[SoC Design]

Lab 2: HW Design

Chester Sungchung Park (박성정)

SoC Design Lab, Konkuk University

Webpage: http://soclab.konkuk.ac.kr

Teaching Assistants

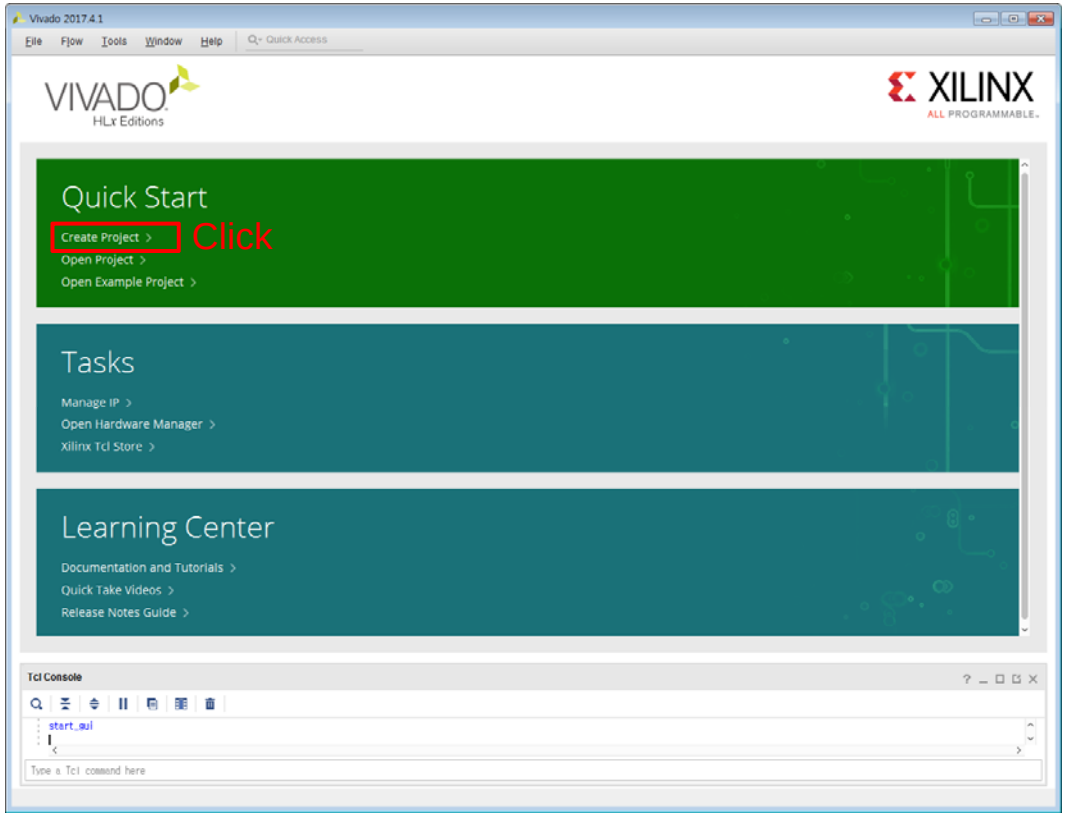
- Youngho Seo (younghoseo@konkuk.ac.kr), M.S. candidate
- Sanghun Lee (sanghunlee@konkuk.ac.kr), M.S. candidate

Outline

- Creating RTL projects
- Programming in RTL
- Running Behavioral Simulation
- Running Synthesis
- Running Implementation

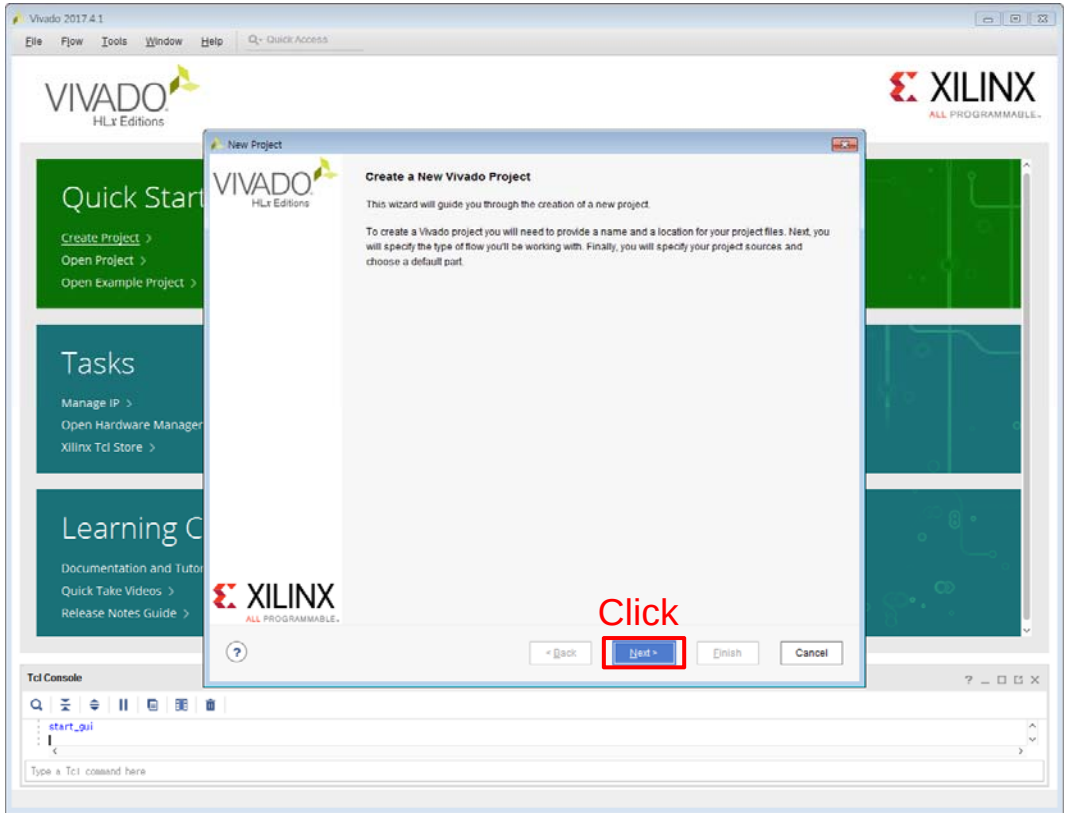
Creating RTL Projects

- Getting Started
  - Click '**Quick Start > *Create Project***'



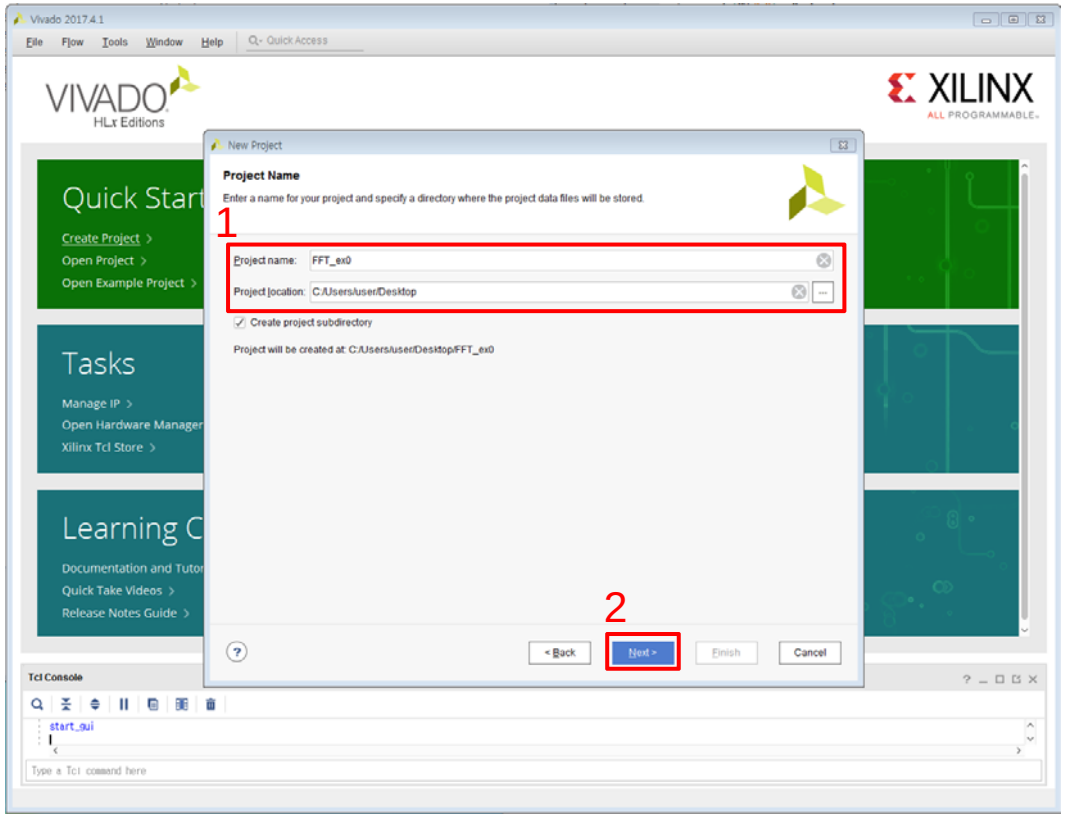
Creating RTL Projects

- Create a New Vivado Project
  - Click ***'Next'***



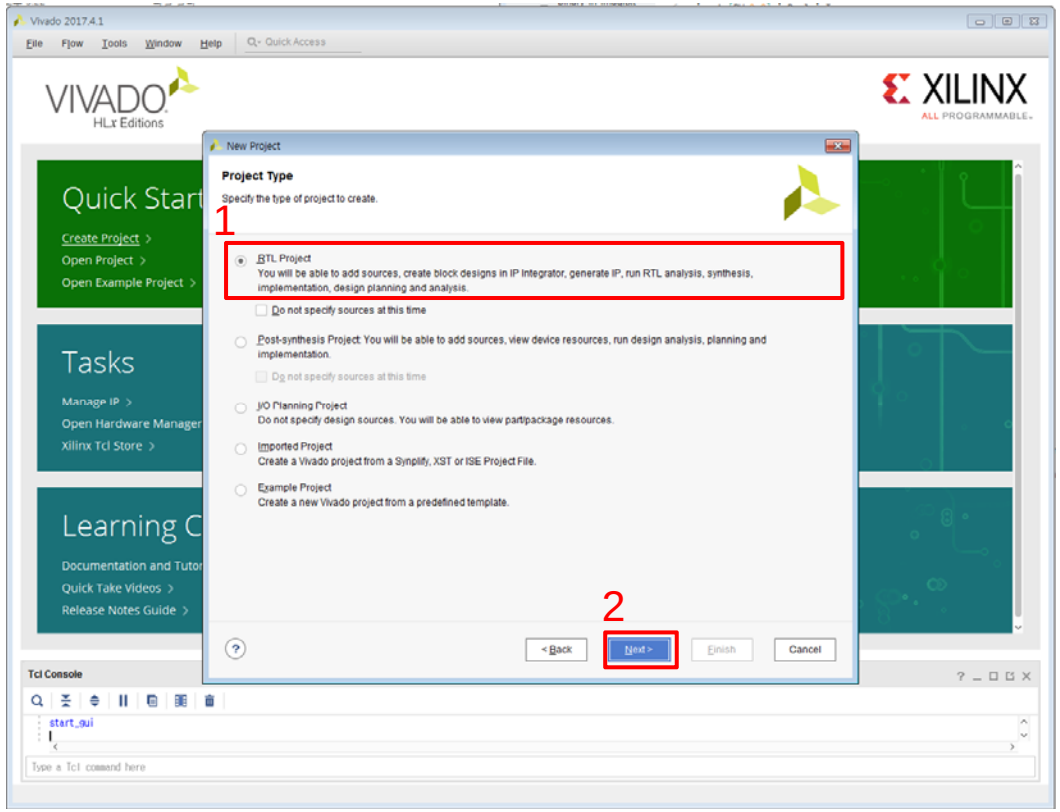
Creating RTL Projects

- Enter Project Name
  - Type ‘***Project name***’ and choose ‘***Project location***’
  - Click ‘***Next***’



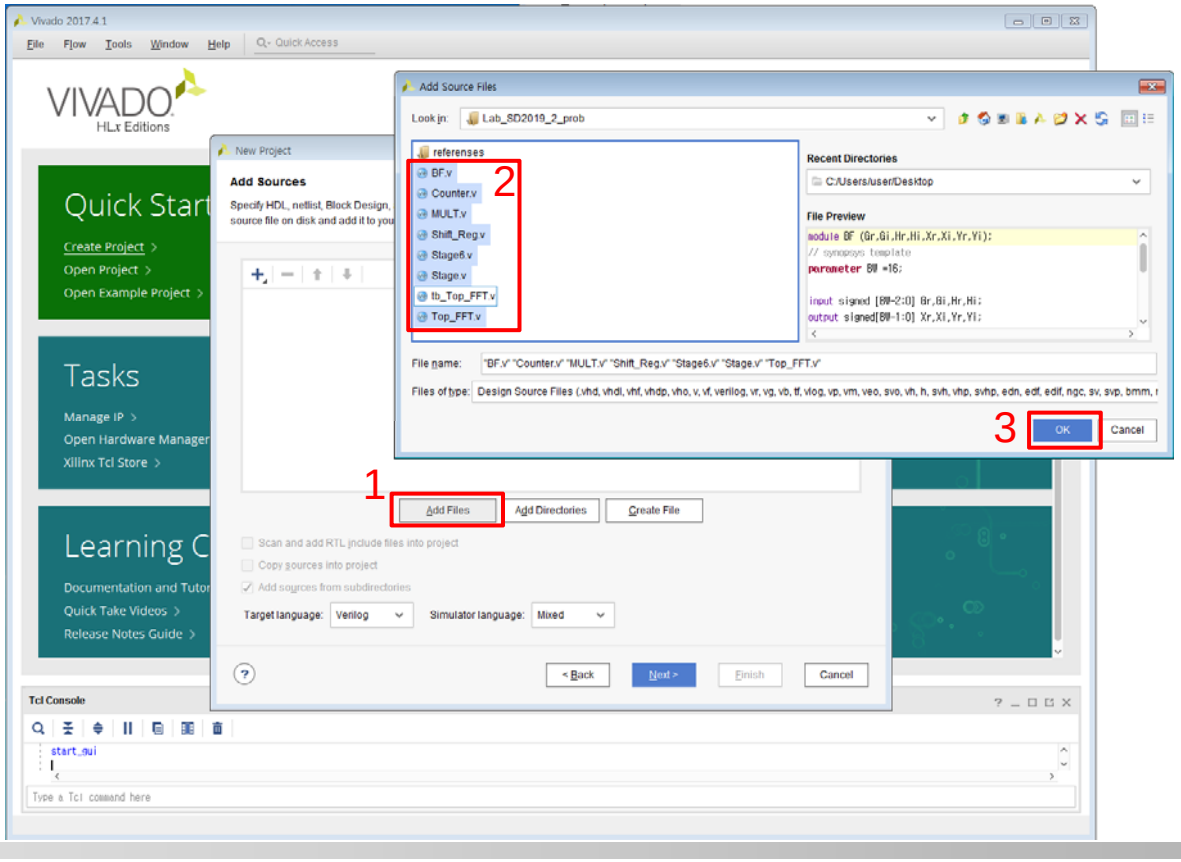
Creating RTL Projects

- Choose Project Type
  - Click ***'RTL Project'*** > Click ***'Next'***



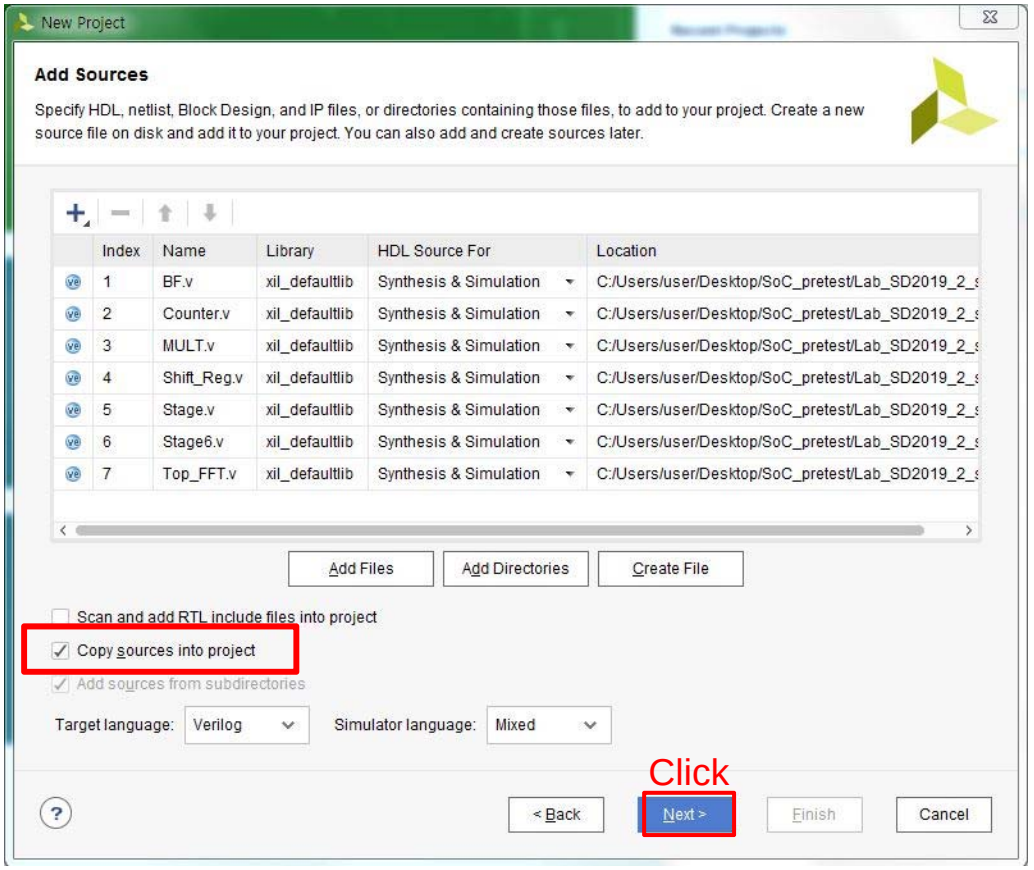
Creating RTL Projects

- Add Sources
  - Click ***'Add Files'*** and add all the source files except the testbench file ***'tb_Top_FFT.v'*** that will be added later



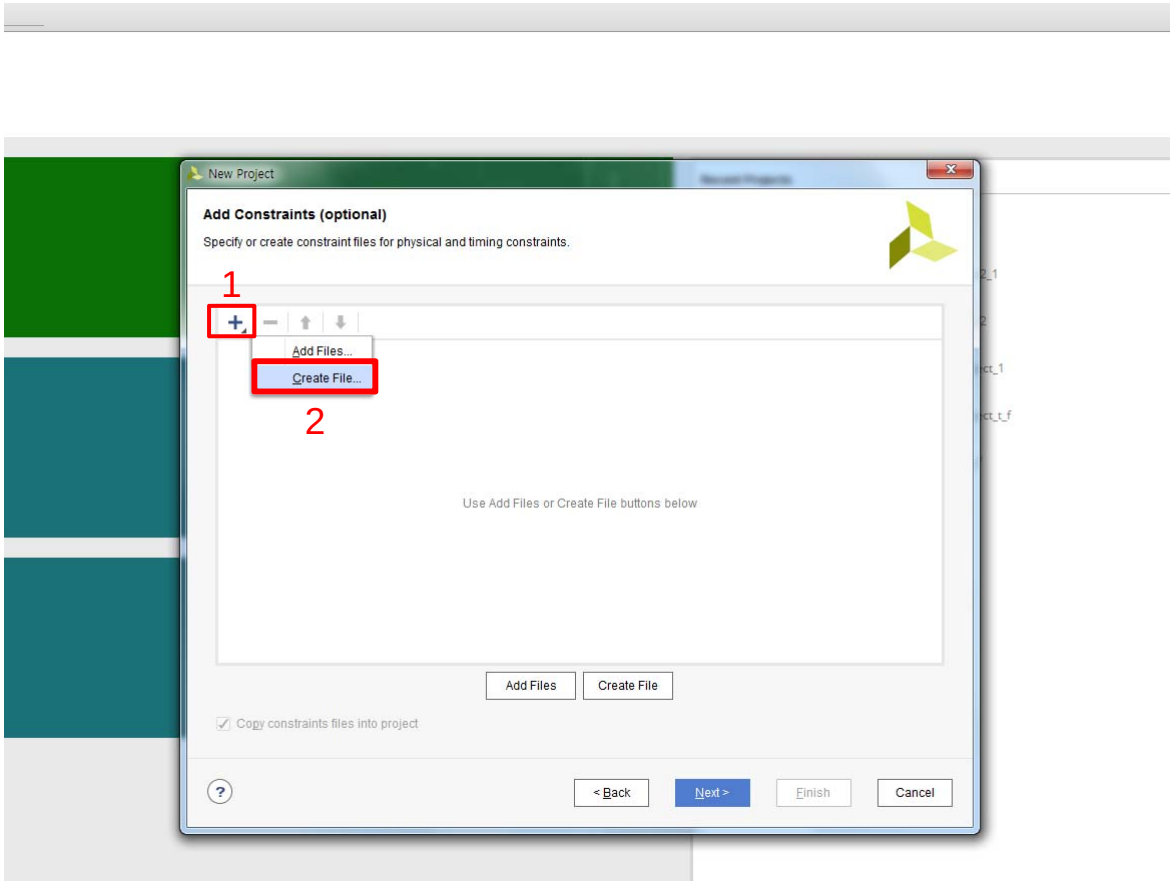
Creating RTL Projects

- Add Sources (cont'd)
  - Check ***'Copy sources into project'*** and the click ***'Next'***



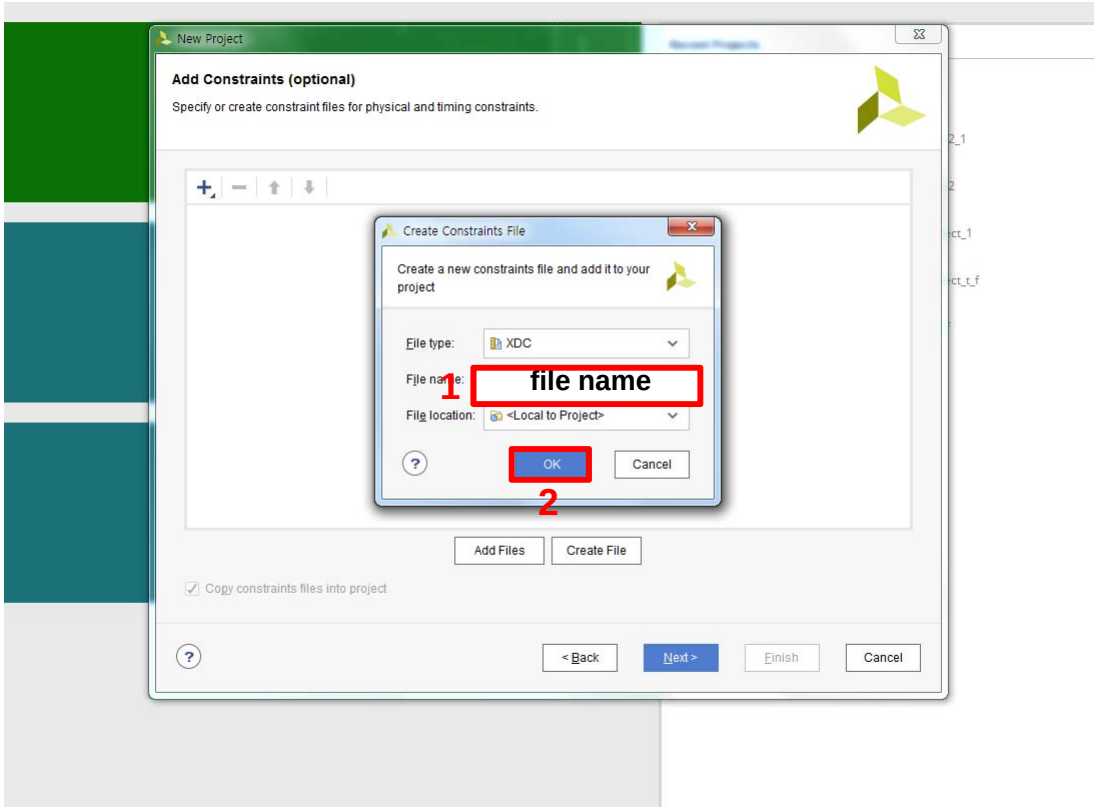
Creating RTL Projects

- Add Constraints
  - Click ***'Add > Create Files'***



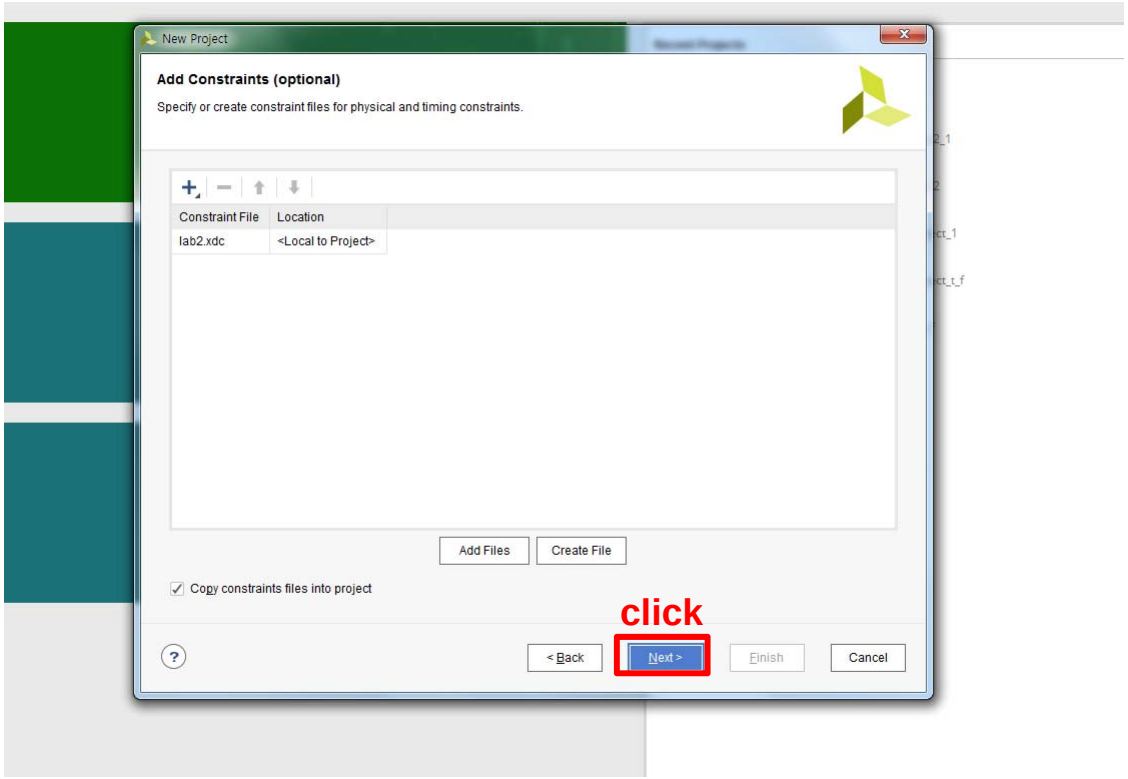
Creating RTL Projects

- Add Constraints (cont'd)
  - Type ***'File name'*** and then click ***'OK'***



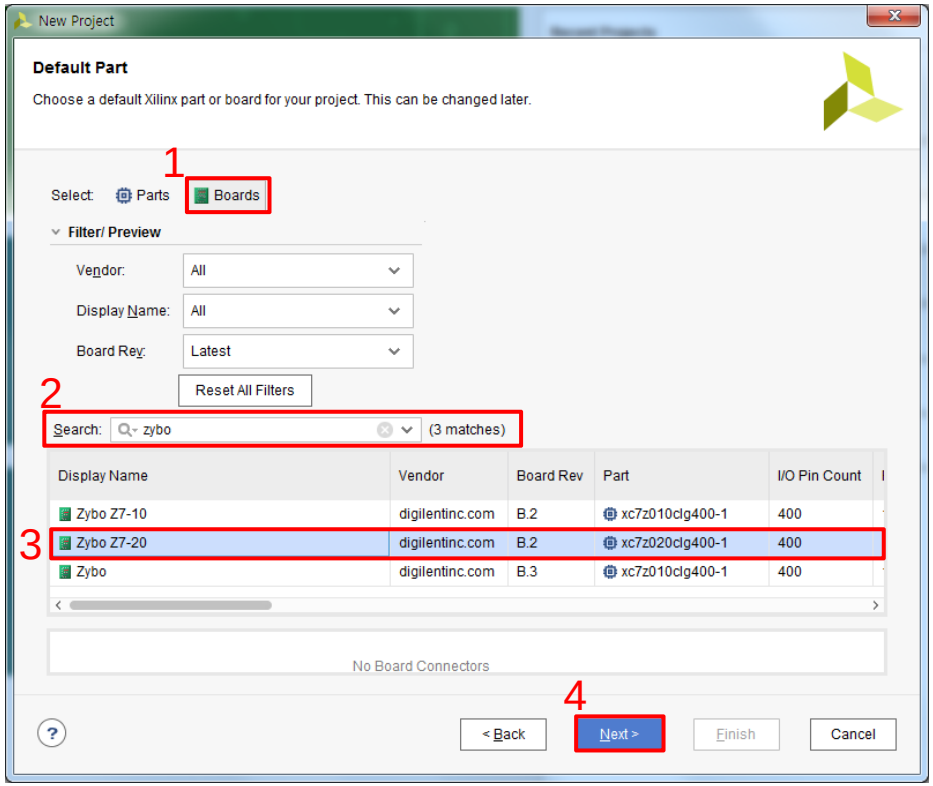
Creating RTL Projects

- Add Constraints(cont'd)
  - Click ***'Next'***



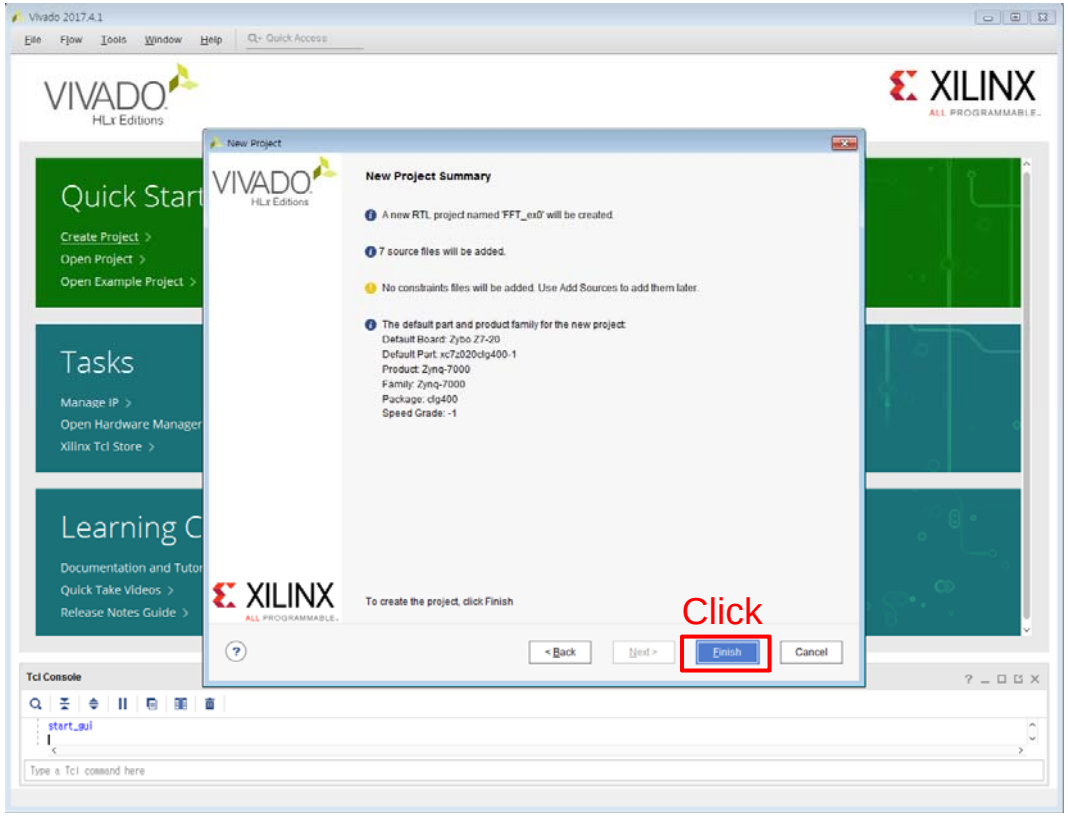
Creating RTL Projects

- Choose Default Part
  - Select the ***'Boards'***
  - Search the ***'zybo'***
  - Select the ***'Zybo Z7-20'***
  - Click '***Next***'



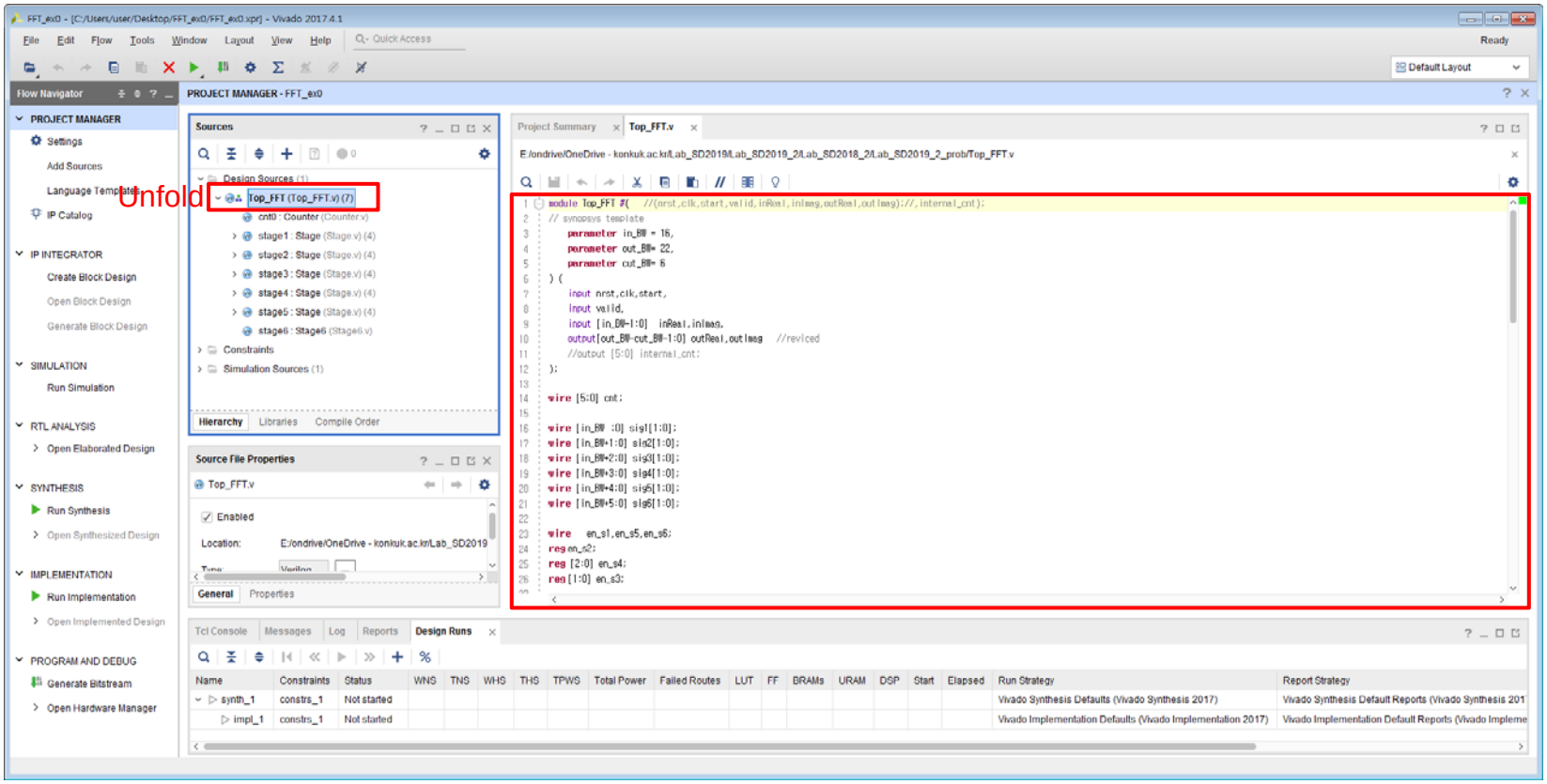
Creating RTL Projects

- Check New Project Summary
  - Click ***'Finish'***



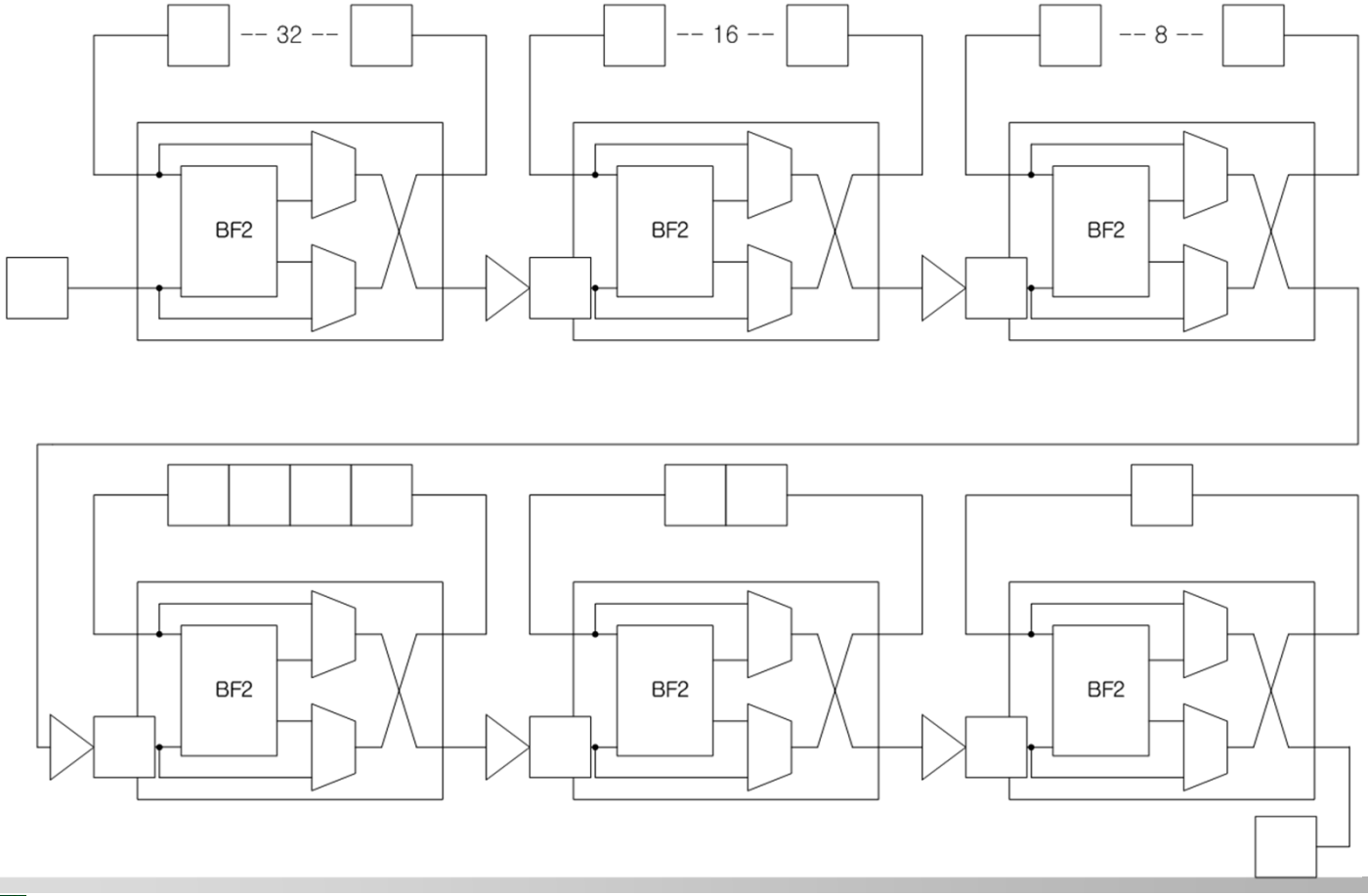
Programming in RTL

- Edit the RTL source codes
  - Unfold a module if necessary, and double-click the module name to edit.



Programming in RTL

- Complete the RTL source codes
  - Single-path delay feedback: 64-pt Radix-2

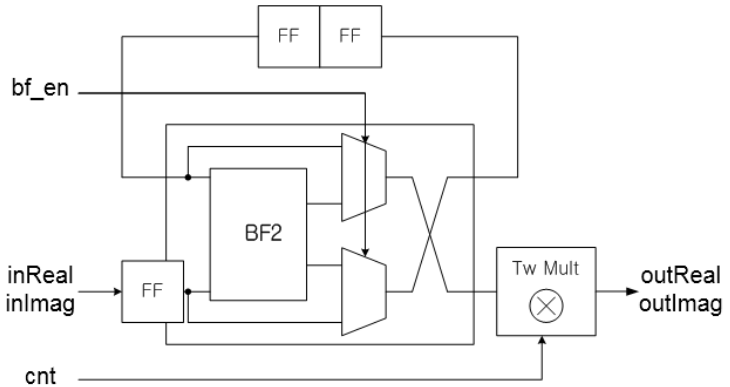


Programming in RTL

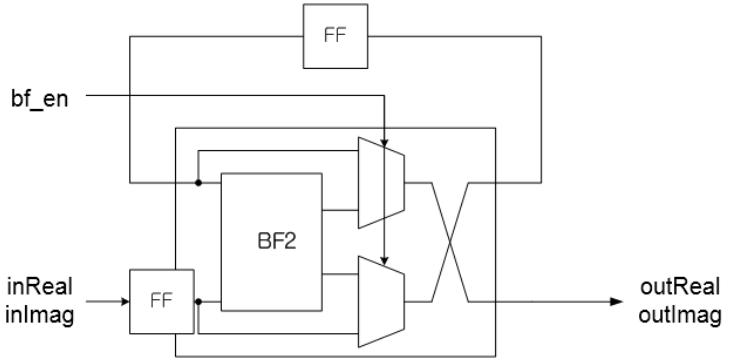
- Complete the RTL source codes (cont'd)
  - Add lines to the ***'Fill Your Code Here'*** section in ***'Stage6.v'***

```
module Stage6(nrst,clk,bf_en,inReal,inImag,valid,outReal,outImag);
parameter BW=16;
parameter N =1;

input           nrst,clk,bf_en;
input [BW-2:0]  inReal,inImag;
input           valid;
output[BW-1:0]  outReal,outImag;

reg   [BW-2:0]  rReal,rImag;

wire  [BW-1:0]  bf_x[1:0];
wire  [BW-1:0]  bf_y[1:0];

reg   [BW-1:0]  sr_out[1:0];

wire  [BW-1:0]  mux0[1:0];
wire  [BW-1:0]  mux1[1:0];

assign mux0[0] = bf_en? bf_x[0] : sr_out[0];
assign mux0[1] = bf_en? bf_x[1] : sr_out[1];

assign mux1[0] = bf_en? bf_y[0] : {rReal[BW-2],rReal};
assign mux1[1] = bf_en? bf_y[1] : {rImag[BW-2],rImag};


//////////////////////////////////////////////////////
//////////     Fill your code here    ////////////////
//////////////////////////////////////////////////////

endmodule
```

Stage 5

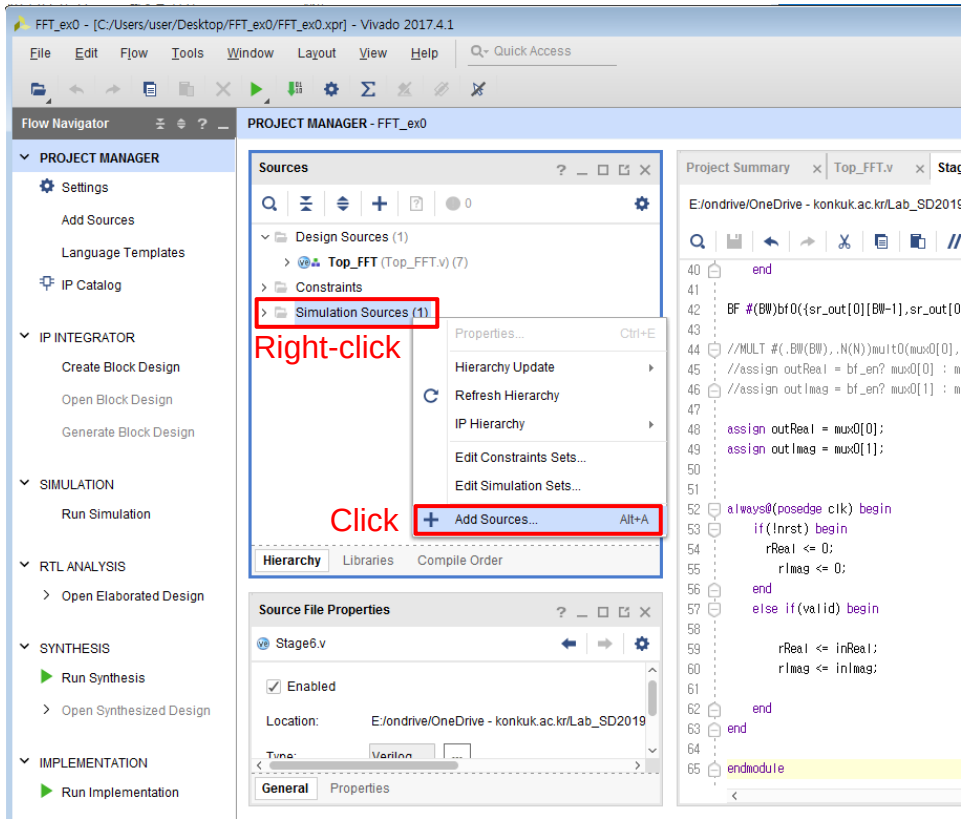


Stage 6



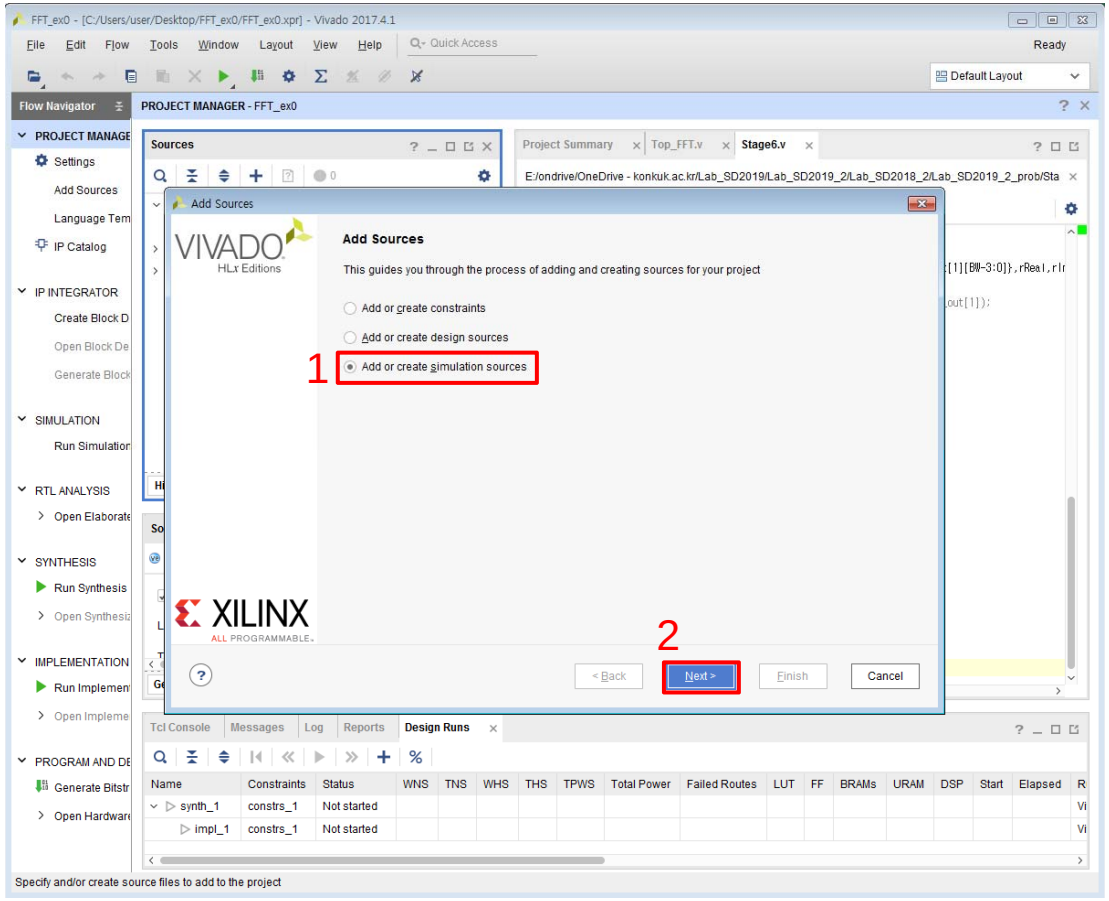
Running Behavioral Simulation

- Add Sources
  - Right-click on ***'Simulation Source'*** in ***'Project Manager (Sources)'***
  - Select ***'Add Sources'***



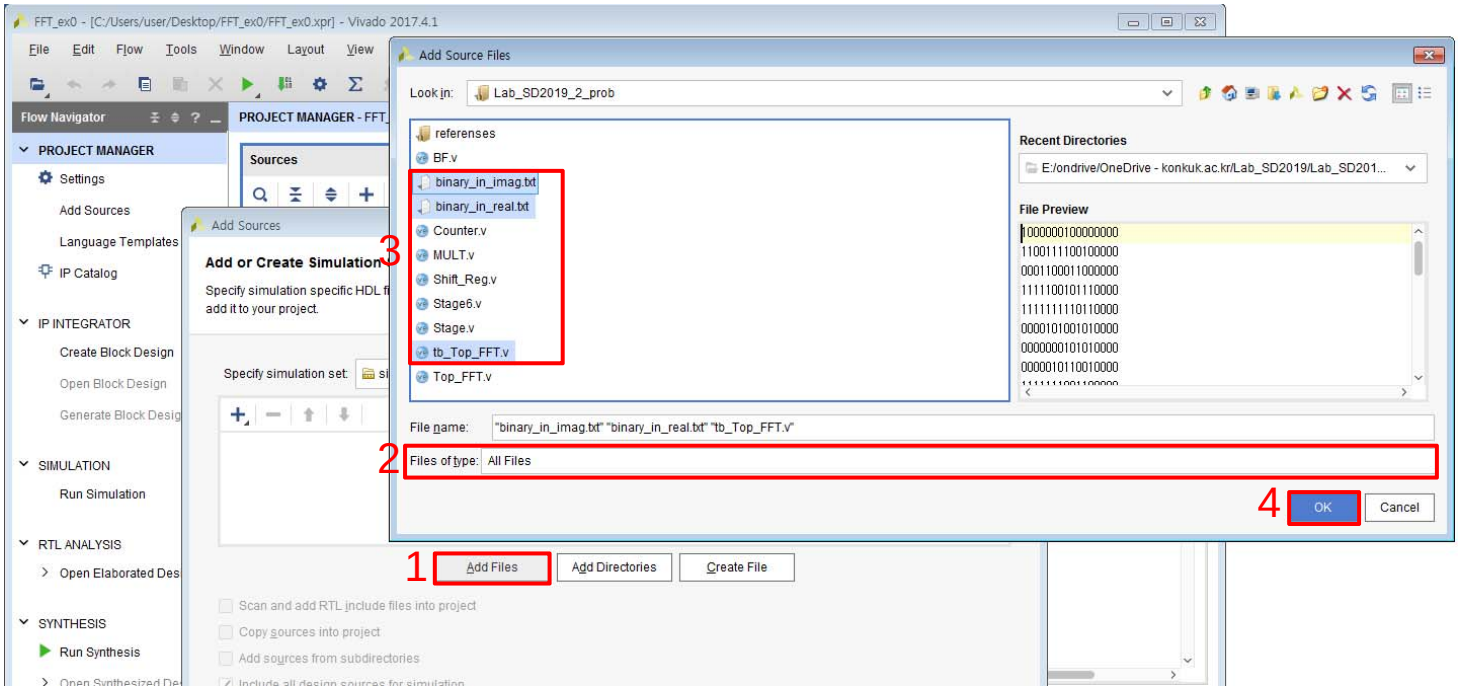
Running Behavioral Simulation

- Add Sources (cont'd)
  - Click ***'Add or create simulation sources'*** and then click ***'Next'***



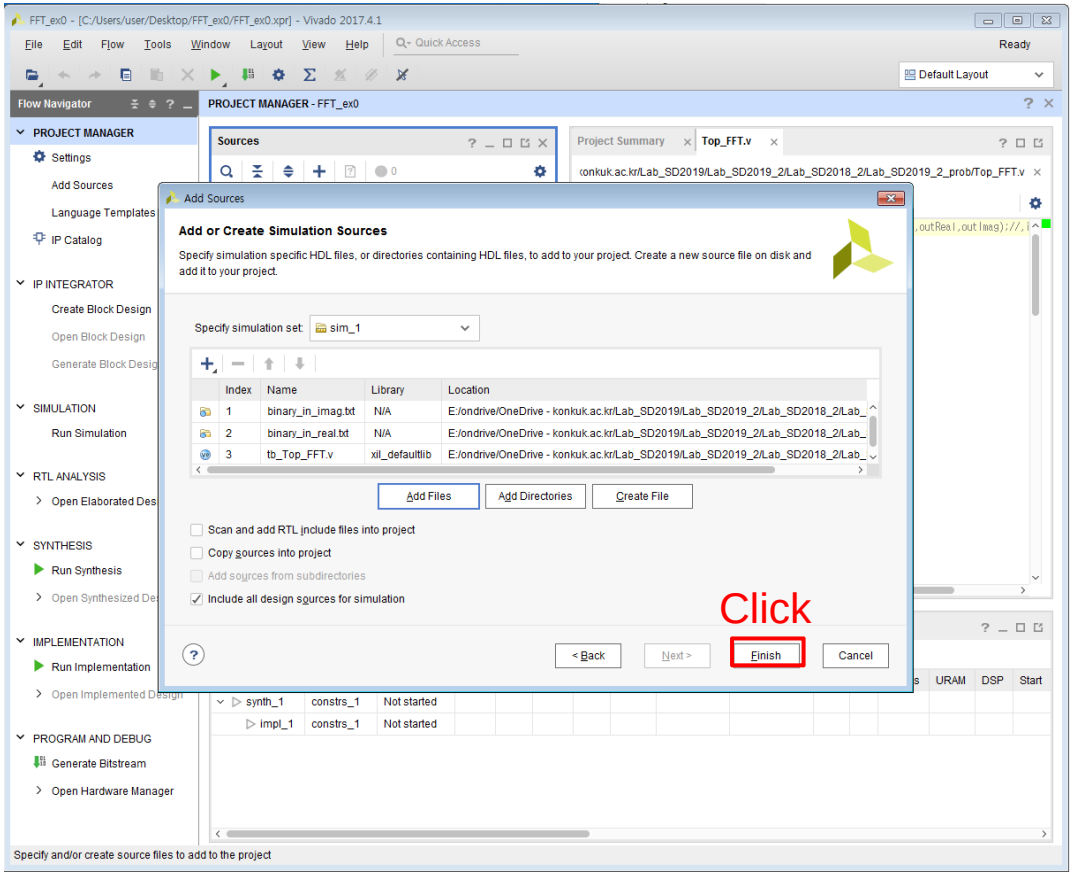
Running Behavioral Simulation

- Add Sources (cont'd)
  - Select ***'Add Files…'*** and choose ***'All Files'*** as Files of type
  - Add the testbench file ***'tb_Top_FFT.v'*** and the input files ***'binary_in_real.txt'*** and ***'binary_in_imag.txt'***



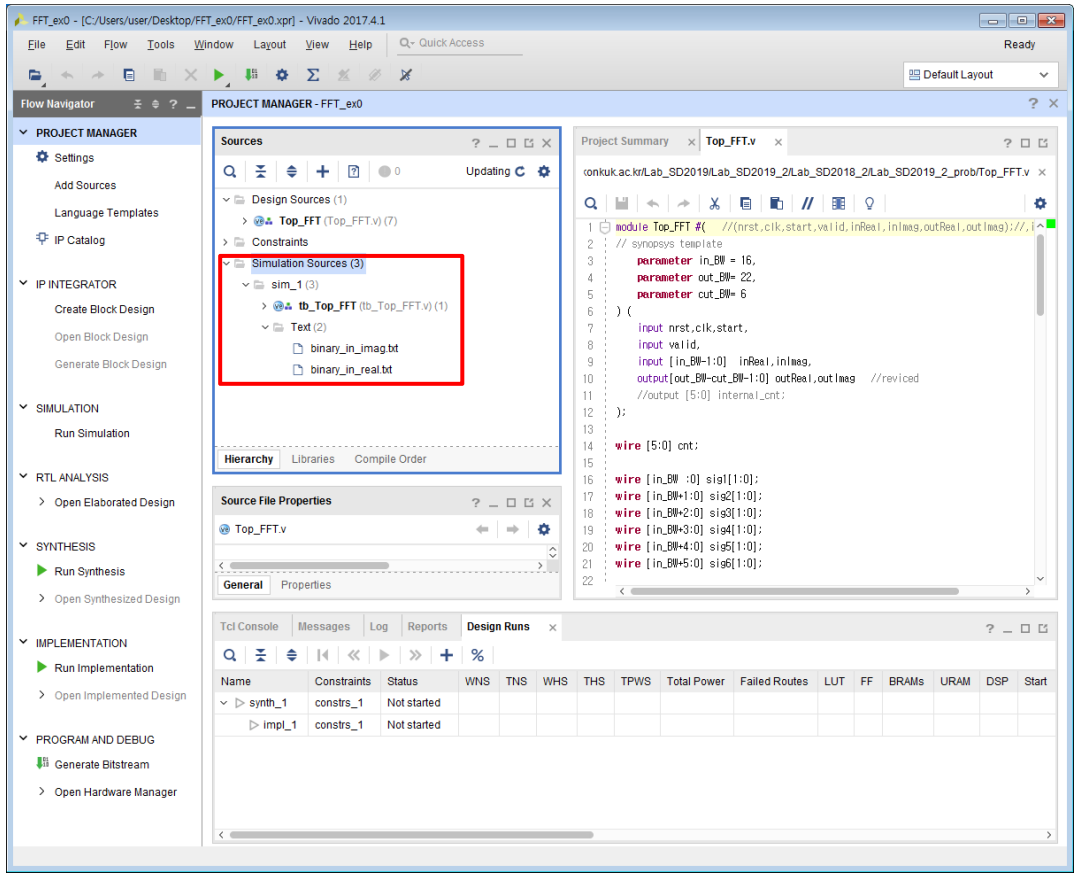
Running Behavioral Simulation

- Add Sources (cont'd)
  - Click ***'Finish'***



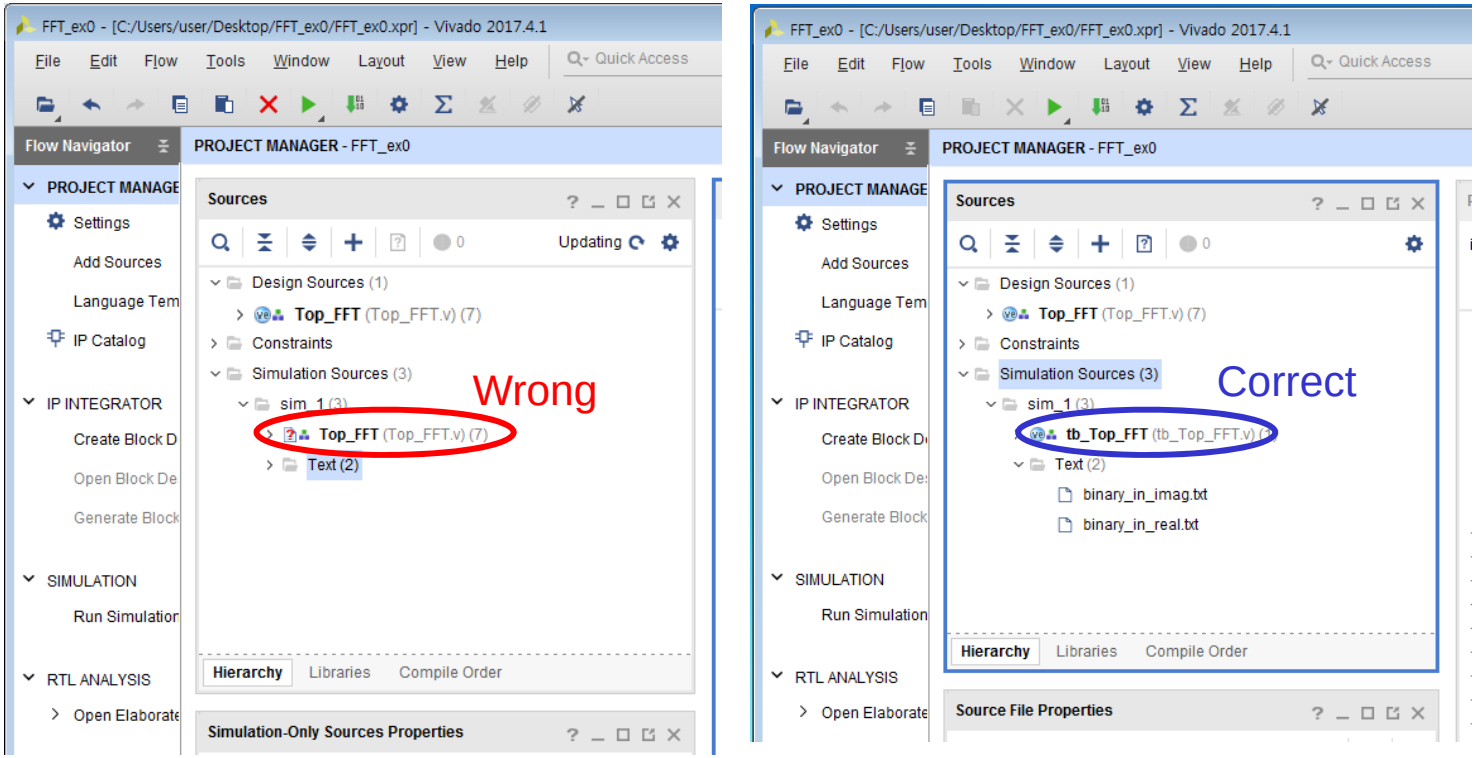
Running Behavioral Simulation

- Check Project Manager (Source)
  - Make sure that all the files have been correctly added.



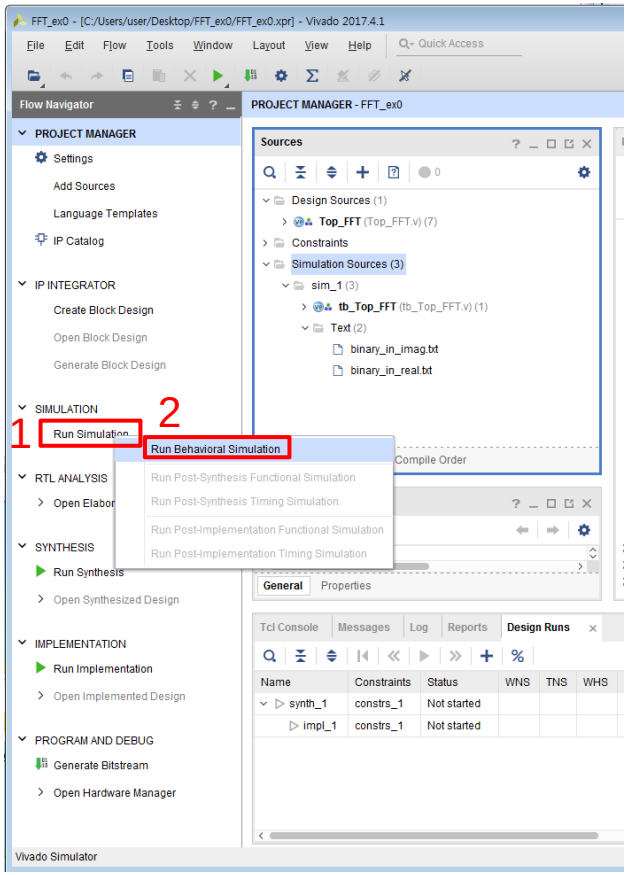
Running Behavioral Simulation

- Check Project Manager (Source) (cont'd)
  - Note that some of the files may fail to be recognized. Then repeat the steps described in pp. 16 ~19.



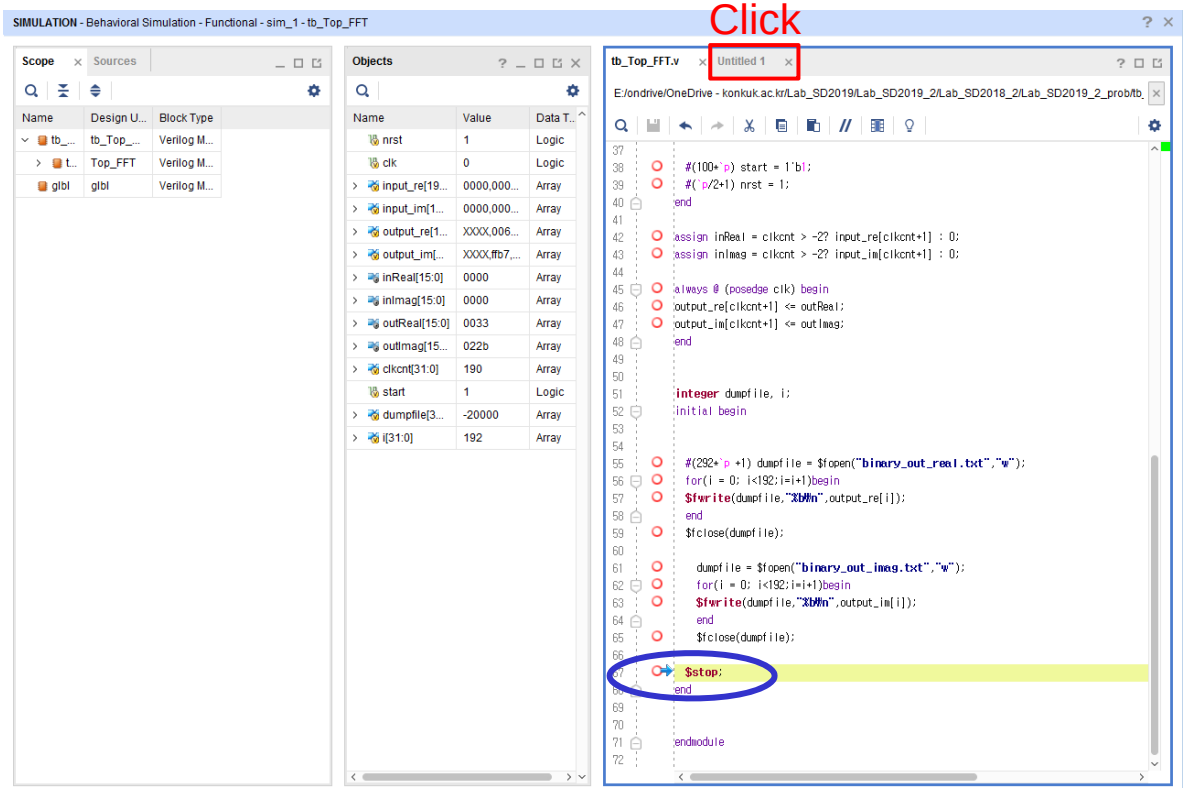
Running Behavioral Simulation

- Run Behavioral Simulation
  - Select ***'Simulation'*** > ***'Run Simulation'*** in ***'Flow Navigator'***
  - Click ***'Run Behavioral Simulation'***



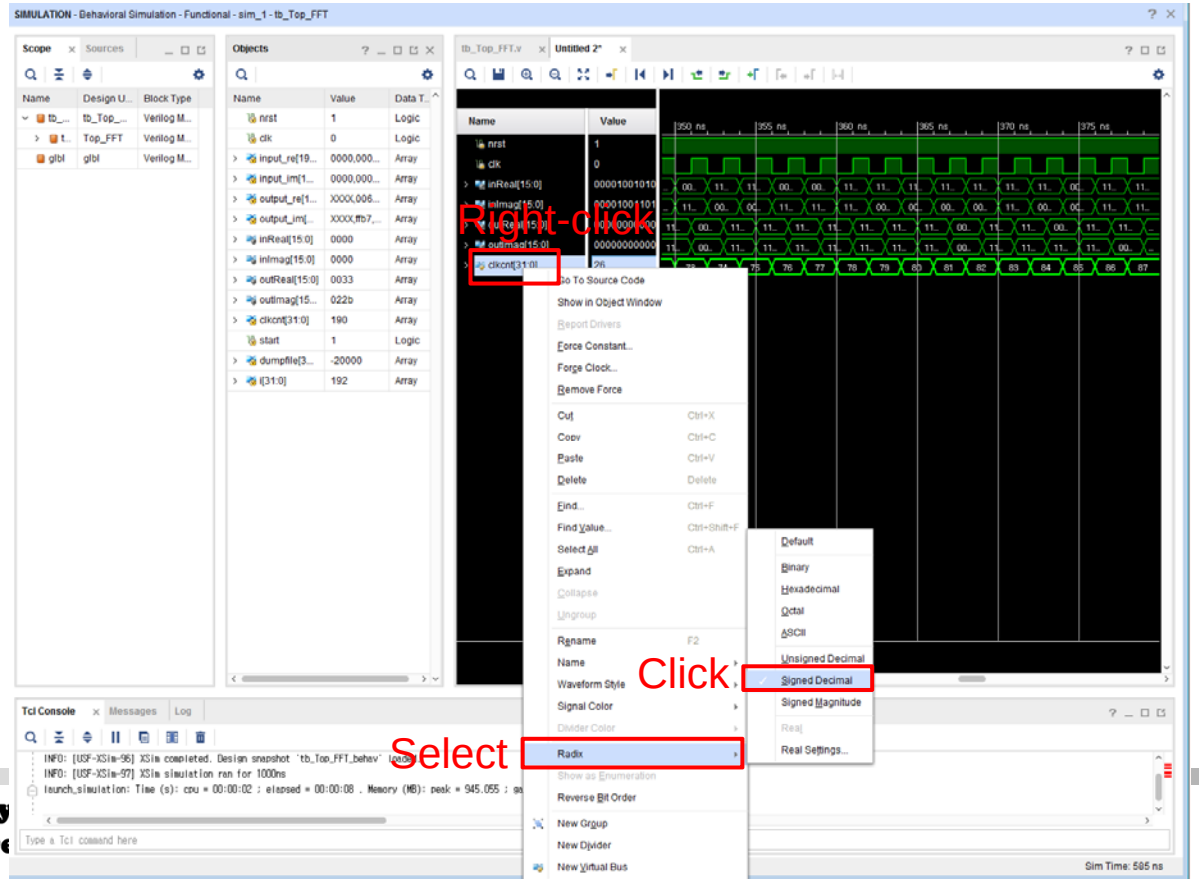
Running Behavioral Simulation

- Run Behavioral Simulation (cont'd)
  - Check that a breakpoint is added on ***'$stop'***
  - Click the waveform viewer (e.g., ***'Untitled 1'***)



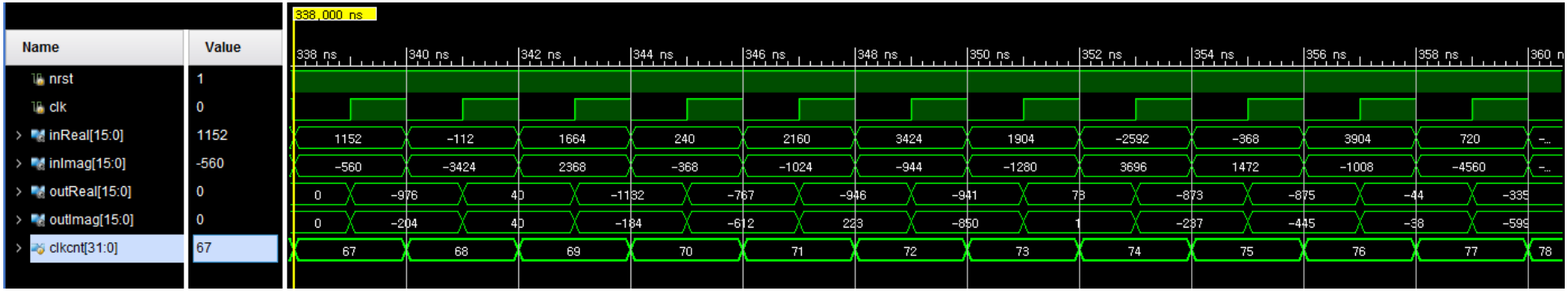
Running Behavioral Simulation

- Check simulation results
  - Select a multi-bit signal and then right-click ***'Radix'>'Signed Decimal'***
  - Zoom in/out by scrolling up/down while pressing the *Ctrl* key.



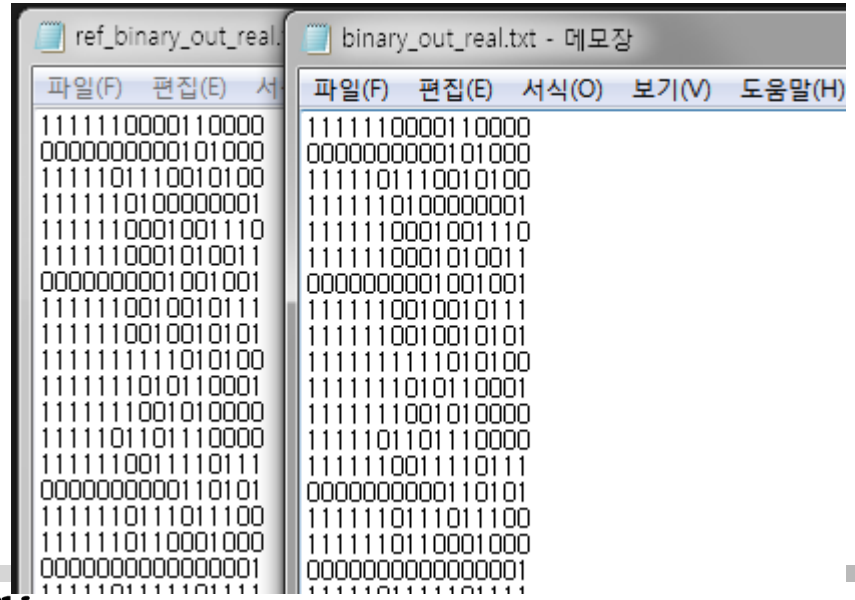
Running Behavioral Simulation

- Check simulation results (cont'd)
  - Check the output signals **outReal** and **outImag**.



Running Behavioral Simulation

- Check simulation results (cont'd)
  - Testbench outputs
    - ✓ '***binary_out_real.txt'*** and ***'binary_out_imag.txt'*** located in {Project folder path}\{project name}.sim\sim_1\behav\xsim
  - Reference outputs
    - ✓ ***'ref_binary_out_real.txt'*** and ***'ref_binary_out_imag.txt'*** located in '***Lab_SD2019_2_prob.zip***'



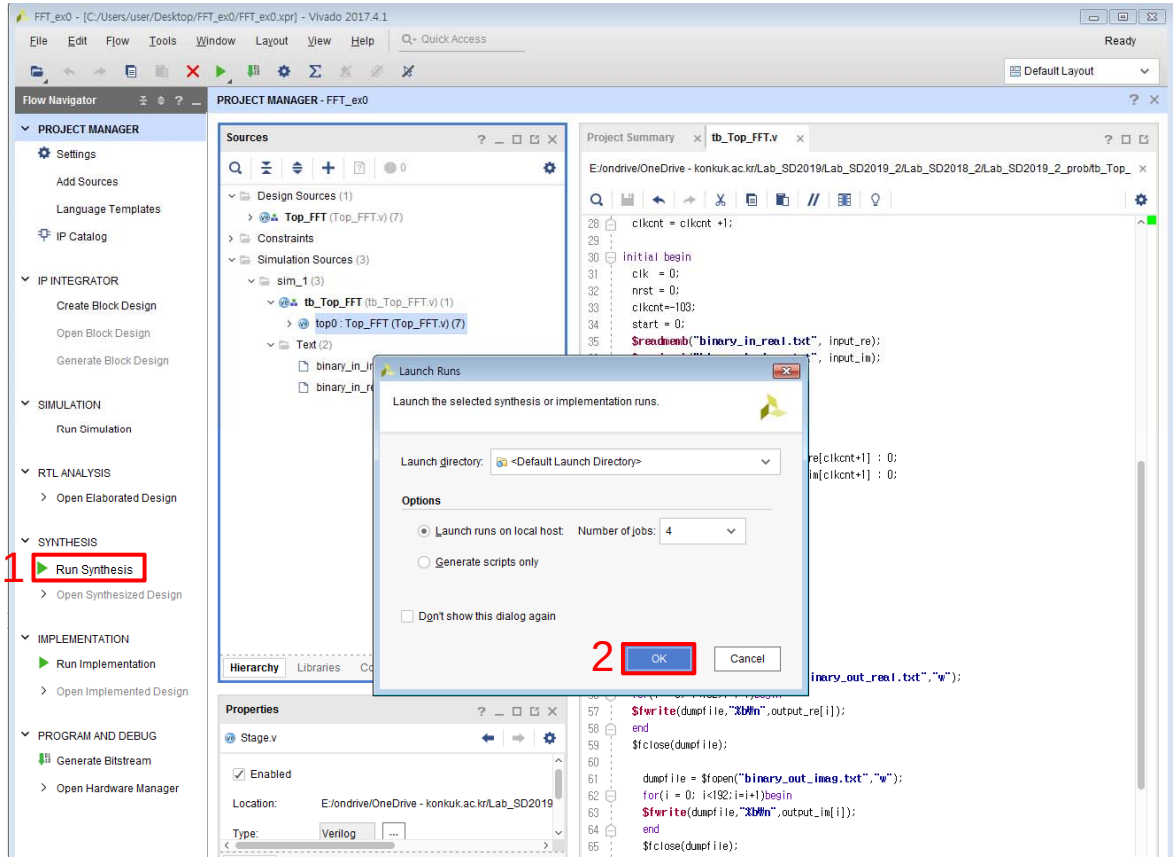
Running Synthesis

- Synthesis is a process by which a set of **RTL source codes** is converted into the equivalent **netlist**.

- Synthesized design consists of instances of modules/entities such as
  - LUTs, flip-flops, carry chain elements, wide MUXs
  - Block RAMs, DSP cells
  - Clocking elements (BUFG, BUFR, MMCM, …)
  - I/O elements (IBUF, OBUF, I/O flip-flops)

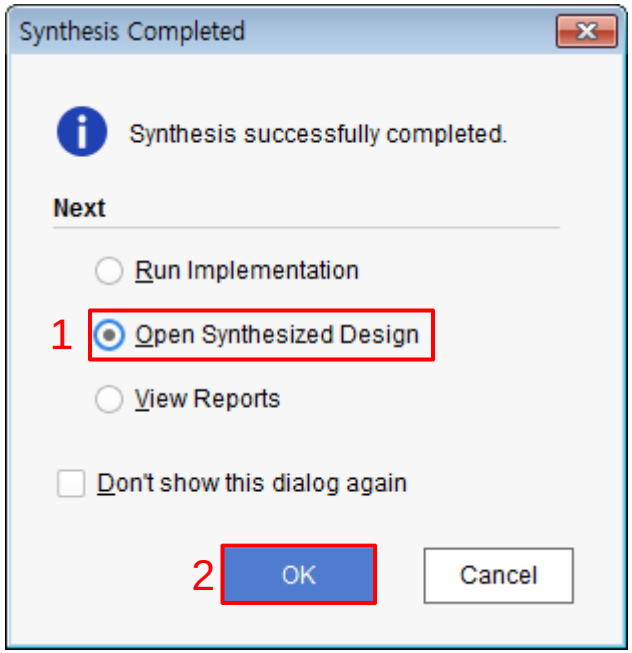
Running Synthesis

- Run Synthesis
  - Select ***'Synthesis' > 'Run Synthesis'*** in ***'Flow Navigator'***
  - Click ***'OK'***



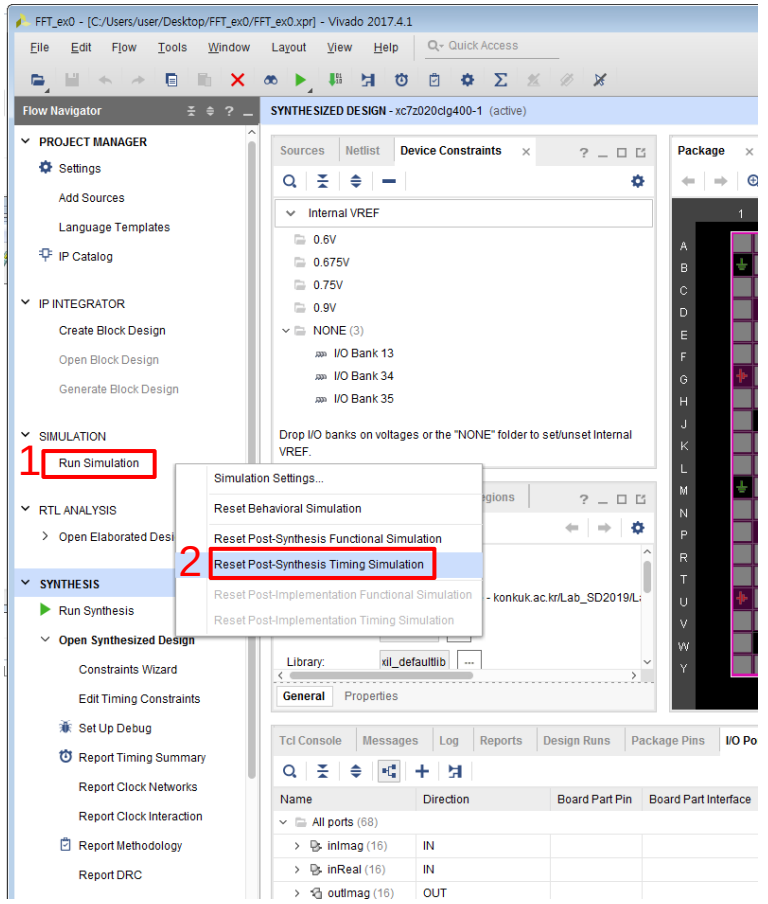
Running Synthesis

- Complete Synthesis
  - A pop-up window will appear if an error occurs. Fix the error and run Synthesis again.
  - Click ***'Open Synthesis Design'*** and then click ***'OK'***



Running Synthesis

- Run Post-Synthesis Timing Simulation
  - Click ***'Run Simulation' >'Run Post-Synthesis Timing Simulation'***

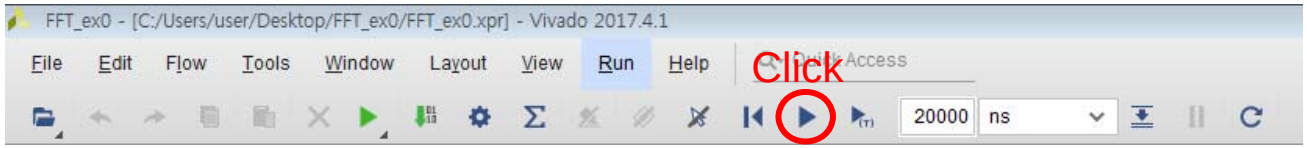


Running Synthesis

- Run Post-Synthesis Timing Simulation (cont'd)
  - Change the clock period to 20 ns and run the simulations again
  - If the simulation is not completed, click ***'Run All'***.

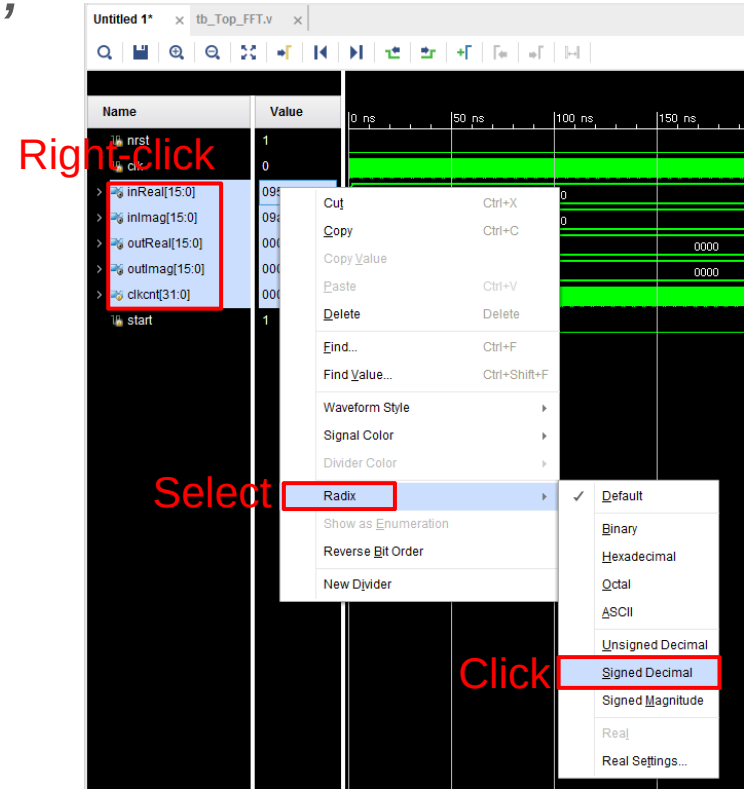
tb_Top_FFT.v

```
`timescale 1ns/1ps
`define p 2
module tb_Top_FFT;

reg nrst,clk;


reg [15:0] input_re[191:0];
reg [15:0] input_im[191:0];
```



Running Synthesis

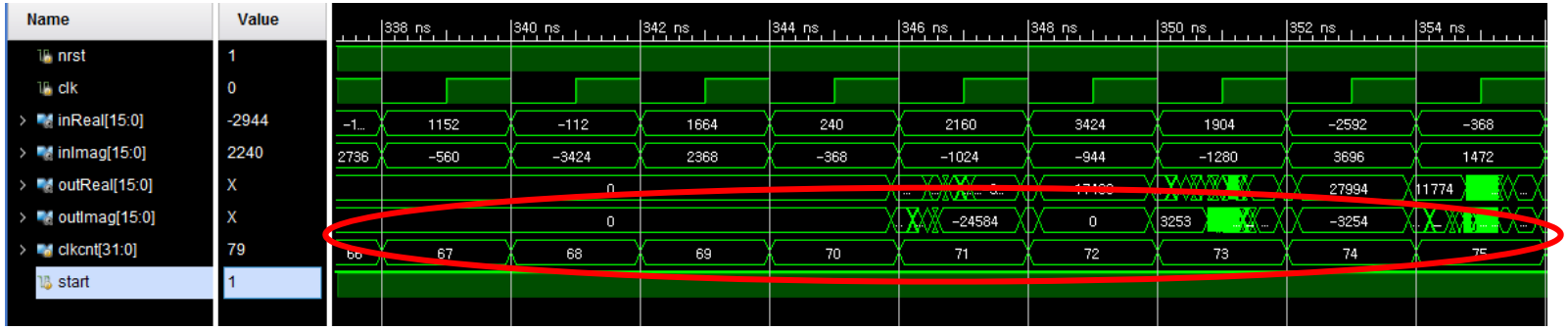
- Run Post-Synthesis Timing Simulation (cont'd)
  - Right-click signal name.
    (***inReal, inImag, outReal, outImag*** or ***clkcnt***)
  - Select ***'Radix > Signed Decimal'***



Running Synthesis

- Check simulation results
  - Check whether the simulation results match the behavioral simulations.

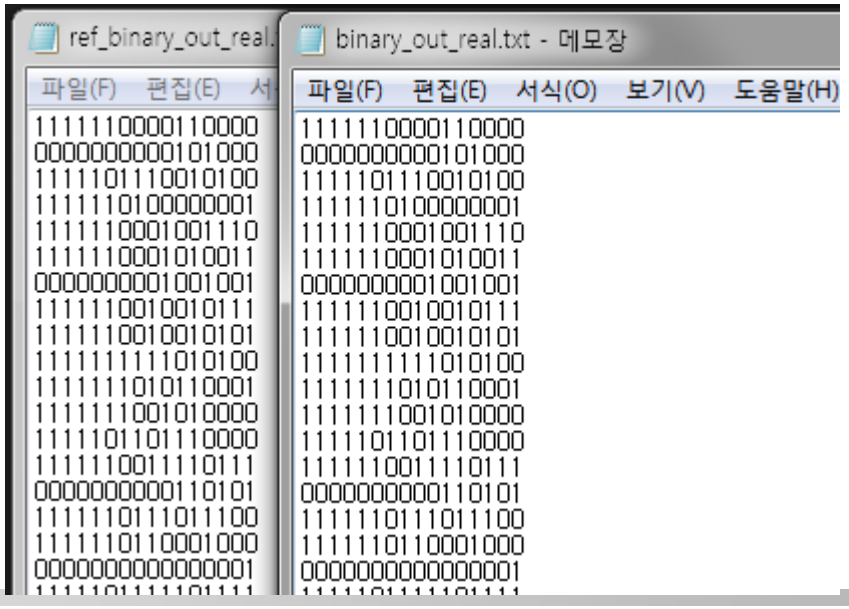
Period: 2 ns (Wrong)



Period: 20 ns (Correct)

Running Synthesis

- Check simulation results (cont'd)
  - Testbench outputs
    - ✓ '***binary_out_real.txt'*** and ***'binary_out_imag.txt'*** located in {Project folder path}\{project name}.sim\sim_1\synth\xsim
  - Reference outputs
    - ✓ ***'ref_binary_out_real.txt'*** and ***'ref_binary_out_imag.txt'*** located in '***Lab_SD2019_2_prob.zip***'

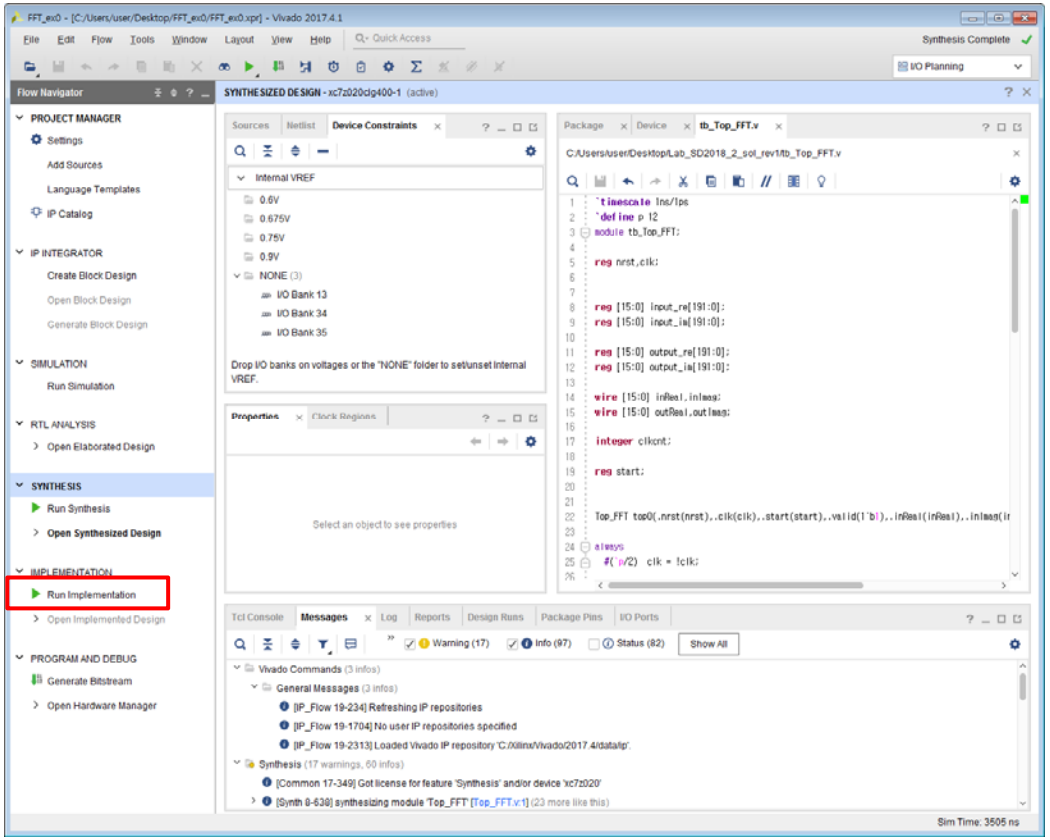


Running Implementation

- Implementation is a process by which the **cells** and **nets** from synthesis are physically **placed** and **routed** in the FPGA.
- Implement Design is similar to Run Synthesis
- Implement Design uses one of the options
  - Opt Design (default)
  - Power Opt Design
  - Place Design
  - Post-Place Power Opt Design
  - Phys Opt Design
  - Route Design
  - Write Bitstream

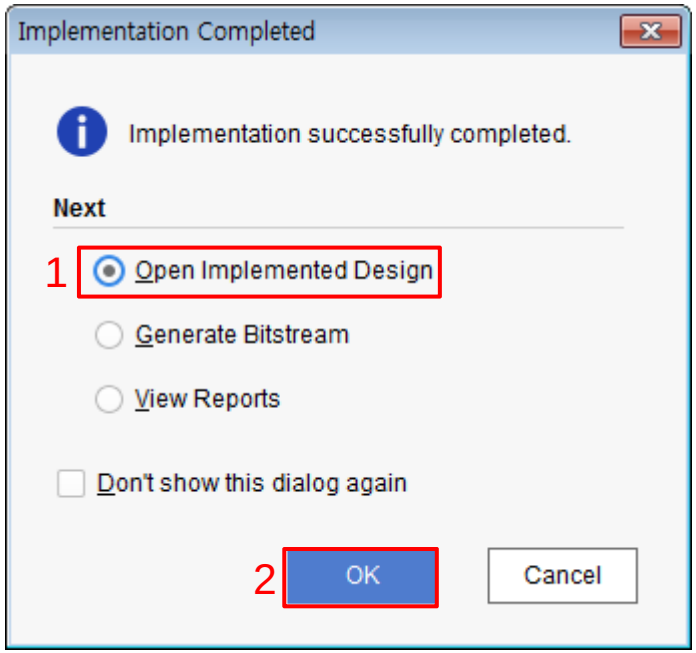
Running Implementation

- Run Implementation
  - ***'Implementation'*** > ***'Run Implementation'*** in ***'Flow Navigator'***



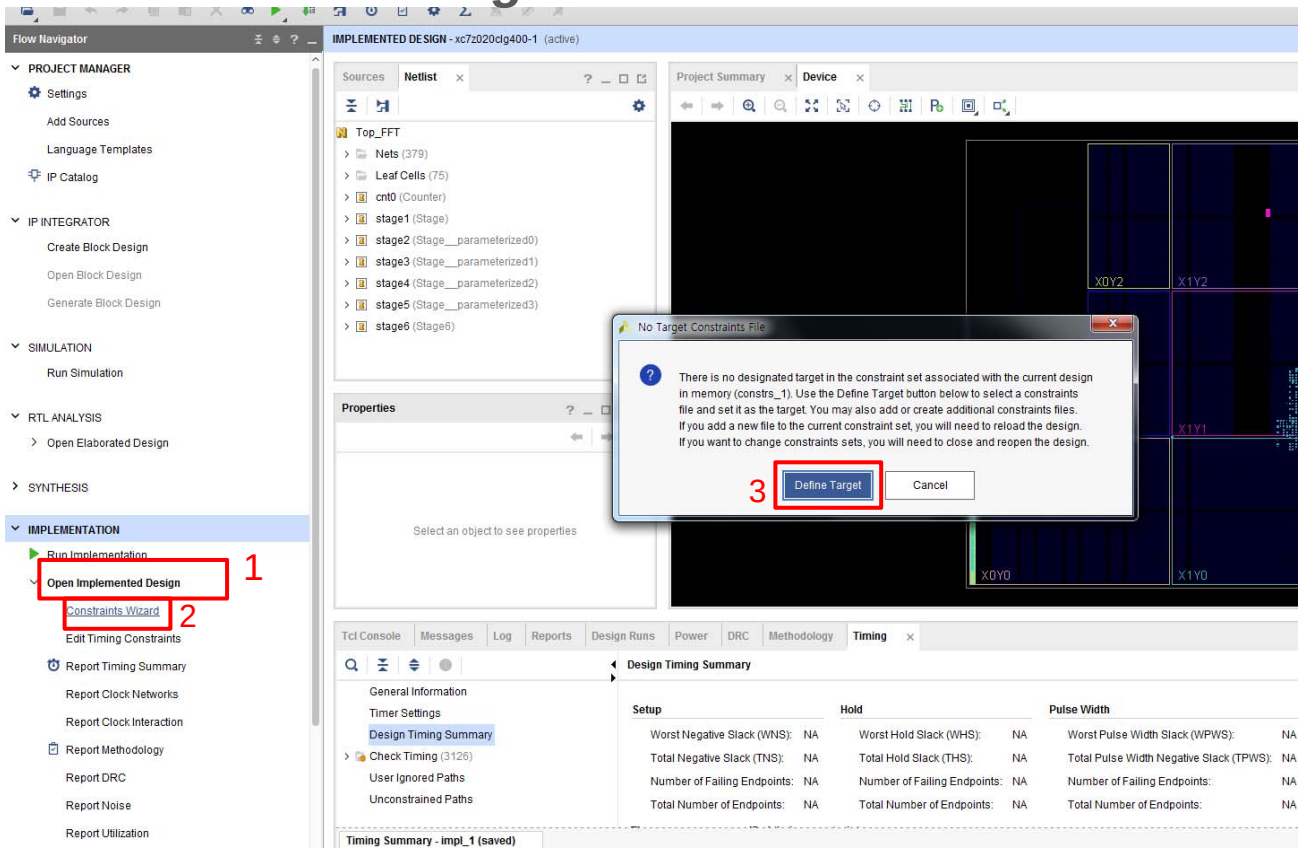
Running Implementation

- Complete Implementation
  - A pop-up window will appear if an error occurs. Fix the error and run Implementation again.
  - Select ***'Open Implementation Design'*** and then click ***'Ok'***



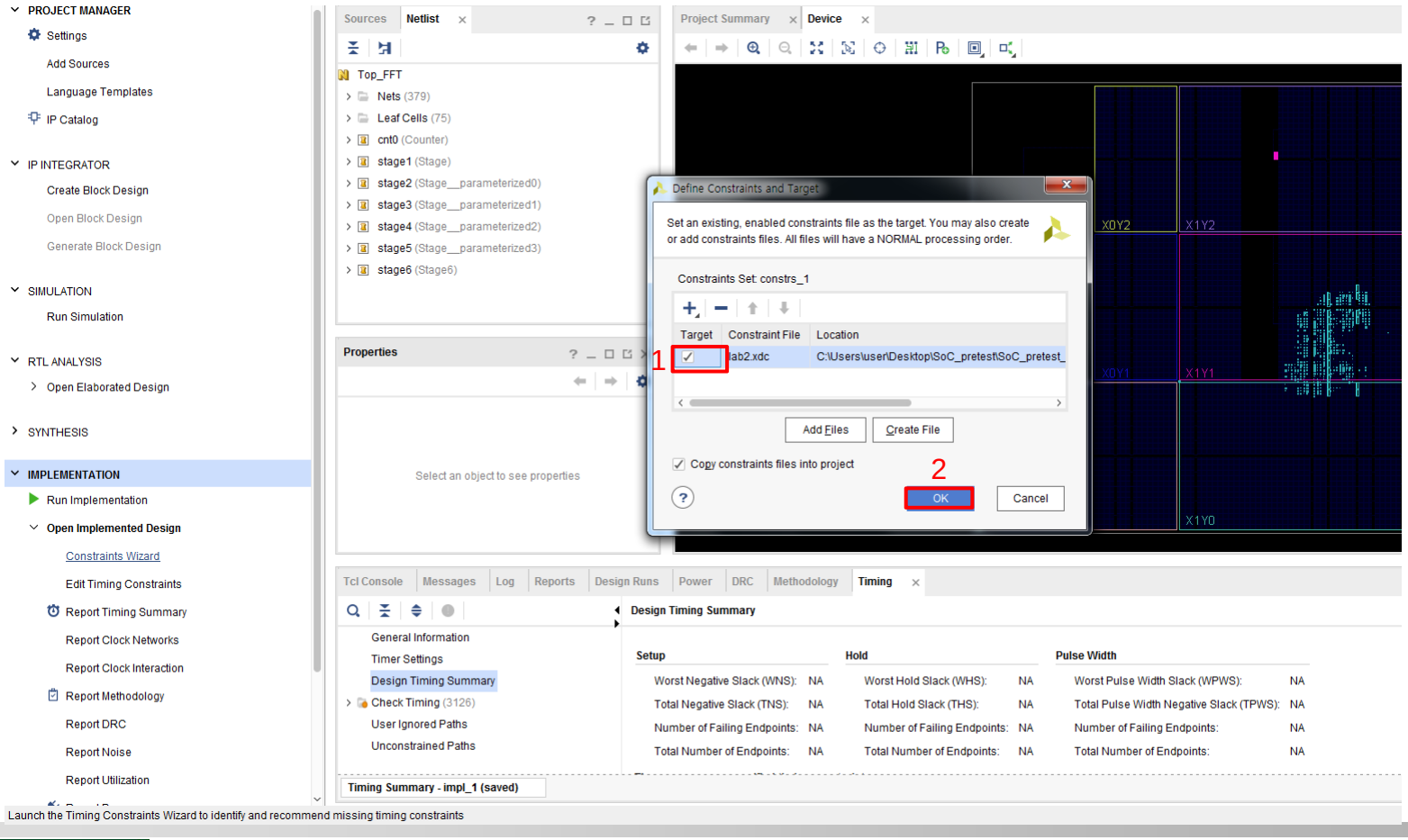
Running Implementation

- Set the constraint
  - Click ***'Open Implemented Design' > 'Constraints Wizard' > 'Define Target'***



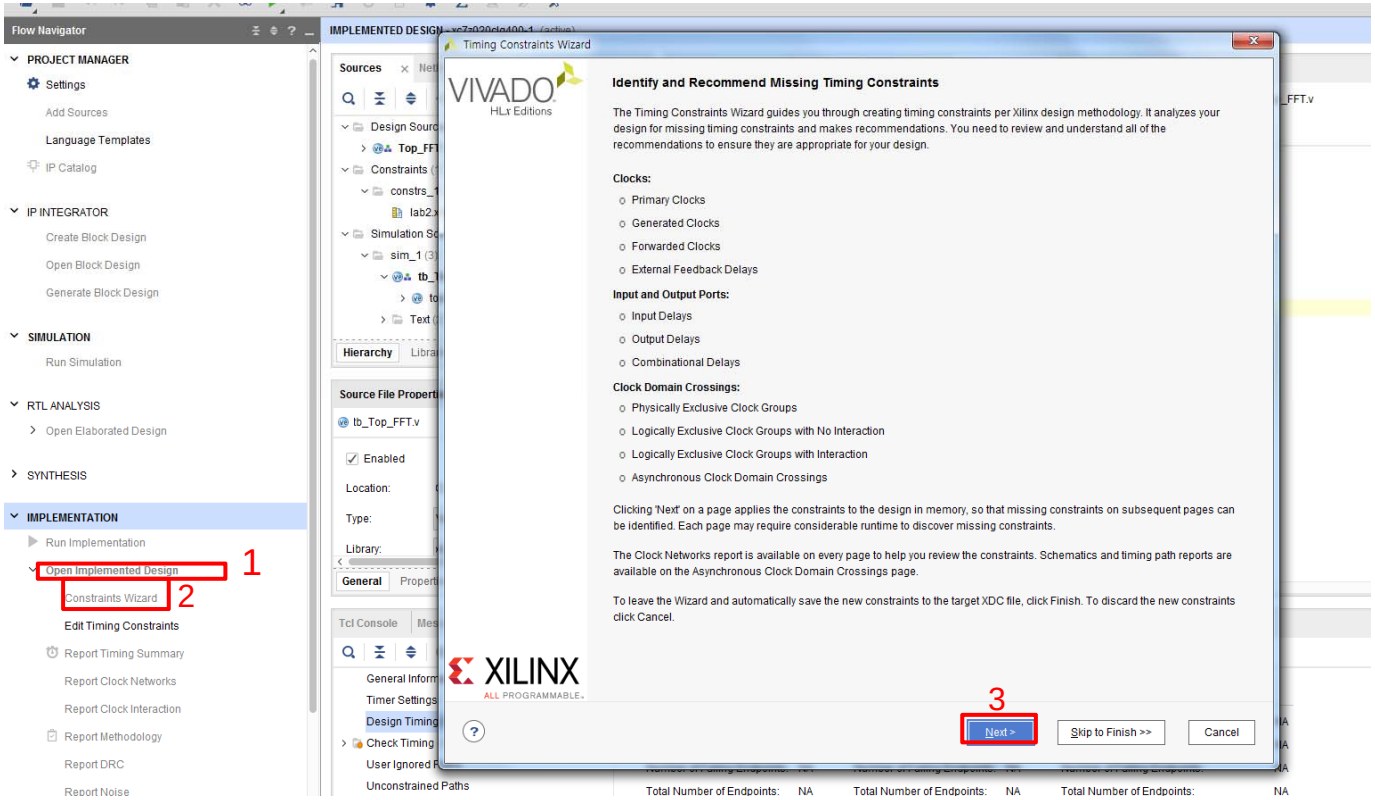
Running Implementation

- Set the constraint (cont'd)
  - Choose ***'Target'*** and then click ***'OK'***



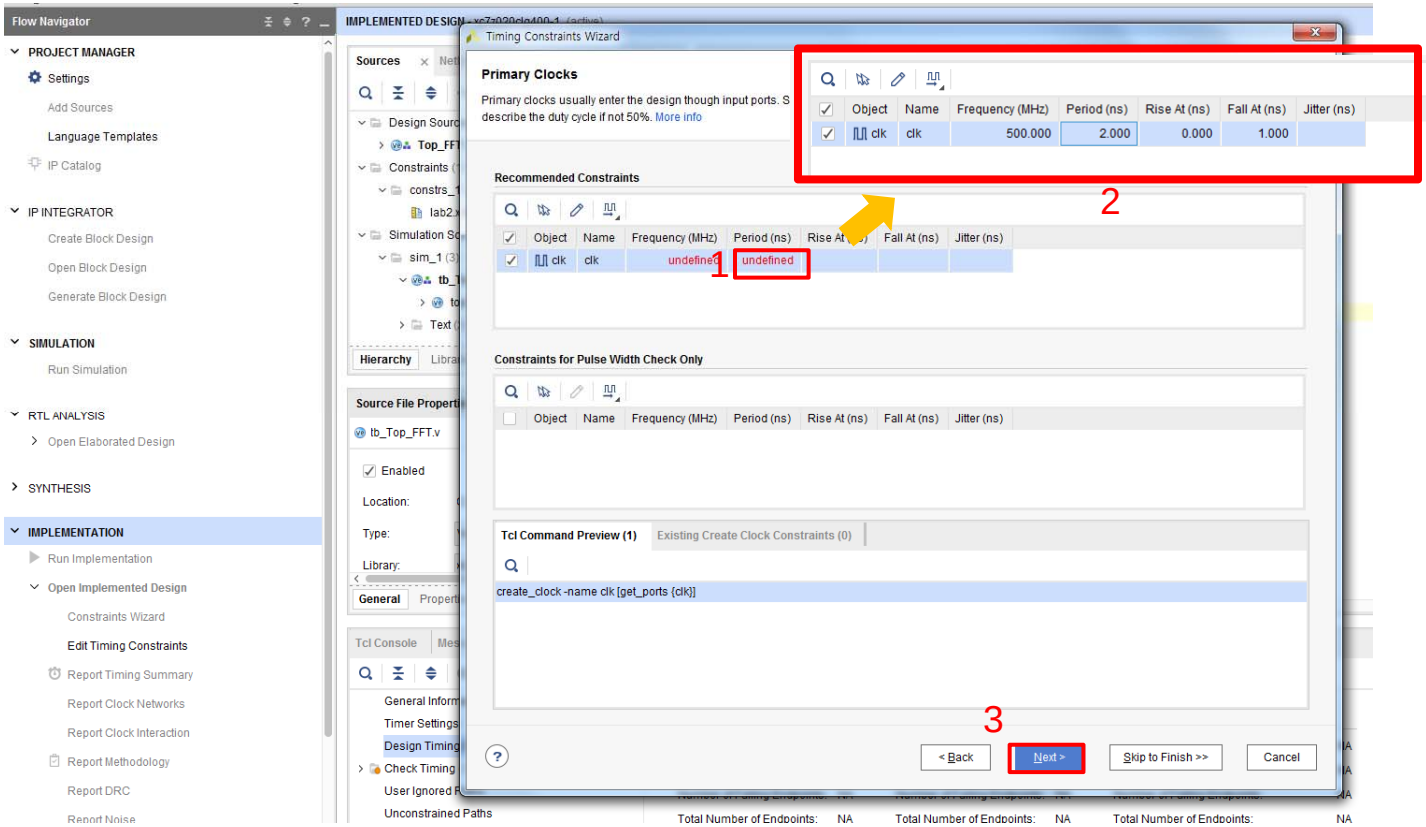
Running Implementation

- Set the constraint (cont'd)
  - Click ***'Open Implemented Design' > 'Constraints Wizard' > 'Next'***



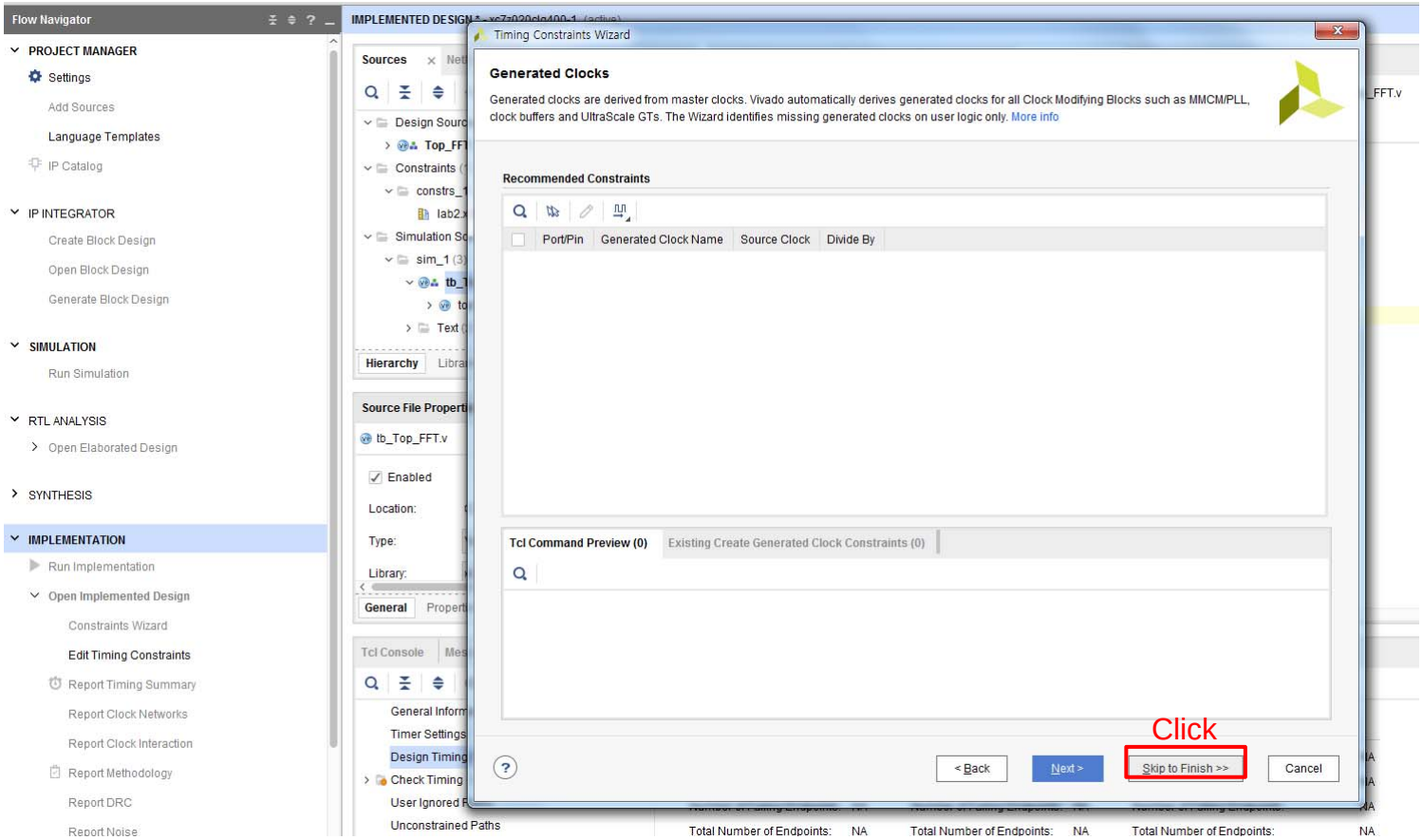
Running Implementation

- Set the constraint (cont'd)
  - Type '***2 ns'*** in the ***'Period'*** tap and then click ***'Next'***



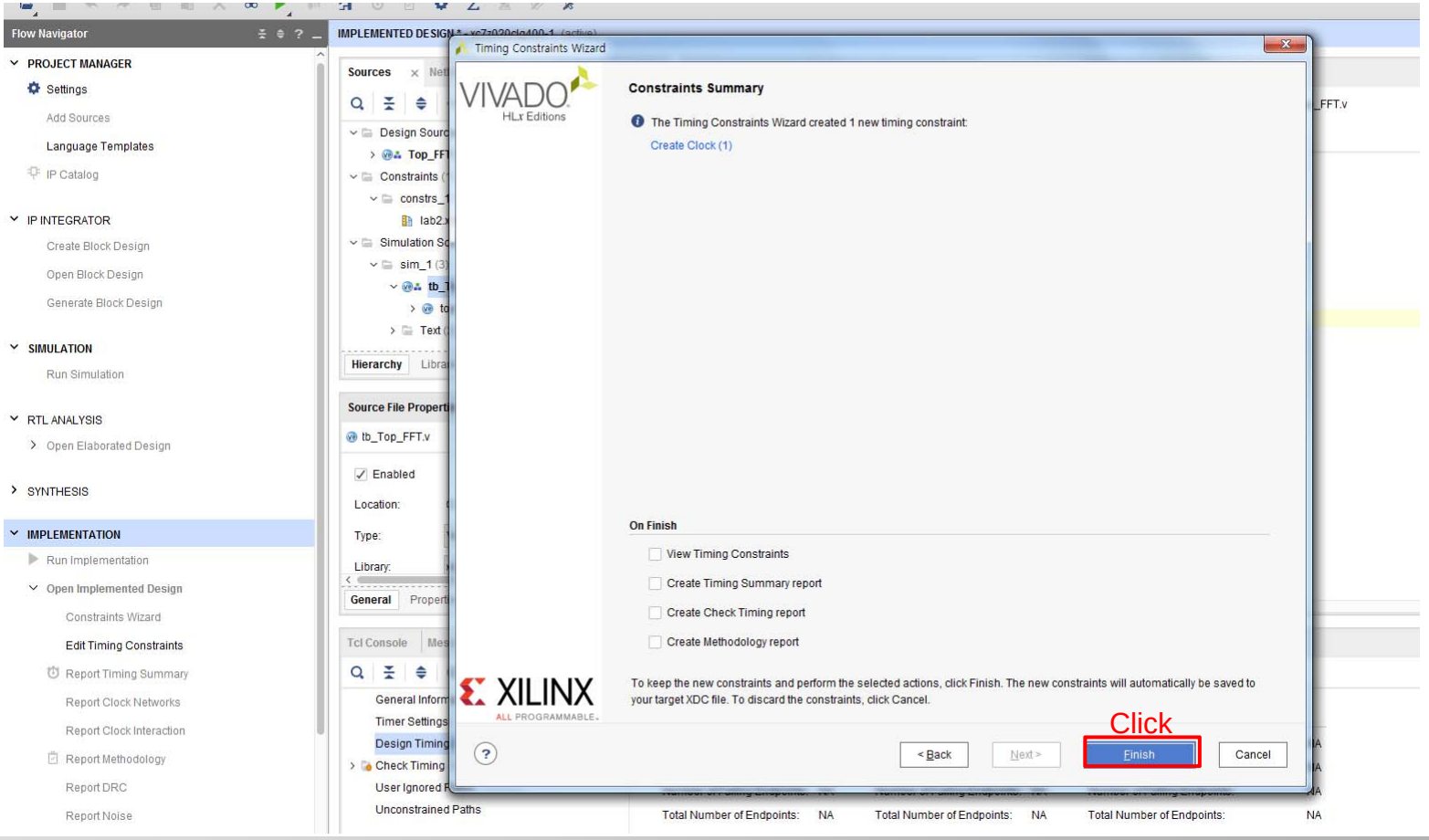
Running Implementation

- Set the constraint (cont'd)
  - Click ***'Skip to Finish'***



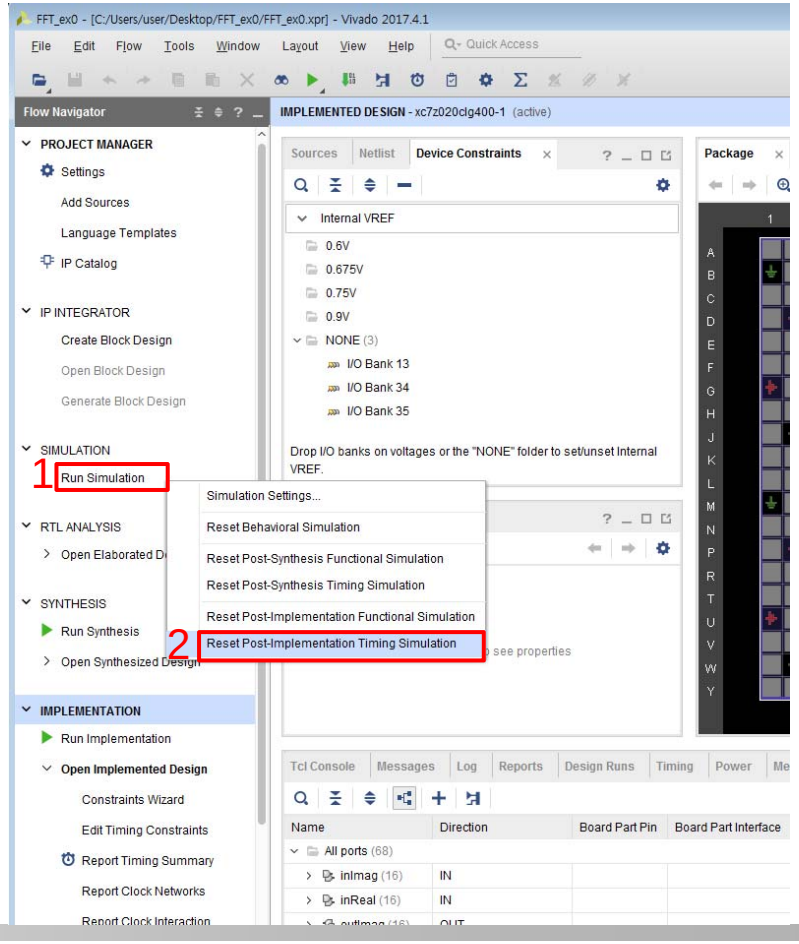
Running Implementation

- Set the constraint (cont'd)
  - Click ***'Finish'***



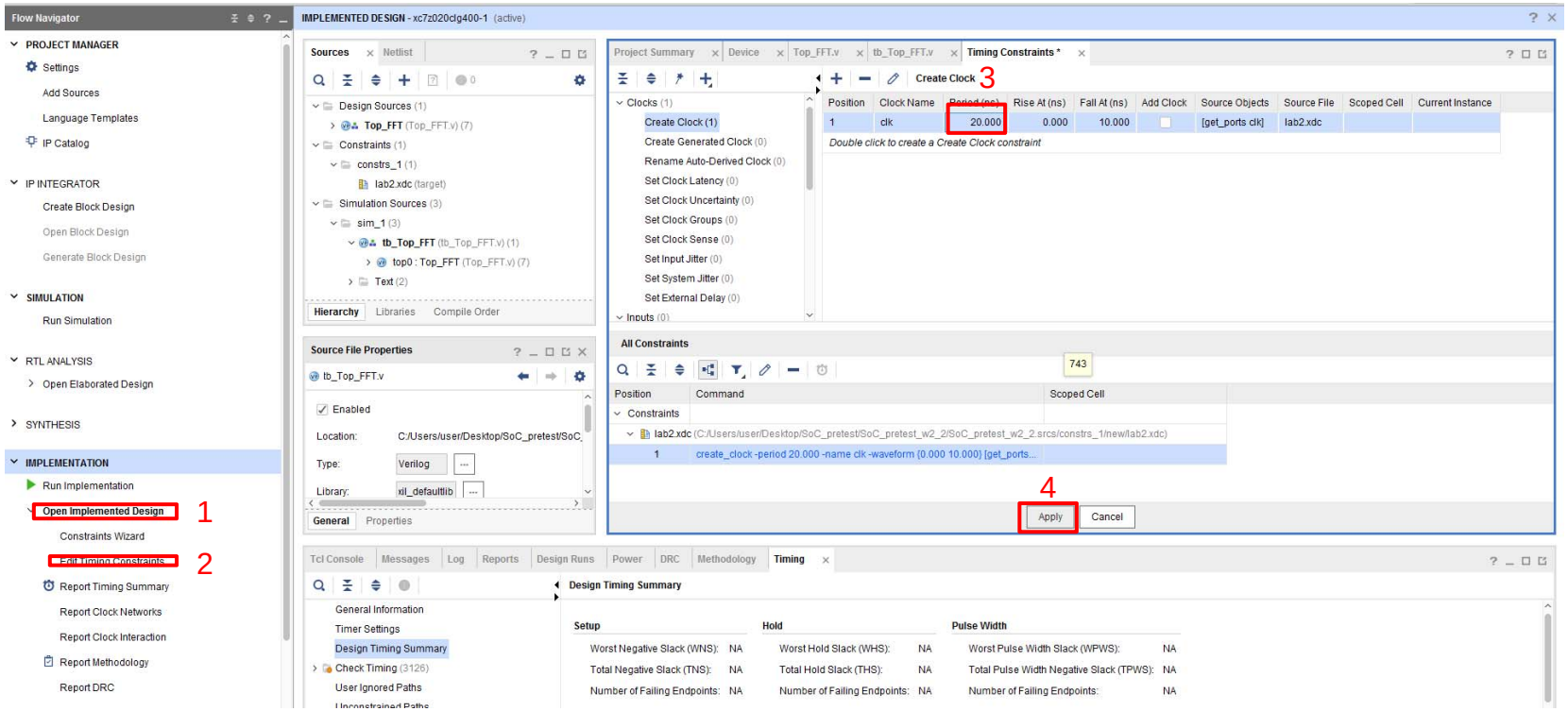
Running Implementation

- Run Post-Implementation Timing Simulation
  - Click ***'Run Simulation' >'Run Post-Implementation Timing Simulation'***



Running Implementation

- Run Post-Implementation Timing Simulation (cont'd)
  - Click ***'Open Implemented Design' > 'Edit Timing Constraints'***,
  - Type ***'20 ns'*** in the ***'Period'*** tap and then click ***'Apply'***

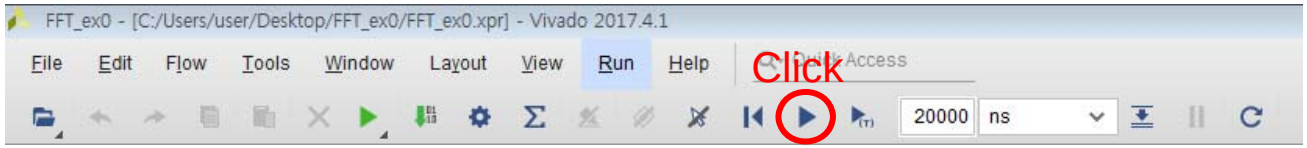


Running Implementation

- Run Post-Implementation Timing Simulation (cont'd)
  - Change the clock period to 20 ns and run the simulations again
  - If the simulation is not completed, click ***'Run All'***.

tb_Top_FFT.v

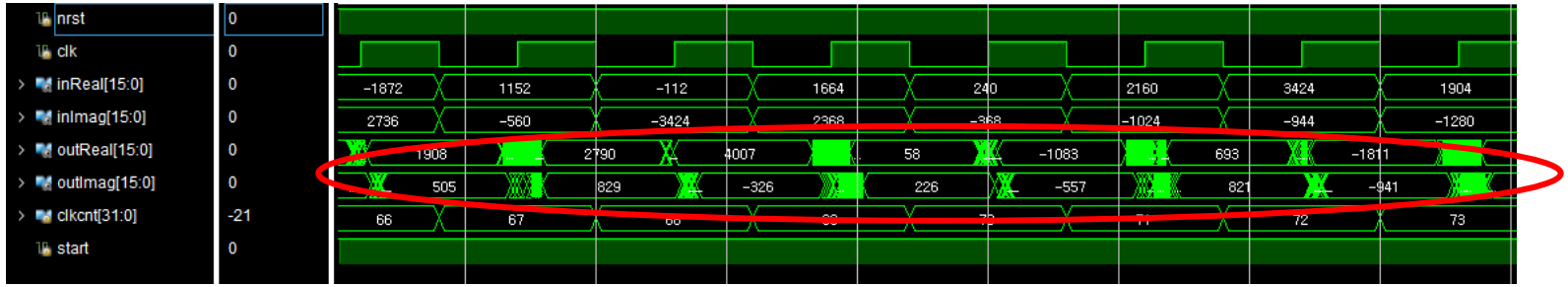
```
`timescale 1ns/1ps
`define p 2
module tb_Top_FFT;

reg nrst,clk;


reg [15:0] input_re[191:0];
reg [15:0] input_im[191:0];
```



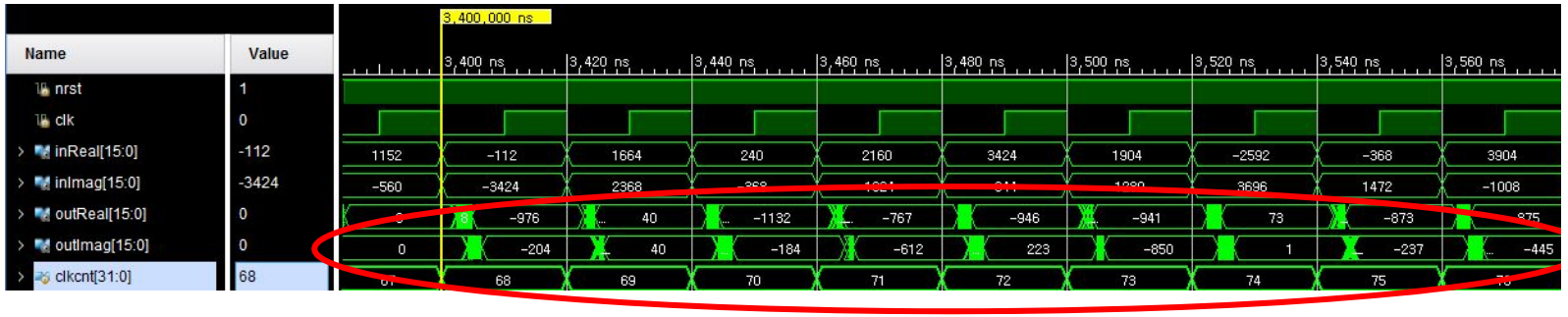
Running Implementation

- Check simulation results
  - Check whether the simulation results match the behavioral simulations.

Period: 2 ns (Wrong)

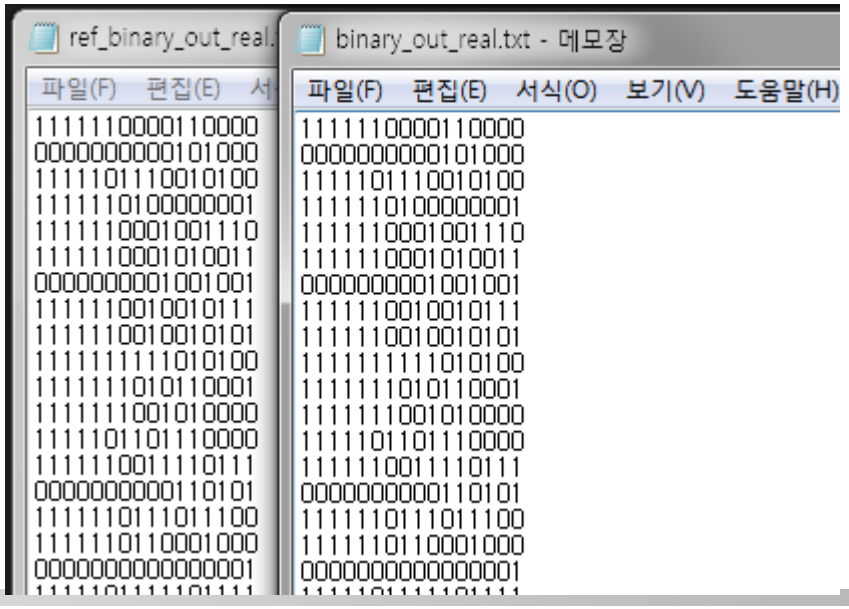


Period: 20 ns (Correct)



Running Implementation

- Check simulation results (cont'd)
  - Testbench outputs
    - ✓ '***binary_out_real.txt'*** and ***'binary_out_imag.txt'*** located in {Project folder path}\{project name}.sim\sim_1\impl\timing\xsim
  - Reference outputs
    - ✓ ***'ref_binary_out_real.txt'*** and ***'ref_binary_out_imag.txt'*** located in '***Lab_SD2019_2_prob.zip***'



Assignments

- Repeat the previous steps for each of the following designs
  - **128-pt** FFT with **interpolated** inputs
    - ✓ The same WL settings: [1,15] input, [8,8] output

    Original Input: **x[0],x[1],x[2],x[3],…, x[63]**

    Interpolated Input: **x[0],0,x[1],0,x[2],0,x[3],0,…, x[63],0**

  - 64-pt FFT with **reordering** with the **same** inputs
    - ✓ The same WL settings: [1,15] input, [7,9] output

Assignments

- Repeat the previous steps for each of the following designs (cont'd)
  - **128-pt** FFT with **reordering** with **interpolated** inputs
    - ✓ The same WL settings: [1,15] input, [8,8] output

Original Input: **x[0],x[1],x[2],x[3],…, x[63]**

Interpolated Input: **x[0],0,x[1],0,x[2],0,x[3],0,…, x[63],0**