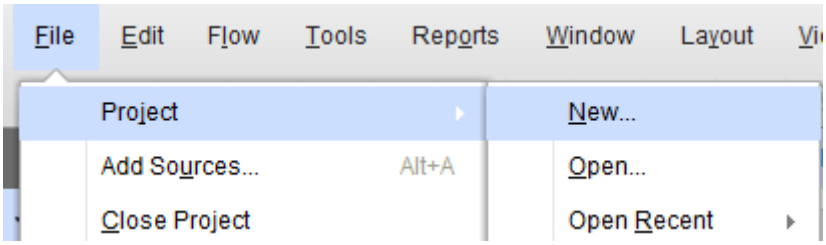
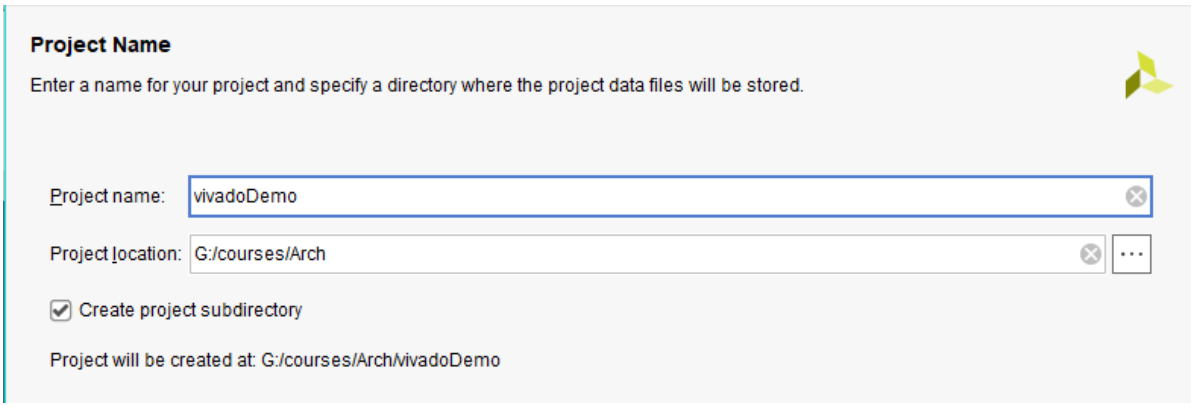
Use Vivado Step by Step

Create a project

New project

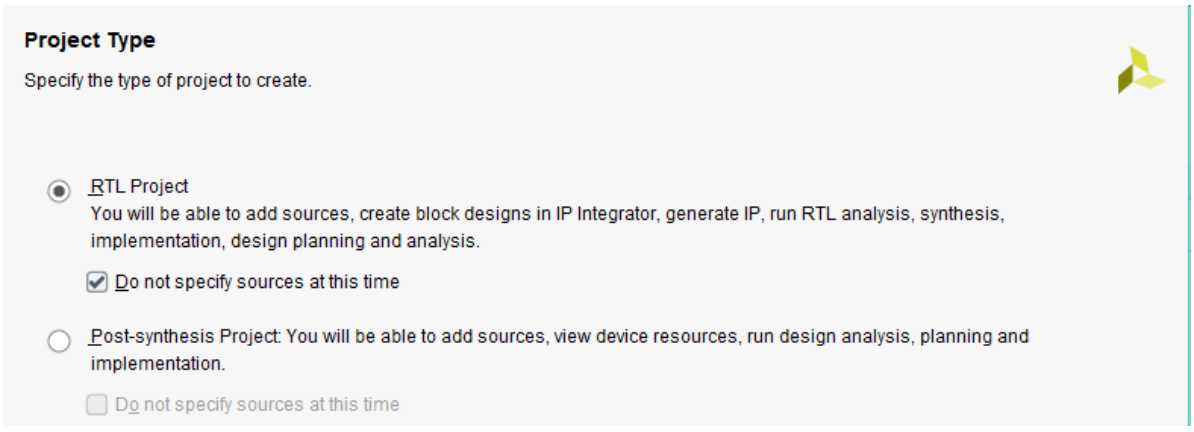


Define project name



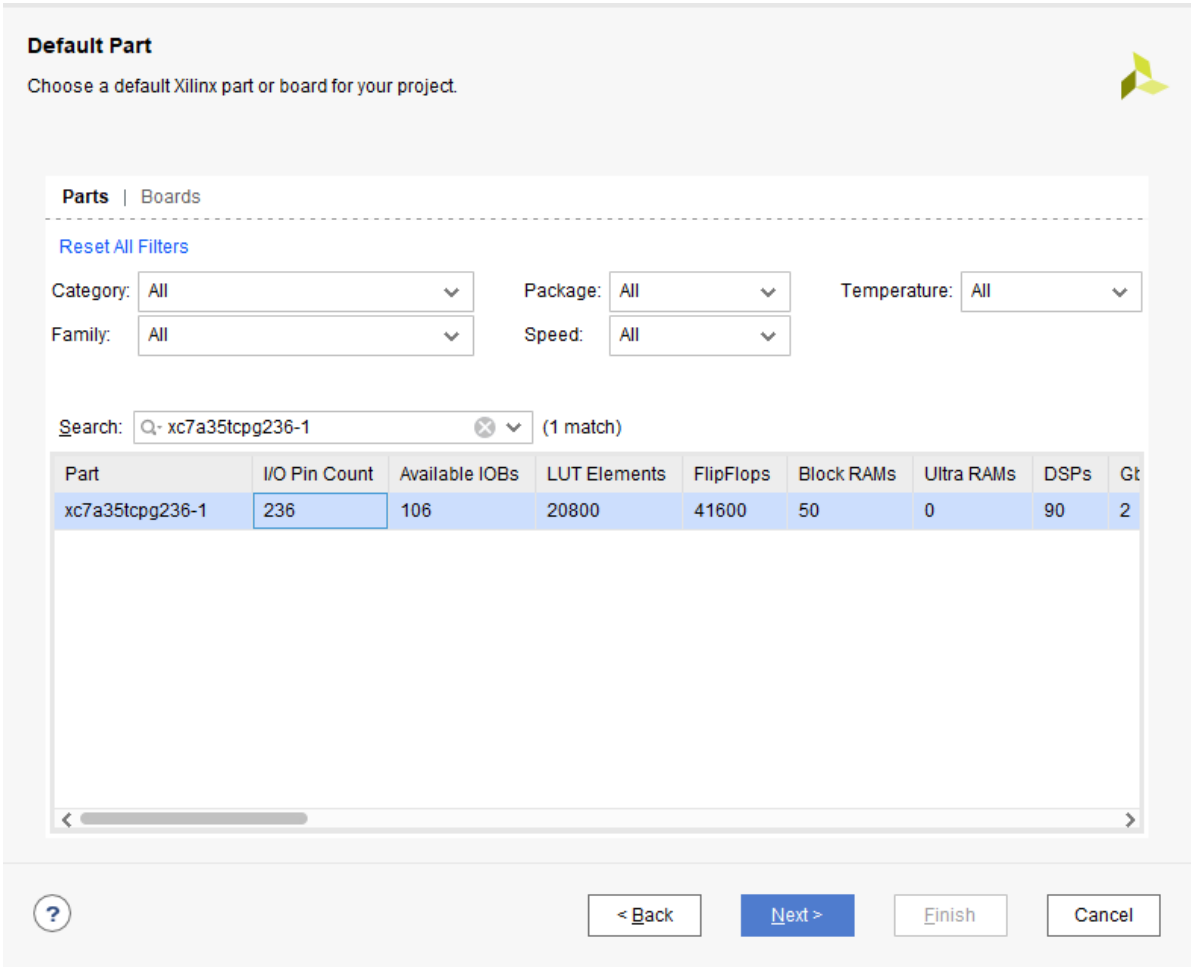
Choose project type to be RTL project

since you need to add some provided code into the project, we recommend you add all sources after creating this project



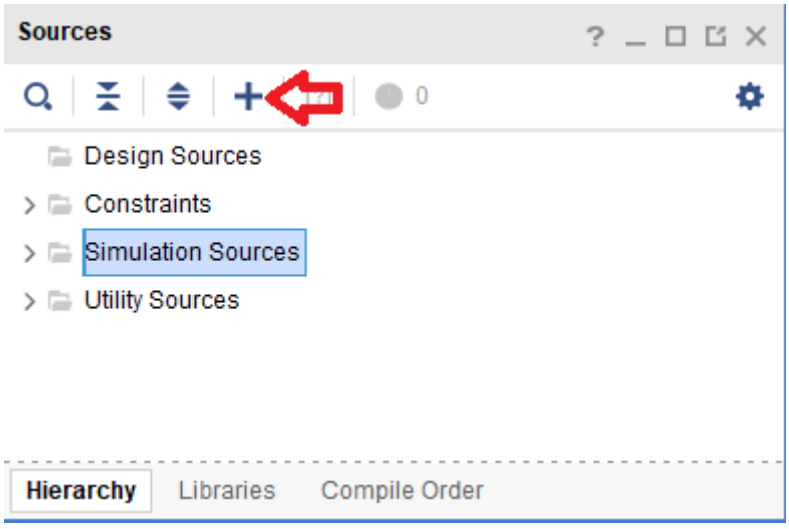
Select Part

Important : Select the part name we wrote in the project doc, otherwise you may fail to run your CPU on FPGA. (but you can still change this part name after creating whole project, so whatever)



Add provided code and other resources

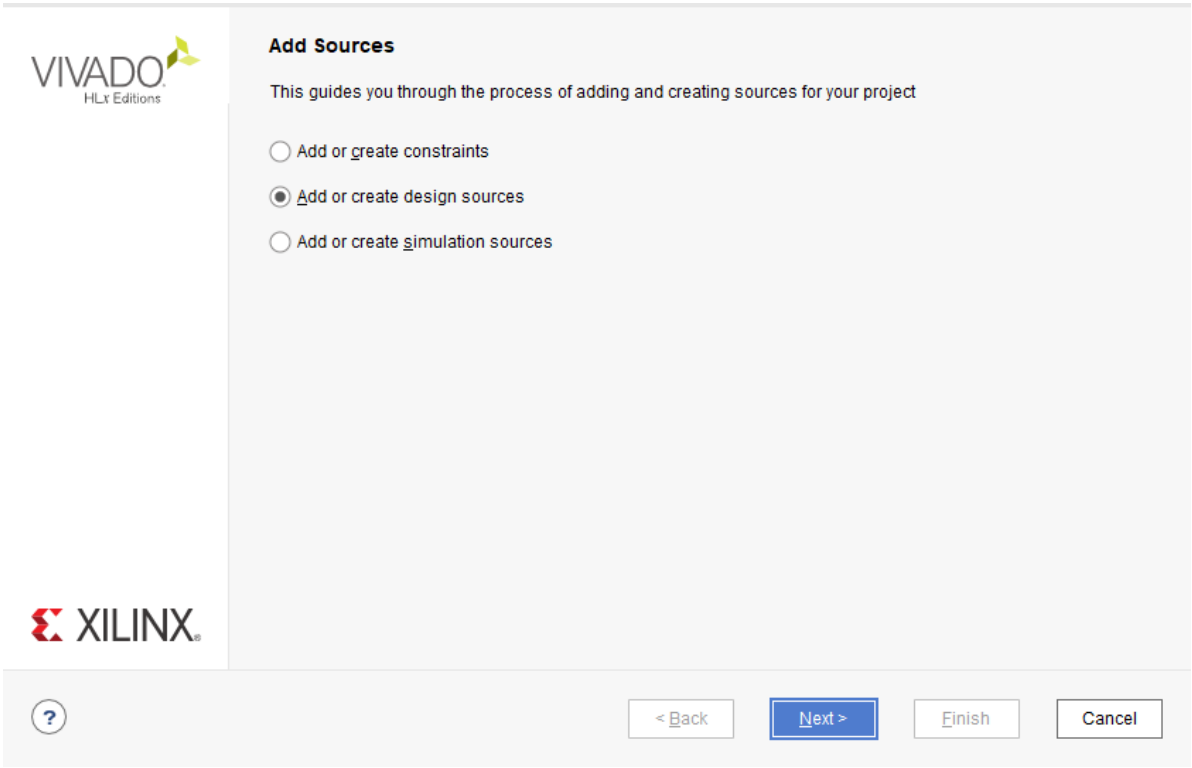
Click on the '+' button



You may see three kinds of resource type,

Here 'constraints' are for .xdc files, i.e. Basys-3-Master.xdc we provided

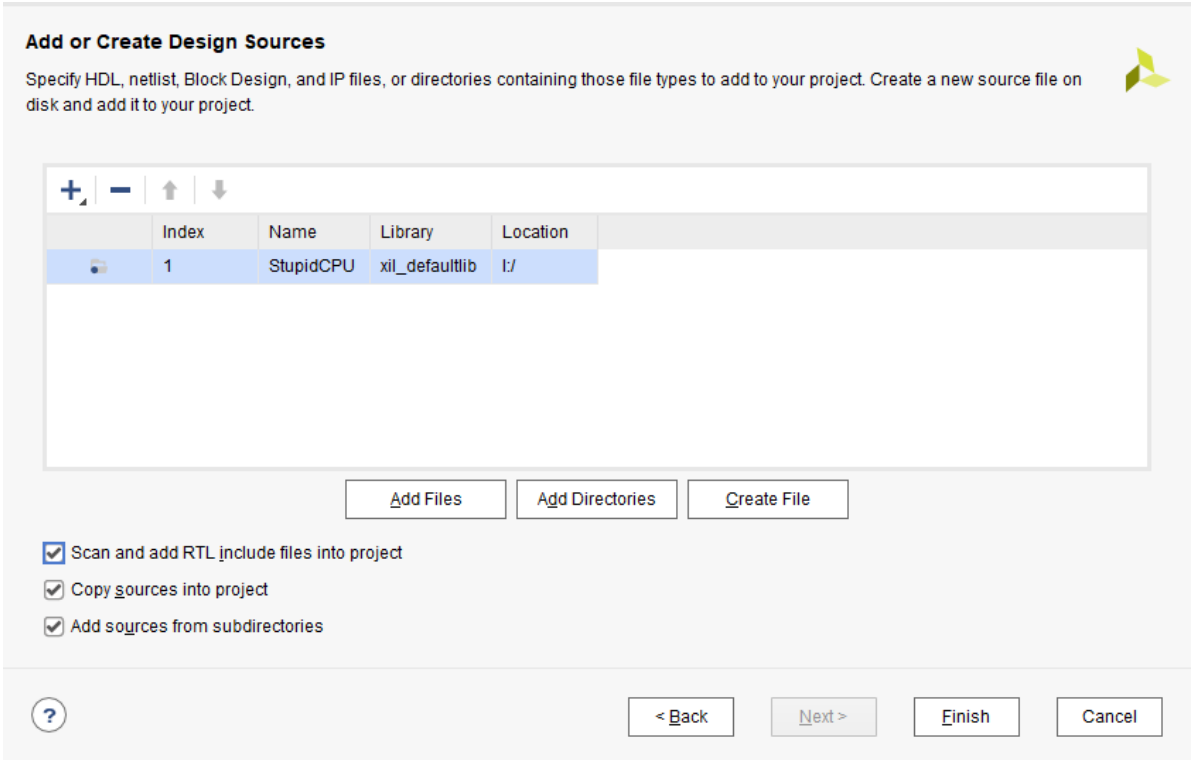
and 'design sources' are for .v files, i.e. cpu.v, hci.v, riscv_top.v we provided and your own .v files



Add your files

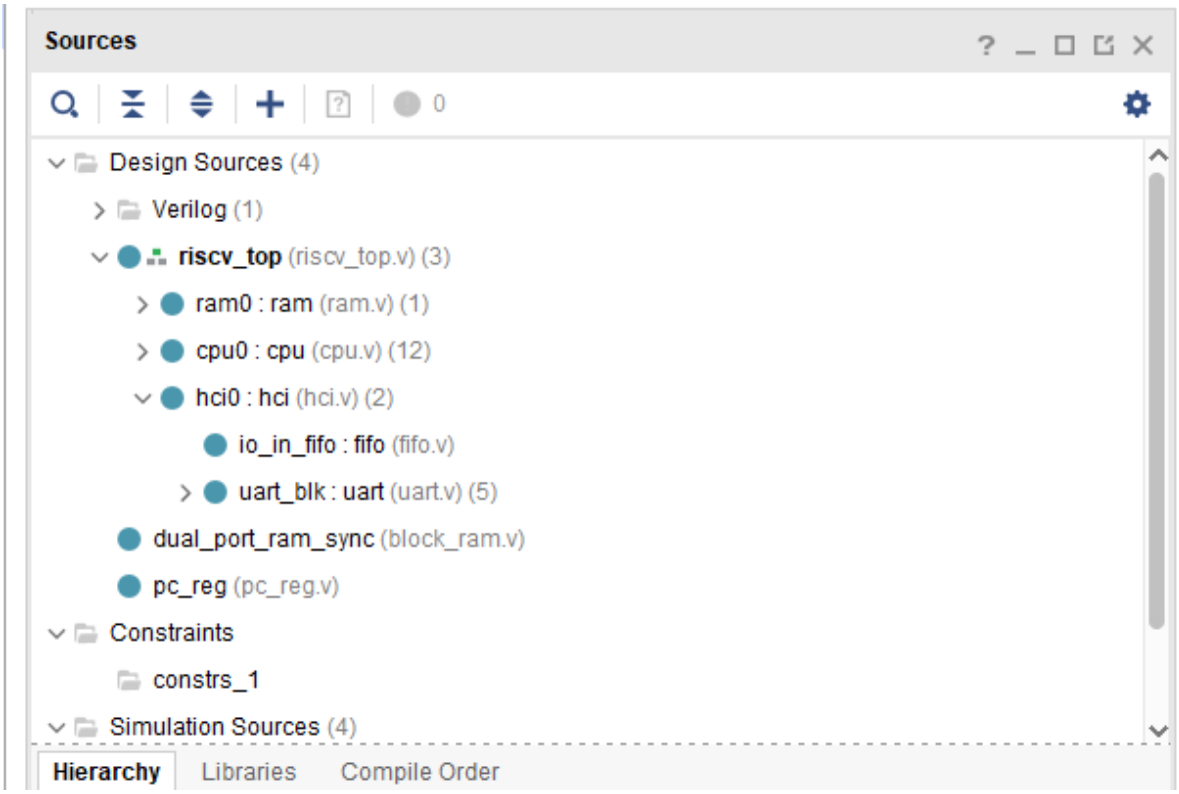
You can choose add directories to add all the provided code for convenience.

But you need to add .xdc file after clicking "Add or create constraints" in the last step.



Check sources

After adding the design files, you will see your design files organized by Vivado automatically. This also demonstrates the hierarchy of your .v files.



Run SIMULATION

We simulate the running of CPU in Vivado to check basic logic of our design.

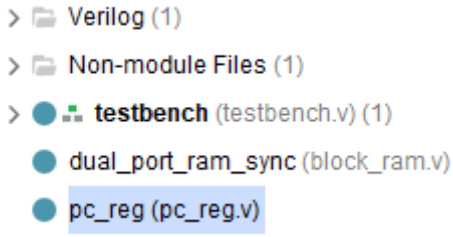
Note: Passing simulation doesn't necessarily mean your design is workable on FPGA.

Before running simulation

- You must add testbench.v into your project and enable this file to run simulation. This is the file that provides fake clk signals in simulation.

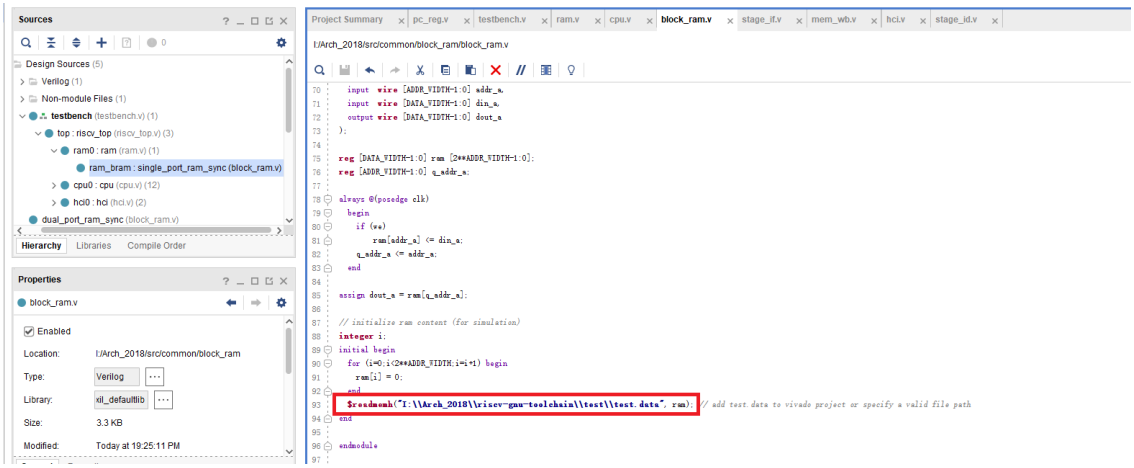
  In other words, you shall find testbench.v the top of your design sources.

  If you want to test your design on FPGA, you must disable 'testbench.v' since it can't be synthesised.



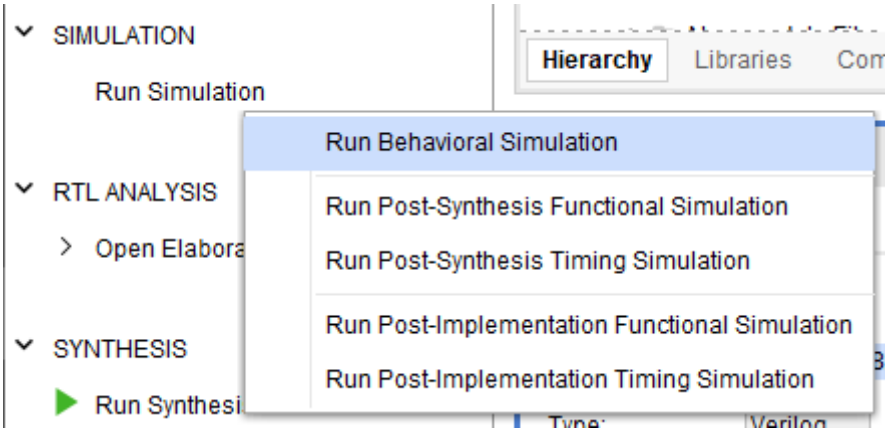
- You must tell Vivado where to find memory file, which is the input file for a test.

  In 'block_ram.v', change the path here to the path of your 'test.data'. If there's some problem with relative path, just use absolute path.

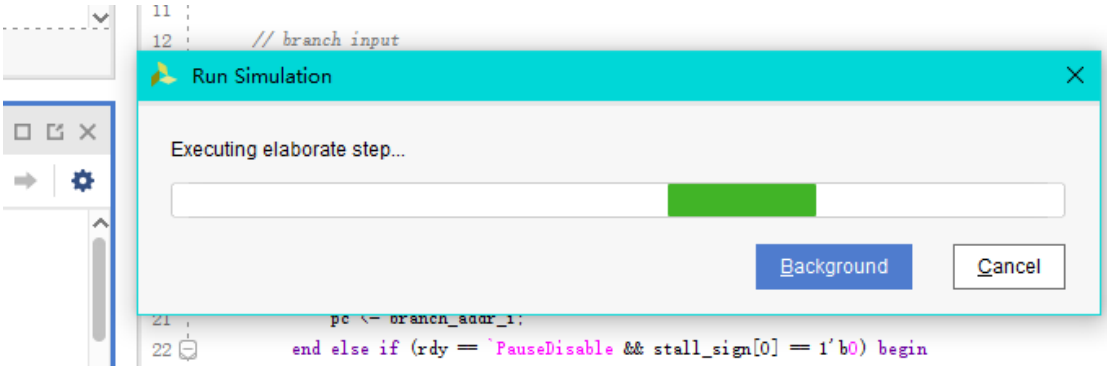


Start simulation

Click 'Run Simulation' and click 'Run Behavioral Simulation'

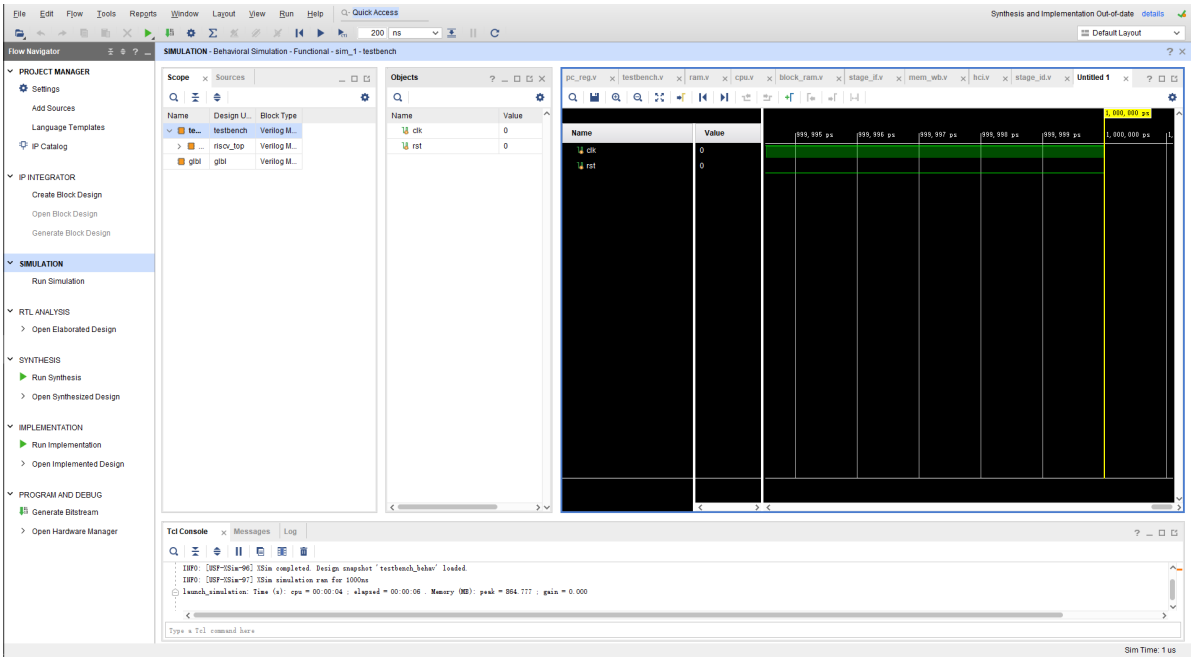


After this

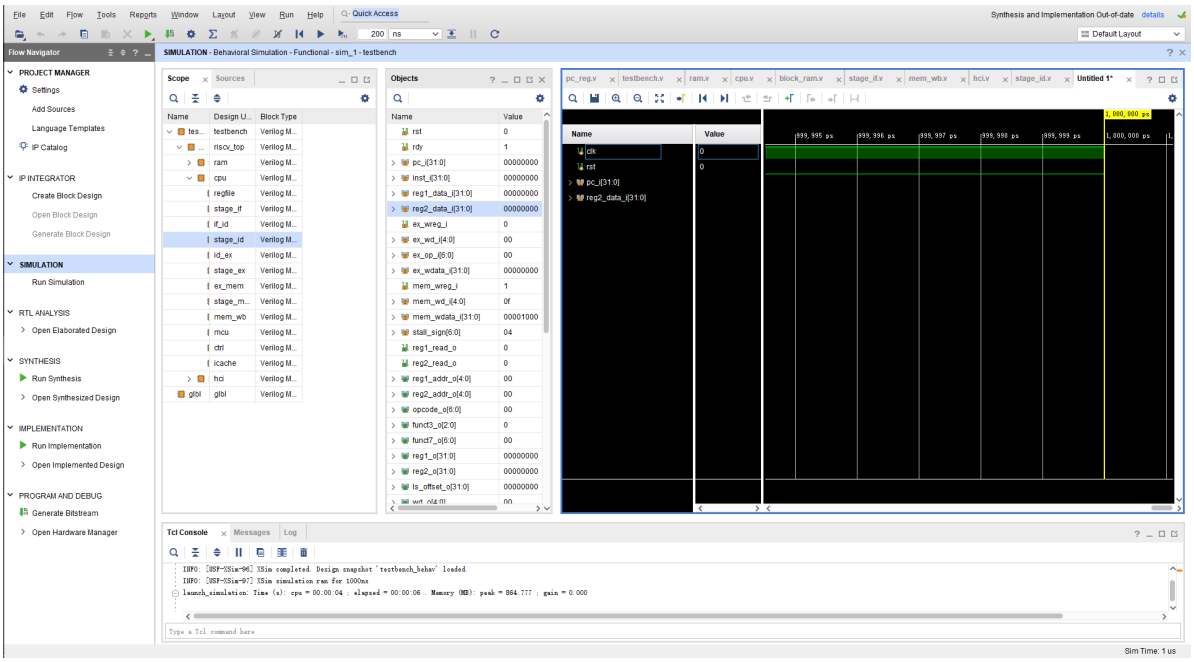


Simulation Layout

You may find Vivado change into simulation layout

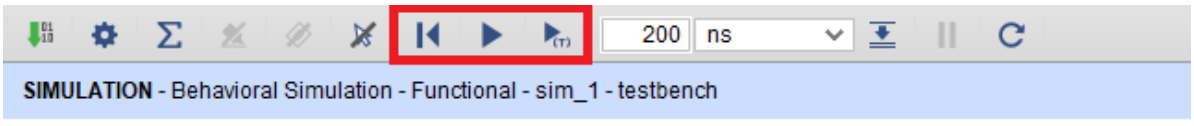


In which the right side is the wave panel, and you may drag some objects into it to watch their status. Also, you can right-click on these objects and then add them into wave panel.



Run Simulation

Notice these 3 buttons.

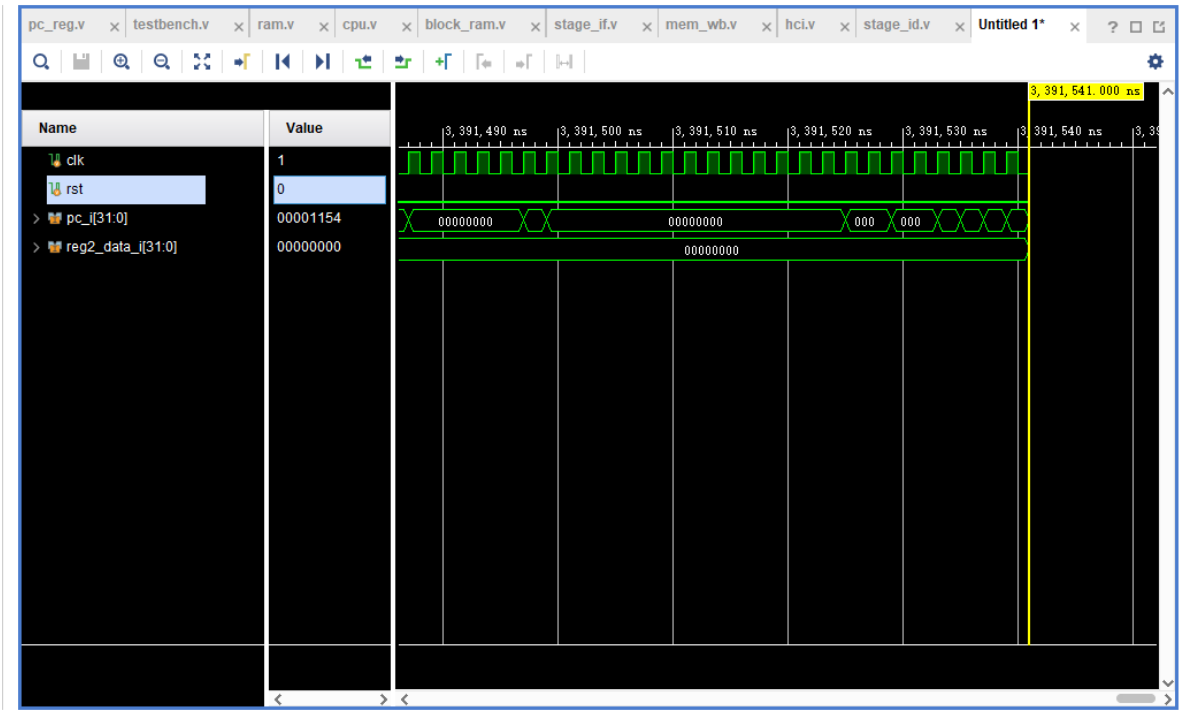


Among these 3, the first one is 'Restart', the middle one is 'Run All' and the last one is 'Run For *'.

Here 'Run All' means simulating until the program stops. As for 'Run For *', you can control the simulation time by changing the '200' in this photo, for example, to '2000' or changing 'ns' to 's'.

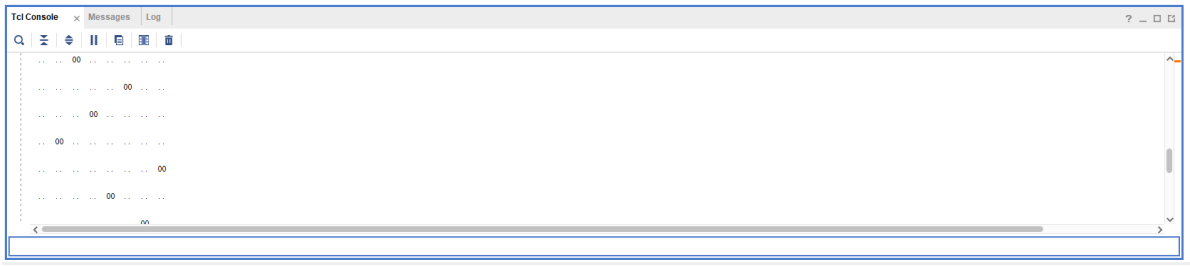
Running status

During simulation, the wave panel is like this



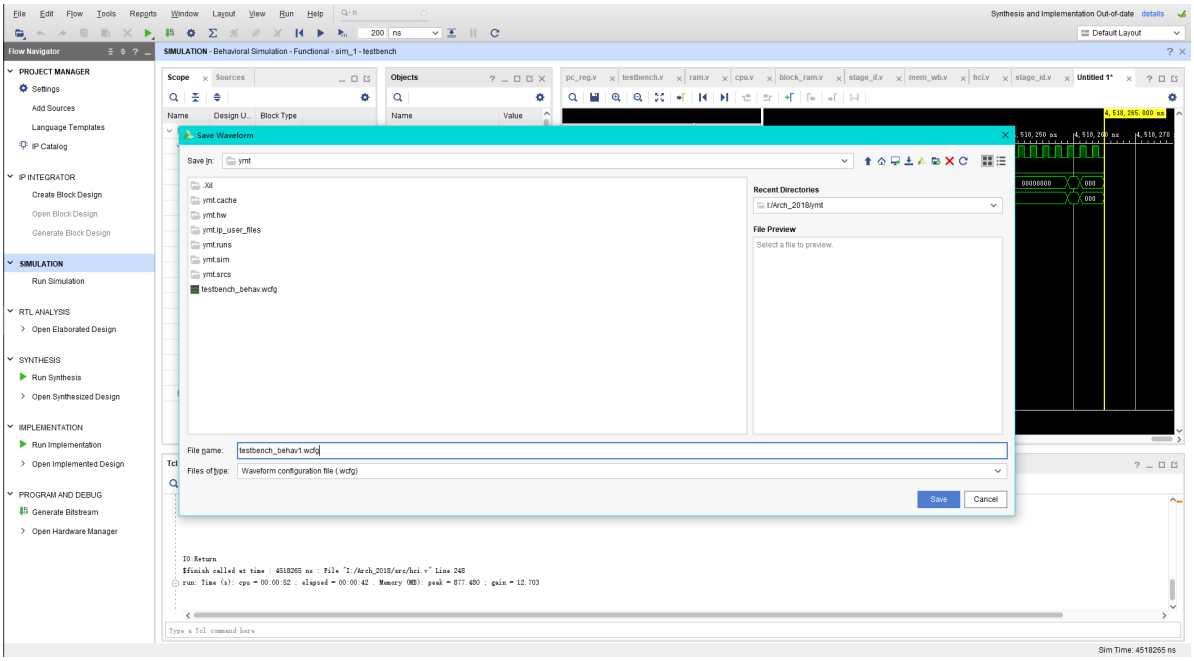
Where you can see the value(decimal by default) of some signals, and their changes.

And you can see the output in TCL console.



Save your simulation

In order to store SIMULATION status for the next use, you can just save this simulation by pressing 'CTRL+S'.



Then you will get rid of trouble of adding all signals you want to watch when you start another simulation.

Run on FPGA

Install RISCV-Toolchain

You can compile the toolchain as the project README says.

Or you can just download the compiled toolchain and extract it anywhere you feel happy to place it and modify the prefix dir in 'build_test.sh'.

MS108-2020 > riscv > build_test.sh

```
#!/bin/sh
set -e
prefix='/mnt/g/courses/Arch/riscv-tool-chain/riscv-tool-chain/riscv'
```

Install serial

You need to install following package listed in the official github page before 'make' .

Dependencies

Required:

- catkin - cmake and Python based buildsystem
- cmake - buildsystem
- Python - scripting language
  - empy - Python templating library
  - catkin_pkg - Runtime Python library for catkin

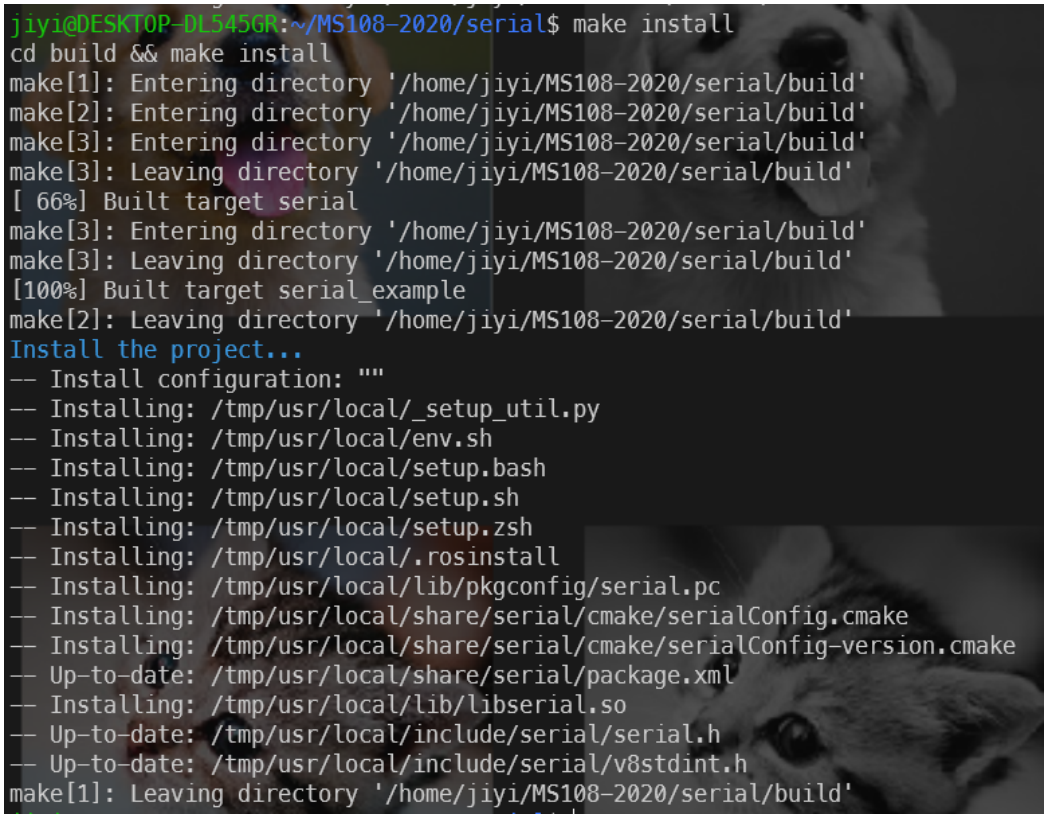
You can use following script to install these dependencies.

```
# for catkin
sudo apt install python-catkin-tools catkin

# for cmake
sudo apt install cmake

# for python
# WSL has python3 within it

pip install empy catkin_pkg
```

Then:

```
cd serial
make
make install
```

You should see something like this



Modify .sh files

Give execute and read permission to all .sh files in the project folder by:

```
sudo chmod +x *.sh
```

Build controller in directory 'ctrl'

```
./build.sh
```

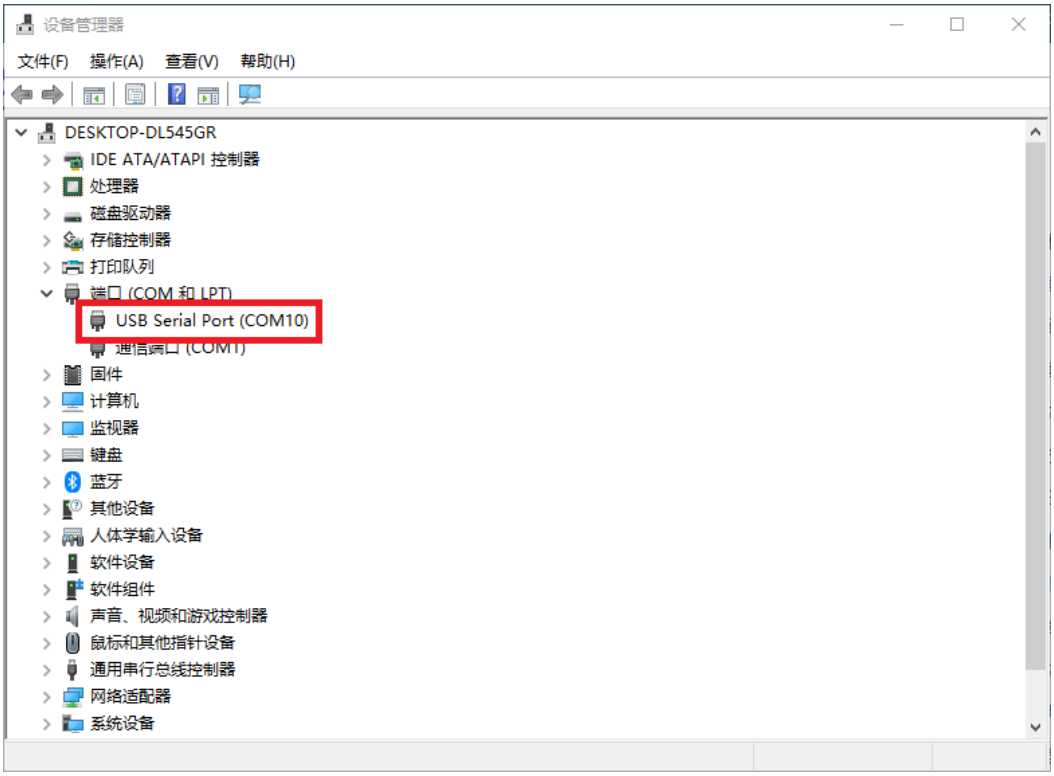
Then modify 'run_test_fpga.sh'.

You should modify the port number

```
./ctrl/build.sh
./ctrl/run.sh ./test/test.bin ./test/test.in /dev/ttyS10 -I
```

You can see the port number in device manager on Windows:

Note: port number may change if you use different FPGA port, so you should use the port number you find in device manager after connecting FPGA with your computer



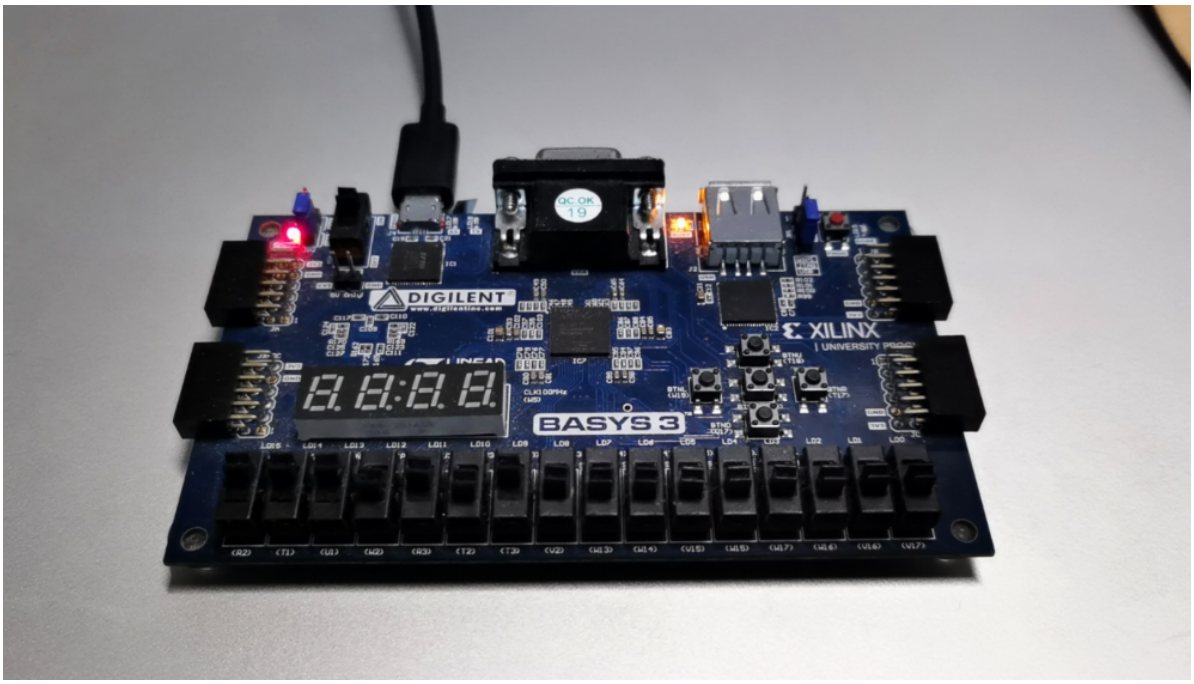
Run test by

```
./run_test_fpga.sh testname
```

Connect FPGA with computer

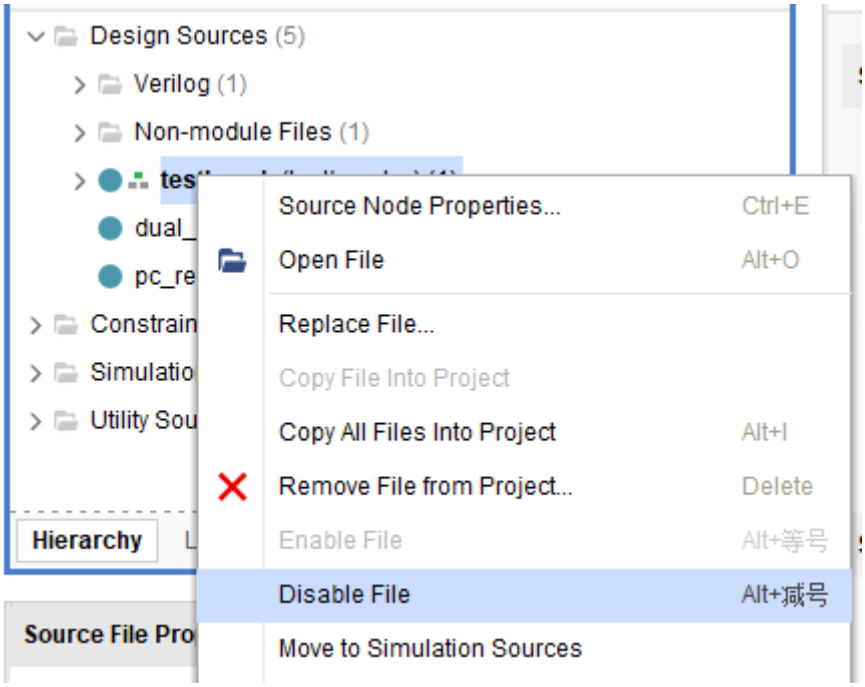
To connect FPGA, you only need a USB-A to micro-USB cable (the same cable Kindle use).

You should see power led on and a Breathing LED blinking after connecting it with your computer.

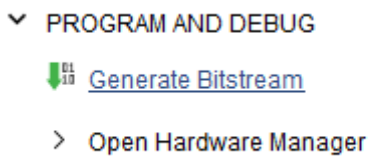


Generate Bitstream

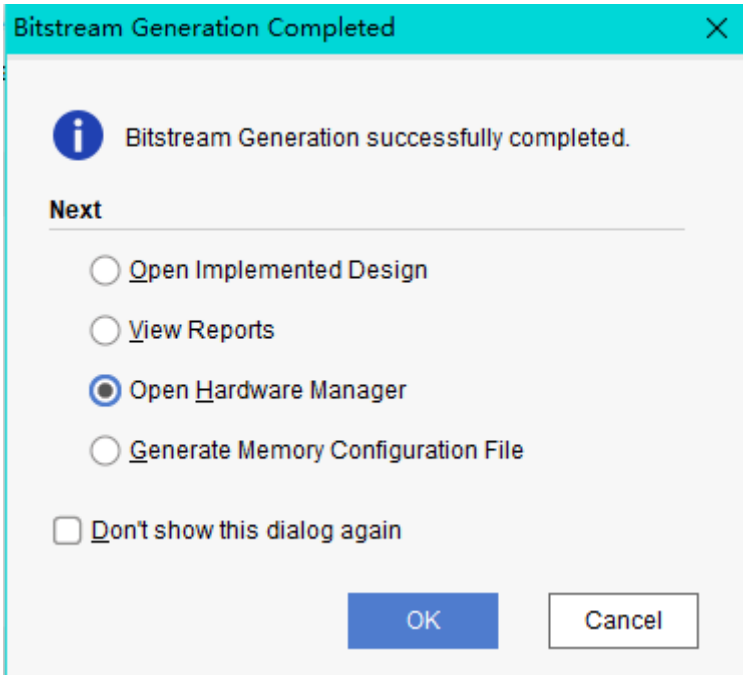
You should first disable testbench.v to run implementation.



Then generate bitstream

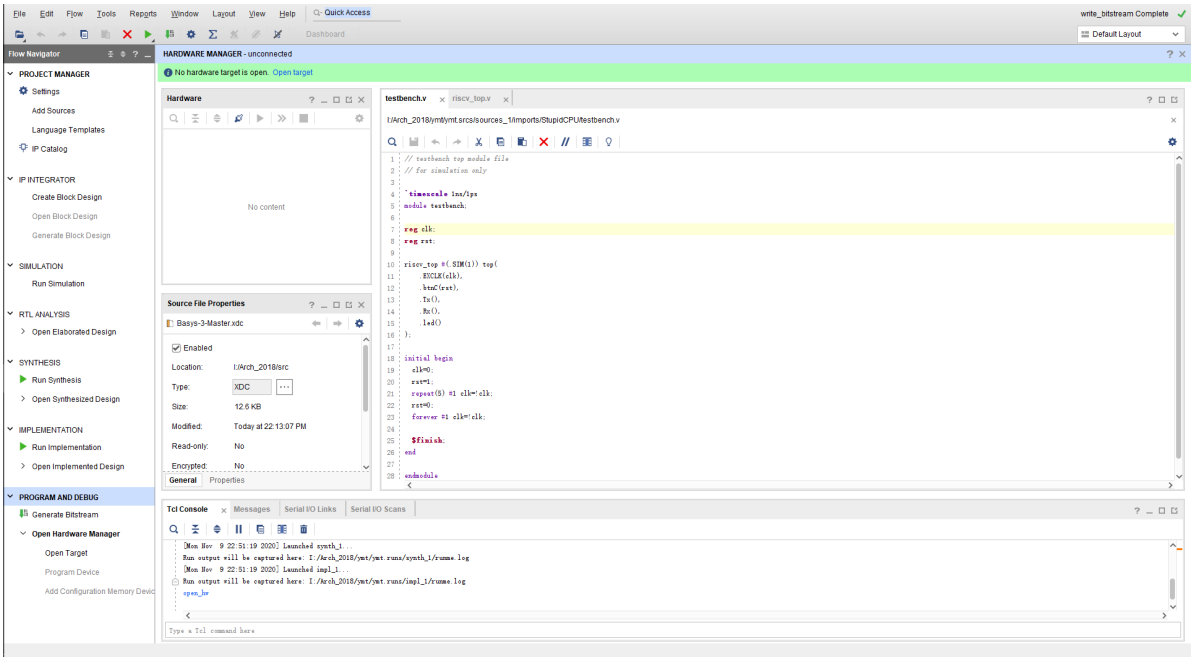


After several minutes, you will see

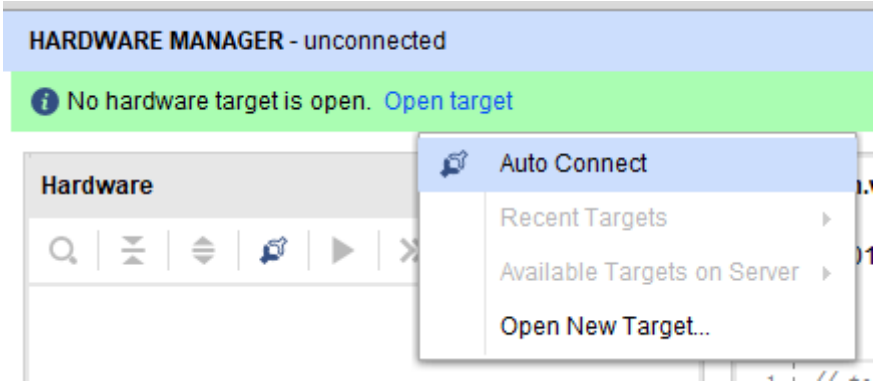


Program device

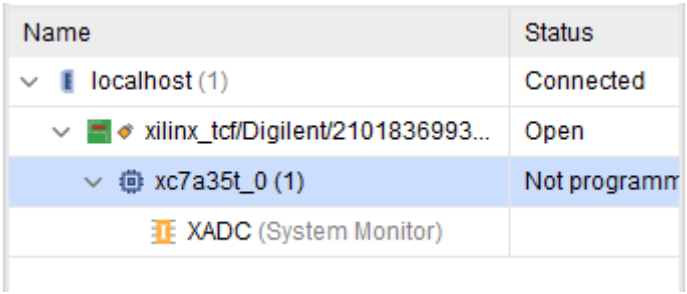
Open hardware manager, and you see hardware layout



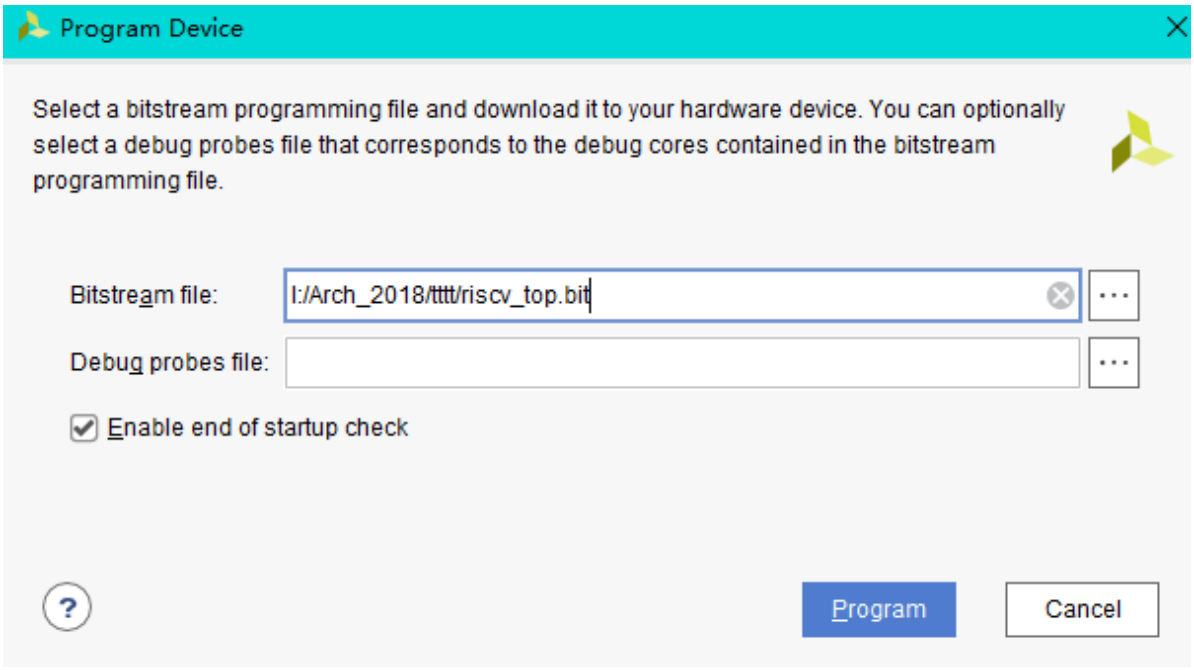
Connect Vivado with your FPGA



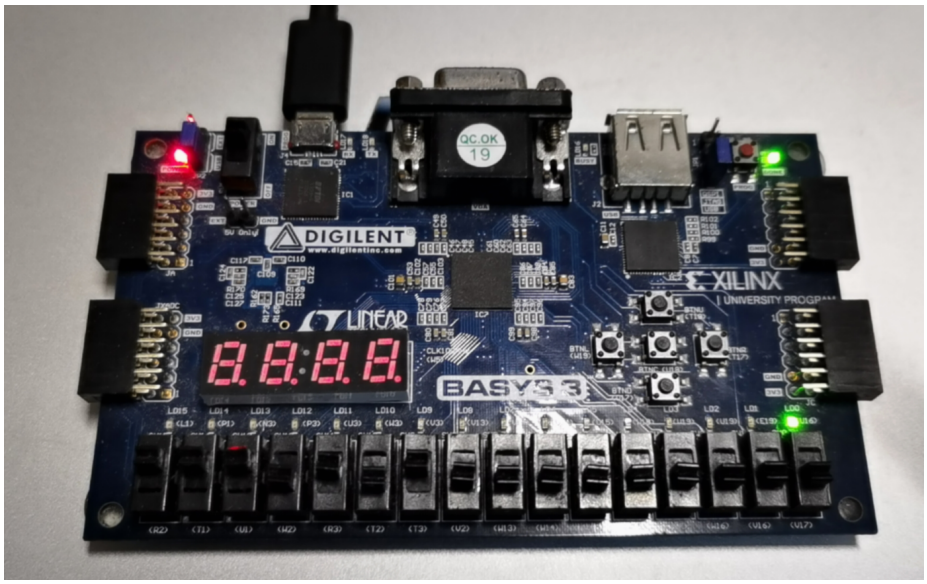
Seeing your device connected, you can program it



Click on program device and choose the .bit file you want to download



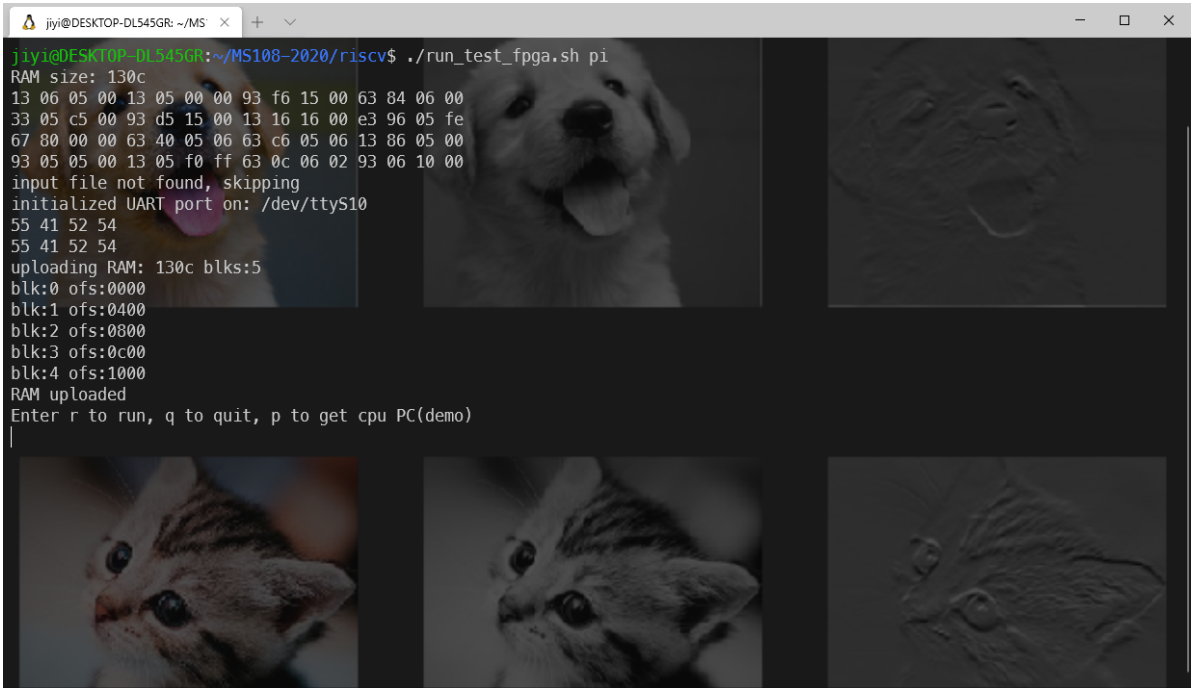
Program done, the breathing LED stops blinking



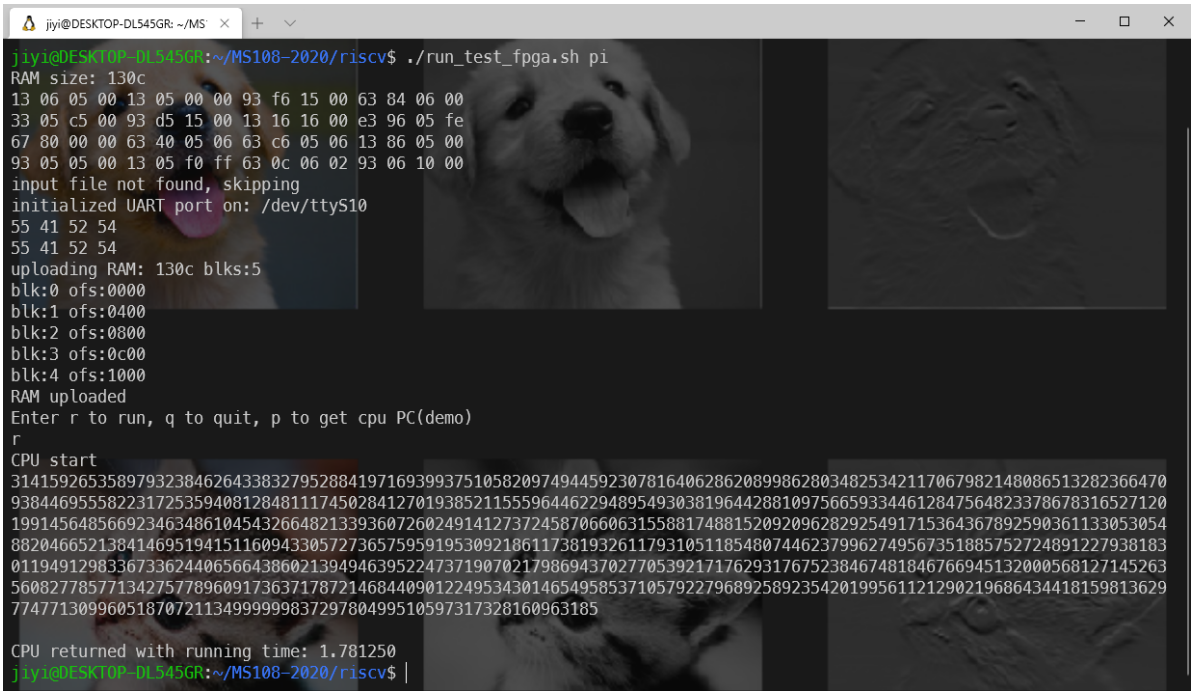
Then you can run test.

Run test

After programming the FPGA, run 'run_test_fpga.sh', we run pi.c as an example

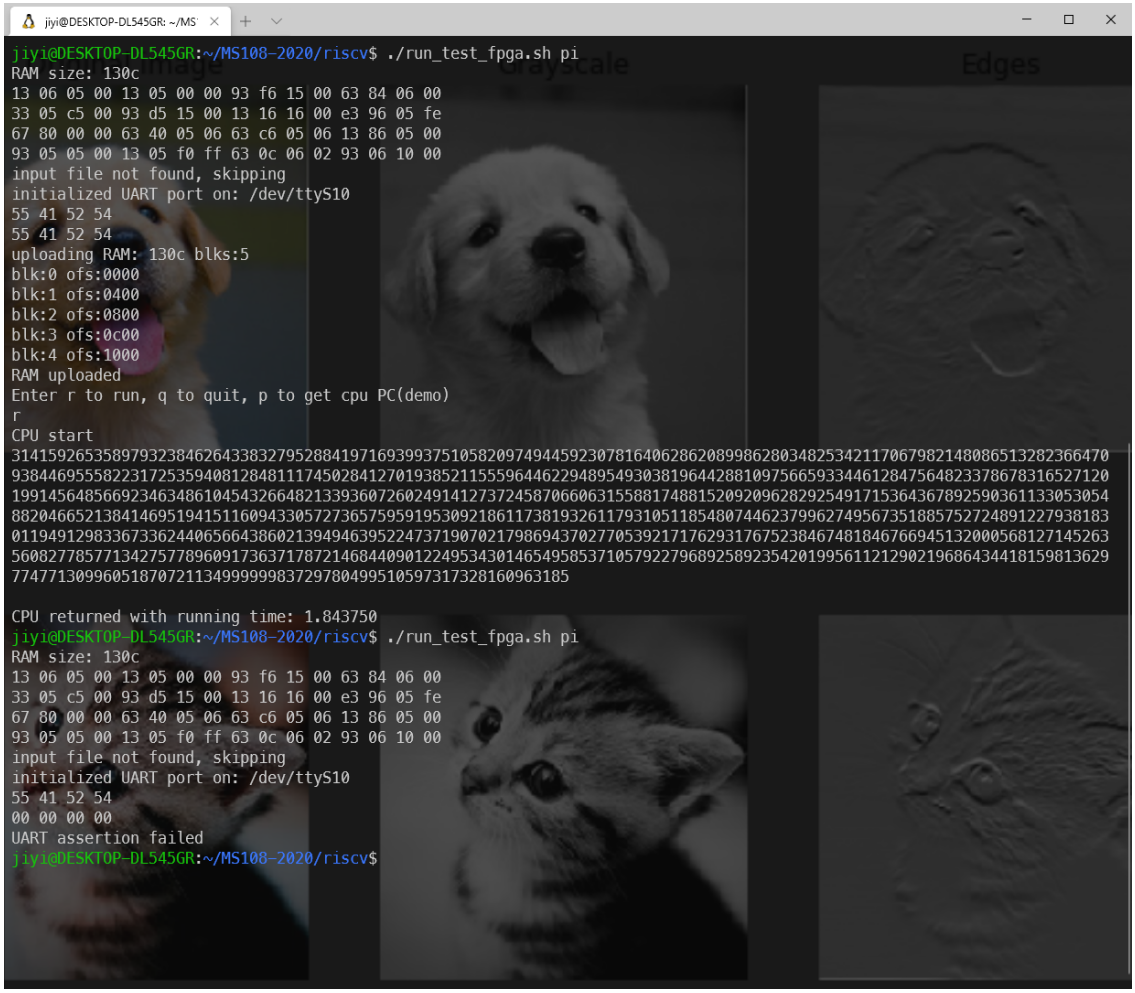


You can see that the test program has been uploaded to FPGA and CPU is paused waiting to run, press 'r' and you will see the output in terminal.



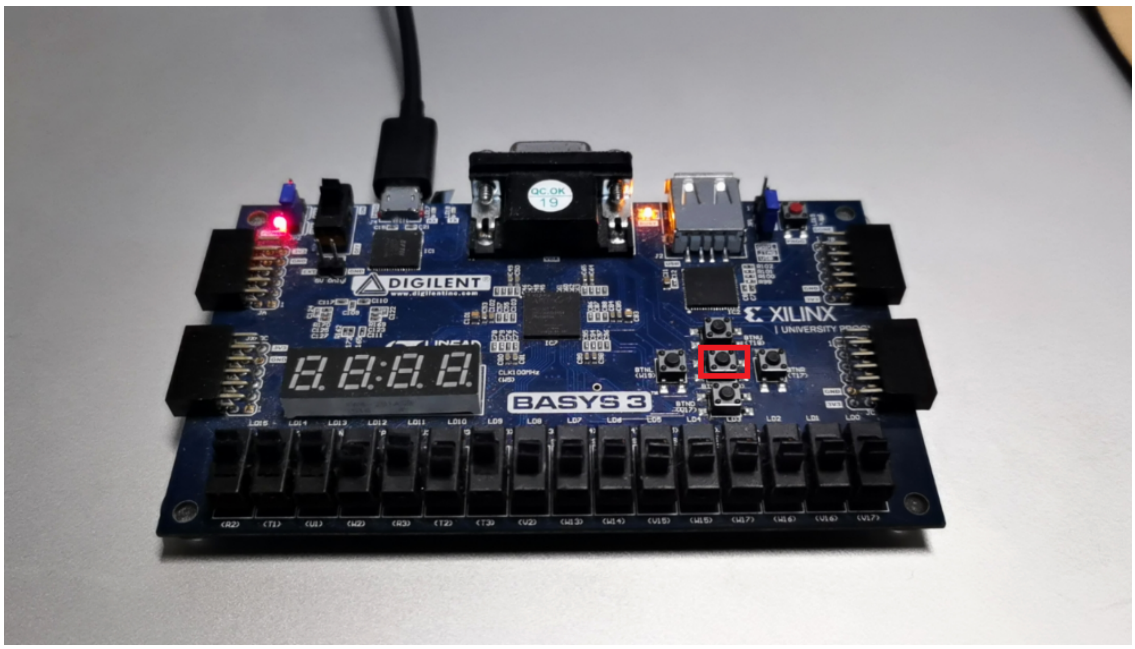
Other little notes on FPGA running

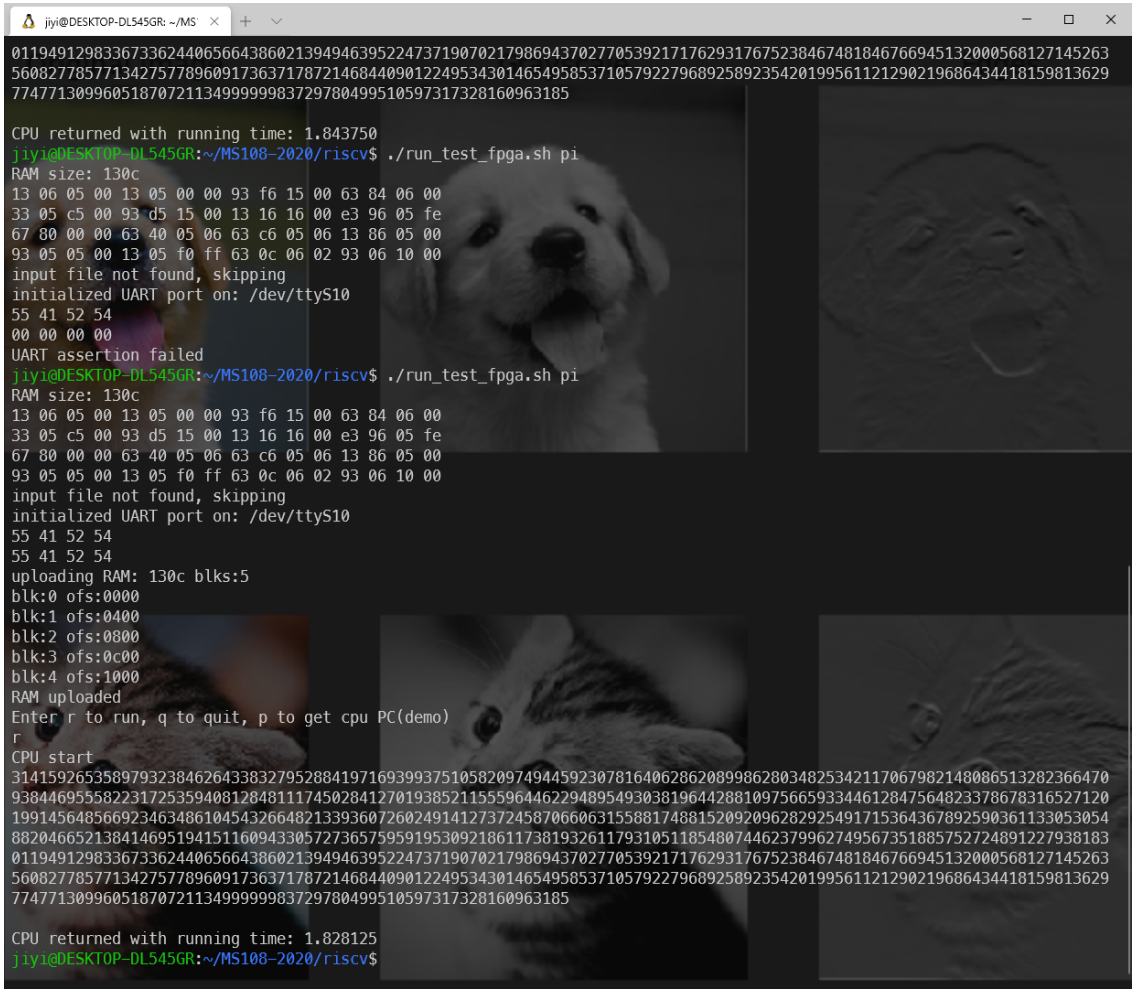
- Once you programmed your CPU, you can run multiple tests. But if you directly run another test, uploading fails like this



Seeing this, you should press a memory reset button.

Then, you can run another test without programming the device again.




jiyi@DESKTOP-DL545GR: ~/MS
01194912983367336244065664386021394946395224737190702179869437027705392171762931767523846748184676694513200056812714526356082778577134275778960917363717872146844090122495343014654958537105792279689258923542019956112129021968643441815981362977477130996051870721134999999837297804995105973173281609631850
CPU returned with running time: 1.843750
jiyi@DESKTOP-DL545GR:~/MS108-2020/riscv$ ./run_test_fpga.sh pi
RAM size: 130c
13 06 05 00 13 05 00 00 93 f6 15 00 63 84 06 00
33 05 c5 00 93 d5 15 00 13 16 16 00 e3 96 05 fe
67 80 00 00 63 40 05 06 63 c6 05 06 13 86 05 00
93 05 05 00 13 05 f0 ff 63 0c 06 02 93 06 10 00
input file not found, skipping
initialized UART port on: /dev/ttyS10
55 41 52 54
00 00 00 00
UART assertion failed
jiyi@DESKTOP-DL545GR:~/MS108-2020/riscv$ ./run_test_fpga.sh pi
RAM size: 130c
13 06 05 00 13 05 00 00 93 f6 15 00 63 84 06 00
33 05 c5 00 93 d5 15 00 13 16 16 00 e3 96 05 fe
67 80 00 00 63 40 05 06 63 c6 05 06 13 86 05 00
93 05 05 00 13 05 f0 ff 63 0c 06 02 93 06 10 00
input file not found, skipping
initialized UART port on: /dev/ttyS10
55 41 52 54
55 41 52 54
uploading RAM: 130c blks:5
blk:0 ofs:0000
blk:1 ofs:0400
blk:2 ofs:0800
blk:3 ofs:0c00
blk:4 ofs:1000
RAM uploaded
Enter r to run, q to quit, p to get cpu PC(demo)
r
CPU start
31415926535897932384626433832795288419716939937510582097494459230781640628620899862803482534211706798214808651328236647093844695558223172535940812848111745028412701938521155596446229489549303819644288109756659334461284756482337867831652712019914564856692346348610454326648213393607260249141273724587066063155881748815209209628292549171536436789259036113305305488204665213841469519415116094330572736575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664386021394946395224737190702179869437027705392171762931767523846748184676694513200056812714526356082778577134275778960917363717872146844090122495343014654958537105792279689258923542019956112129021968643441815981362977477130996051870721134999999837297804995105973173281609631850
CPU returned with running time: 1.828125
jiyi@DESKTOP-DL545GR:~/MS108-2020/riscv$