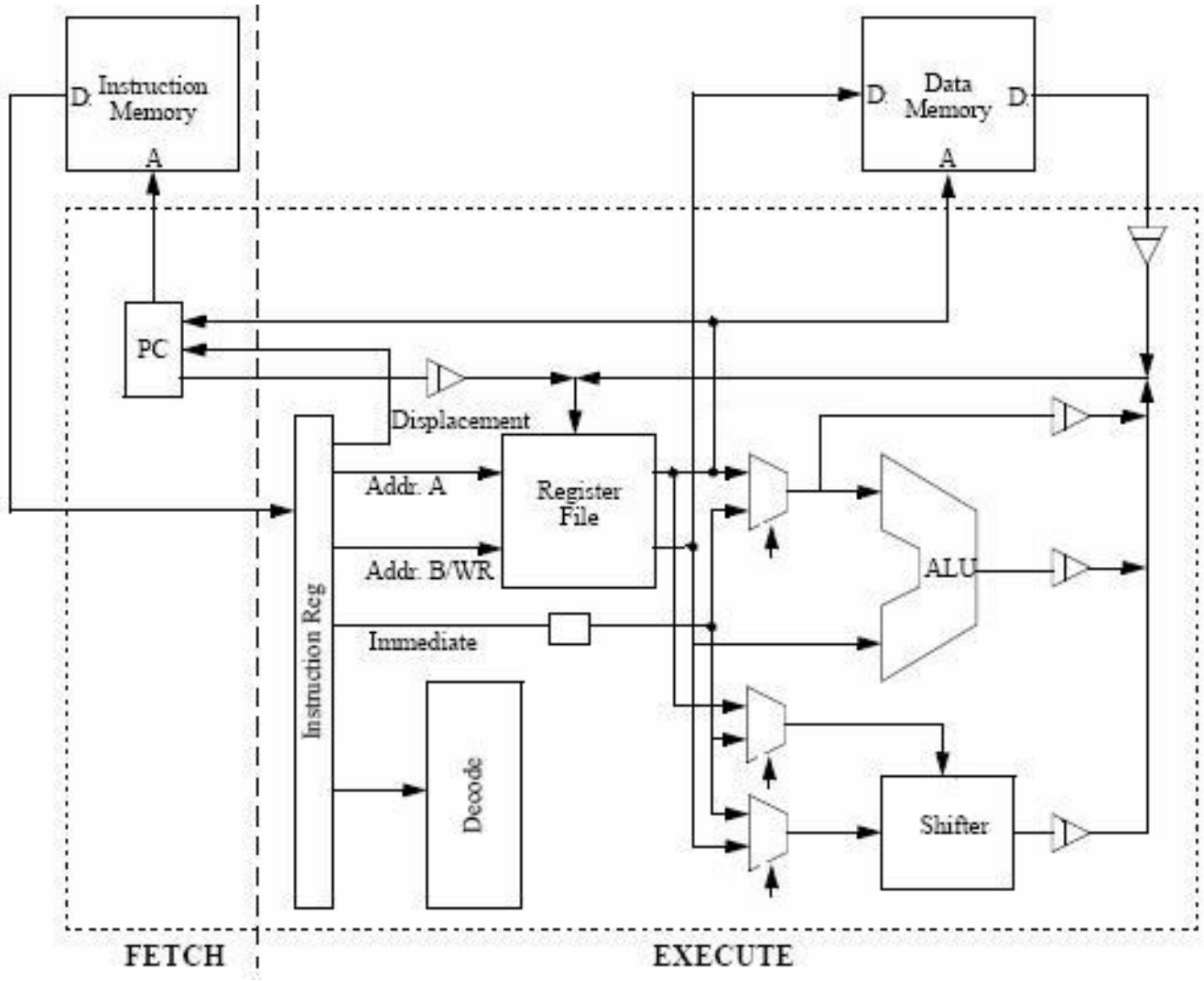
Project Introduction

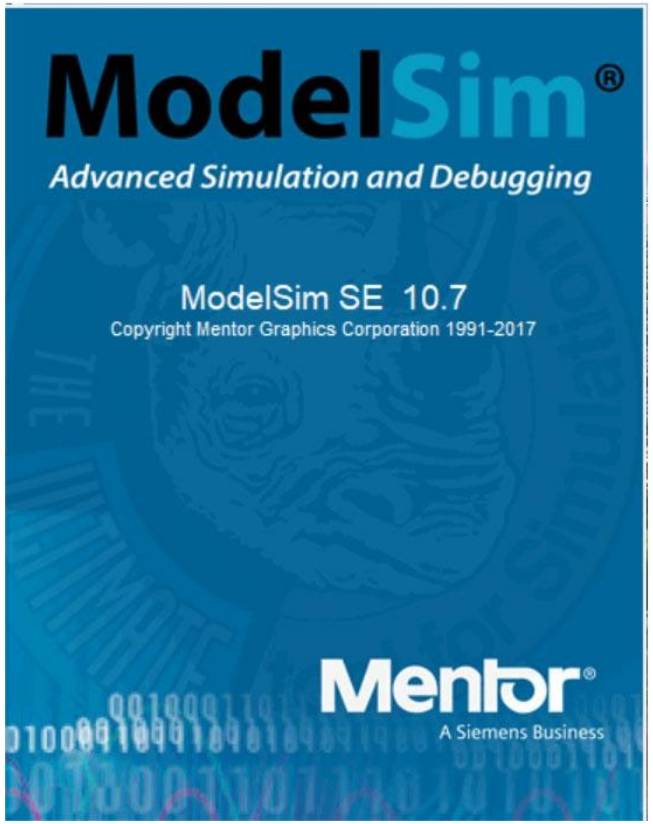
In this Semester…

- Very simple in-order RISC processor
- Flow
  - Register file
  - ALU & Status registers
  - Program counter & Instruction register
  - Instruction decoder
  - Integration
  - Done!



HDL Simulator in this Class's Lab

- Simulation tool for validating hardware design
- Support various HDLs: Verilog, VHDL, SystemVerilog, etc.
- Can be used as 3rd-party simulator along with other synthesizing tools (e.g., Vivado, Quartus)
- Able to do both pre- and post-synthesis simulations



Please refer to backup slides regarding installation!

Task Summary

- You need to implement ***TODO***s commented in skeleton code
  - The corresponding part in slide is noted as “TODO:…” as well
  - There required code files are listed in “*Assignment*” page in each lab
  - The reference waveforms are illustrated right after “*Assignment*” page
- You need to include all requirements that are listed in “*Assignment*” page in your report!
  - The report must be PDF, otherwise, 0 point will be conferred
  - Please submit ***single report file***! Not separately
- The score of whole project is 400
  - Detailed score criteria are also listed in “Assignment” as well

Regulations

- Delay
  - 10% cut-down of score for each delayed day
  - Final score = obtained score * (100% - N*10%)
    - N: # of delayed days, where 10 is the maximum

- Submit two items
  - Zipped file of your ModelSim project folder (format must be: student#_name_code.zip)
  - Report formatted in pdf file (format must be: student#_name_report.pdf)

- Please note that…
  - ***This is training, please implement by yourself***
  - ***We can answer ambiguous points***
  - ***But, we do not service to debug your codes and analyze your codes***
  - ***We DO NOT accept ANY request changing criteria***
  - ***Since it is lab assignment/project, no solution will be shared***

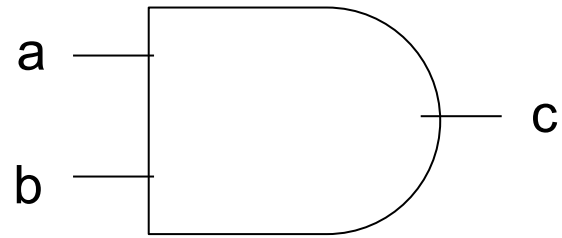
Backup slides for Installations

Backup Slides: Combinational Logic

Coding of Combinational Logic

```
wire a;

assign a = 2'b00;
```

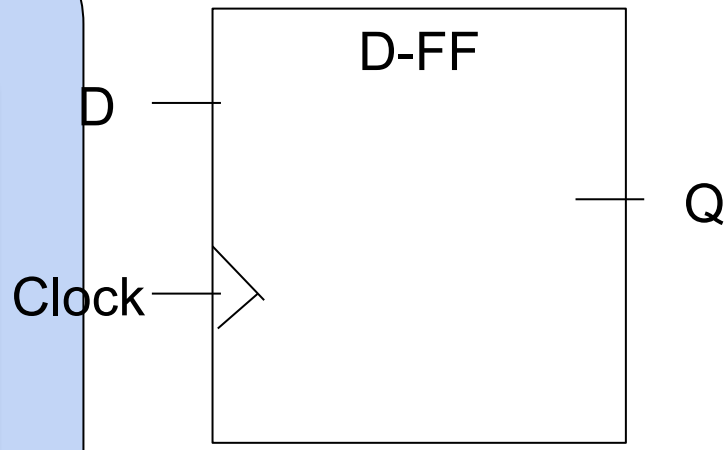
```
reg b;

always @* begin

  b = 2'b00;

end
```



Backup Slides: Sequential Logic

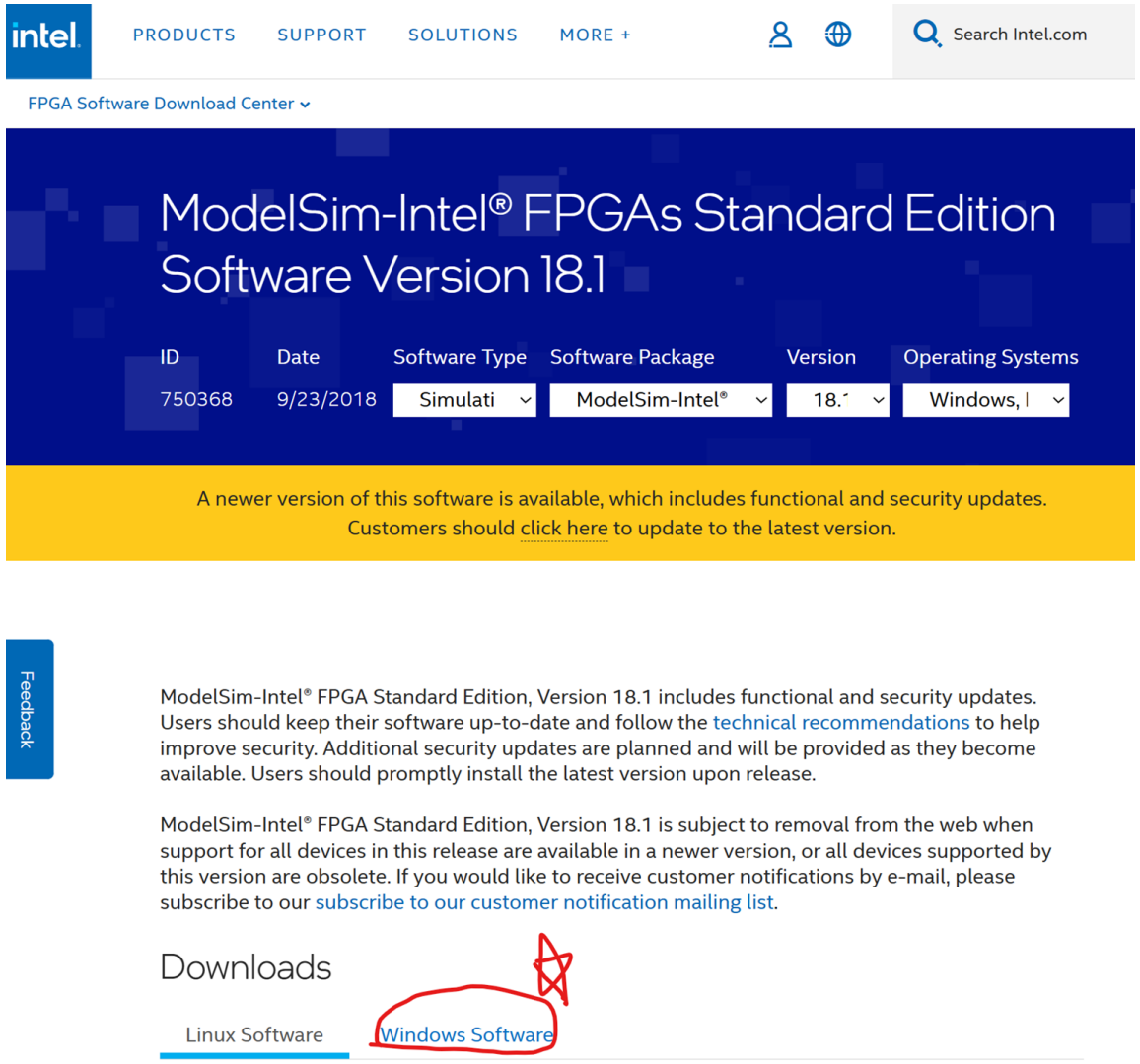
Coding of Sequential Logic

```
reg c;

always@(posedge CLK or negedge RSTn)
begin

  c <= 2'b00;

end
```



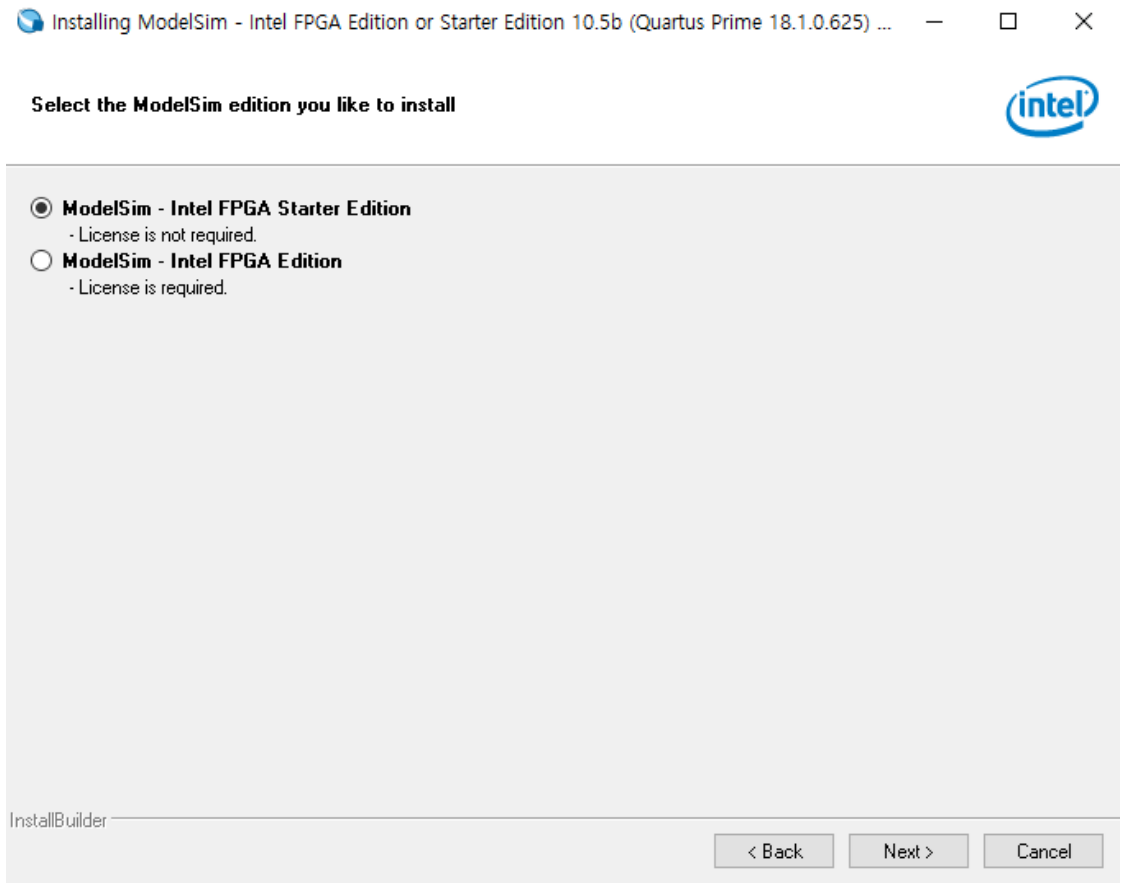
Set up Your Environment (1)

- We are going to use “Starter Edition” of ModelSim
- Link:
  - https://www.intel.com/content/www/us/en/software-kit/750368/modelsim-intel-fpgas-standard-edition-software-version-18-1.html
- Download the software depending on your operating systems (OS)



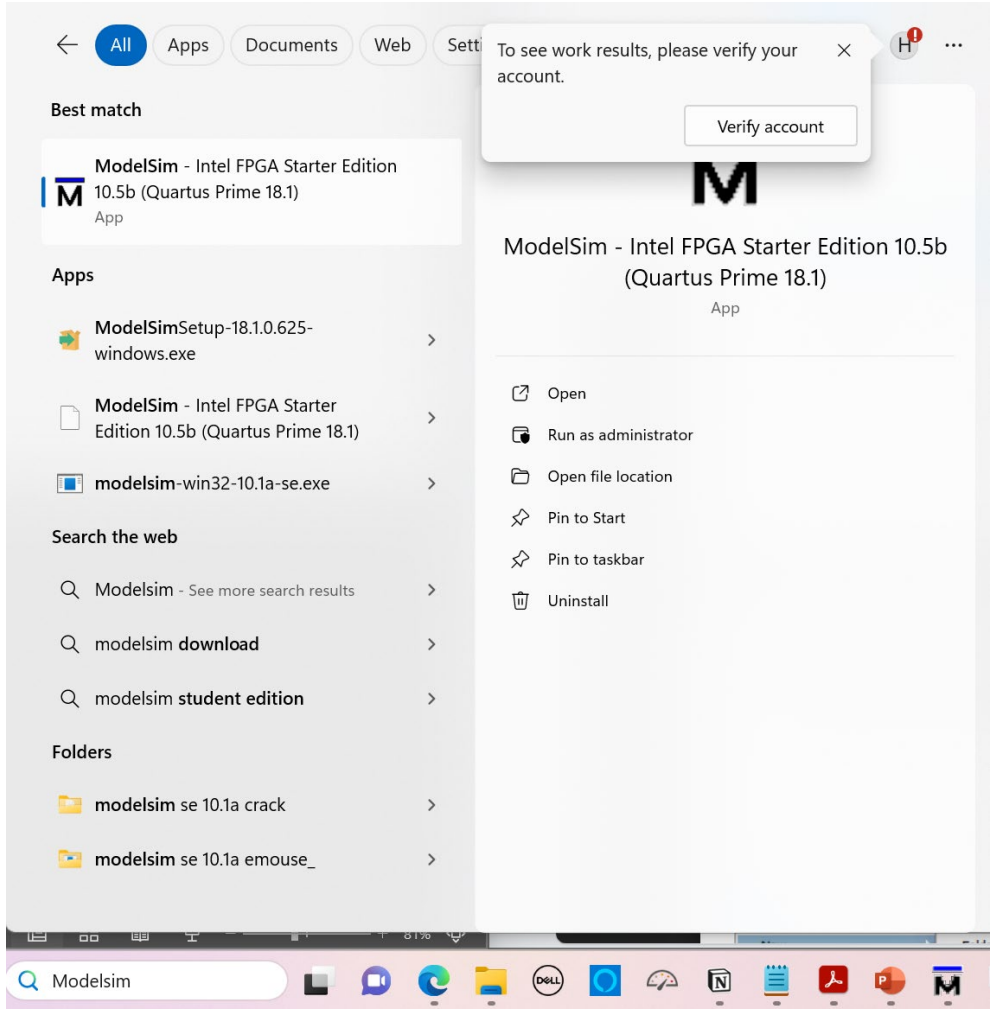
Set up Your Environment (2)

- Please proceed with “Next”
- Make sure to click on “Starter Edition”!
- After this step, installation will be done!



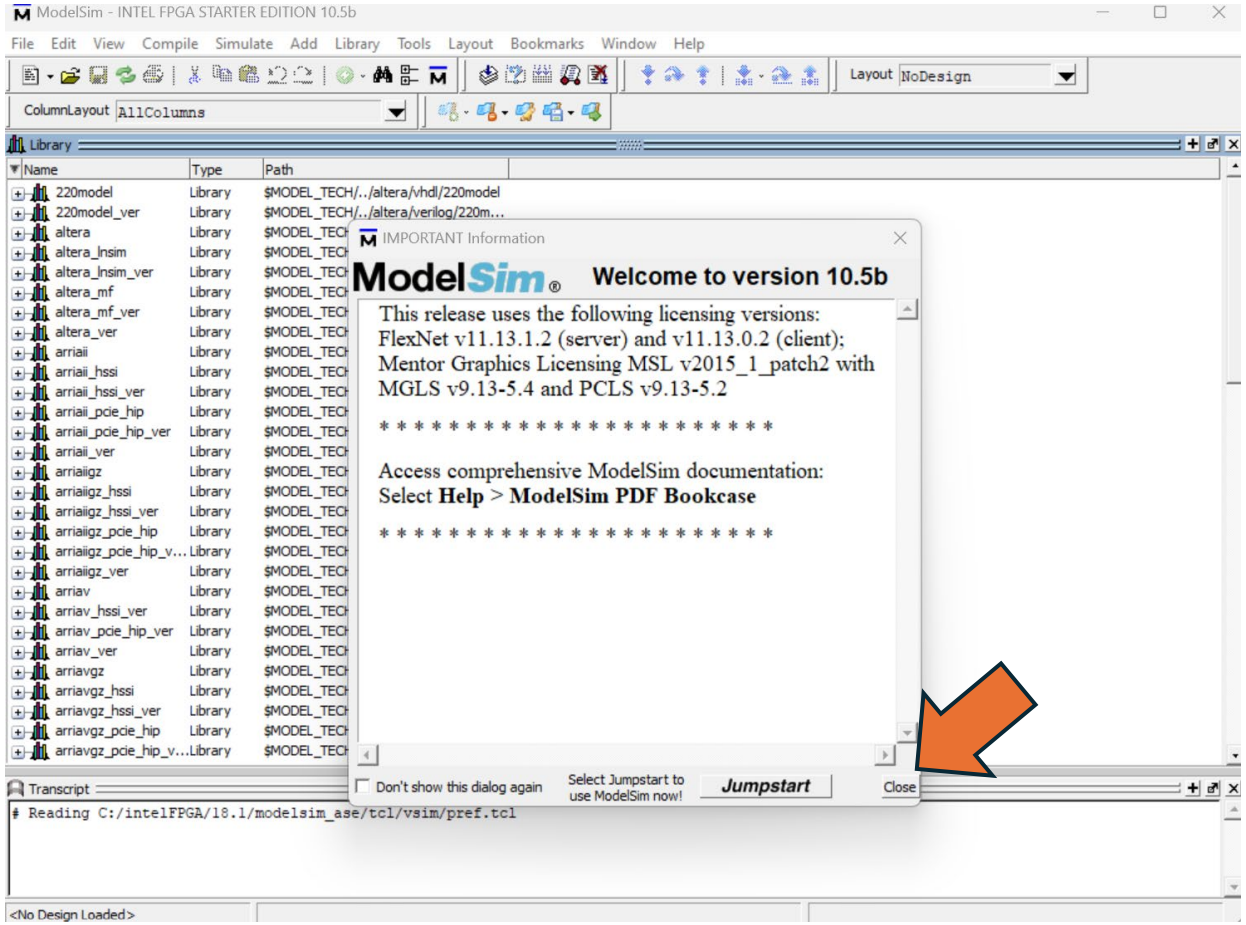
Set up Your Environment (3)

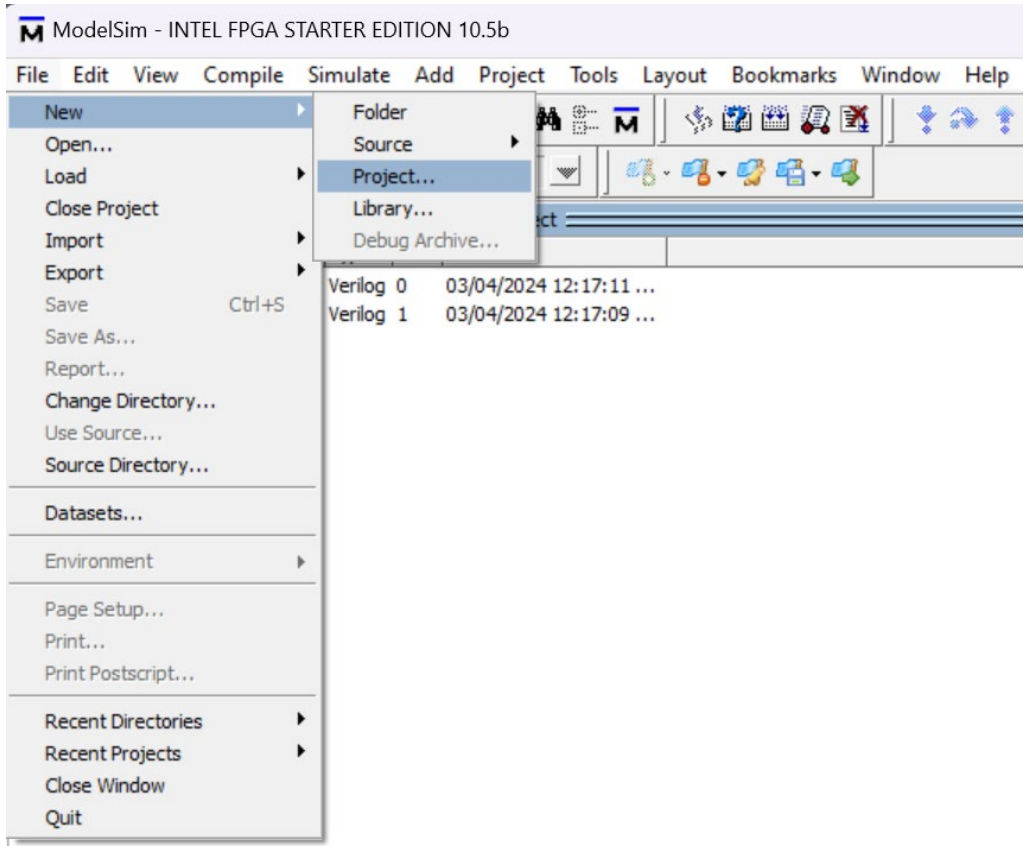
- Tab “windows” key on keyboard
- Type down “Modelsim”
- Launch ModelSim



Set up Your Environment (4)

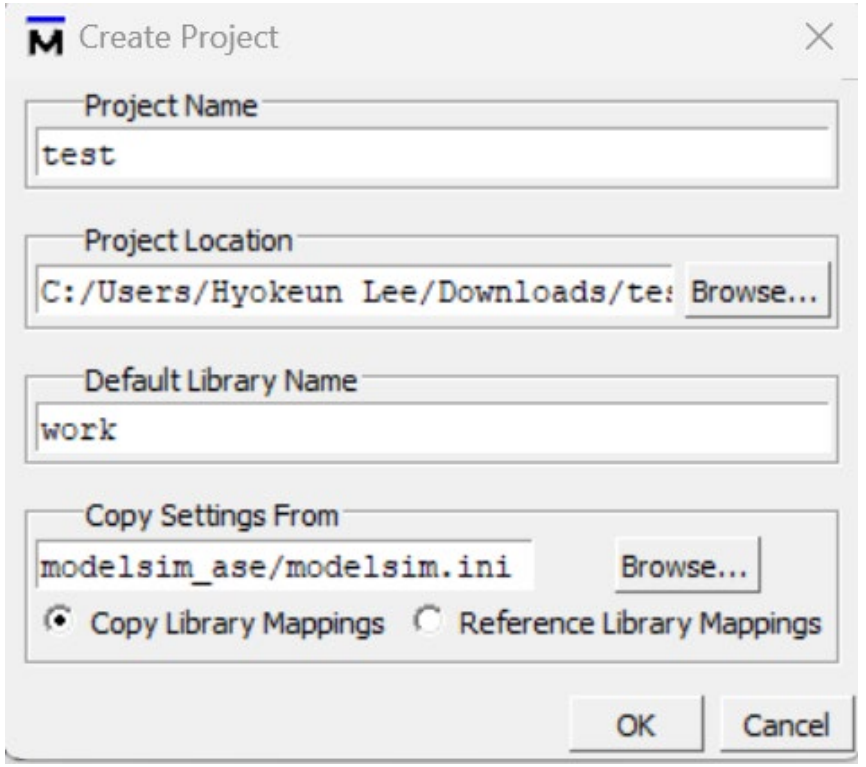
- Create new project as below





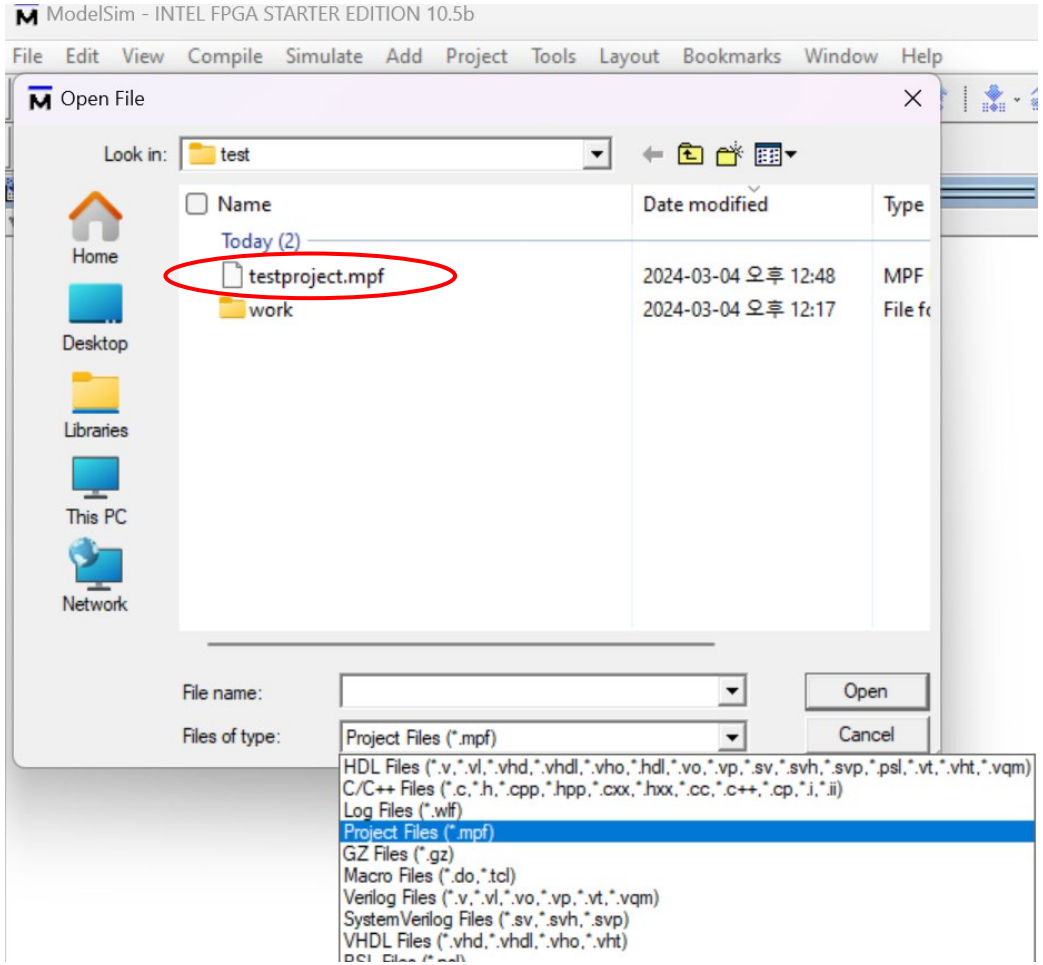
Set up Your Environment (5)

- You will see a project creation setting window
- Project name should be in English
- Just feel free to set the project location named in English
- Click “OK”



Set up Your Environment (6)

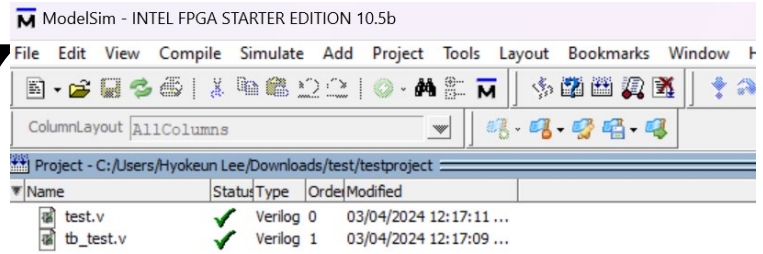
- Sometimes, you want to open up existing project
- Click “File – Open”
- Drag down “Files of type”
- Select “Project Files (*.mpf)”
- Click the .mpf to open the project

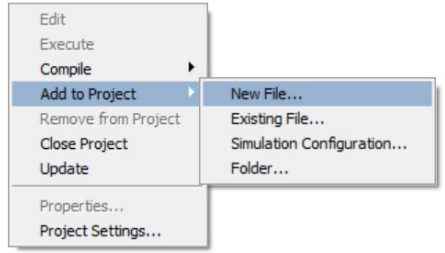


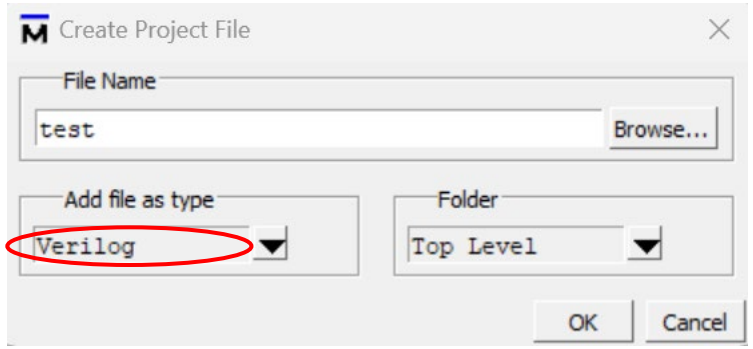
Set up Your Environment (7

- Now, we need to create source file
- Right click at “Project” board
- Select “Add to Project – New File”
- Type down file name as you want
- But make sure to have “Verilog” format

- Make two files: “test” and “tb_test”







Set up Your Environment (8)

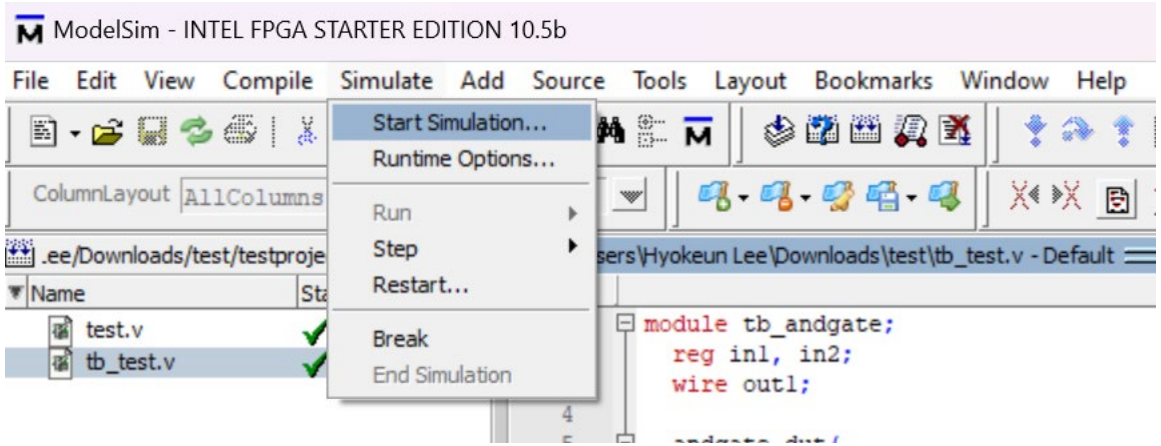
- Code “test” and “tb_test” as follows, respectively

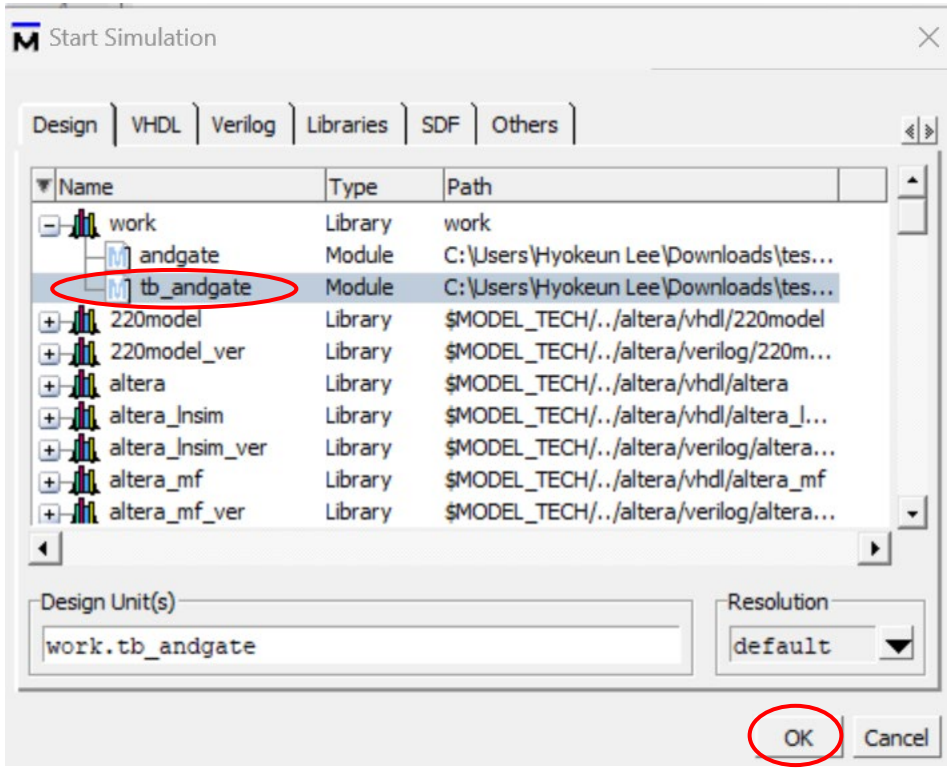
```
1   module andgate(in1, in2, out1);
2
3   input in1, in2;
4   output out1;
5
6   and(out1, in1, in2);
7
8   endmodule
9
```

```
1   module tb_andgate;
2     reg in1, in2;
3     wire out1;
4
5     andgate dut(
6       .in1(in1),
7       .in2(in2),
8       .out1(out1));
9
10    initial begin
11      in1 = 0;
12      in2 = 0;
13      #100;
14    end
15
16    always #(100) begin
17      in1 = ~in1;
18    end
19
20    always #(50) begin
21      in2 = ~in2;
22    end
23
24  endmodule
25
```

Set up Your Environment (9)

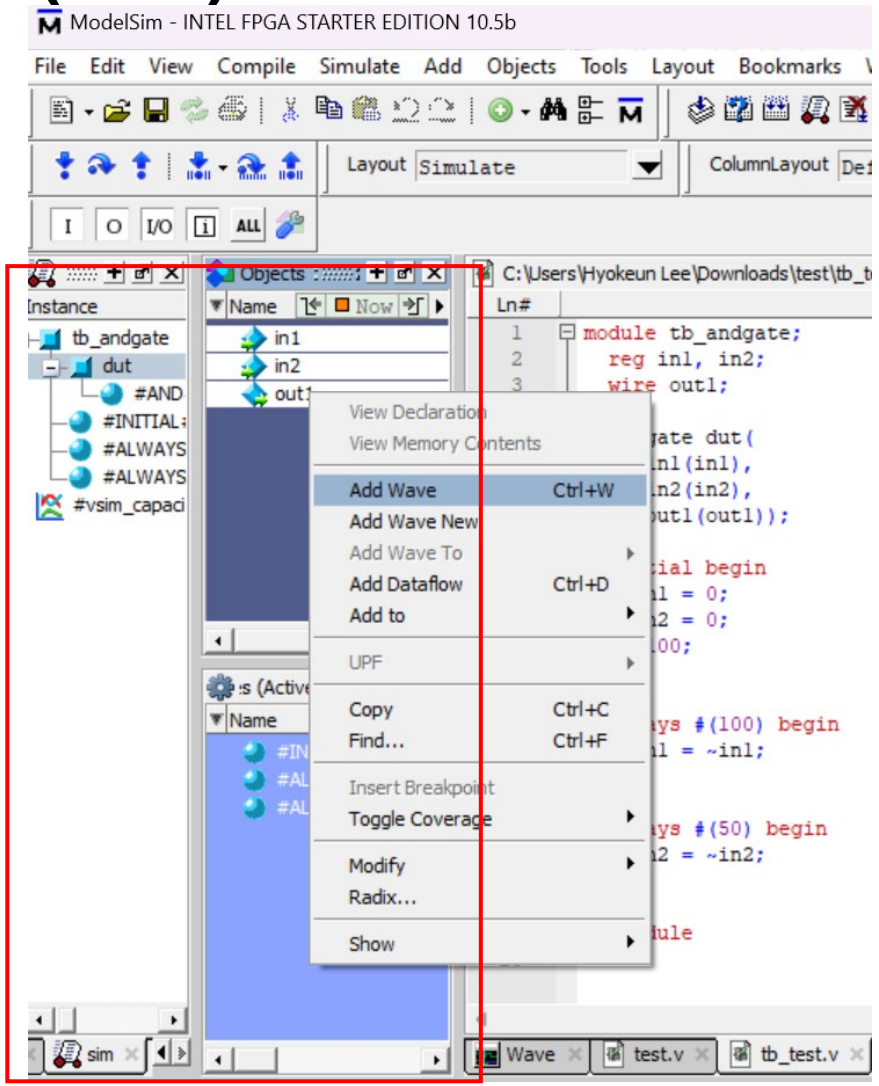
- Let’s do fast simulation!
- Select “work” tab
- Select testbench, “tb_andgate”





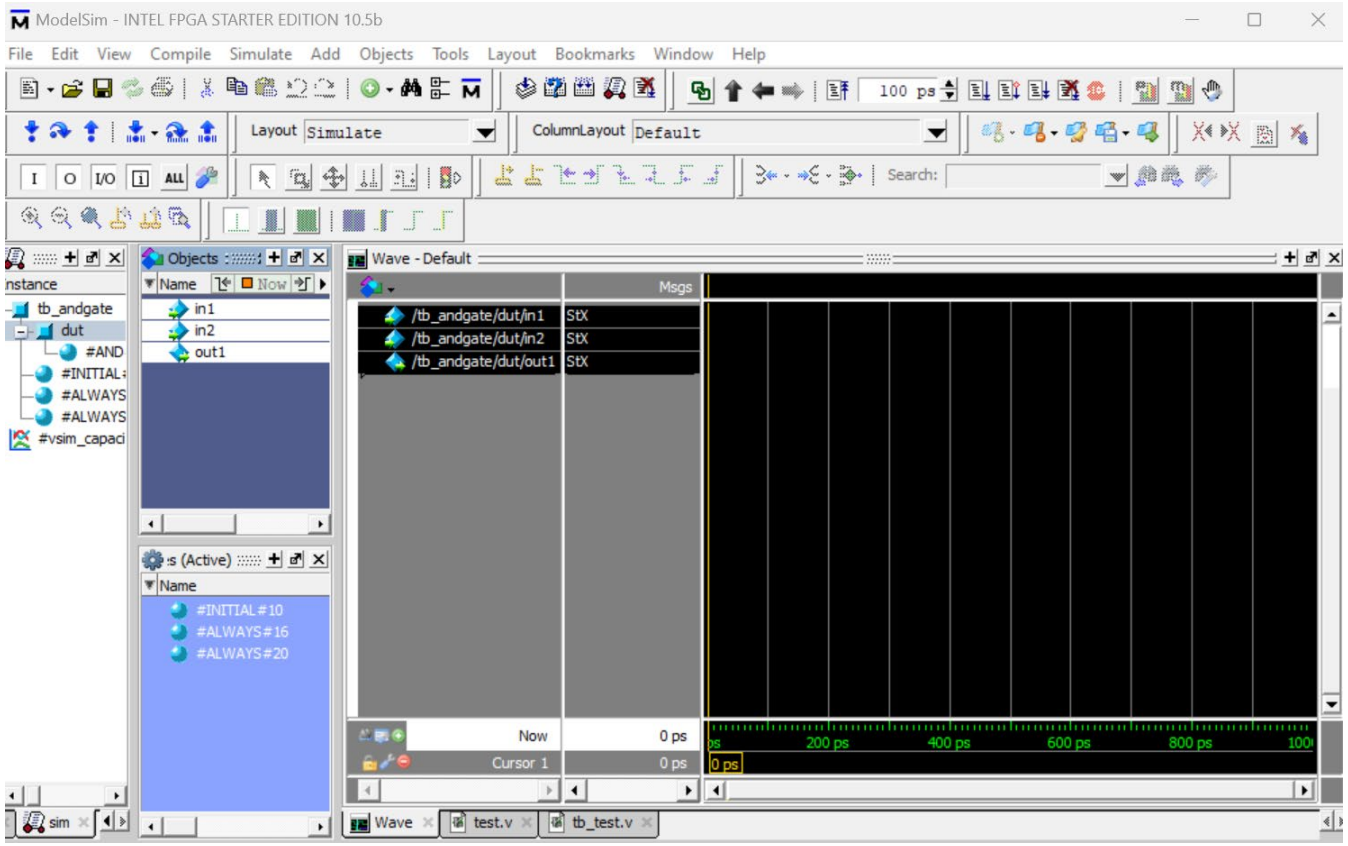
Set up Your Environment (10)

- You'll see three popped out windows
- In "Instance" window, you will see modules that you created for simulation
  - Clicking it, you can see corresponding signals in "Objects" window
- In "Objects" window, select the signals you interested
  - Right click, and select "Add Wave"



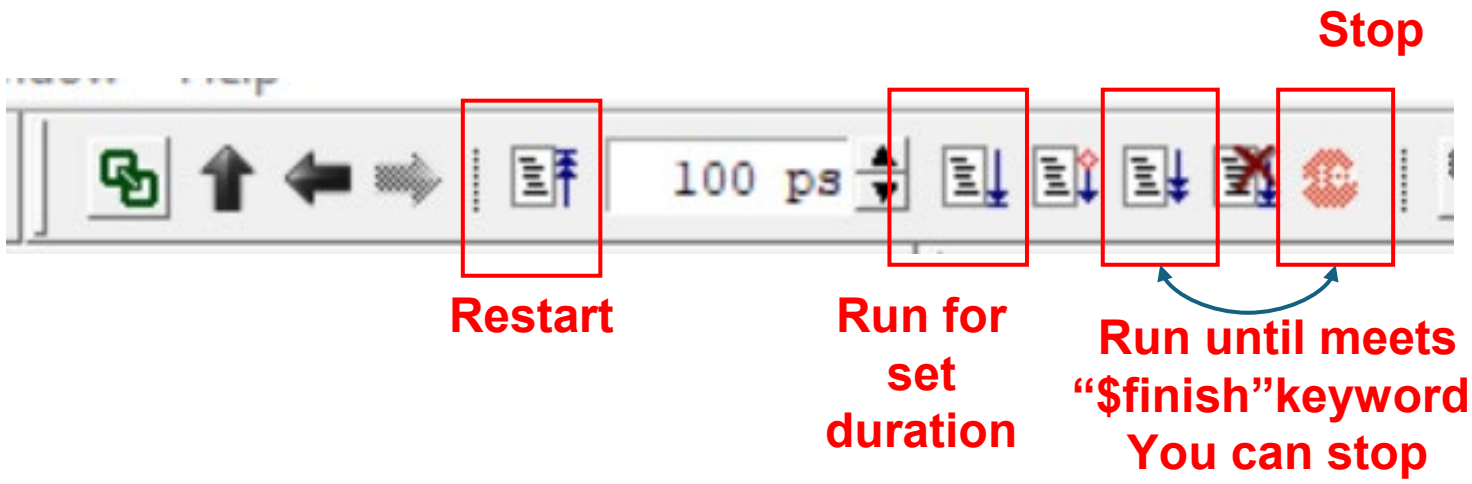
Set up Your Environment (11)

- You can see signals are added to “Wave” tab



Set up Your Environment (12)

- You can type down simulation time, here “100 ps” is set
- Several run options are available



Set up Your Environment (13)

- That’s it! Now you’re ready to do HDL simulation!