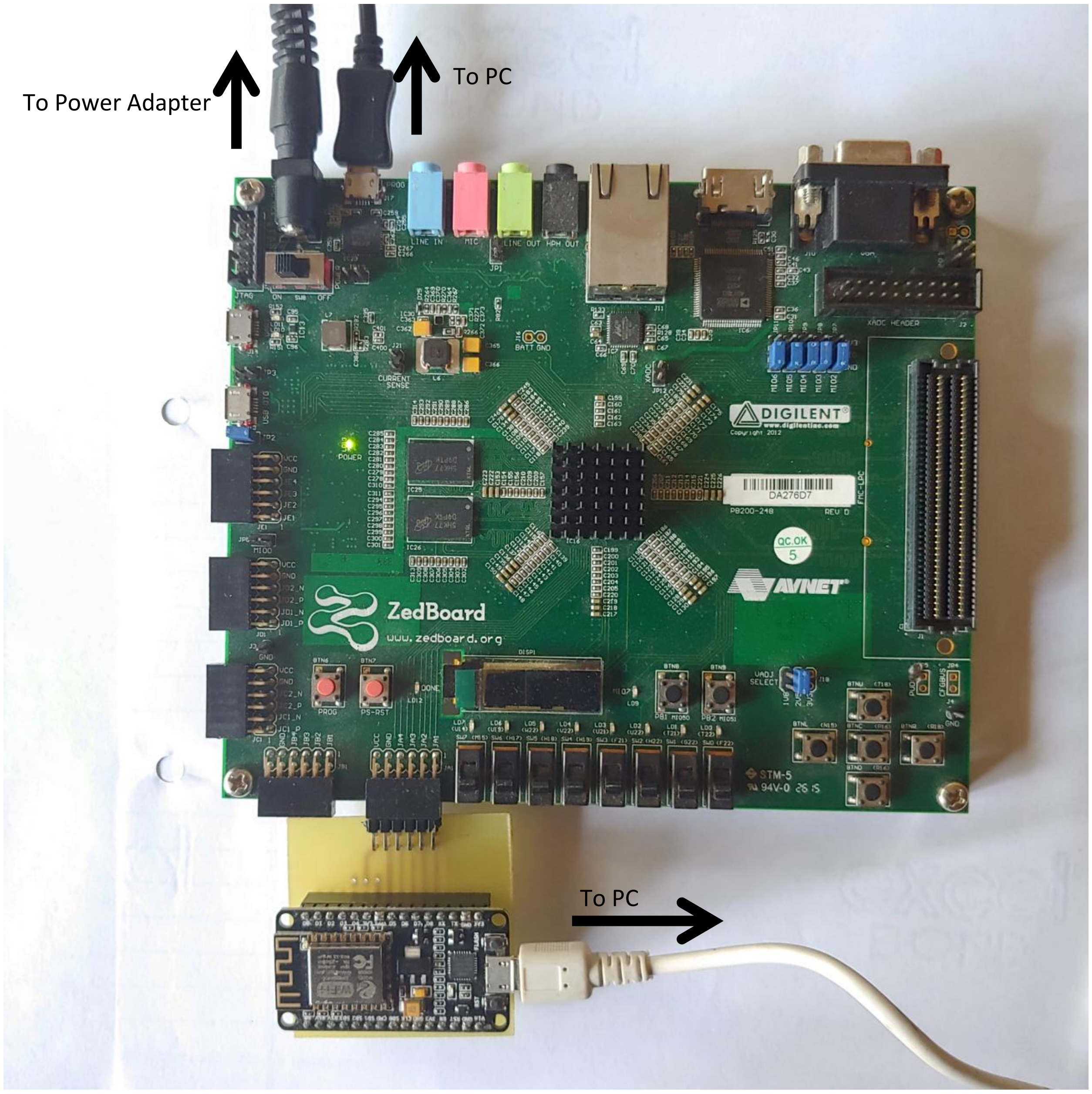


7. Connect a microUSB cable to the port on the NodeMCU. Connect the other end of the cabe to your PC.

8. Connect the NodeMCU PCB to the JA1 PMOD connector as shown in the figure



9. Your Completed connections should look like this



3. Downloading the required files

Download the required files on both Windows and Linux PC

A. Windows

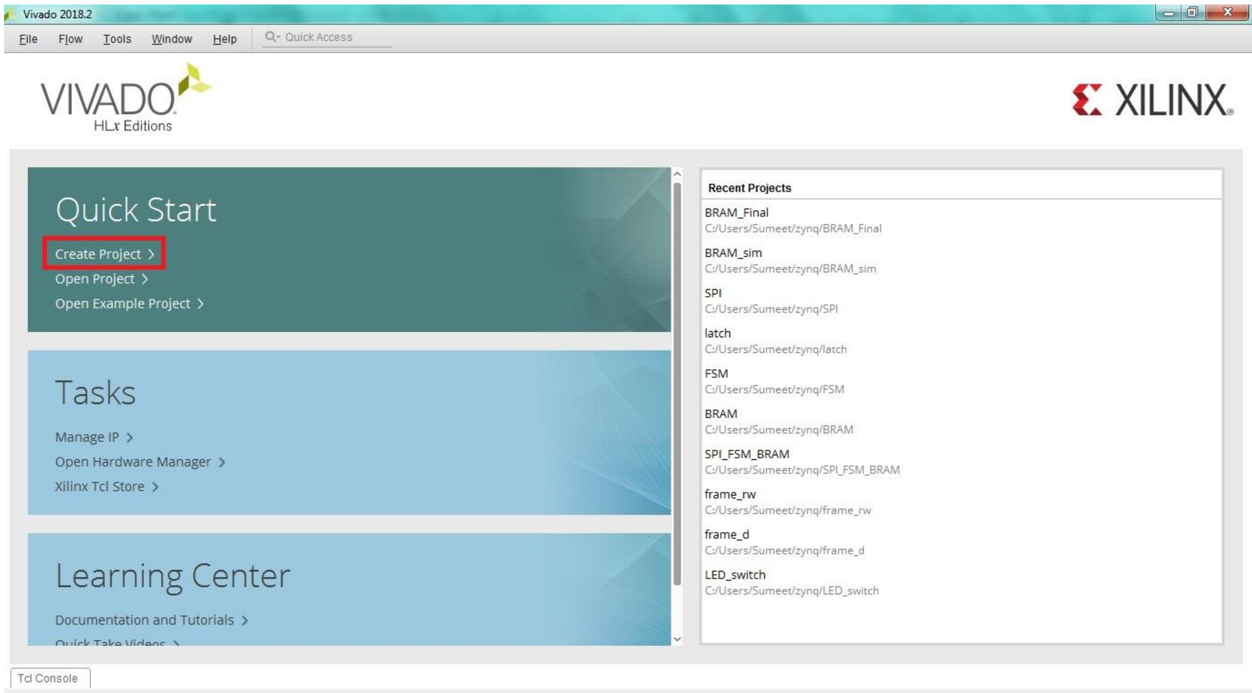
1. You will be using Vivado and Arduino IDE on your Windows PC and Linux Client on Linux PC.
2. On your windows PC, go to https://github.com/Sumeetk96/FPGA_interface
3. Click the green colored “download or clone” button on the top right of the window
4. Click Download ZIP. All the code files will be downloaded as a zip file in your Downloads folder.
5. Extract the zip file to get the main folder FPGA_interface

B. Linux

1. Make sure that your PC is connected to the internet.
2. Open Terminal.
3. Change directory to Downloads by entering command
   cd Downloads
4. Install git on your PC using the command
   sudo apt-get install git
5. Clone the repository using the command
   git clone https://github.com/Sumeetk96/FPGA_interface
6. Go to the folder containing the code using the command
   cd FPGA_interface/Linux_client

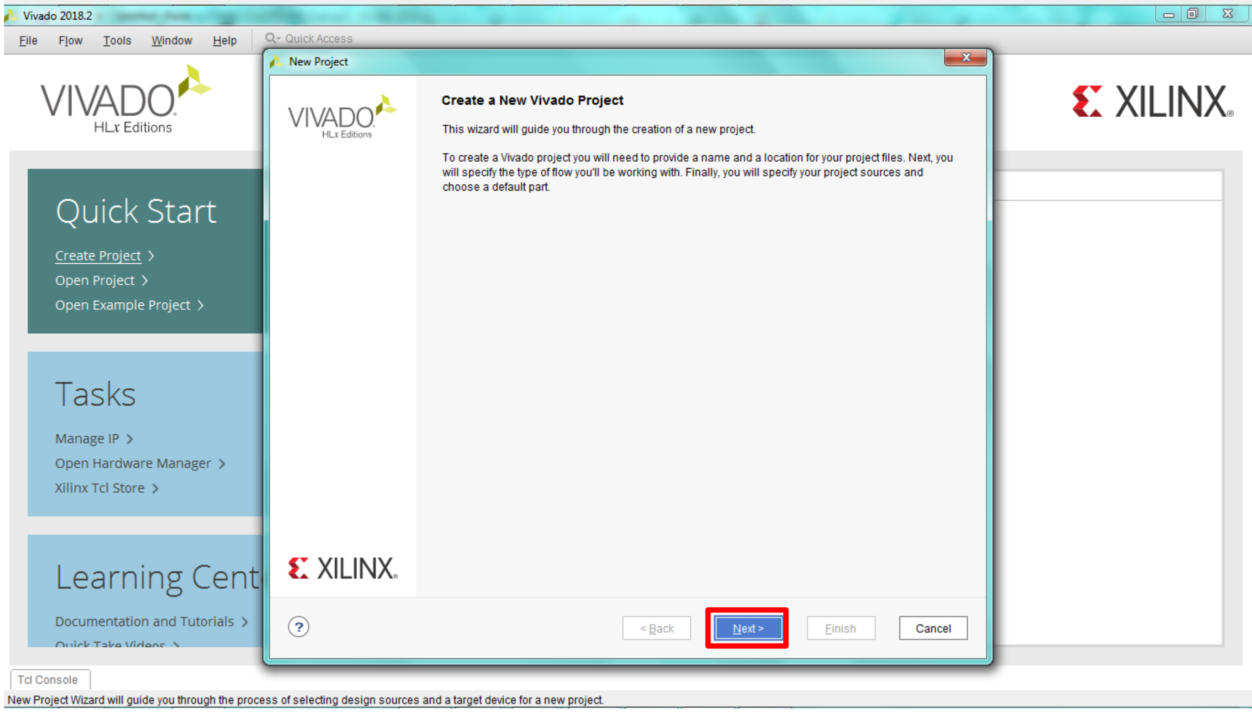
4. Programming the ZedBoard

1. Open Xilinx Vivado.
2. Create Project.



Recent Projects

BRAM_Final
C:/Users/Sumeet/zynq/BRAM_Final
BRAM_sim
C:/Users/Sumeet/zynq/BRAM_sim
SPI
C:/Users/Sumeet/zynq/SPI
latch
C:/Users/Sumeet/zynq/latch
FSM
C:/Users/Sumeet/zynq/FSM
BRAM
C:/Users/Sumeet/zynq/BRAM
SPI_FSM_BRAM
C:/Users/Sumeet/zynq/SPI_FSM_BRAM
frame_rw
C:/Users/Sumeet/zynq/frame_rw
frame_d
C:/Users/Sumeet/zynq/frame_d
LED_switch
C:/Users/Sumeet/zynq/LED_switch



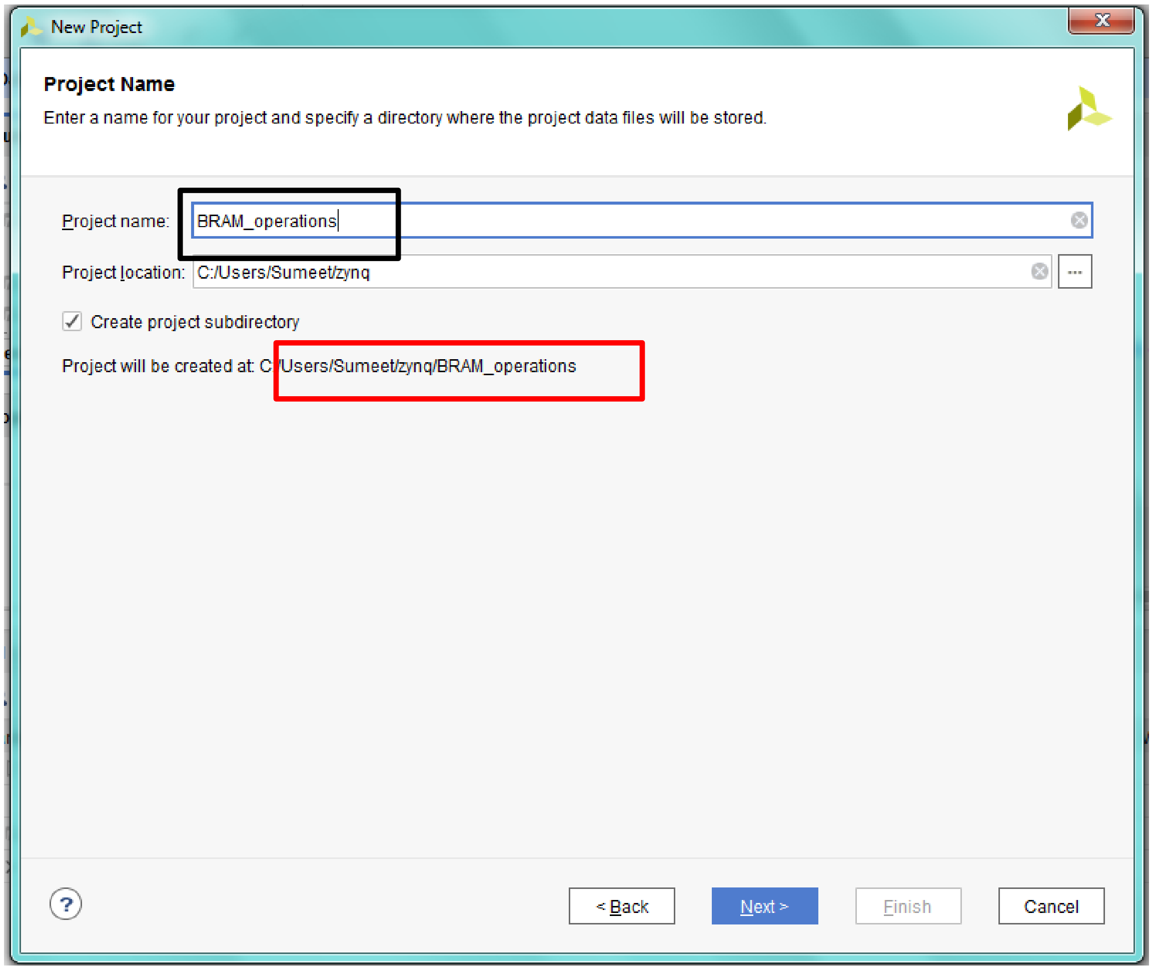
Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

3. In project name, enter BRAM_operations. Note down the location enclosed by the red box in the figure below. This is the location of your project. You will require this in the future.


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

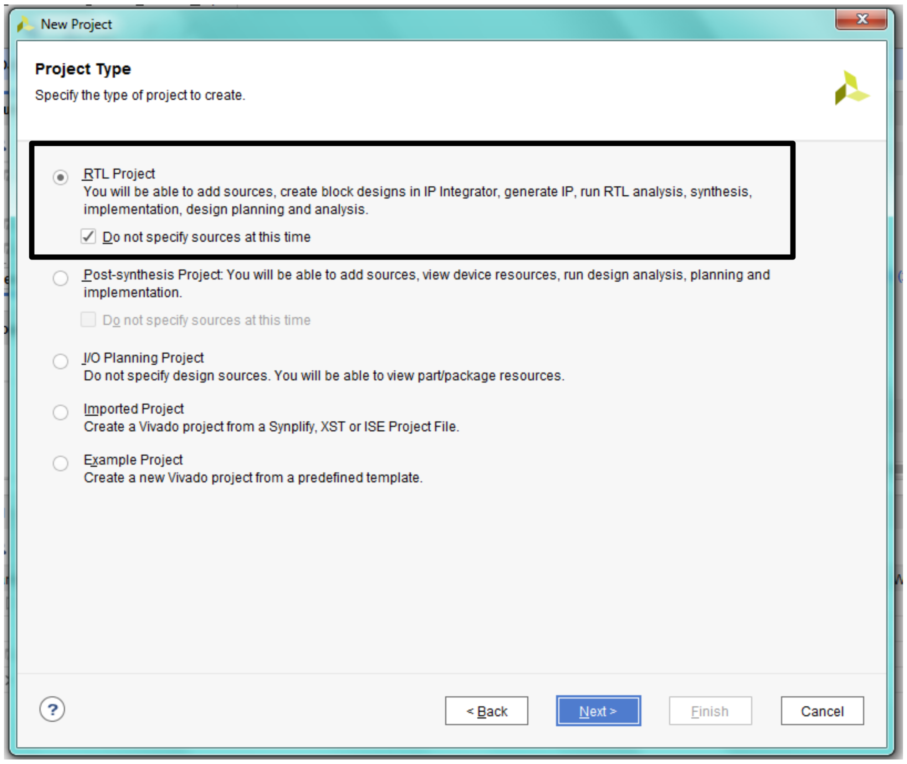
Project name: BRAM_operations

Project location: C:/Users/Sumeet/zynq

☑ Create project subdirectory

Project will be created at: C:/Users/Sumeet/zynq/BRAM_operations

4. Select RTL Project and check “Do not specify sources”. Click next



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

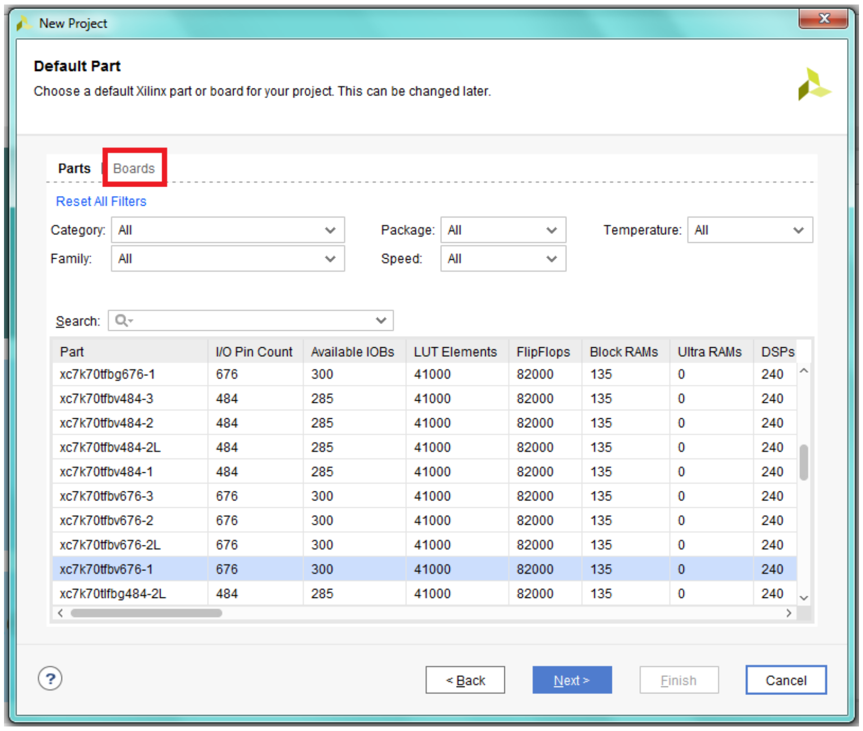
○ Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

○ 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.

5. First, click on Boards, as shown in the red box.



New Project

Default Part

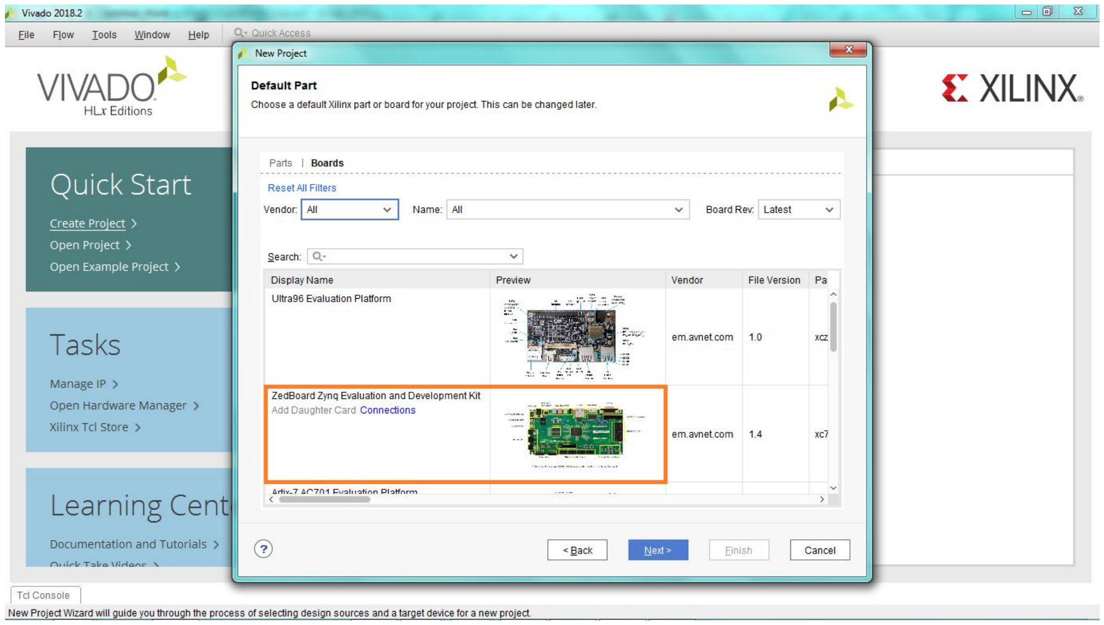
Choose a default Xilinx part or board for your project. This can be changed later.

Category: All
Package: All
Temperature: All
Family: All
Speed: All

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs
xc7k70tfbg676-1	676	300	41000	82000	135	0	240
xc7k70tfbv484-3	484	285	41000	82000	135	0	240
xc7k70tfbv484-2	484	285	41000	82000	135	0	240
xc7k70tfbv484-2L	484	285	41000	82000	135	0	240
xc7k70tfbv484-1	484	285	41000	82000	135	0	240
xc7k70tfbv676-3	676	300	41000	82000	135	0	240
xc7k70tfbv676-2	676	300	41000	82000	135	0	240
xc7k70tfbv676-2L	676	300	41000	82000	135	0	240
xc7k70tfbv676-1	676	300	41000	82000	135	0	240
xc7k70tlfbg484-2L	484	285	41000	82000	135	0	240

6. Click on Zedboard as shown in the orange box. Click Next.



Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

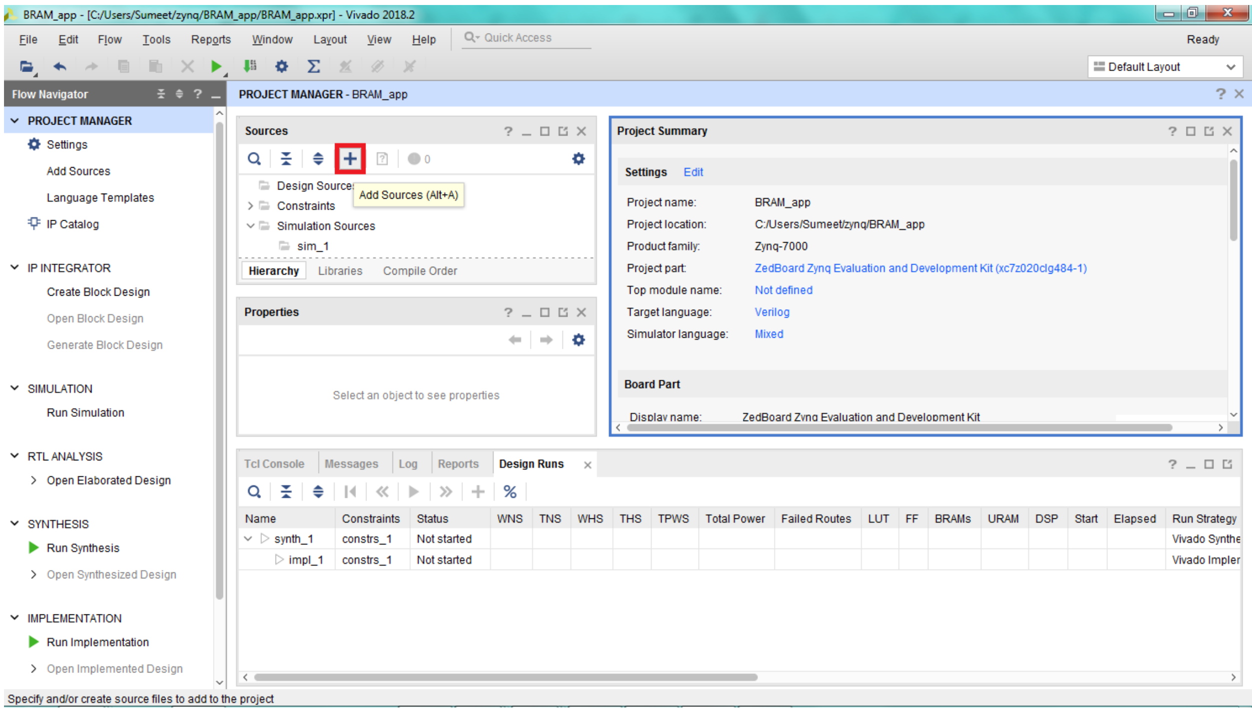
Vendor: All Name: All Board Rev: Latest

Display Name	Preview	Vendor	File Version	Pa
Ultra96 Evaluation Platform		em.avnet.com	1.0	xcz
ZedBoard Zynq Evaluation and Development Kit Add Daughter Card Connections		em.avnet.com	1.4	xc7
Artix-7 AC701 Evaluation Platform				

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

7. Click Finish. Your project will be created.
8. In the downloaded folder, go to FPGA_interface->Zedboard->Design_files
9. Copy all the files from the folder.
10. Now navigate to the project folder location you noted down here
11. Paste all the files in the folder: BRAM_operations.
12. Go to Vivado window.

13. Click on the small '+' sign



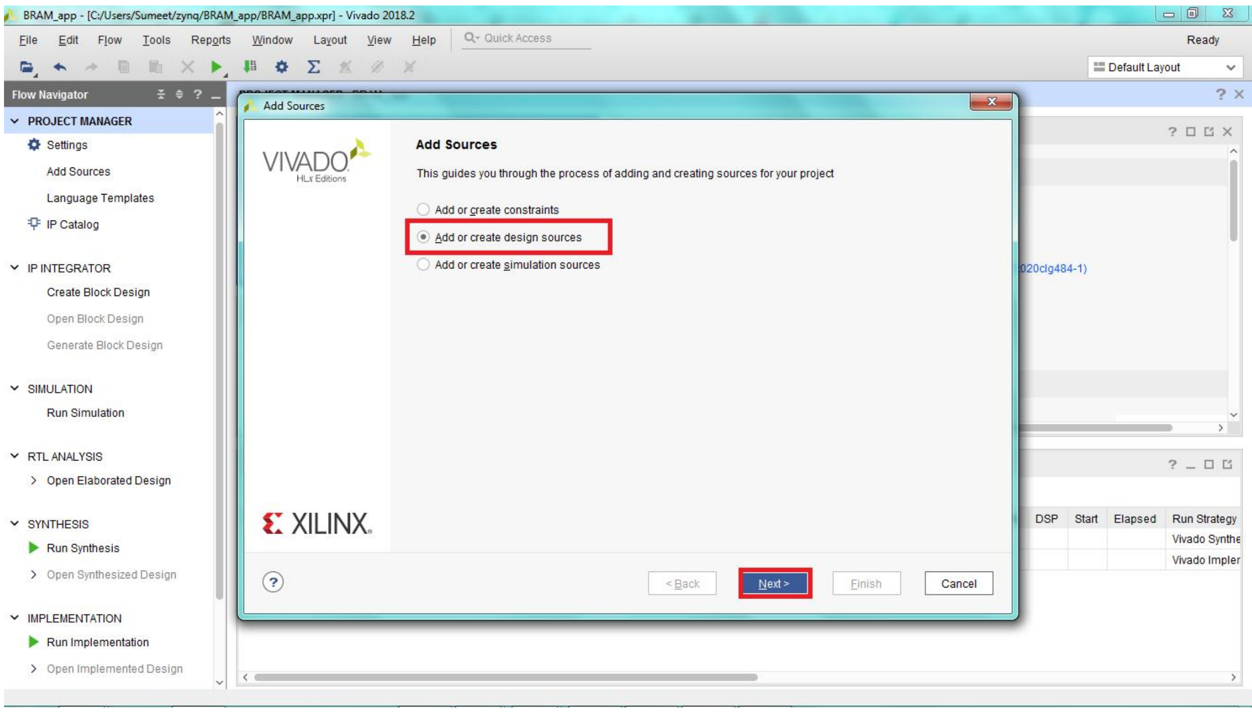
Select an object to see properties

Project name:	BRAM_app
Project location:	C:/Users/Sumeet/zynq/BRAM_app
Product family:	Zynq-7000
Project part:	ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed
Display name:	ZedBoard Zynq Evaluation and Development Kit

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthe
impl_1	constrs_1	Not started															Vivado Impler

Specify and/or create source files to add to the project

14. Click on "Add or create design sources" and click next

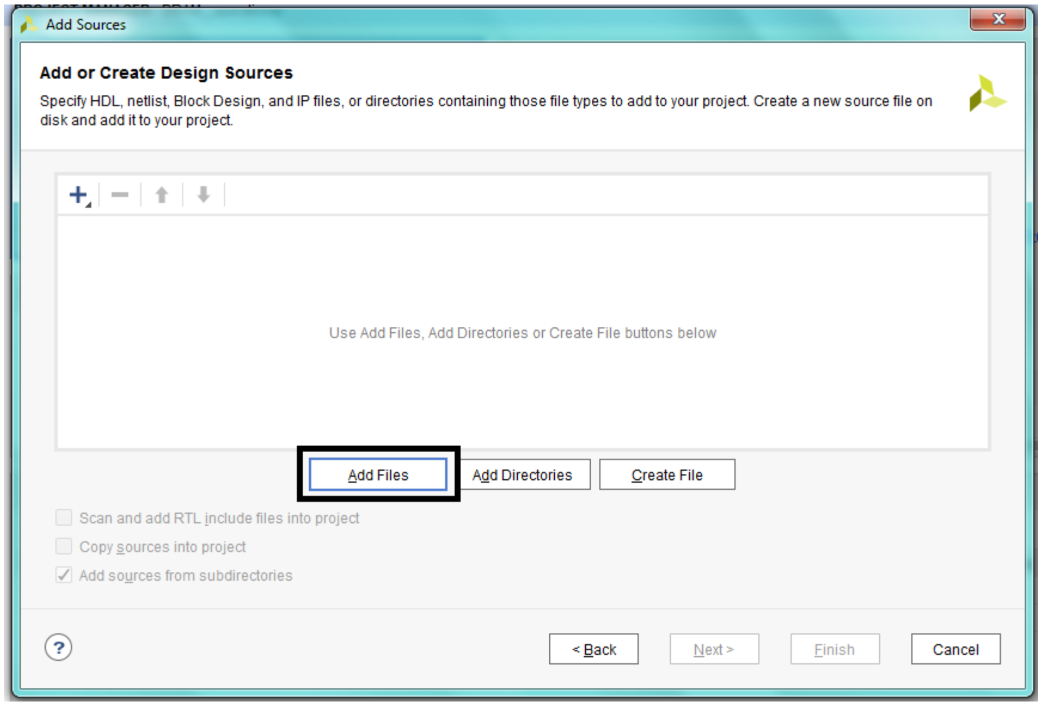


Add Sources

This guides you through the process of adding and creating sources for your project

( ) Add or create constraints
[✓ Add or create design sources]
( ) Add or create simulation sources

15. Click on Add Files


Add Sources

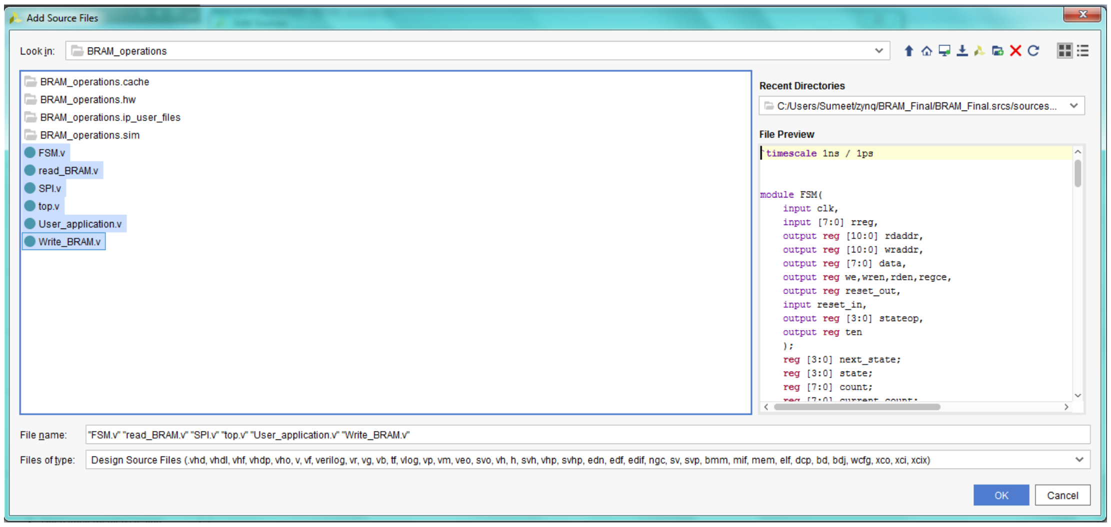
Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

Use Add Files, Add Directories or Create File buttons below

Add Files
Add Directories
Create File
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
< Back
Next >
Finish
Cancel

16. Navigate to your project folder BRAM_operations and select all .v files using shift+down arrow and click OK.


Add Source Files

Look in: BRAM_operations

- BRAM_operations.cache
- BRAM_operations.hw
- BRAM_operations.ip_user_files
- BRAM_operations.sim
- FSM.v
- read_BRAM.v
- SPI.v
- top.v
- User_application.v
- Write_BRAM.v

Recent Directories: C:/Users/Sumeet/zynq/BRAM_Final/BRAM_Final.srcs/sources...

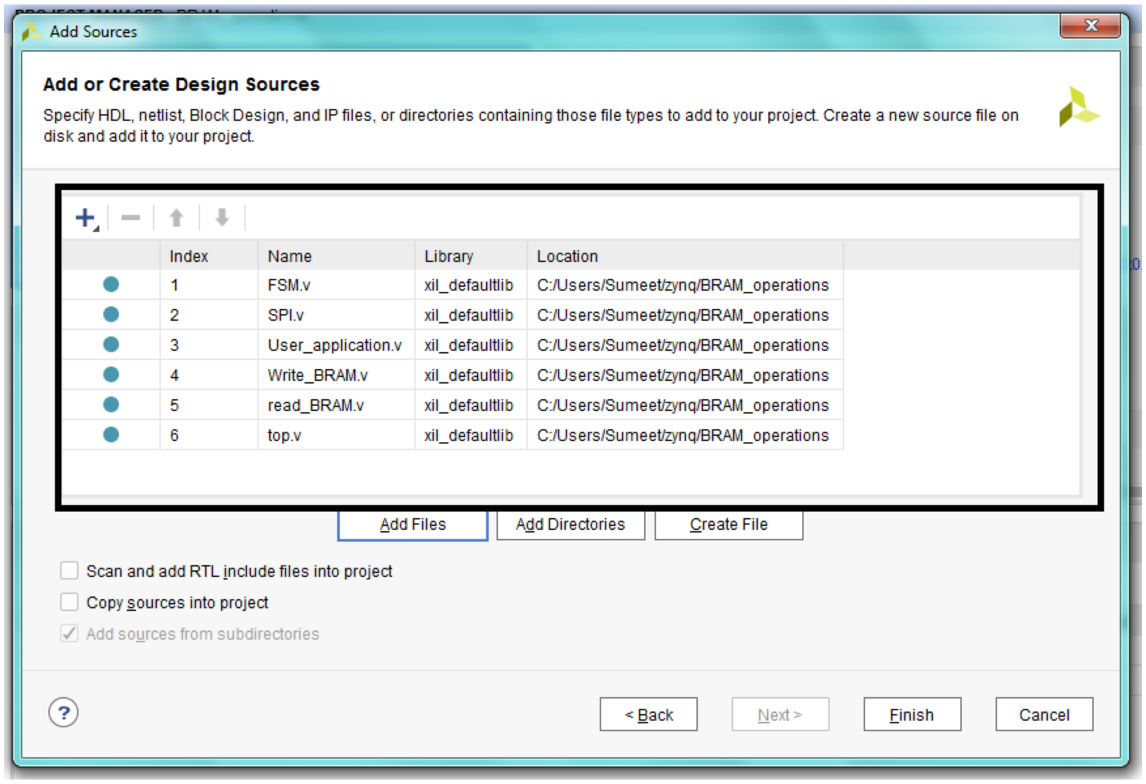
File Preview

```
`timescale 1ns / 1ps


module FSM(
    input clk,
    input [7:0] rreg,
    output reg [10:0] rdaddr,
    output reg [10:0] wraddr,
    output reg [7:0] data,
    output reg we,wren,rden,regce,
    output reg reset_out,
    input reset_in,
    output reg [3:0] stateop,
    output reg ten
    );
    reg [3:0] next_state;
    reg [3:0] state;
    reg [7:0] count;
```

File name: "FSM.v" "read_BRAM.v" "SPI.v" "top.v" "User_application.v" "Write_BRAM.v"

Files of type: Design Source Files (.vhd, vhdl, vhf, vhdp, vho, v, vf, verilog, vr, vg, vb, tf, vlog, vp, vm, veo, svo, vh, h, svh, vhp, svhp, edn, edf, edif, ngc, sv, svp, bmm, mif, mem, elf, dcp, bd, bdj, wcfg, xco, xci, xcix)

17. You can see that the files have been added. Click Finish.



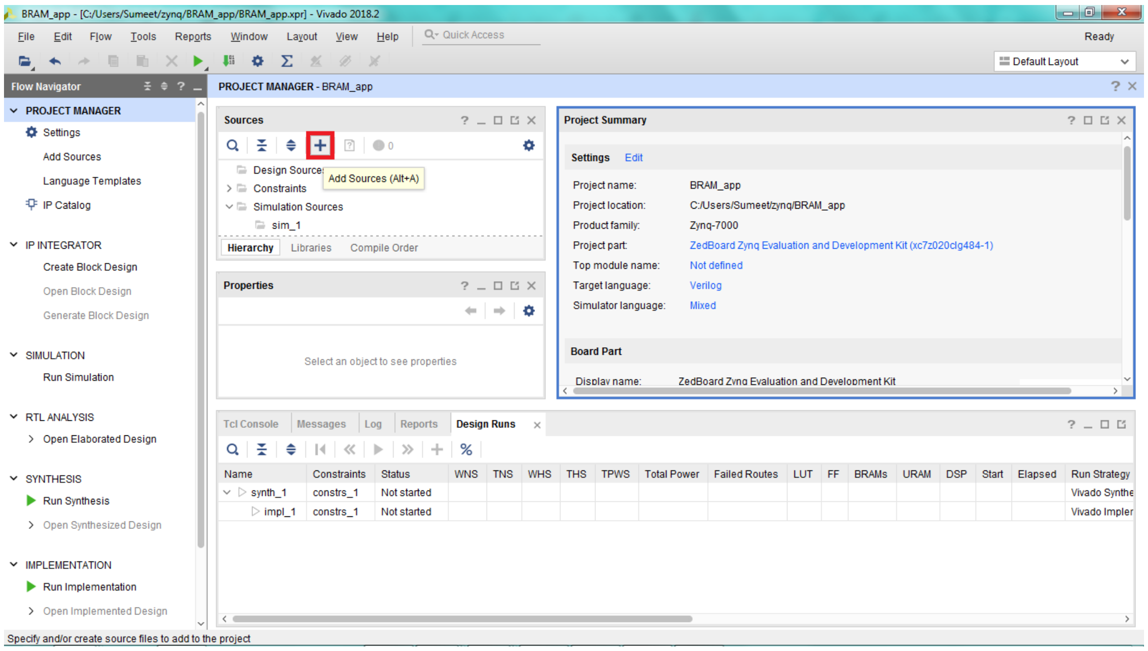
Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

	Index	Name	Library	Location
●	1	FSM.v	xil_defaultlib	C:/Users/Sumeet/zynq/BRAM_operations
●	2	SPI.v	xil_defaultlib	C:/Users/Sumeet/zynq/BRAM_operations
●	3	User_application.v	xil_defaultlib	C:/Users/Sumeet/zynq/BRAM_operations
●	4	Write_BRAM.v	xil_defaultlib	C:/Users/Sumeet/zynq/BRAM_operations
●	5	read_BRAM.v	xil_defaultlib	C:/Users/Sumeet/zynq/BRAM_operations
●	6	top.v	xil_defaultlib	C:/Users/Sumeet/zynq/BRAM_operations

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [x] Add sources from subdirectories

18. Copy the file FPGA_interface-master-> Zedboard -> Constraints-> bram_fsm.xdc
19. Paste the file to your project folder BRAM_operations where you previously copied your .v files
20. Click on the small '+' sign



Settings Edit

Project name:	BRAM_app
Project location:	C:/Users/Sumeet/zynq/BRAM_app
Product family:	Zynq-7000
Project part:	ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

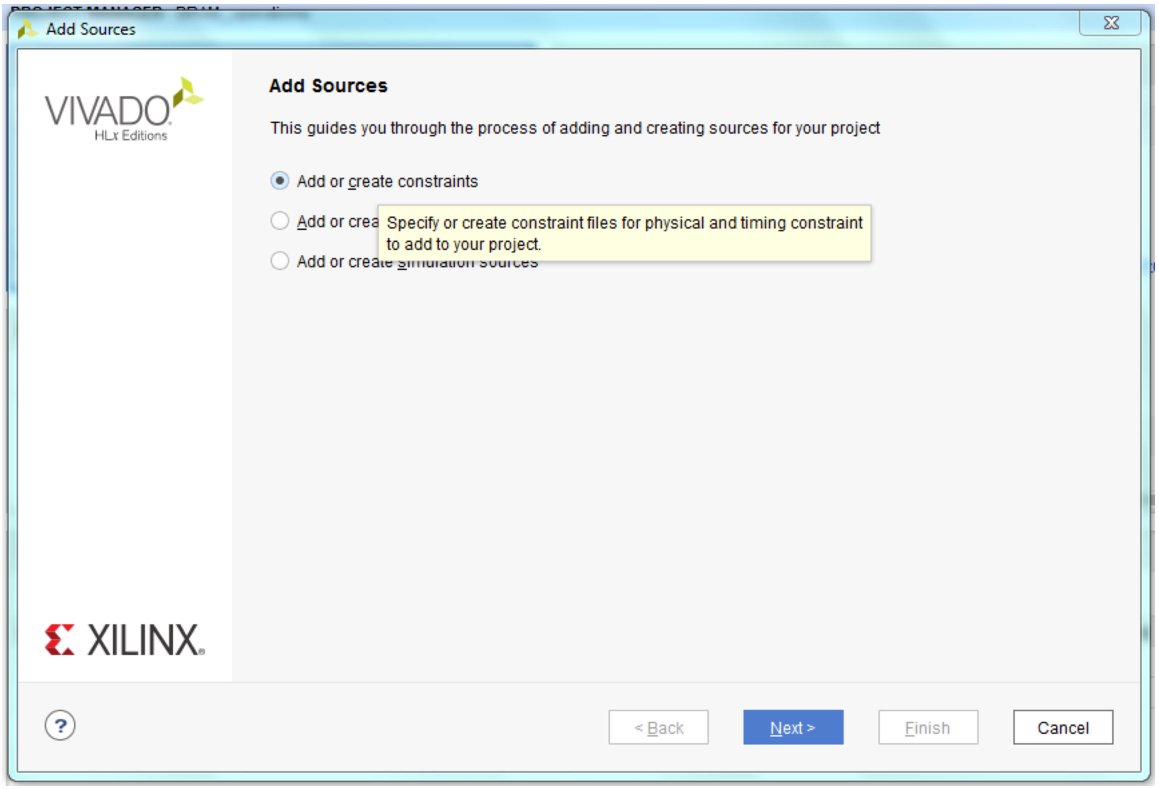
Board Part

Display name: ZedBoard Zynq Evaluation and Development Kit

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthe
impl_1	constrs_1	Not started															Vivado Impler

Specify and/or create source files to add to the project

21. Add or create constraints. Click Next



Add Sources

Add Sources

This guides you through the process of adding and creating sources for your project

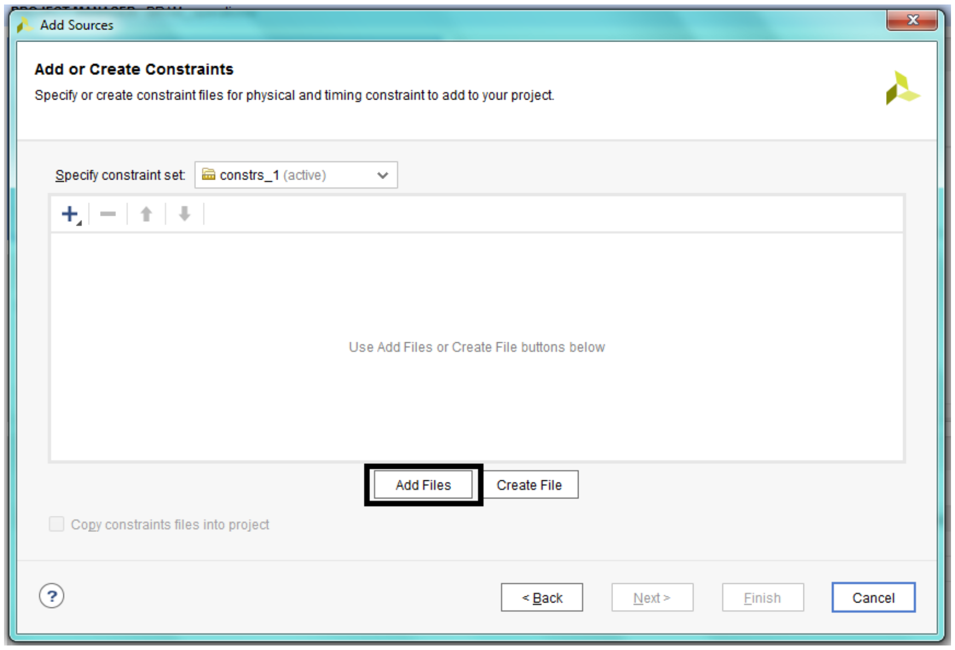
(●) Add or create constraints

( ) Add or crea

( ) Add or create simulation sources

Specify or create constraint files for physical and timing constraint to add to your project.

22. Click on Add files.



Add Sources

Add or Create Constraints

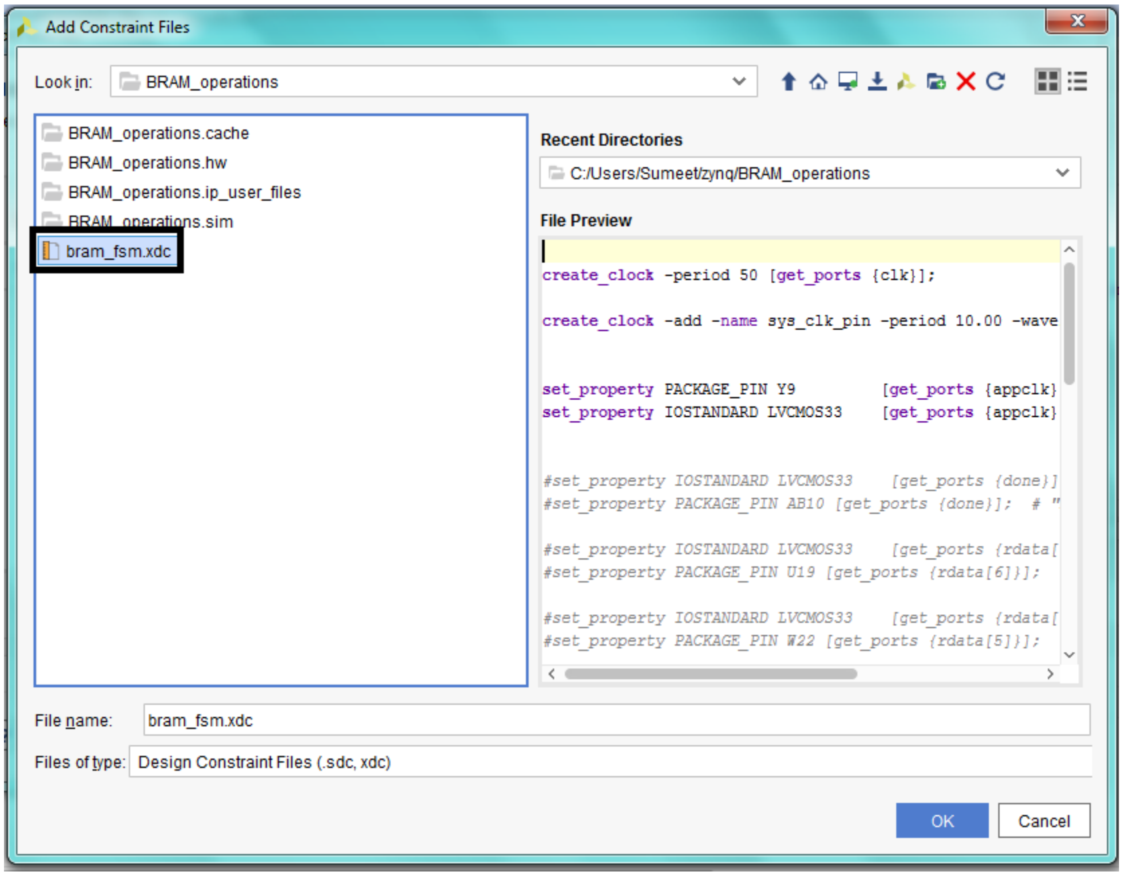
Specify or create constraint files for physical and timing constraint to add to your project.

Specify constraint set: constrs_1 (active)

Use Add Files or Create File buttons below

[ ] Copy constraints files into project

23. Select the bram_fsm.xdc file and click OK. Then Click finish.

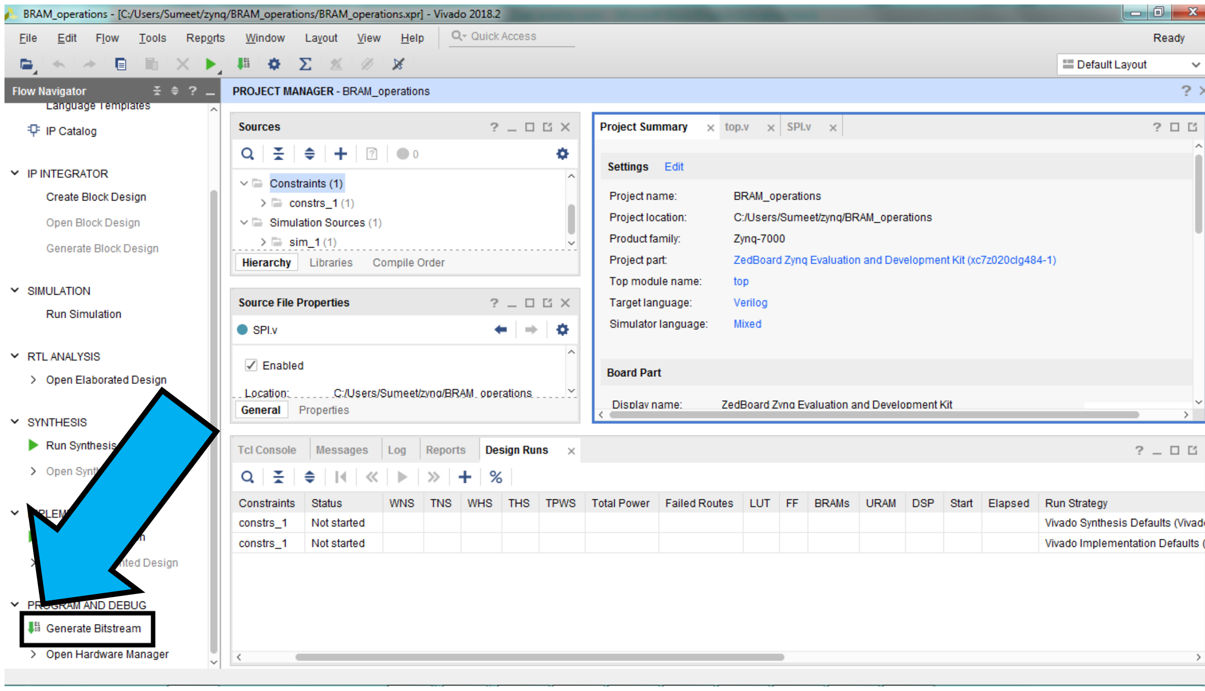


```
create_clock -period 50 [get_ports {clk}];

create_clock -add -name sys_clk_pin -period 10.00 -wave


set_property PACKAGE_PIN Y9          [get_ports {appclk}
set_property IOSTANDARD LVCMOS33     [get_ports {appclk}


#set_property IOSTANDARD LVCMOS33    [get_ports {done}]
#set_property PACKAGE_PIN AB10 [get_ports {done}];  # "

#set_property IOSTANDARD LVCMOS33    [get_ports {rdata[
#set_property PACKAGE_PIN U19 [get_ports {rdata[6]}];

#set_property IOSTANDARD LVCMOS33    [get_ports {rdata[
#set_property PACKAGE_PIN W22 [get_ports {rdata[5]}];
```

File name: bram_fsm.xdc

Files of type: Design Constraint Files (.sdc, xdc)

24. Click on “Generate Bitstream”



Settings Edit

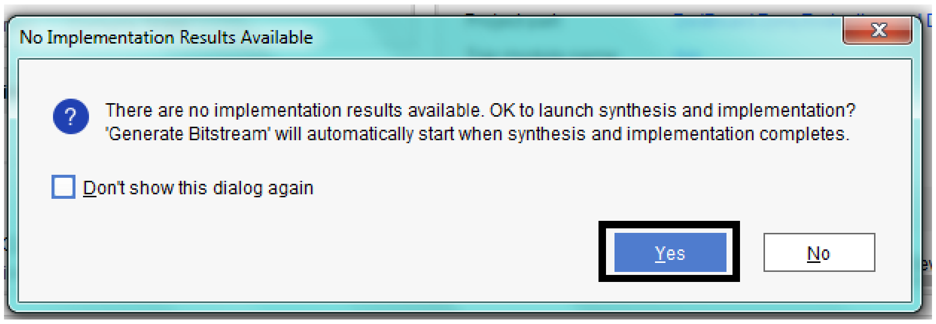
Project name:	BRAM_operations
Project location:	C:/Users/Sumeet/zynq/BRAM_operations
Product family:	Zynq-7000
Project part:	ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)
Top module name:	top
Target language:	Verilog
Simulator language:	Mixed

Board Part

Display name:	ZedBoard Zynq Evaluation and Development Kit

Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
constrs_1	Not started															Vivado Synthesis Defaults (Vivad
constrs_1	Not started															Vivado Implementation Defaults (

25. Click Yes on this popup

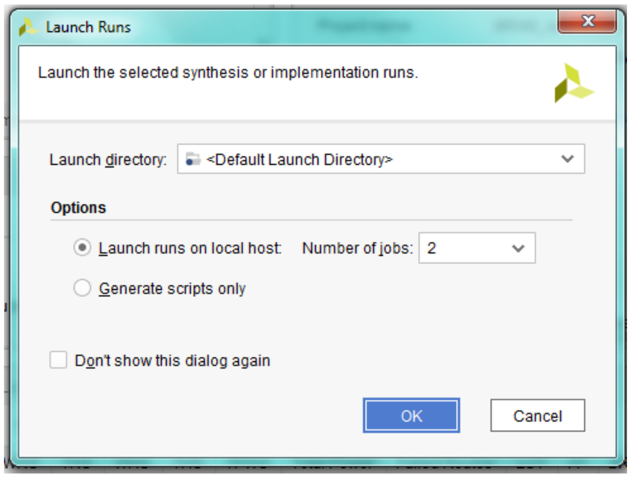


No Implementation Results Available

There are no implementation results available. OK to launch synthesis and implementation?
'Generate Bitstream' will automatically start when synthesis and implementation completes.

Don't show this dialog again

26. Click OK



Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

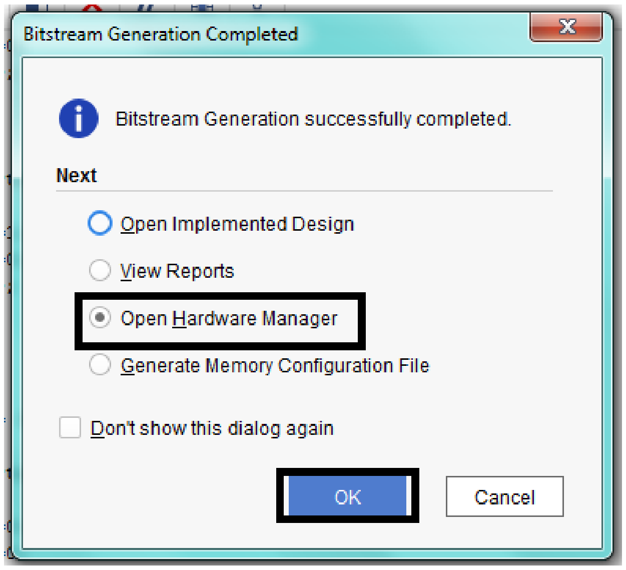
Options

Launch runs on local host: [✓] Number of jobs: 2

Generate scripts only

Don't show this dialog again

27. Vivado will now generate the output products required to program the FPGA. You can track the progress in the top right of the window. This will take some time.
28. On completion of bitstream generation, you will see the following dialog box. Click on “Open Hardware Manager” and press OK.



Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

Open Implemented Design

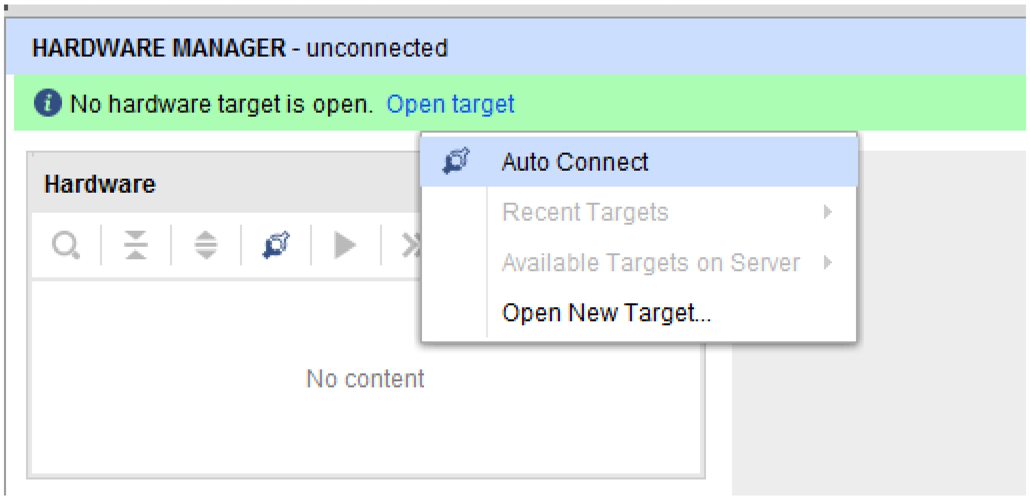
View Reports

[✓ Open Hardware Manager]

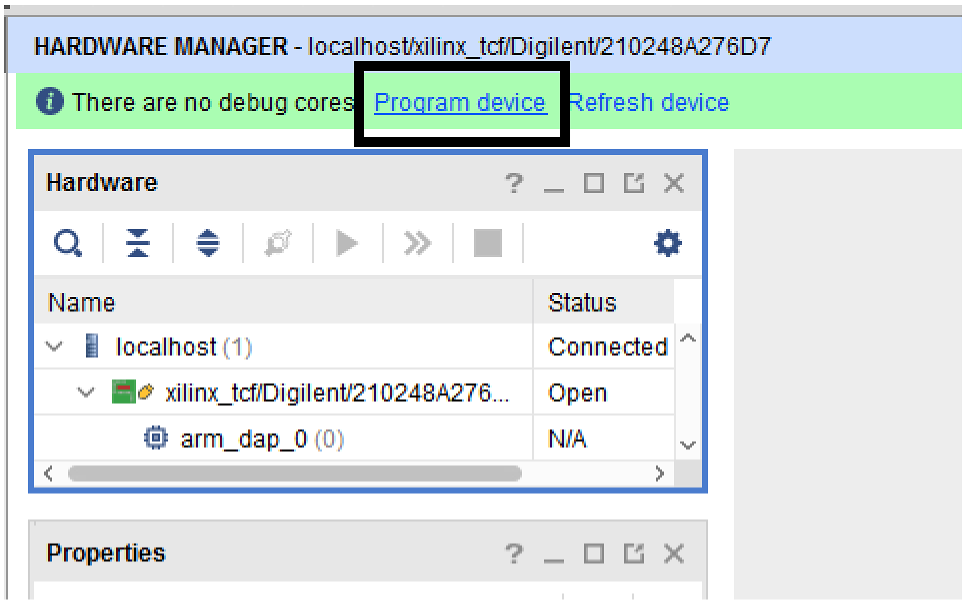
Generate Memory Configuration File

Don't show this dialog again

29. Click Open Target -> Auto Connect

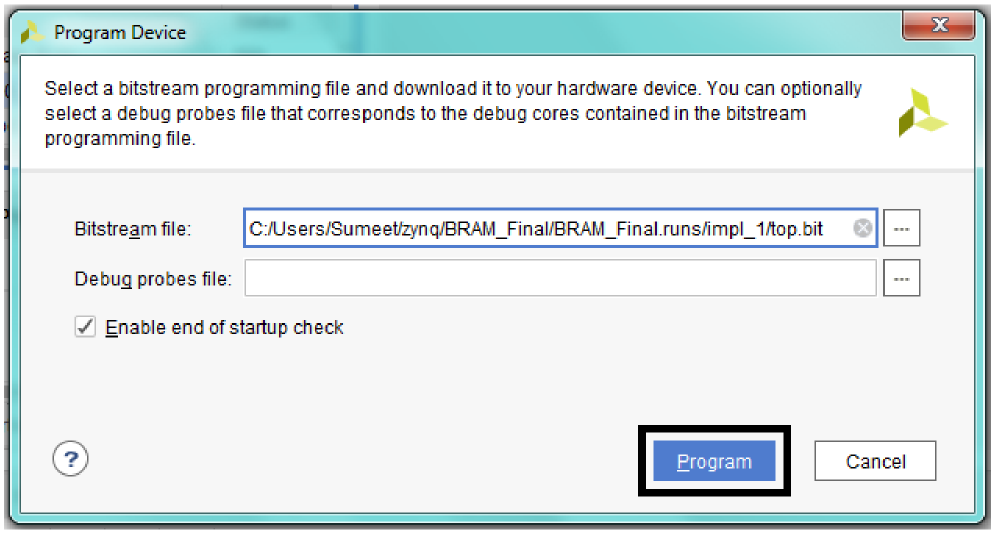

HARDWARE MANAGER - unconnected
No hardware target is open. Open target
Hardware
No content
Auto Connect
Recent Targets
Available Targets on Server
Open New Target...

30. Click Program Device


HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210248A276D7
There are no debug cores. Program device Refresh device
Hardware
Properties

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210248A276...	Open
arm_dap_0 (0)	N/A

31. Click Program


Program Device

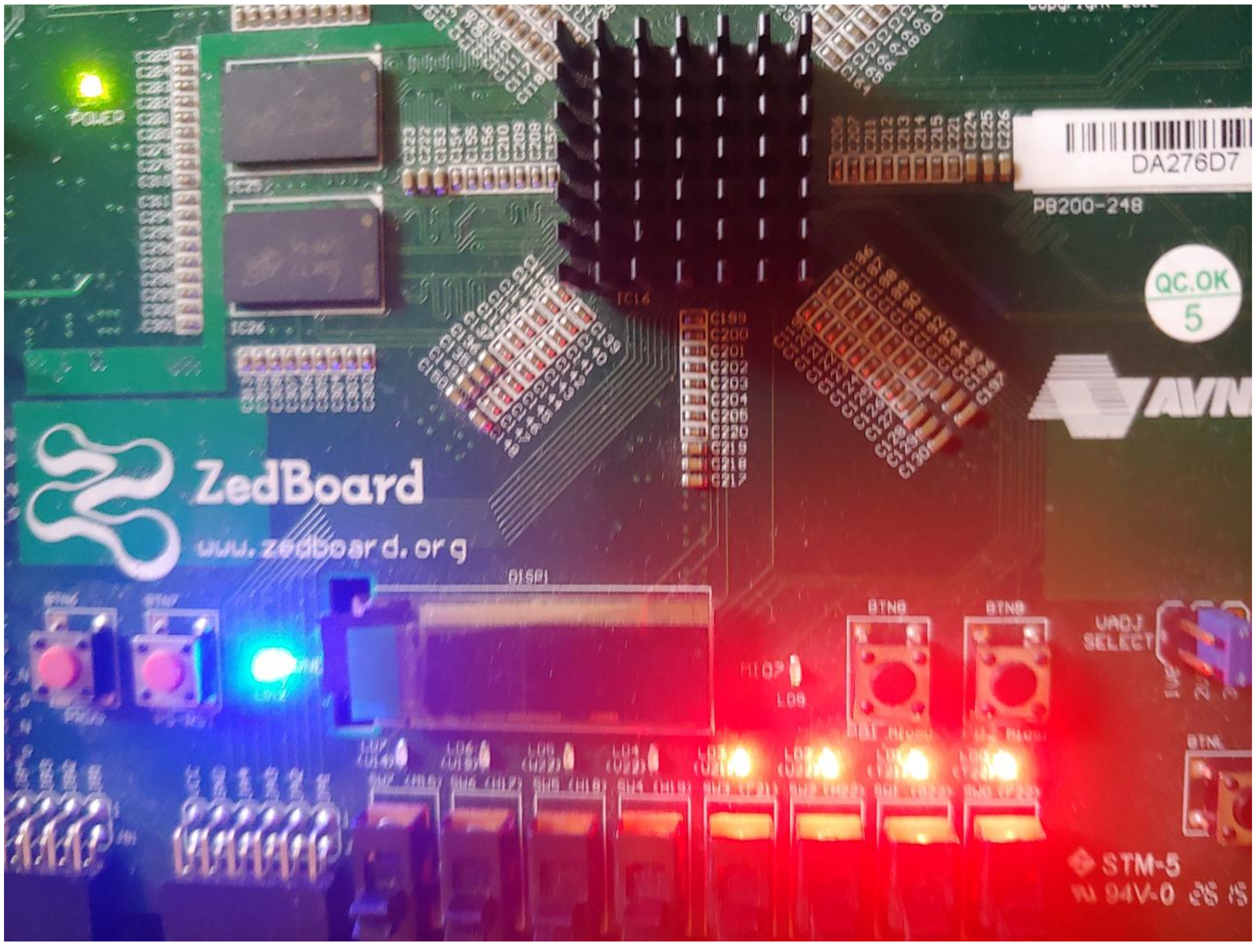
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: C:/Users/Sumeet/zynq/BRAM_Final/BRAM_Final.runs/impl_1/top.bit

Debug probes file:

☑ Enable end of startup check

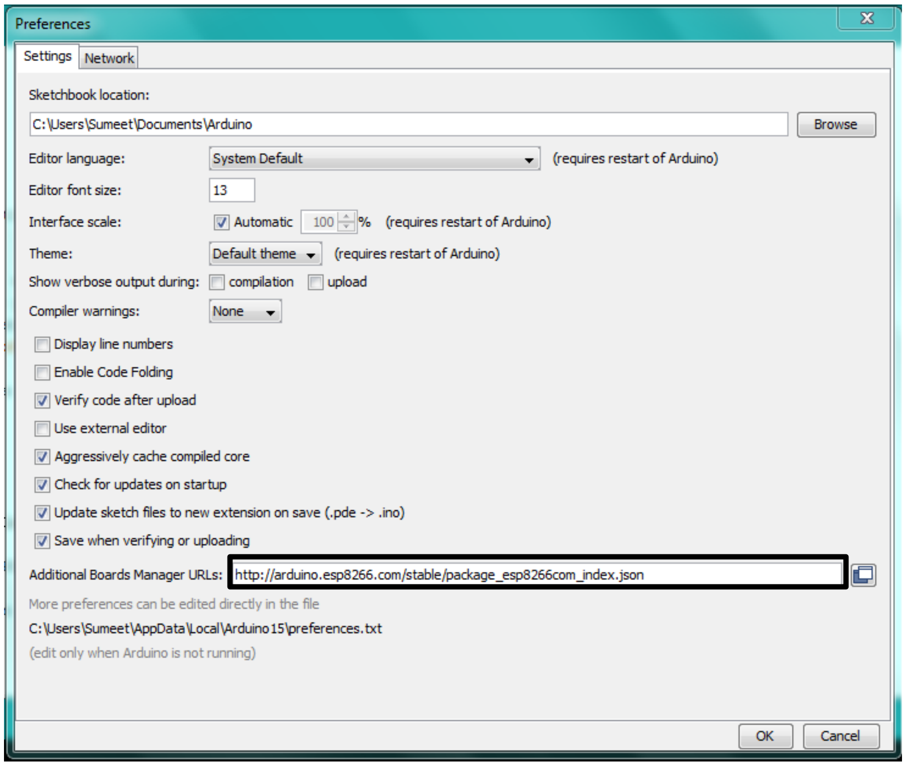
32. Once Programmed, the Blue 'DONE' LED on the Zedboard along with 4 RED LEDs will turn ON.



5. Programming the NodeMCU

1. Before starting the procedure, make sure you have a stable internet connection.
2. The first step in setting up NodeMCU is downloading board files in Arduino IDE. To do so, first open Arduino IDE and go to File -> Preferences.

3. In the Additional Boards Manager URLs field, paste the following URL

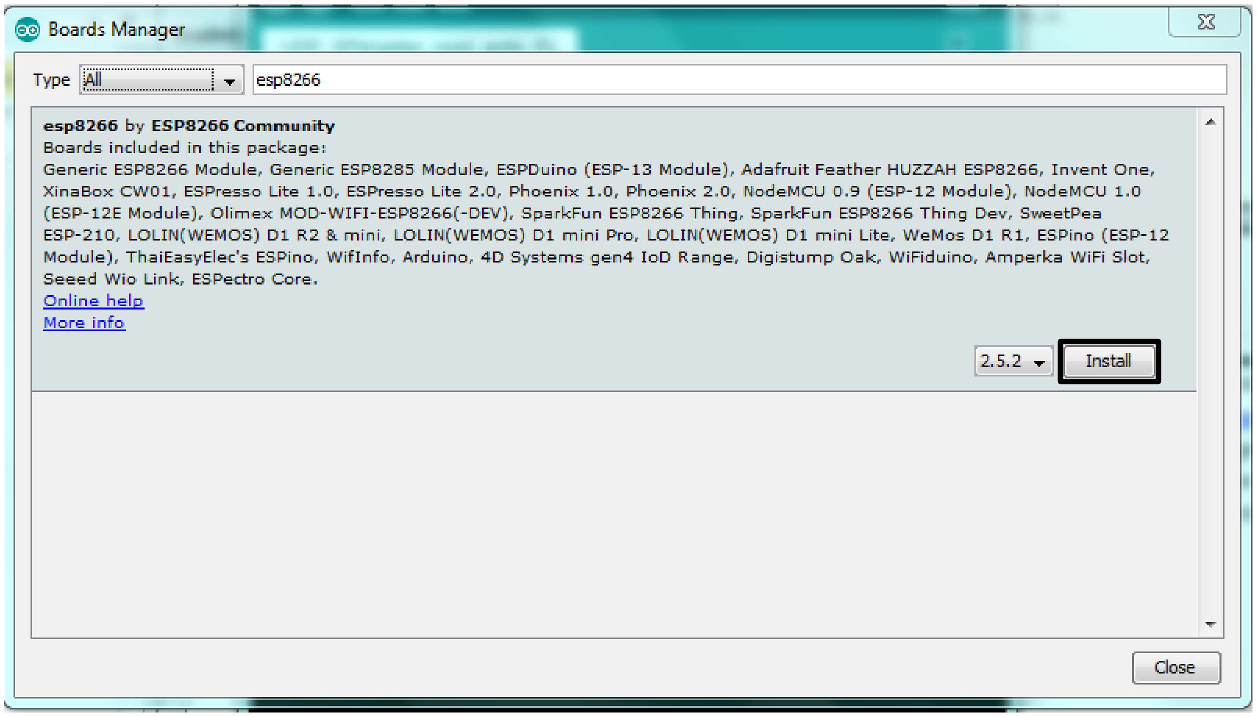


Sketchbook location:
C:\Users\Sumeet\Documents\Arduino
Editor language: System Default (requires restart of Arduino)
Editor font size: 13
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: [ ] compilation [ ] upload
Compiler warnings: None
[ ] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Aggressively cache compiled core
[✓] Check for updates on startup
[✓] Update sketch files to new extension on save (.pde -> .ino)
[✓] Save when verifying or uploading
Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json
More preferences can be edited directly in the file
C:\Users\Sumeet\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

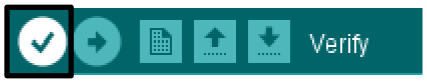
http://arduino.esp8266.com/stable/package_esp8266com_index.json and press OK

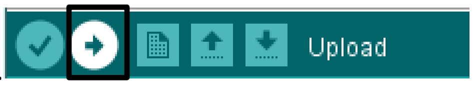
4. Now go to Tools -> Board -> Boards Manager and wait for platforms index to be downloaded. You can track progress at the bottom of the window.

5. Type esp8266 in the search bar of the boards manager. Select the latest version



esp8266 by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.

6. Click install and wait for the board files to be downloaded. You can track the progress at the bottom of the window. Close the window on completion of installation.
7. Go to tools -> Board ->NodeMCU 1.0 (ESP-12E Module).
8. At this point connect your NodeMCU board to your computer with a micro USB cable.
9. To test whether your board is working, go to file->Examples->01. Basics->Blink
10. Once the sketch is opened, click the verify button 

11. On successful compilation, click the Upload button. 

12. Track the progress in the console below the editor.
13. On successful upload, you will see a blue LED on the NodeMCU blinking. This means that you are ready to use the board.

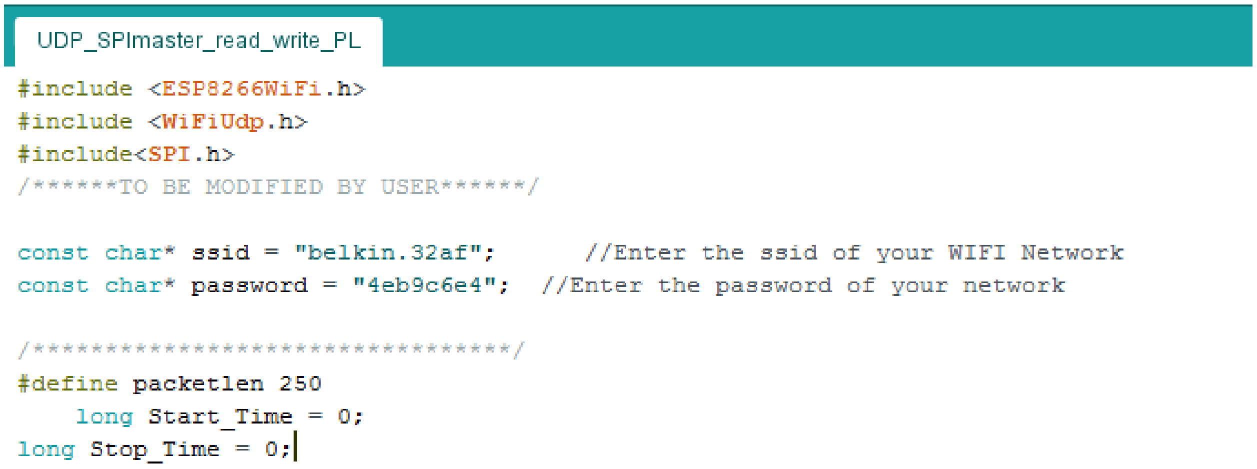
Connecting the NodeMCU to a Wi-Fi Network.

1. It is recommended that you use a wireless router for Wi-Fi. If the Wi-Fi router is unavailable, create a mobile hotspot and note down the network name (SSID) and password. You will need these later.
2. On your PC, open Downloads->FPGA_interface -> NodeMCU -> UDP_SPImaster_read_write_PL.txt
3. Copy the entire code.
4. In Arduino IDE, press Ctrl + N to open a new sketch.

5. Paste the code from UDP_SPImaster_read_write_PL.txt in the editor window.
6. Save the code under an appropriate filename.
7. Near the Top of the code, you will find a field which is to be modified by the user.

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include<SPI.h>
/******TO BE MODIFIED BY USER******/

const char* ssid = "Hotspot";       //Enter the ssid of your WIFI Network
const char* password = "Passwerd";  //Enter the password of your network

/**********************************/
long Start_Time = 0;
```

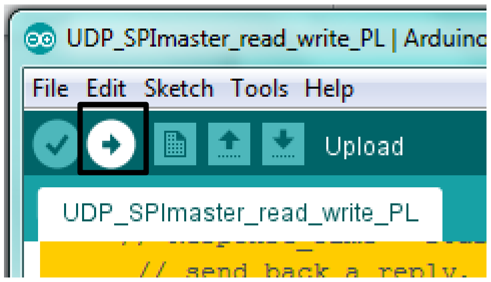
8. In this field, replace the word "Hotspot" by the name of your Wi-Fi network (SSID) and the word "Passwerd" with the password of your network. Make sure your SSID and password are enclosed in double quotes as shown below. These names are case sensitive.



UDP_SPImaster_read_write_PL

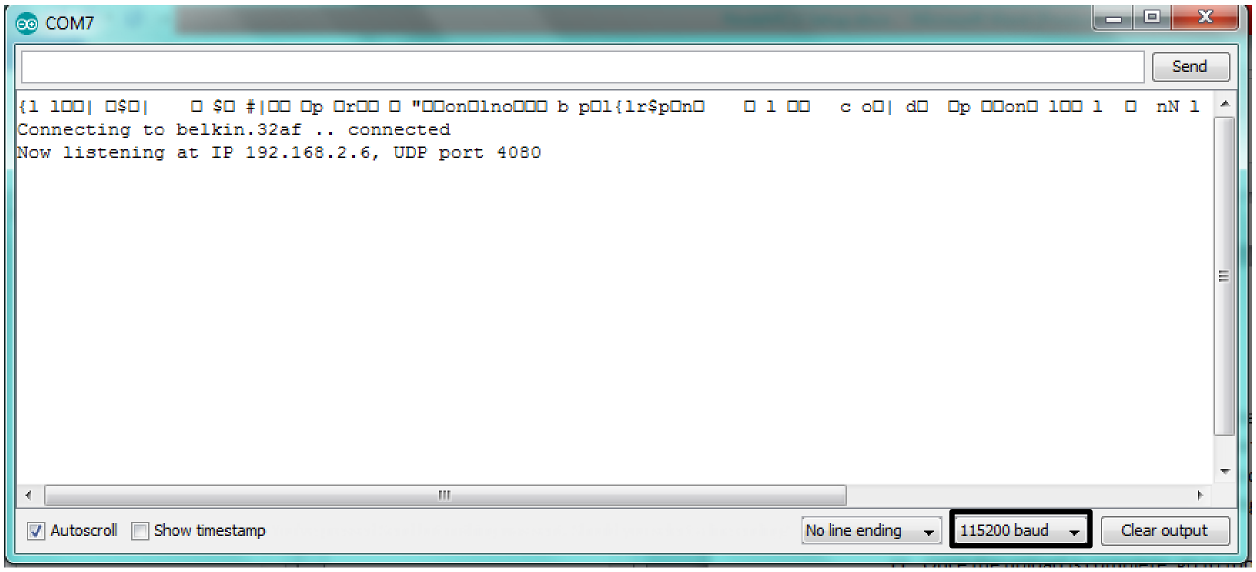
```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include<SPI.h>
/******TO BE MODIFIED BY USER******/

const char* ssid = "belkin.32af";       //Enter the ssid of your WIFI Network
const char* password = "4eb9c6e4";  //Enter the password of your network

/**********************************/
#define packetlen 250
    long Start_Time = 0;
long Stop_Time = 0;
```

9. Once you have made these changes click the upload button.


UDP_SPImaster_read_write_PL | Arduinc
File Edit Sketch Tools Help
Upload
UDP_SPImaster_read_write_PL
// send back a reply.

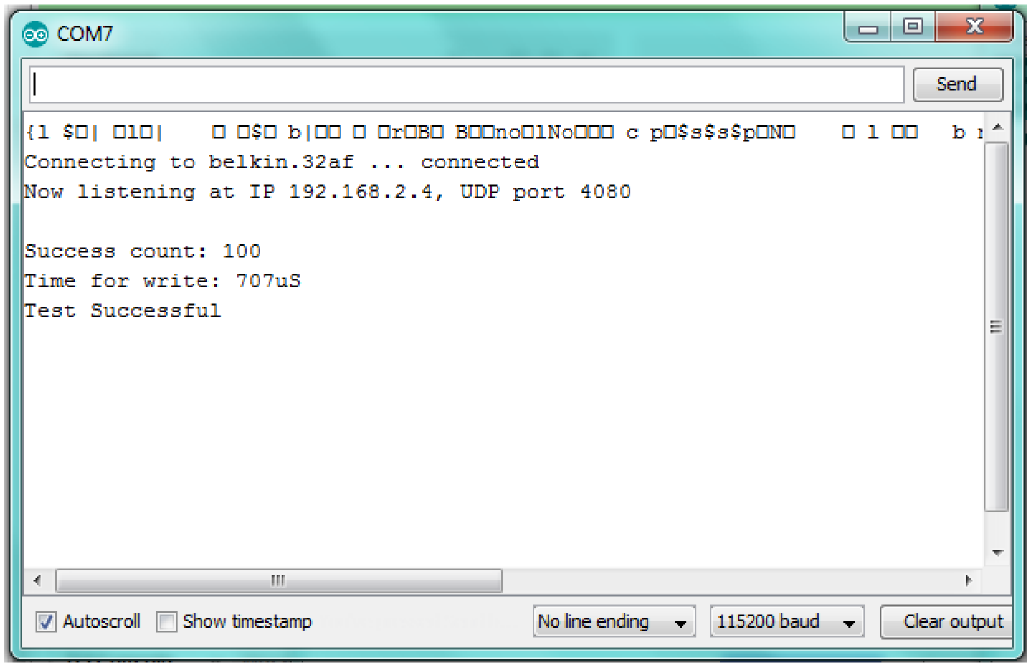
10. You can track the progress of the upload at the bottom of the window. You will see a message: "Uploading..." This means that the sketch has compiled successfully and is being uploaded to the NodeMCU. Wait for the upload to complete.
Note: If you see a message: "Uploading to IO Board" click the upload button again. This will happen very rarely.
11. Once the upload is complete, go to tools->Serial monitor or press Ctrl+shift+M to open serial monitor.

12. As shown in the following figure, set the baud rate at the bottom right to 115200.



```
{l l□□| □$□|    □ $□ #|□□ □p □r□□ □ "□□on□lno□□□ b p□l{lr$p□n□    □ l □□   c o□| d□  □p □□on□ l□□ l   □  nN l
Connecting to belkin.32af .. connected
Now listening at IP 192.168.2.6, UDP port 4080
```

13. Press ‘RST’ button on the NodeMCU.
14. You will see the following text in the Serial monitor window if you enter the SSID and password correctly in the code.
    Connecting to belkin.32af …… connected
    Now listening at IP **192.168.2.6**, UDP port 4080
    Here, “belkin.32af” is the name of your network and “192.168.2.6” is the IP of the server created by the NodeMCU. Your serial monitor will display data other than the one shown above. The highlighted text is the IP of you NodeMCU server.
15. Note down the IP assigned to the NodeMCU. You will need it later.
16. Your serial Monitor should look like this. (Press ctrl+shift+m to open)



```
{l $□| □l□|    □ □$□ b|□□ □ □r□B□ B□□no□lNo□□□ c p□$s$s$p□N□    □ l □□   b ı
Connecting to belkin.32af ... connected
Now listening at IP 192.168.2.4, UDP port 4080

Success count: 100
Time for write: 707uS
Test Successful
```

6. Running the Linux Client

1. Connect the Linux PC to the same network as the NodeMCU.
2. First, we need to make sure that the NodeMCU is accessible. To do this, first open terminal in Linux.
3. On the terminal type the command "ping <IP of NodeMCU>". For example : ping 192.168.2.6
4. This will initiate a ping sequence which will give you the time required by packets to reach the NodeMCU.
5. After a couple of iterations, press ctrl+Z to exit the sequence.
6. Open udp_PL.c using the command
   gedit udp_PL.c
7. Near the top of the code after the include statements, there is a field to be modified by user.
8. Change the #define SERVER to the IP of the NodeMCU server you noted down here
9. Save the file.
10. Compile the C file using the command:
    cc udp_PL.c –o udp
11. On success, an executable file with the name udp will be created on the in the directory.
12. To run this file, use the command:*./udp*
13. Press the RST button on your NodeMCU and observe Serial Monitor.
14. When the program is run, you will be made to choose between read and write. Enter 'w' to write data. Choose a location, for example, 1 and enter your desired data.
15. On successful write, you will be given a choice to read or write again. This time, enter 'r' to read data. When an address is requested, enter the same address where you wrote the data, i. e. 1. You will be asked the number of bytes to be read. Ask for the number of bytes same as the length of the message you sent. For example, if you sent "Hello" as a message, you will need to ask for 5 bytes to be read as "Hello" contains 5 characters.
16. On a successful read, you will be presented with the fetched data.