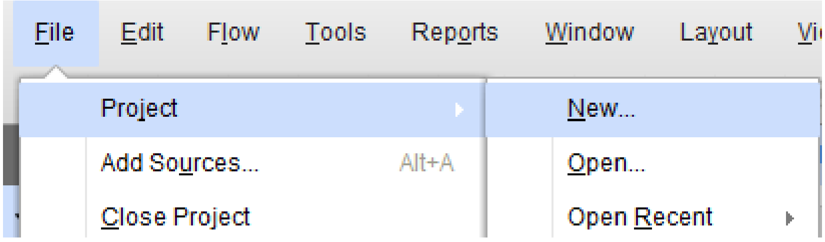
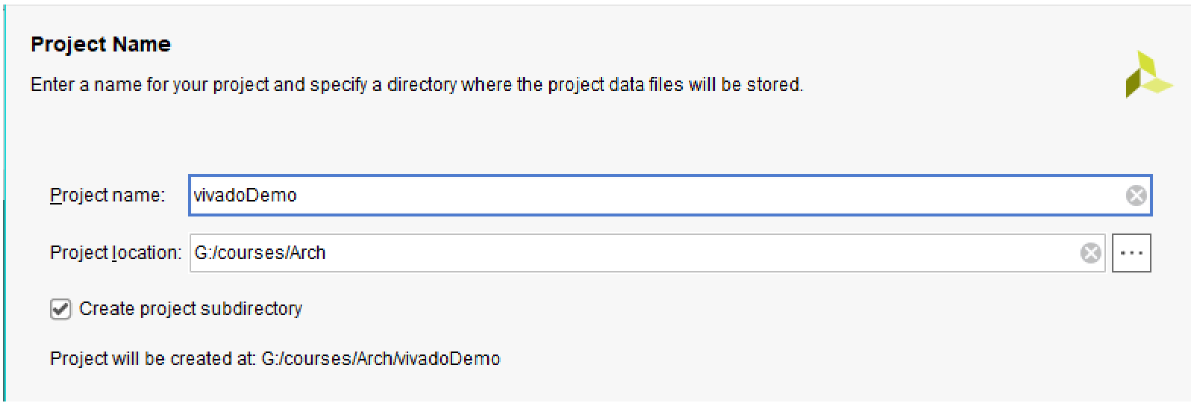
Use Vivado Step by Step

Create a project

New project



Define project name



Project Name

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

Project name: vivadoDemo

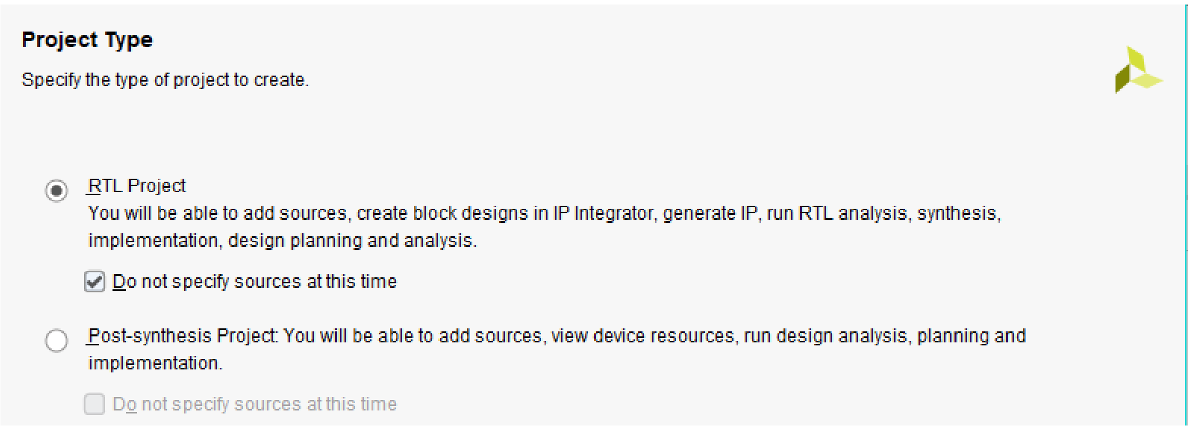
Project location: G:/courses/Arch

☑ Create project subdirectory

Project will be created at: G:/courses/Arch/vivadoDemo

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



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.

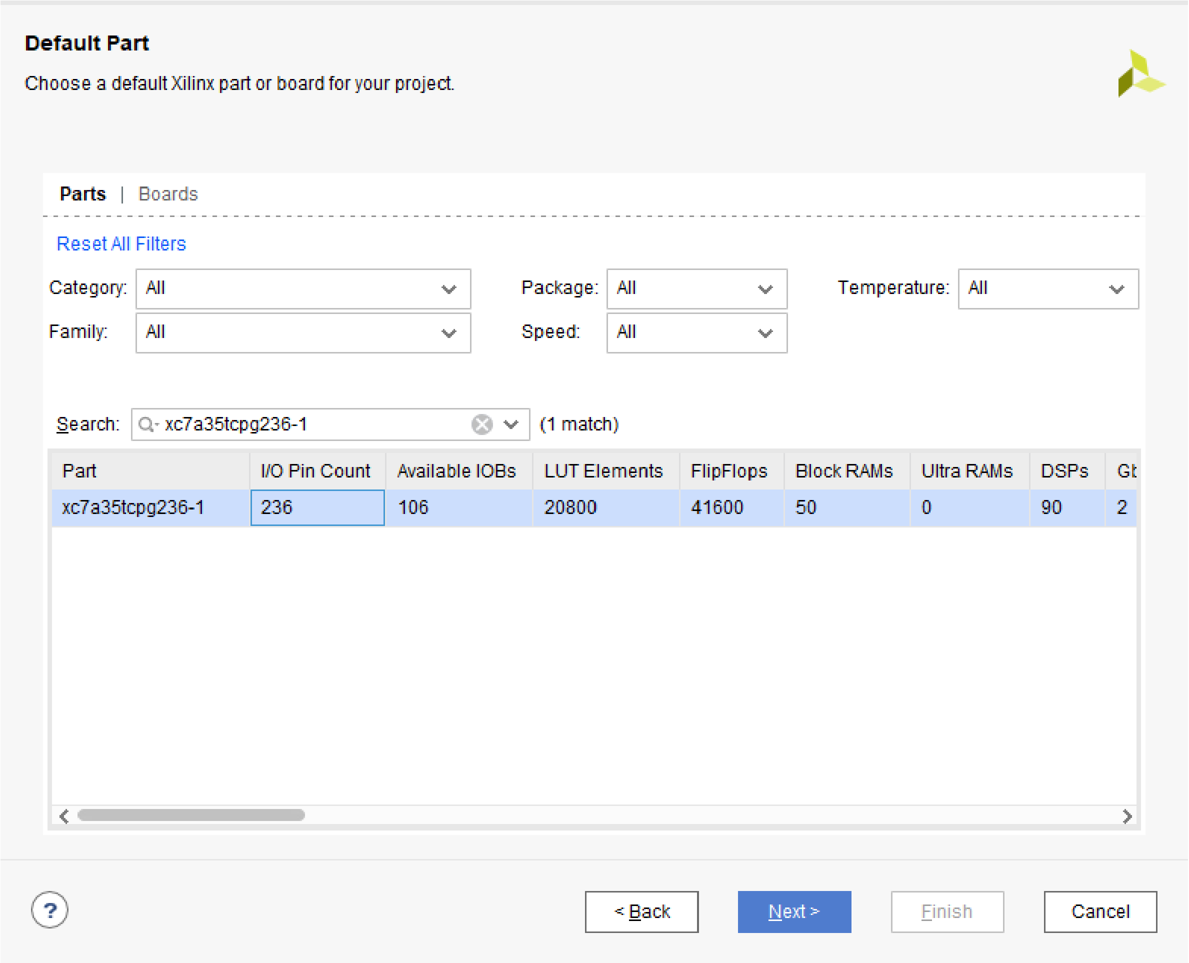
☑ Do not specify sources at this time

○ 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

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)



Default Part

Choose a default Xilinx part or board for your project.

Parts | Boards

Reset All Filters

Category: All

Package: All

Temperature: All

Family: All

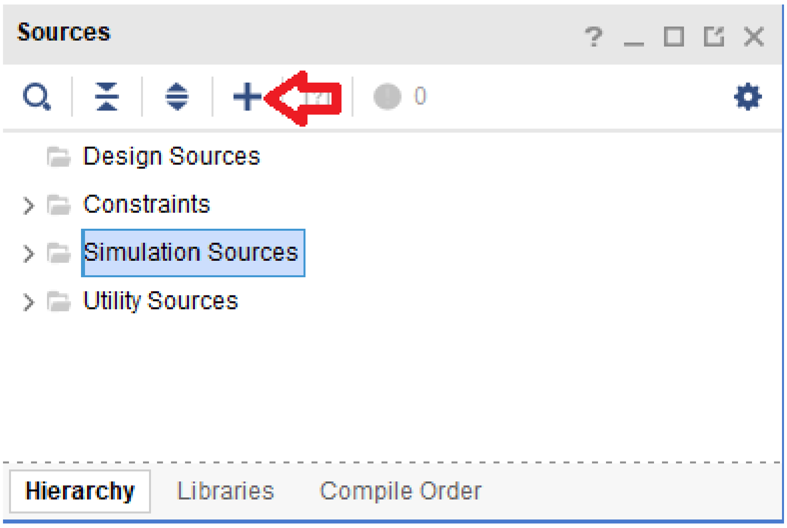
Speed: All

Search: xc7a35tcpg236-1 (1 match)

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7a35tcpg236-1	236	106	20800	41600	50	0	90	2

Add provided code and other resources

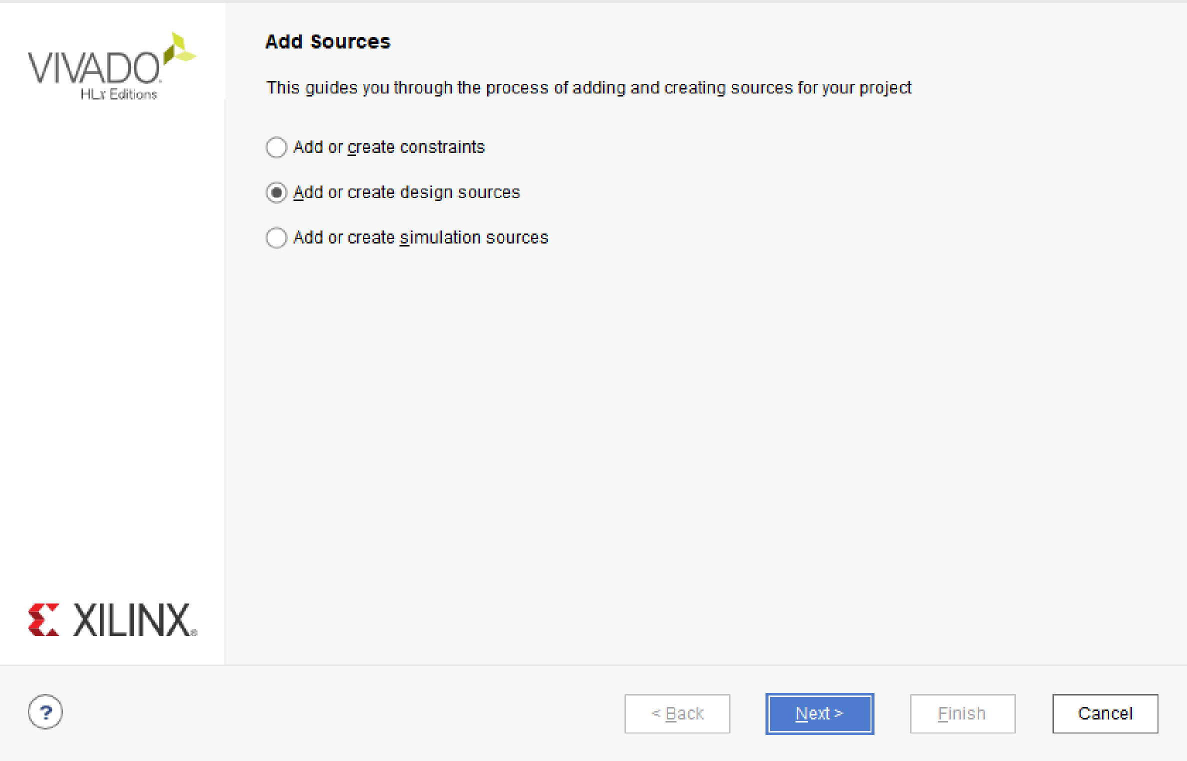
Click on the '+' button



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 Sources

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

○ Add or create constraints

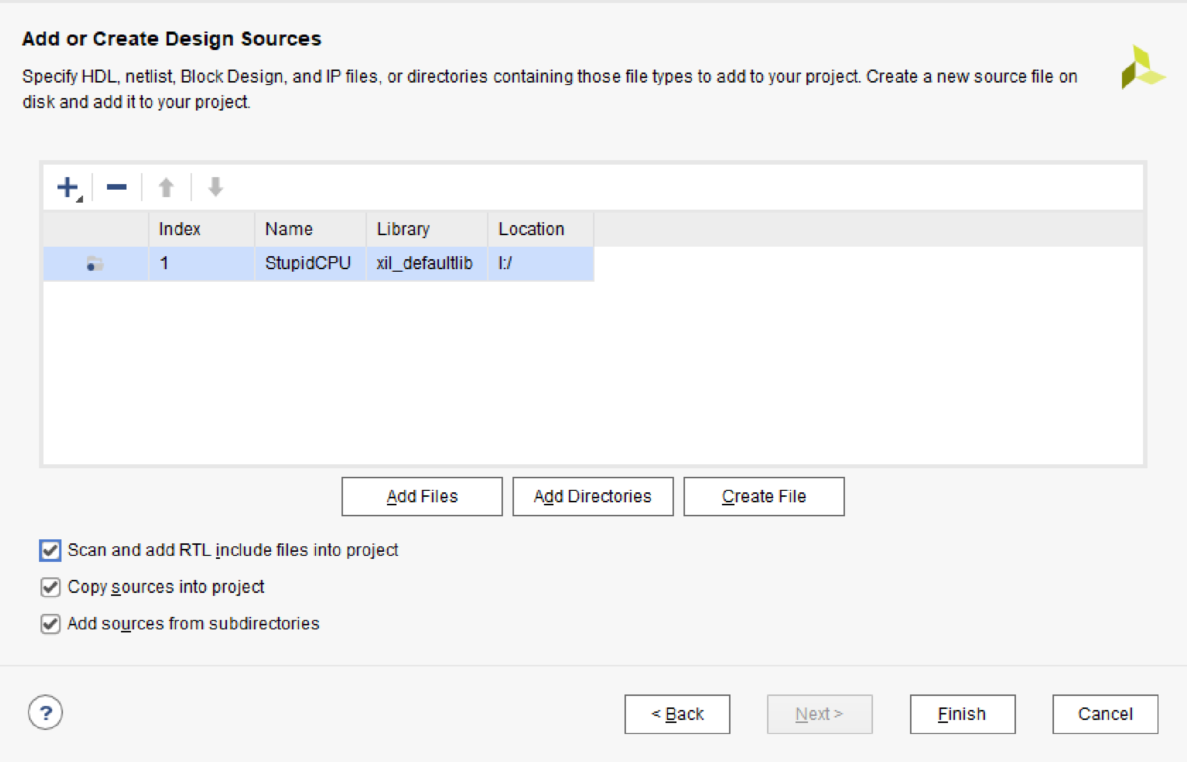
● Add or create design sources

○ Add or create simulation sources

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.



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	StupidCPU	xil_defaultlib	I:/

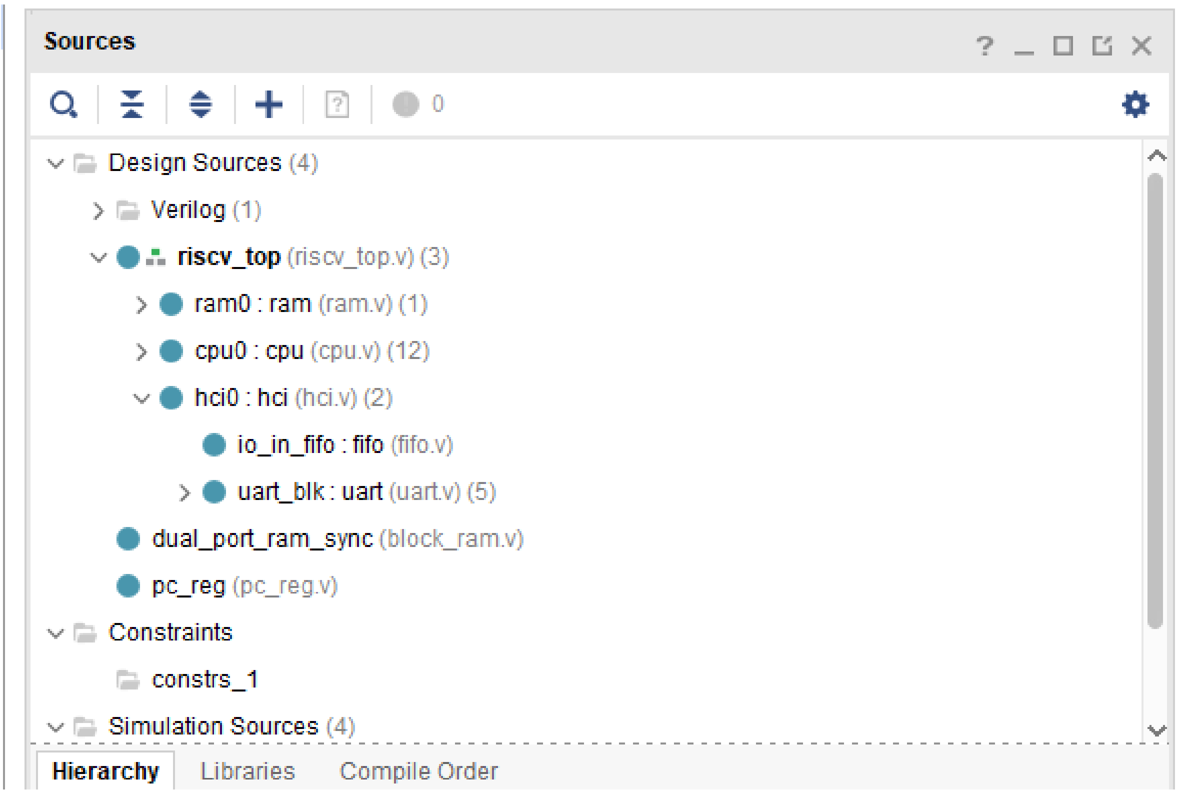
☑ Scan and add RTL include files into project

☑ Copy sources into project

☑ Add sources from subdirectories

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.



Sources

Design Sources (4)
- Verilog (1)
- riscv_top (riscv_top.v) (3)
  - ram0 : ram (ram.v) (1)
  - cpu0 : cpu (cpu.v) (12)
  - hci0 : hci (hci.v) (2)
    - io_in_fifo : fifo (fifo.v)
    - uart_blk : uart (uart.v) (5)
- dual_port_ram_sync (block_ram.v)
- pc_reg (pc_reg.v)

Constraints
- constrs_1

Simulation Sources (4)

Run SIMULATION

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

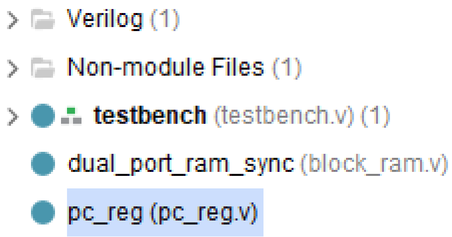
Note: Passing simulation doesn't necessarily mean your design is workable on FPGA.

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.

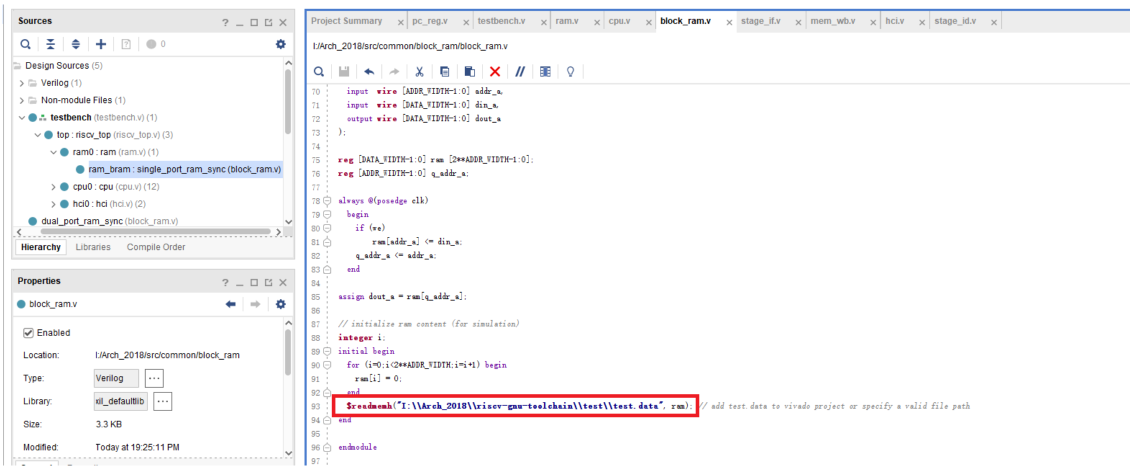
  If you want to test your design on FPGA, you must disable 'testbench.v' since it can't be synthesised.

  - Verilog (1)
  - Non-module Files (1)
  - testbench (testbench.v) (1)
  - dual_port_ram_sync (block_ram.v)
  - pc_reg (pc_reg.v)

- 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.



☑ Enabled

Location: I:/Arch_2018/src/common/block_ram

Type: Verilog

Library: xil_defaultlib

Size: 3.3 KB

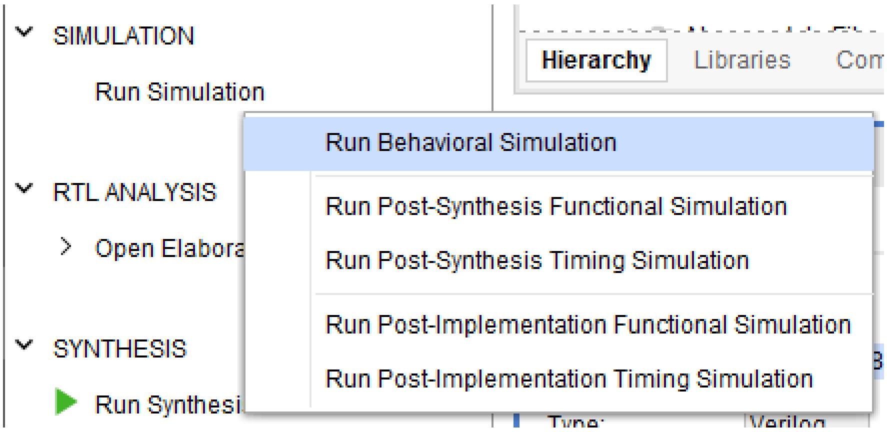
Modified: Today at 19:25:11 PM

I:/Arch_2018/src/common/block_ram/block_ram.v

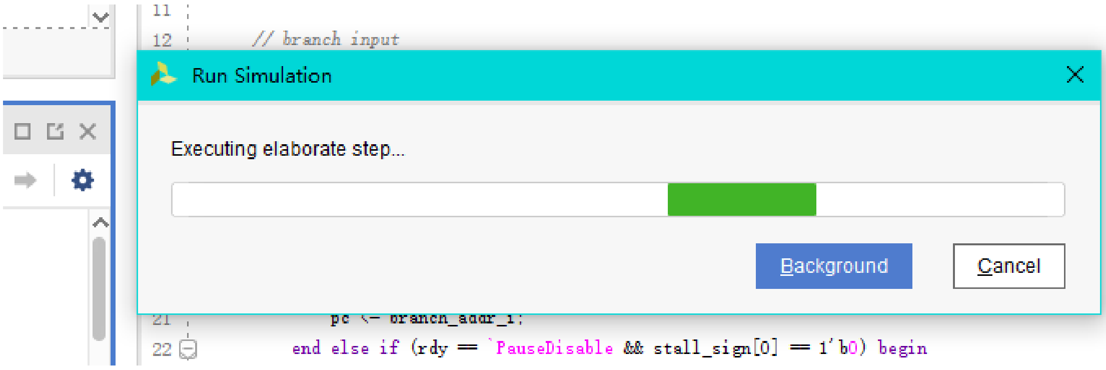
```
  input  wire [ADDR_WIDTH-1:0] addr_a,
  input  wire [DATA_WIDTH-1:0] din_a,
  output wire [DATA_WIDTH-1:0] dout_a
);

reg [DATA_WIDTH-1:0] ram [2**ADDR_WIDTH-1:0];
reg [ADDR_WIDTH-1:0] q_addr_a;

always @(posedge clk)
  begin
    if (we)
        ram[addr_a] <= din_a;
    q_addr_a <= addr_a;
  end

assign dout_a = ram[q_addr_a];

// initialize ram content (for simulation)
integer i;
initial begin
  for (i=0;i<2**ADDR_WIDTH;i=i+1) begin
    ram[i] = 0;
  end
  $readmemh("I:\\Arch_2018\\riscv-gnu-toolchain\\test\\test.data", ram); // add test.data to vivado project or specify a valid file path
end

endmodule
```

Start simulation

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


SIMULATION
Run Simulation
RTL ANALYSIS
Open Elabora
SYNTHESIS
Run Synthesi
Hierarchy
Libraries
Com
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation
Type:
Verilog

After this

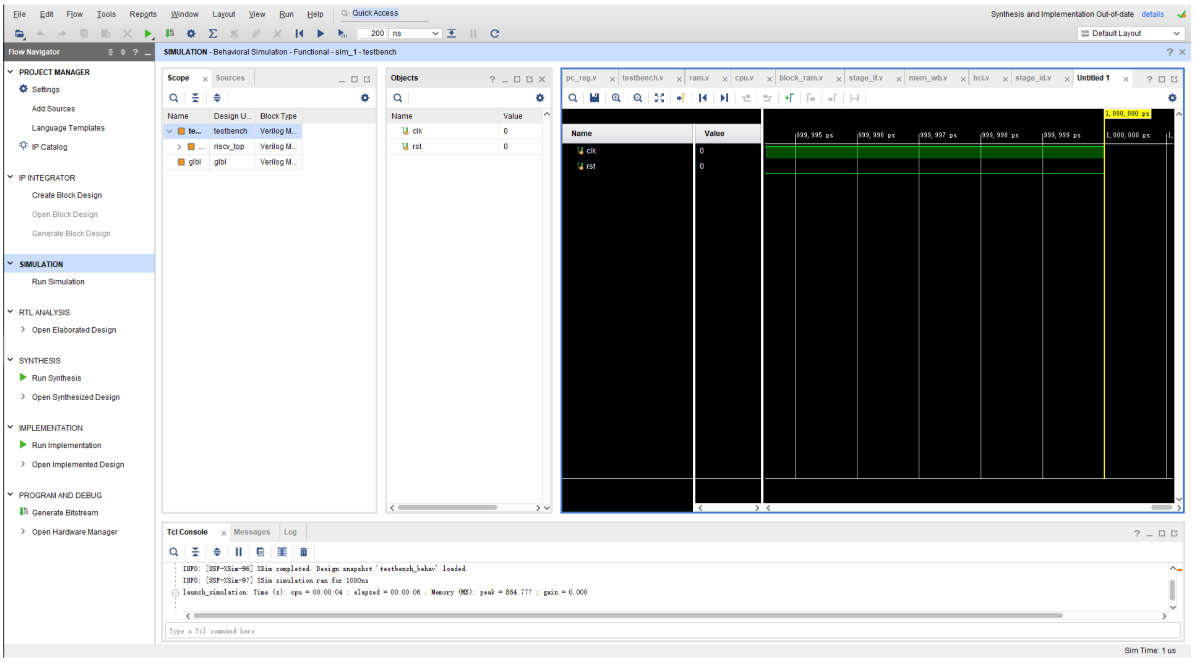


Executing elaborate step...

```
        // branch input
                pc <= branch_addr_i;
            end else if (rdy == `PauseDisable && stall_sign[0] == 1'b0) begin
```

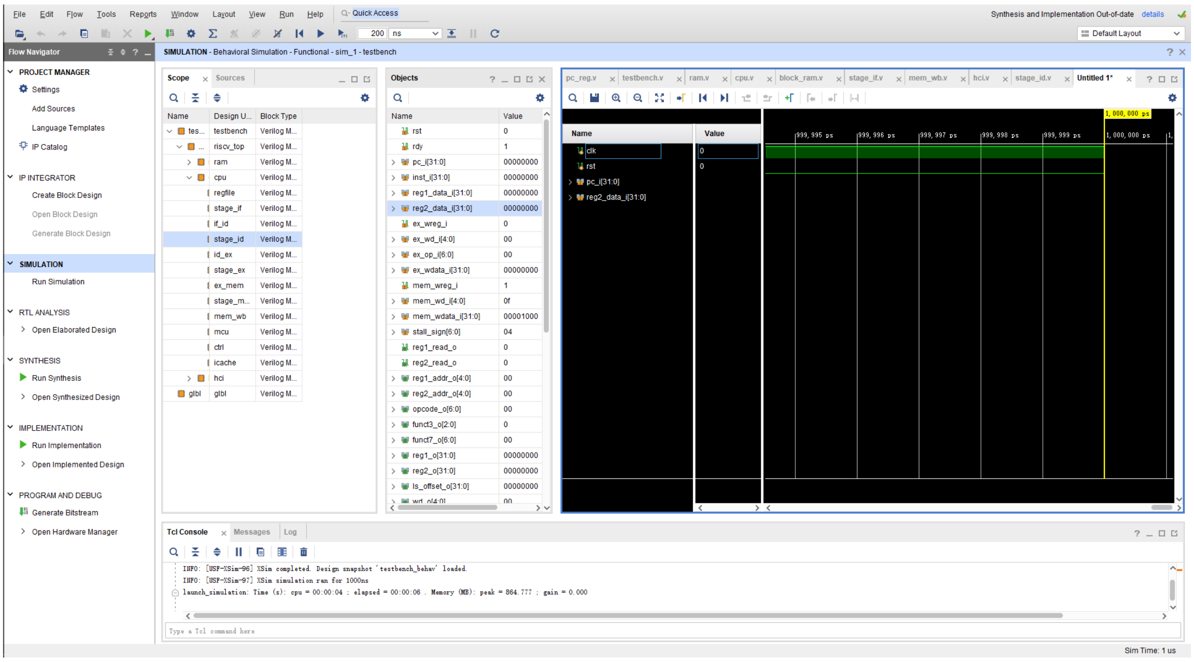
Simulation Layout

You may find Vivado change into simulation layout



```
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testbench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:06 . Memory (MB): peak = 864.777 ; gain = 0.000
```

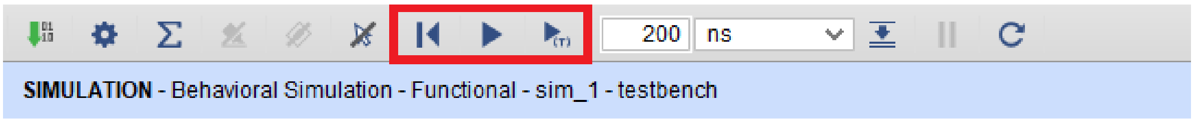
In which the right side is the wave panel, and you may drag some objects into it to watch their status. Also, you can right-click on these objects and then add them into wave panel.


SIMULATION - Behavioral Simulation - Functional - sim_1 - testbench
Scope
Objects
Untitled 1*
Tcl Console
Sim Time: 1 us

```
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testbench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:06 . Memory (MB): peak = 864.777 ; gain = 0.000
```

Run Simulation

Notice these 3 buttons.

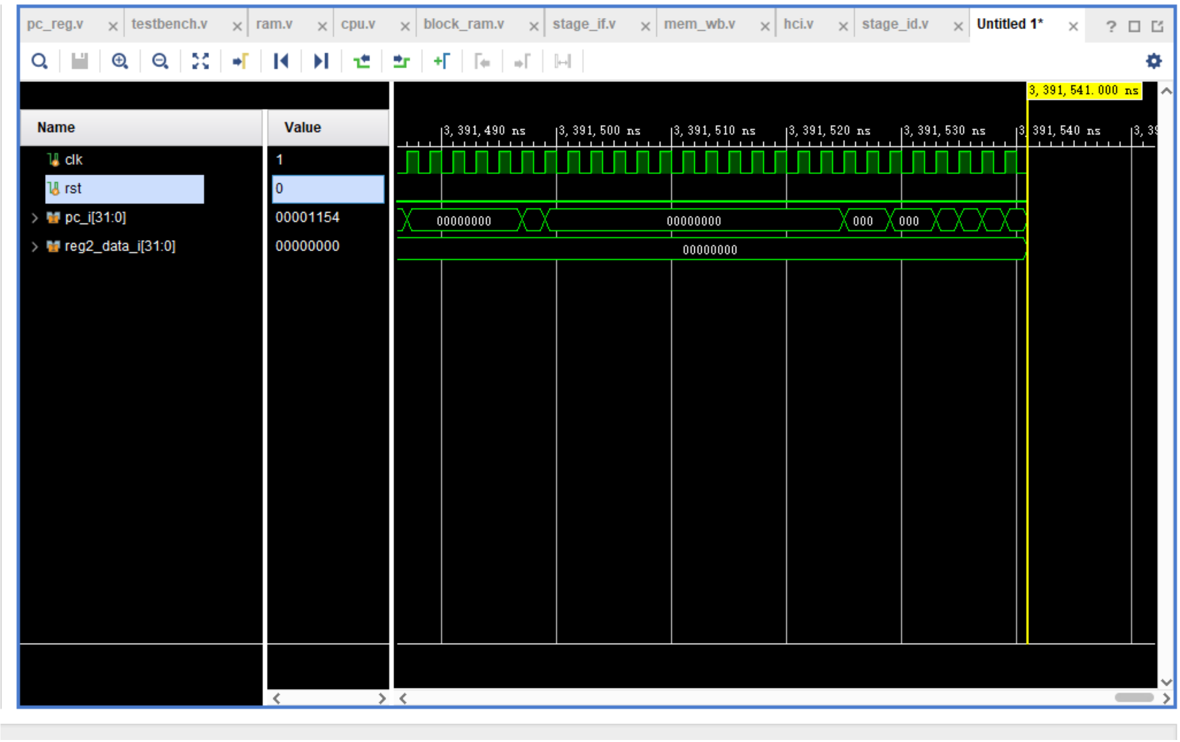

200
ns
SIMULATION - Behavioral Simulation - Functional - sim_1 - testbench

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

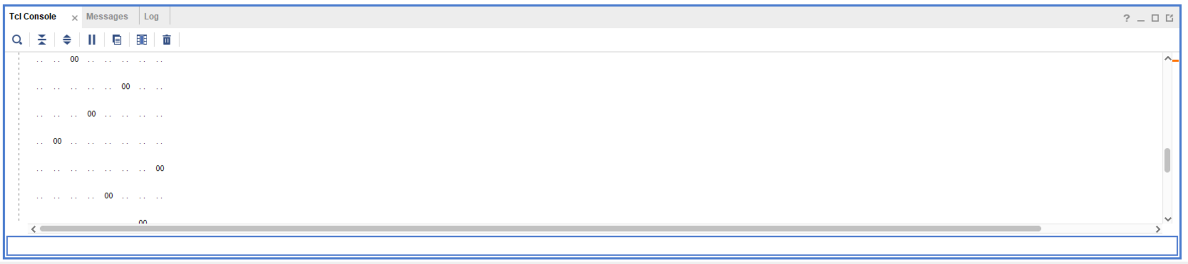
During simulation, the wave panel is like this



Name	Value
clk	1
rst	0
pc_i[31:0]	00001154
reg2_data_i[31:0]	00000000

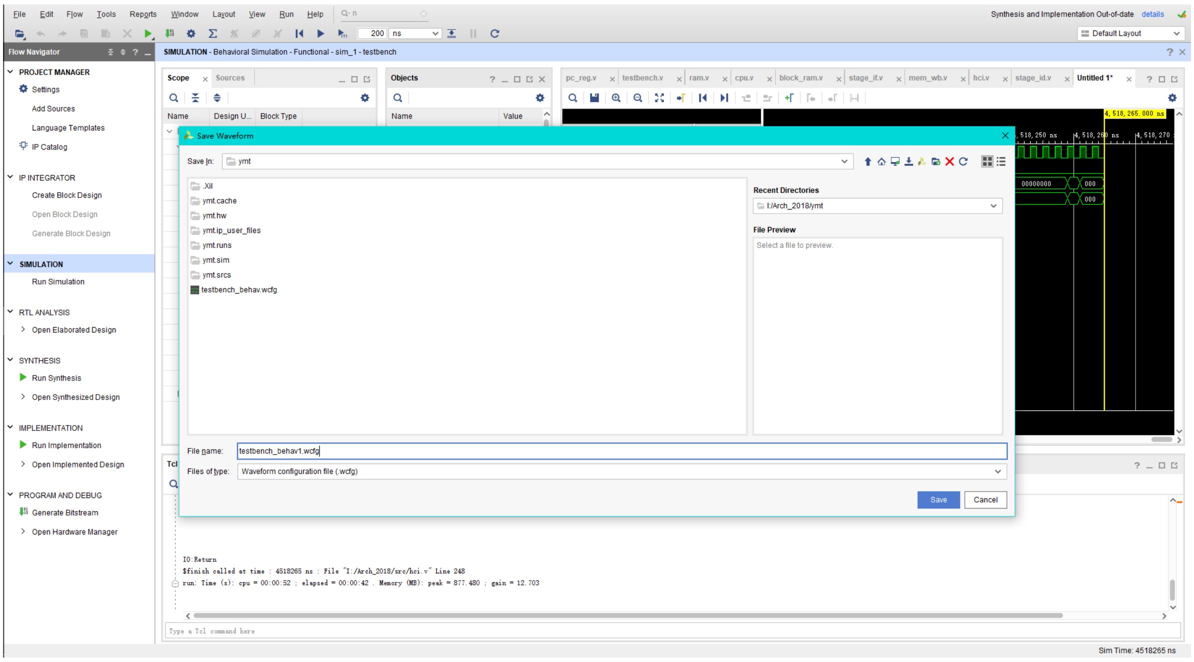
Where you can see the value(decimal by default) of some signals, and their changes.

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'.



SIMULATION - Behavioral Simulation - Functional - sim_1 - testbench

Save Waveform

Save In: ymt

.Xil
ymt.cache
ymt.hw
ymt.ip_user_files
ymt.runs
ymt.sim
ymt.srcs
testbench_behav.wcfg

Recent Directories: I:/Arch_2018/ymt

File Preview: Select a file to preview.

File name: testbench_behav1.wcfg

Files of type: Waveform configuration file (.wcfg)

Save Cancel

```
IO:Return
$finish called at time : 4518265 ns : File "I:/Arch_2018/src/hci.v" Line 248
run: Time (s): cpu = 00:00:52 ; elapsed = 00:00:42 . Memory (MB): peak = 877.480 ; gain = 12.703
```

Type a Tcl command here

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