Lab 3: PS-controlled Accelerator

Chester Sungchung Park (박성정)
SoC Design Lab, Konkuk University
Webpage: http://soclab.konkuk.ac.kr

Teaching Assistants

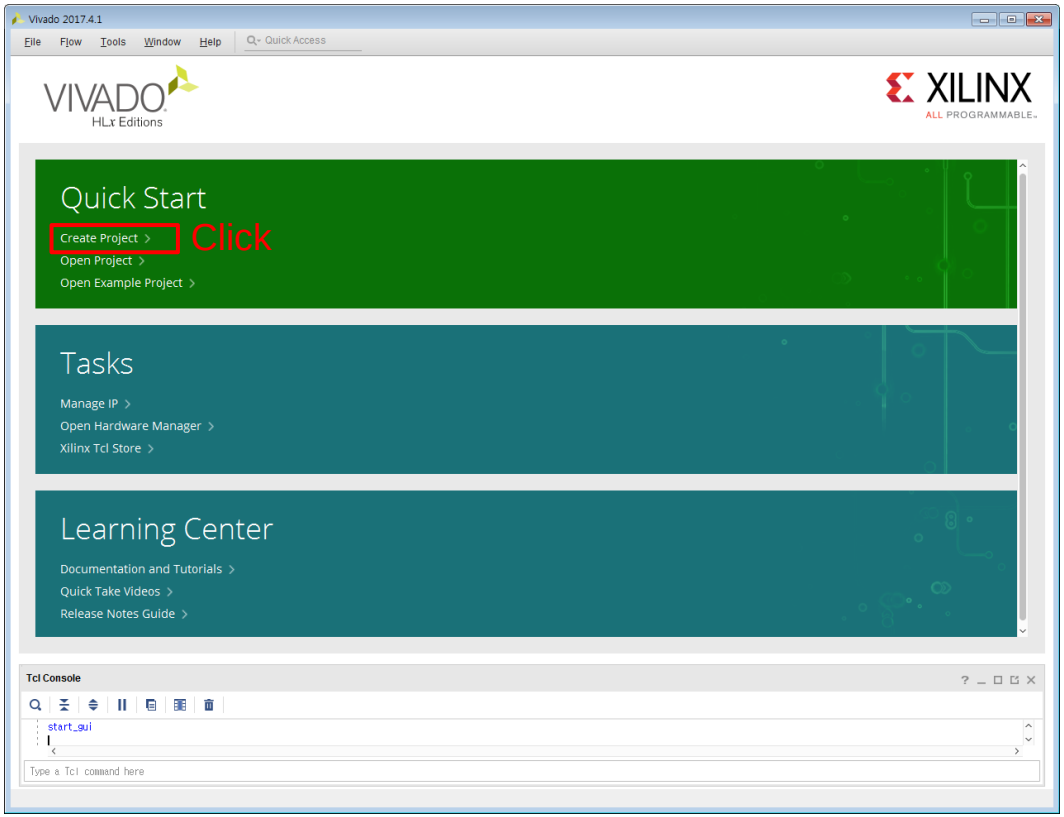
- Check the course website to see who is assigned to this lab.

Outline

- Creating IP projects
- Creating block designs
- Generating bitstream
- Running C applications
- Debugging designs in Integrated Logic Analyzer (ILA)

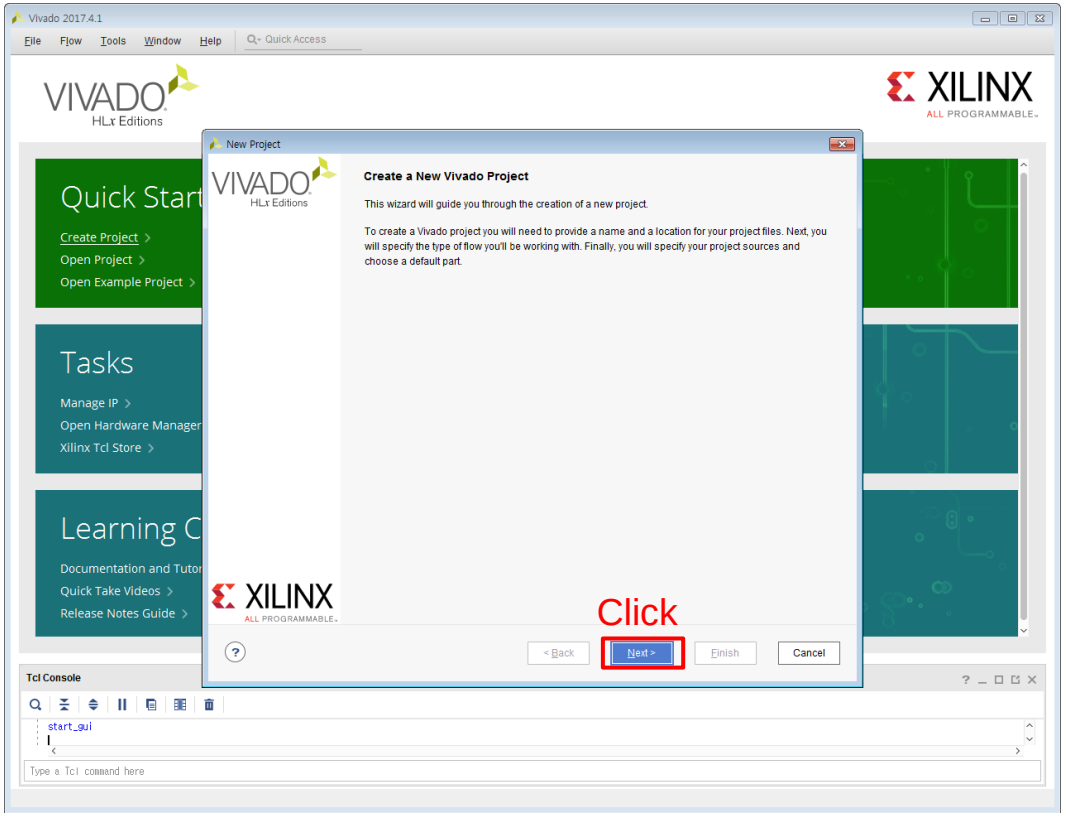
Creating RTL Projects

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



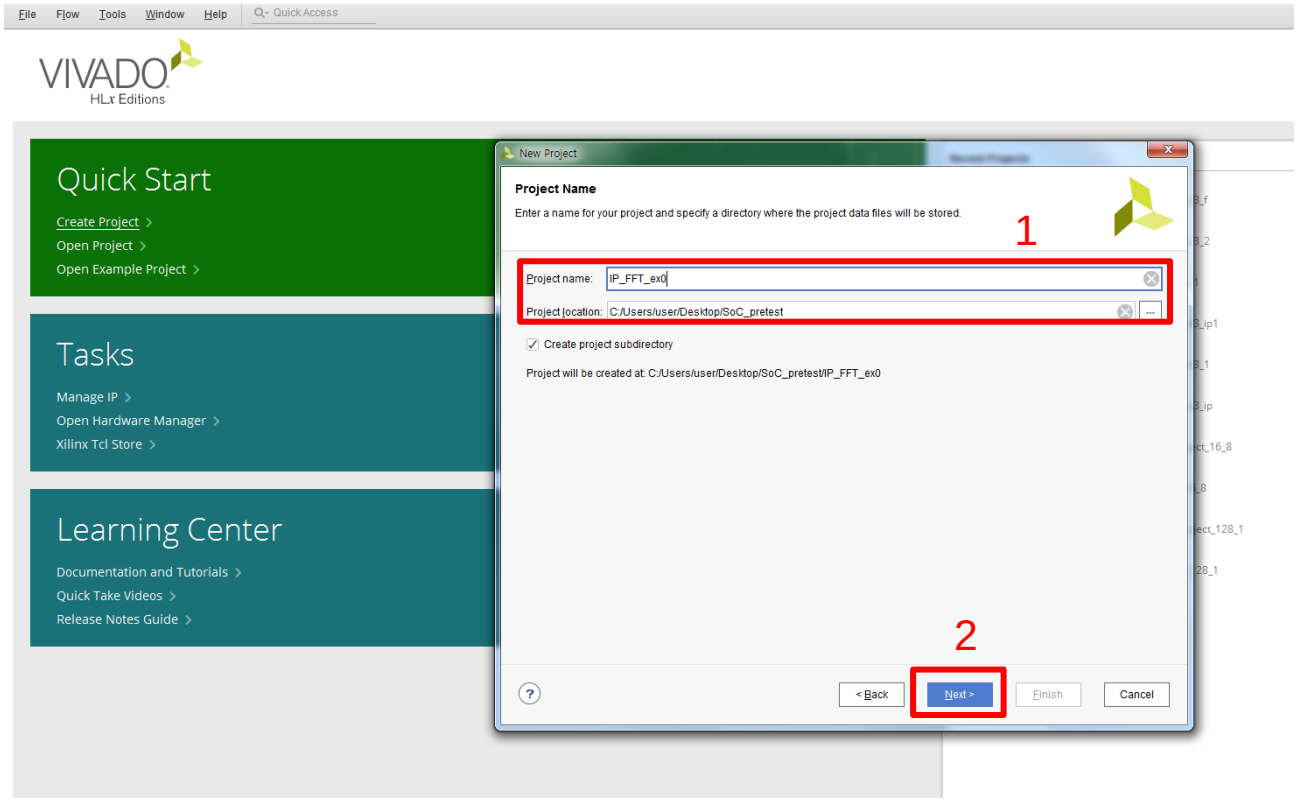
Creating RTL Projects

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



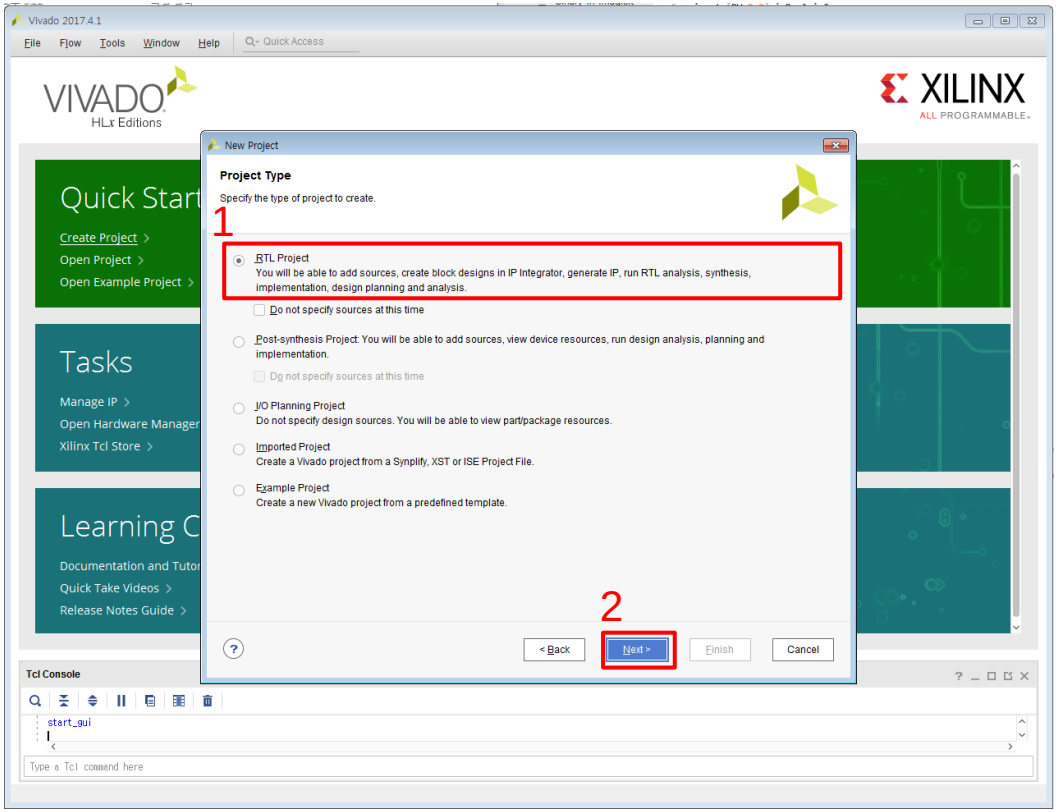
Creating IP Projects

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



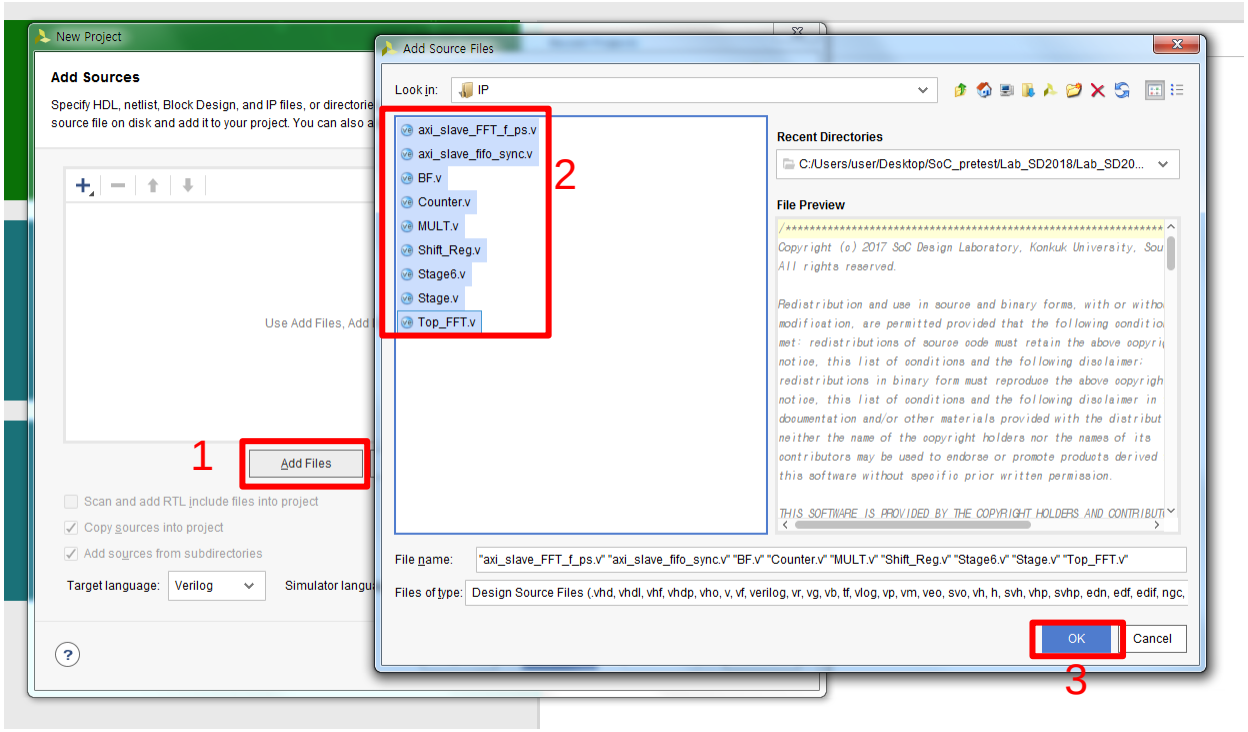
Creating IP Projects

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



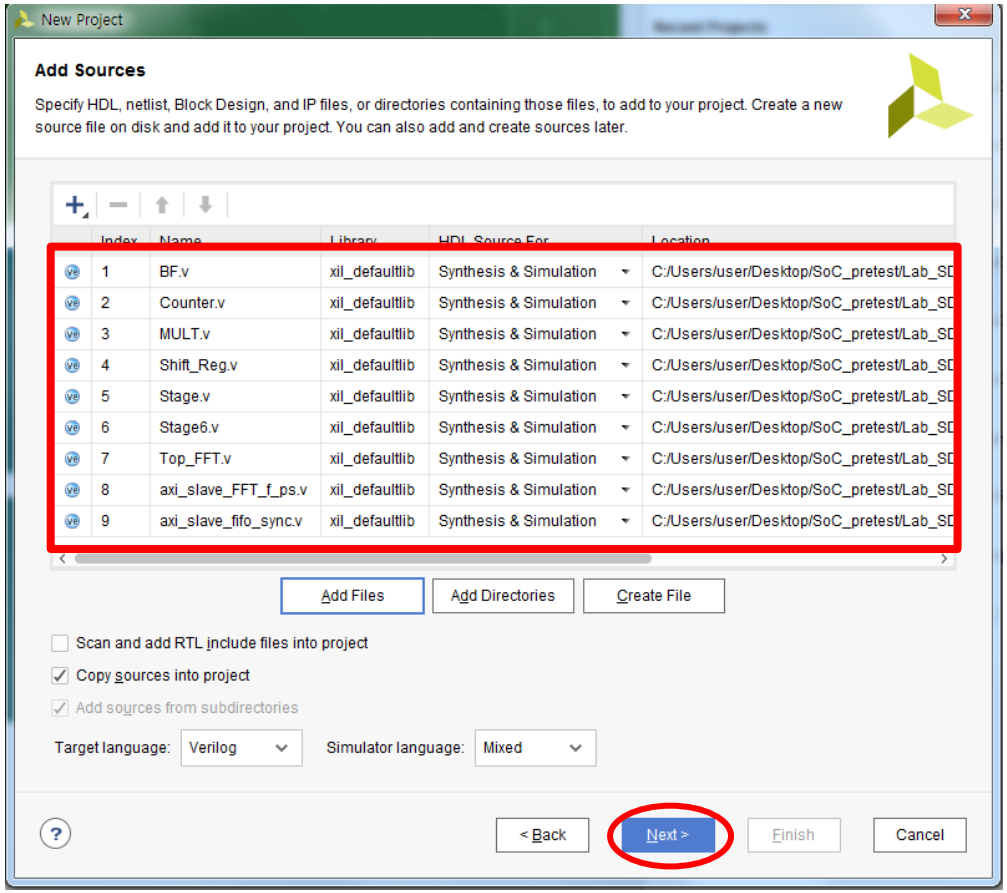
Creating IP Projects

- Add Sources
  - Click ***'Add Files'***
  - Add the HDL source files in the ***'IP'*** folder
  - Click ***'OK'***



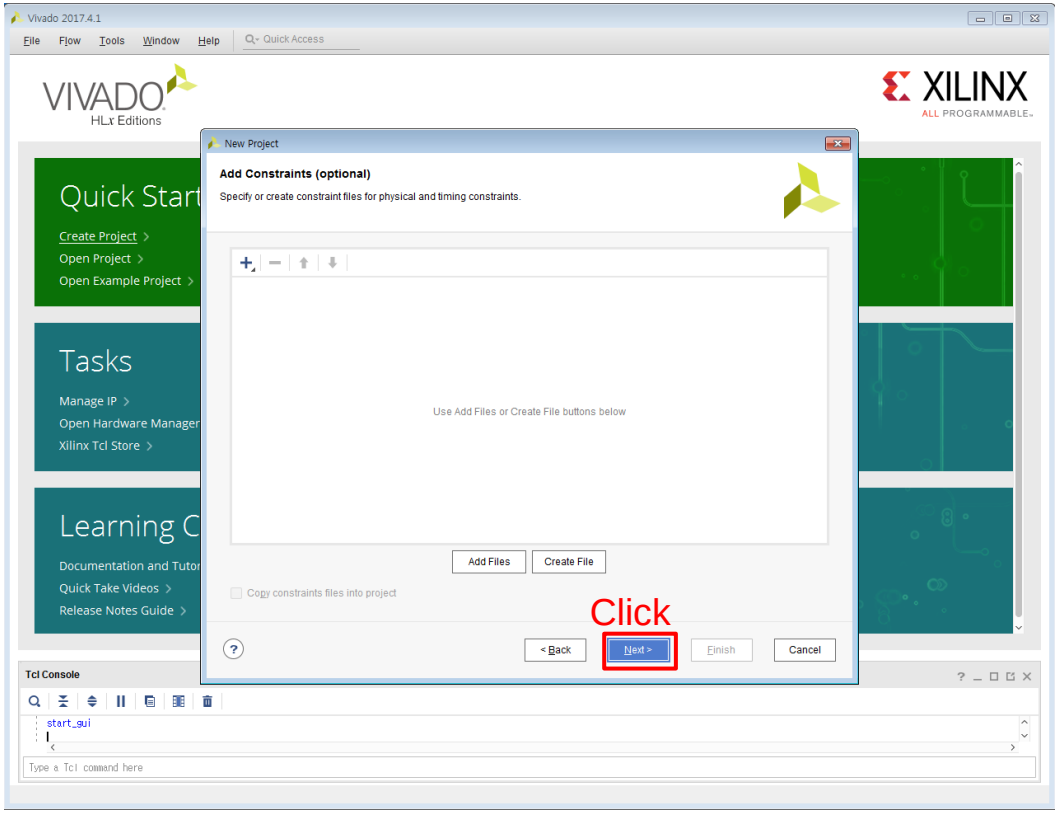
Creating IP Projects

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



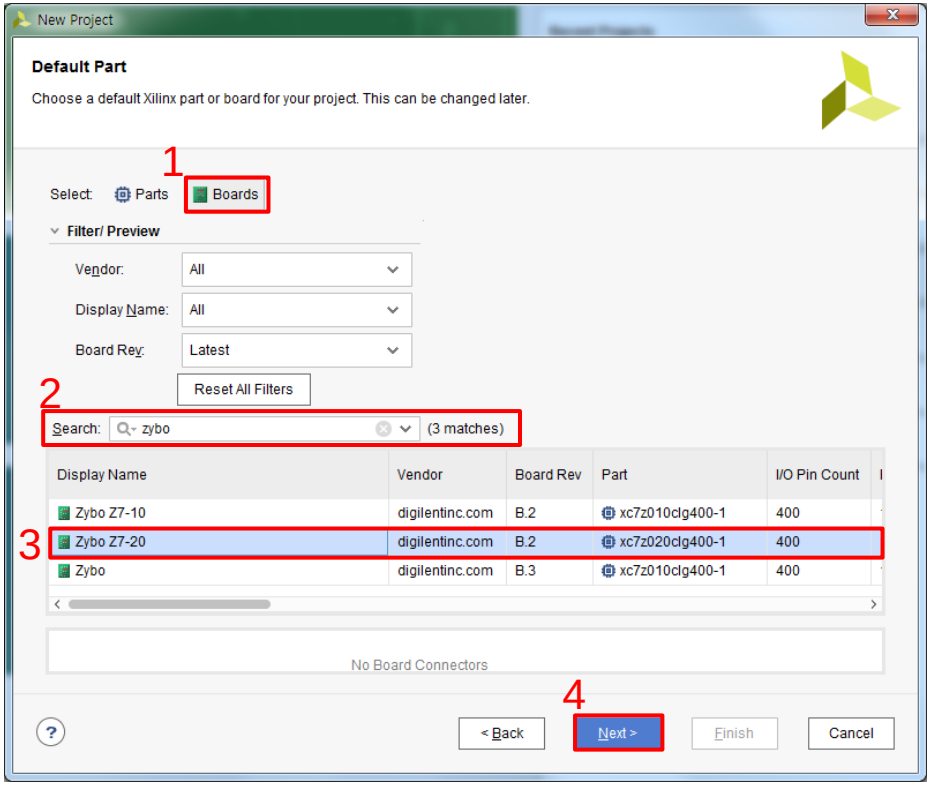
Creating IP Projects

- Add Constraints
  - Click ***'Next'***



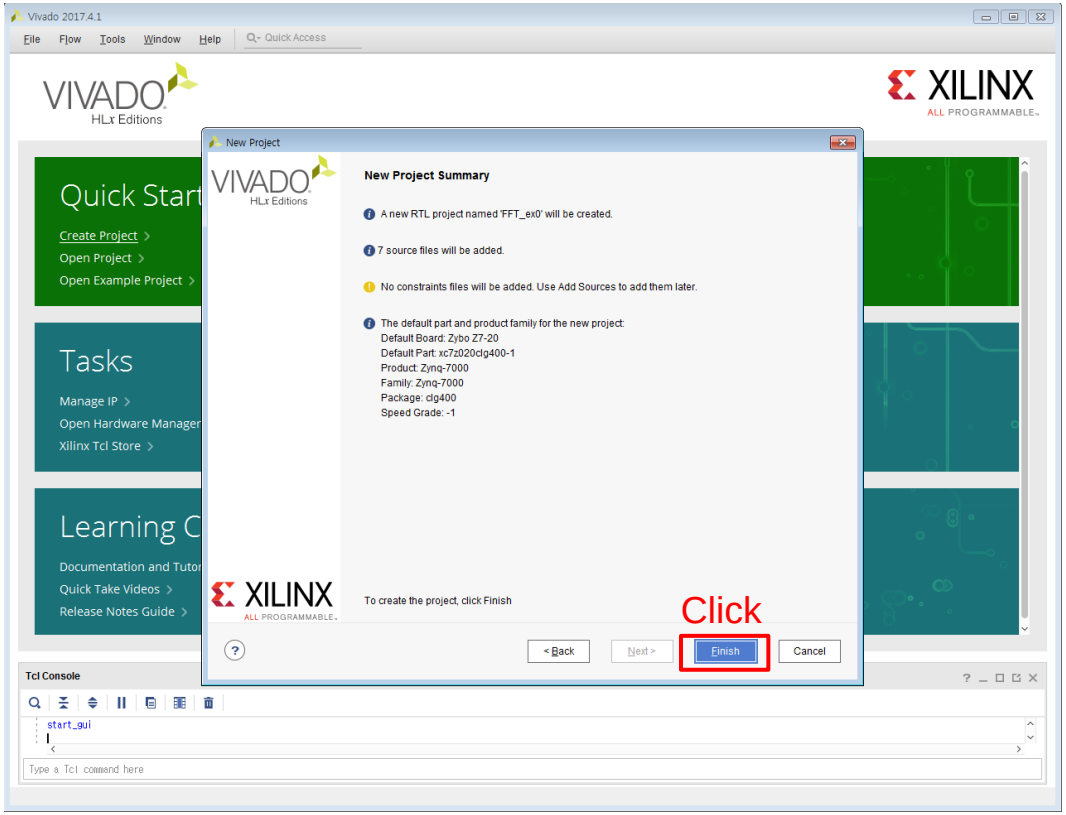
Creating IP Projects

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



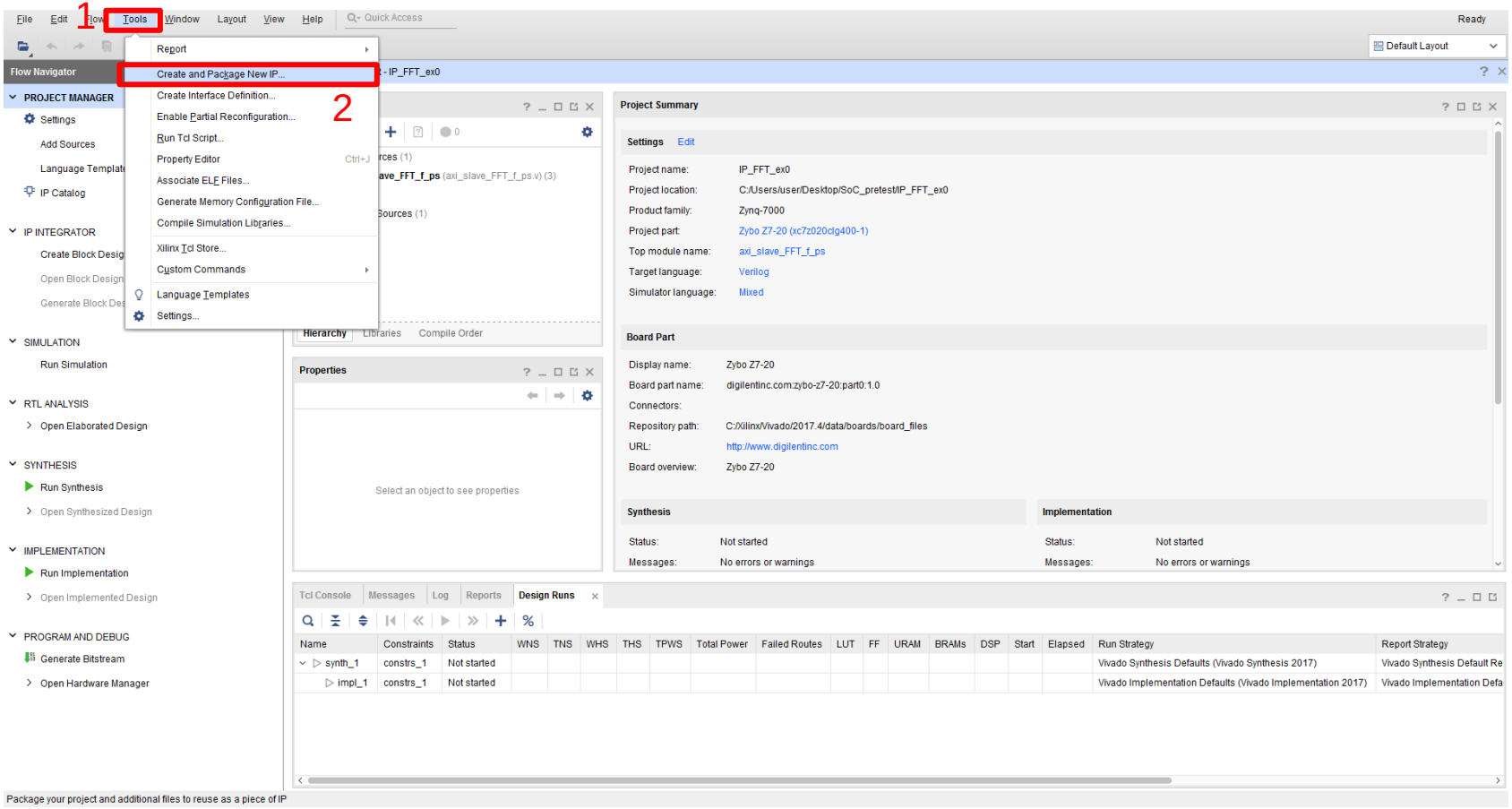
Creating IP Projects

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



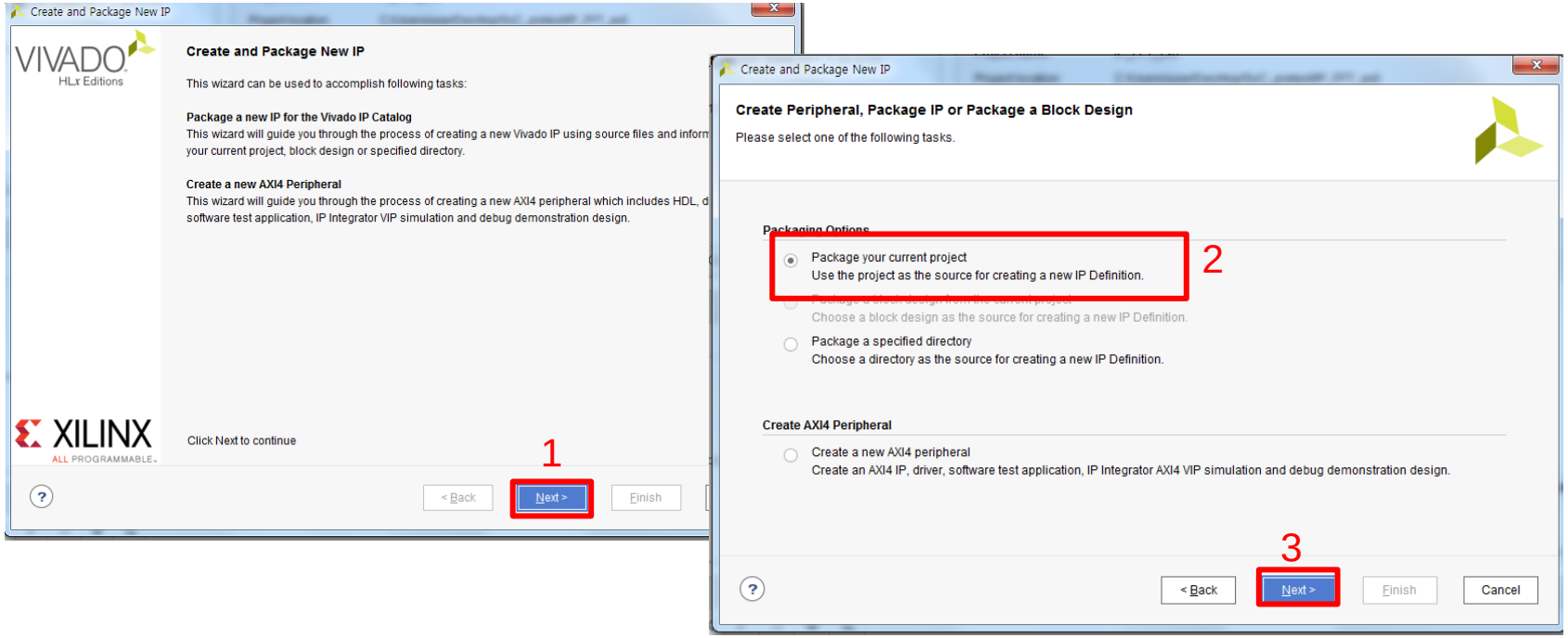
Creating IP Projects

- Create and Package IP
  - Select ***'Tools'*** > ***'Create and Package New IP'***



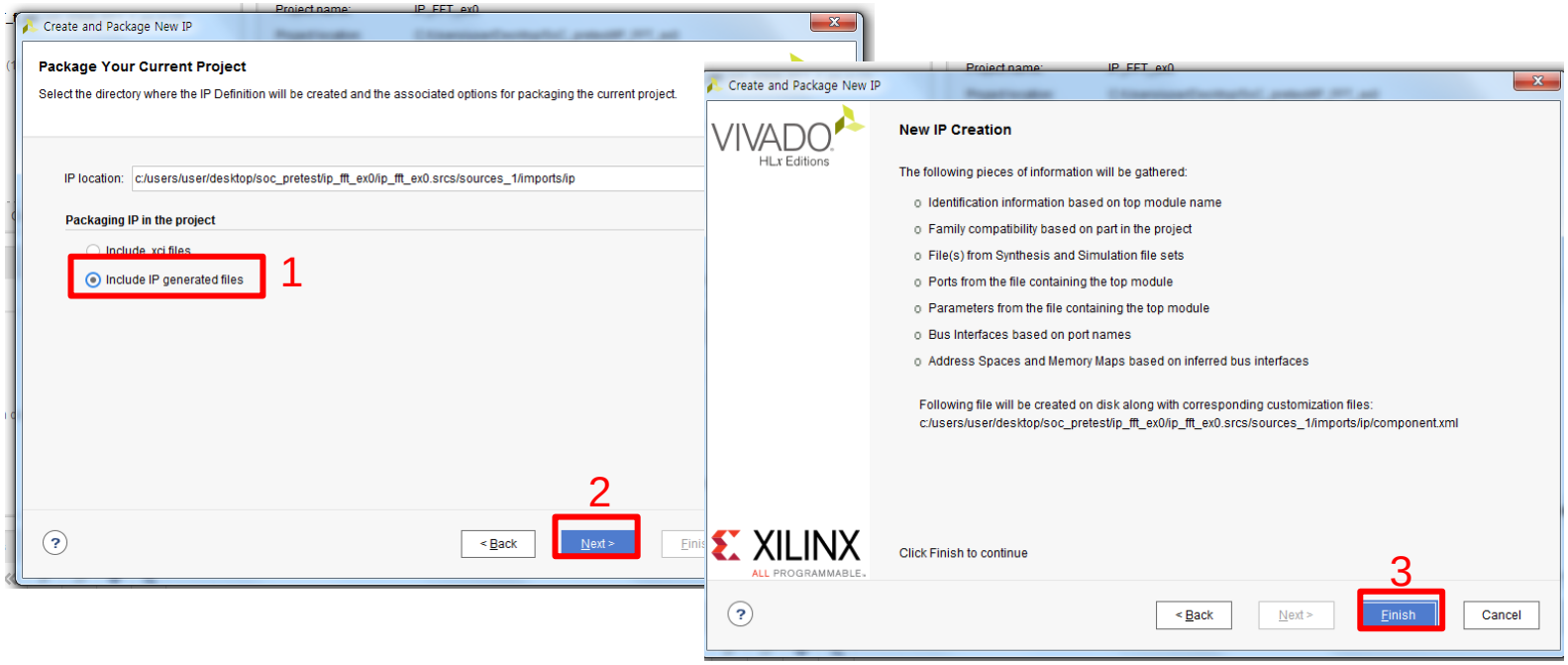
Creating IP Projects

- Create and Package IP (cont'd)
  - Click ***'Next'***
  - Select the ***'Package your current project'*** *and the click* ***'Next'***



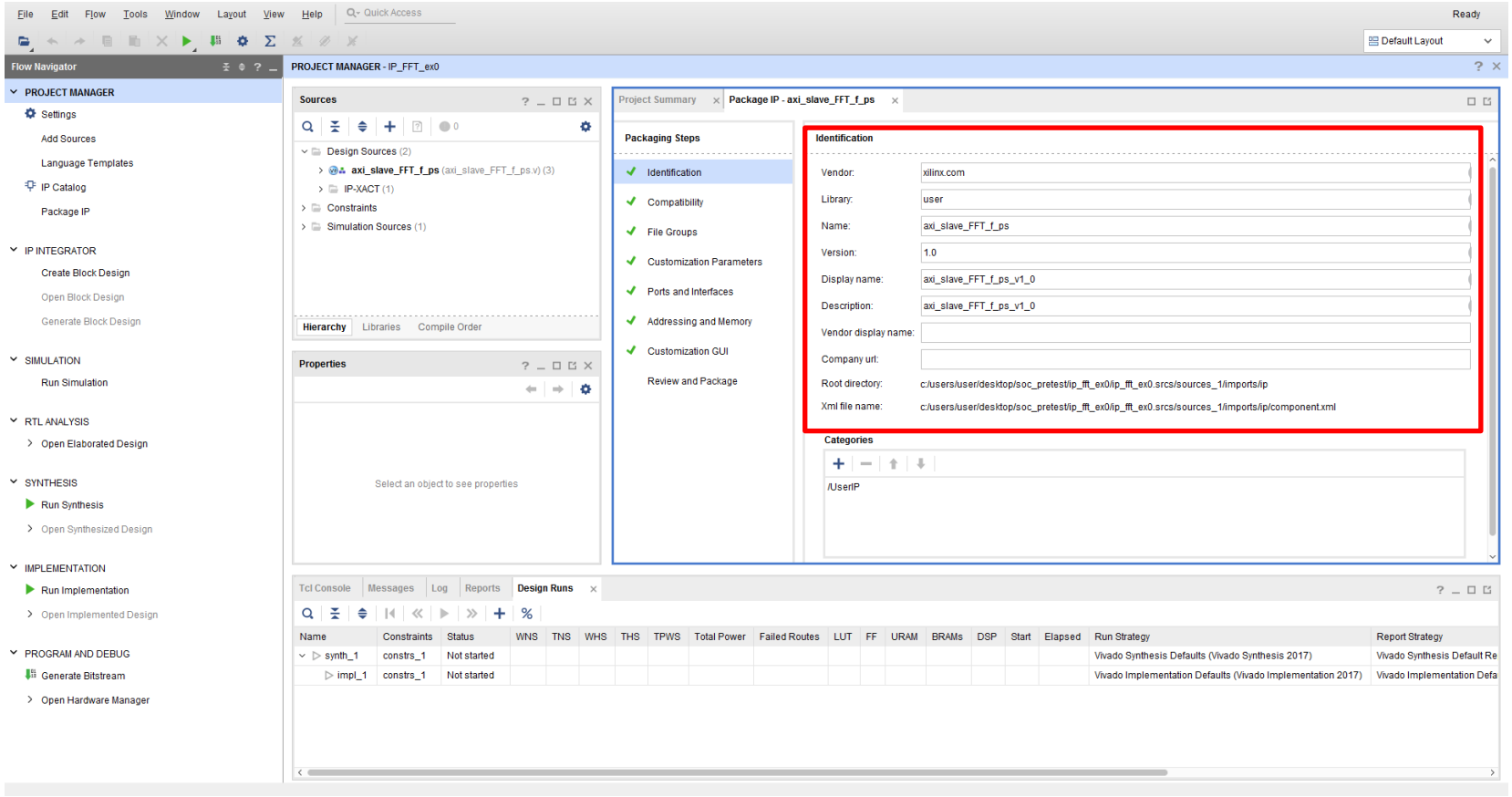
Creating IP Projects

- Create and Package IP (cont'd)
  - Choose ***'Include IP generated files'***
  - Click ***'Next > Finish'***
  - You may wait for a few seconds



Creating IP Projects

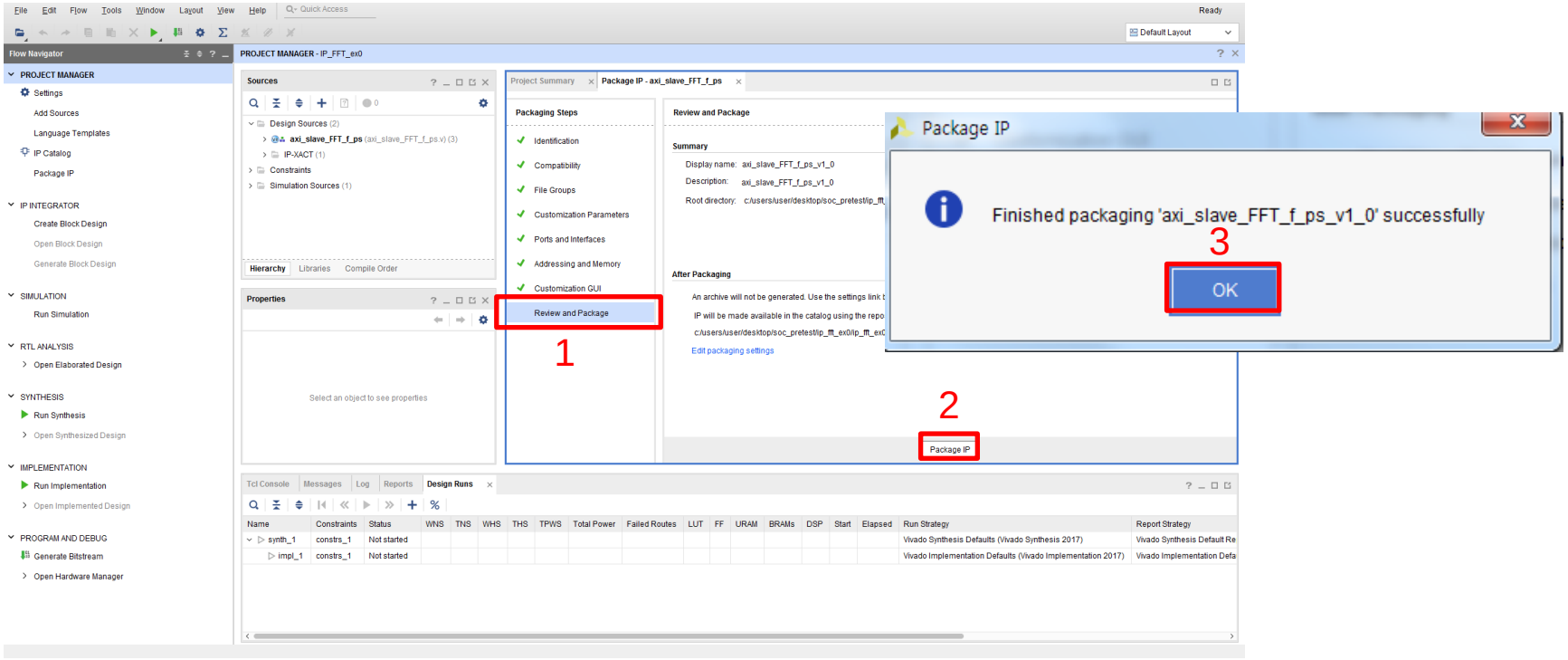
- Check IP Identification
  - You may change the name or category of IP (optional)



Creating IP Projects

❑ Package IP

- Click ***'Review and Package > Package IP > OK'***

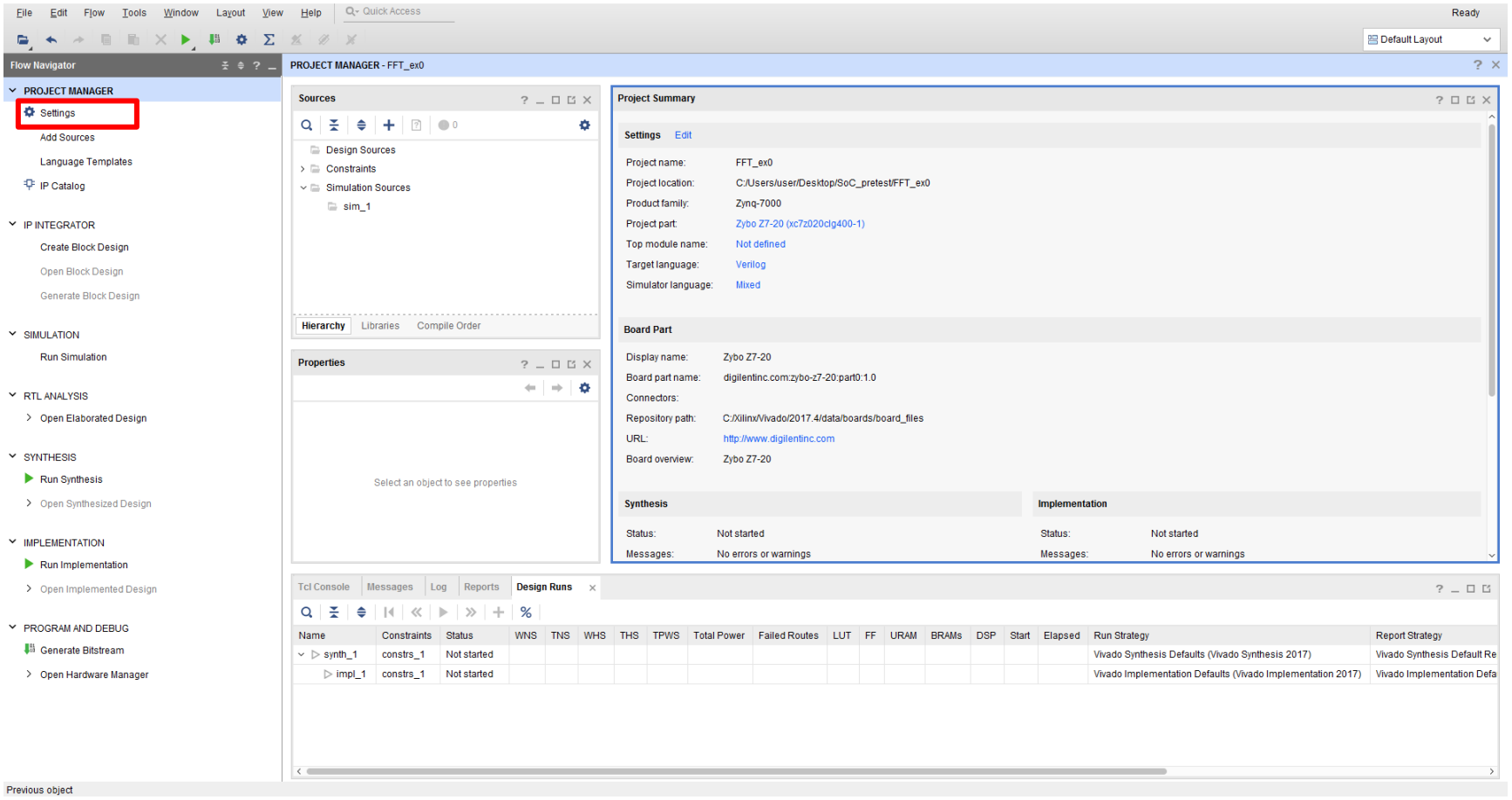


Creating Block Designs

- Repeat the previous steps
  - Get Started
  - Create a New Vivado Project
  - Enter Project Name
    - ✓ A **new** project name and a **new** project location should be used
  - Choose Project Type
  - ~~Add Sources~~ (skipped)
  - Add Existing IP & Constraints
  - Choose Default Part
  - Check New Project Summary

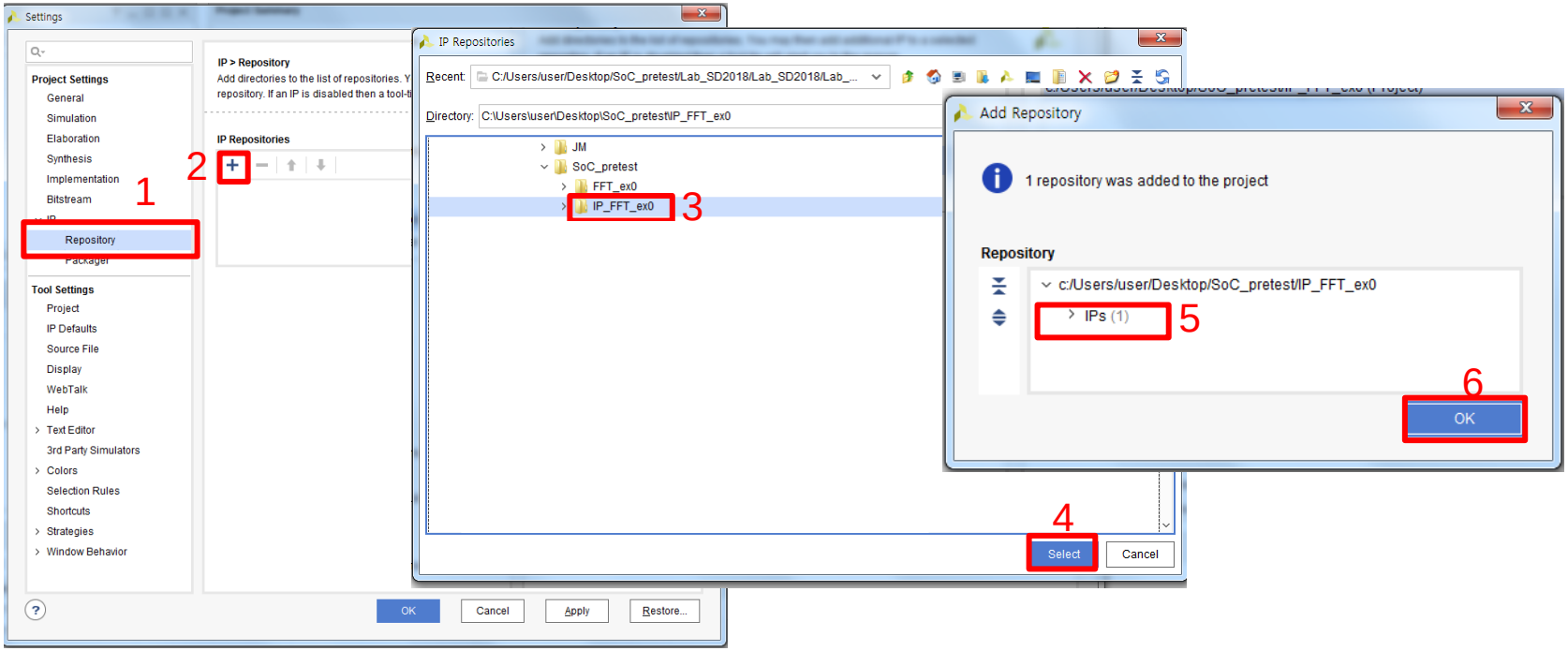
Creating Block Designs

- Add IP repository
  - Click ***'Settings'*** in ***'Project Manager'***



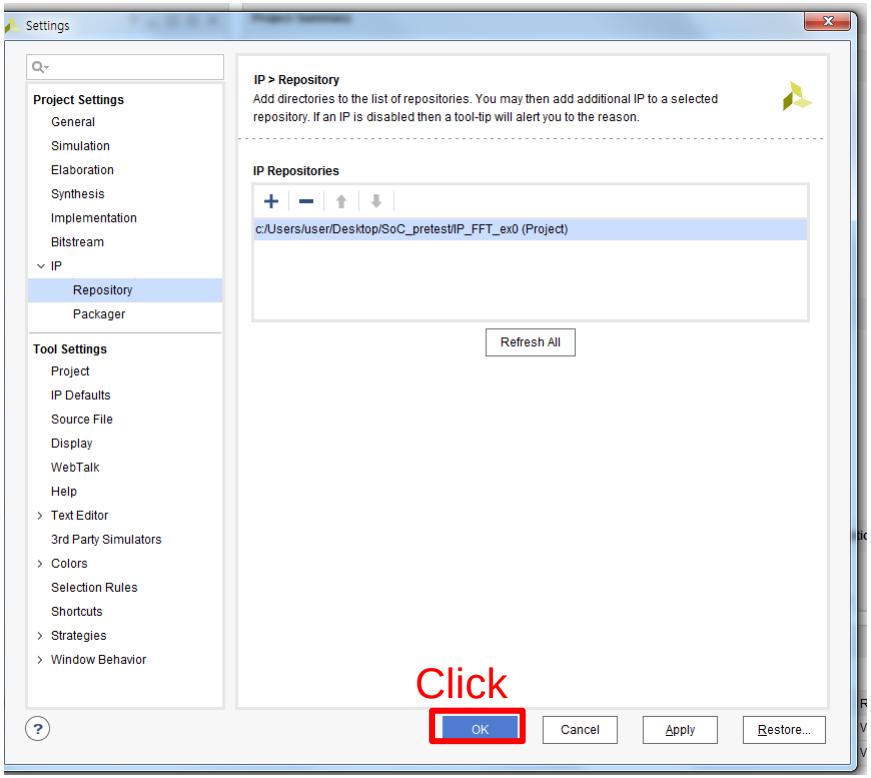
Creating Block Designs

- Add IP repository (cont'd)
  - Unfold the ***'IP'*** and then click ***'Repository*** > ***Add'***
  - Choose the IP project folder that you have already created.
  - Click ***'Select*** > ***OK'***



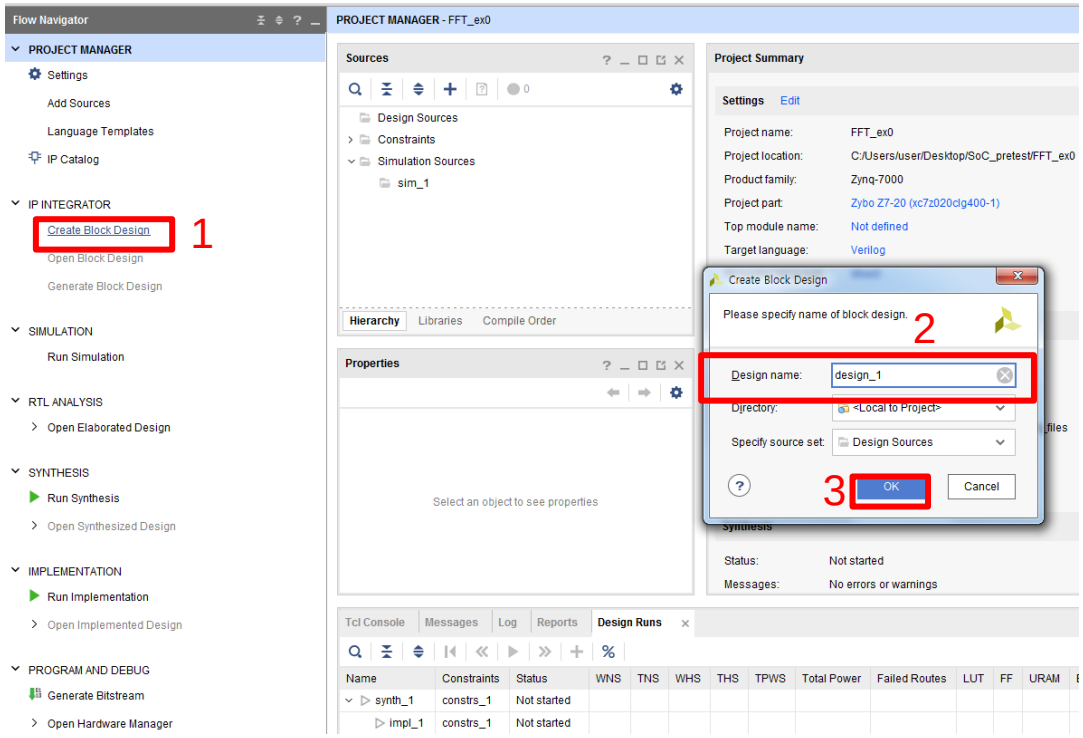
Creating Block Designs

- Add IP repository (cont'd)
  - Click ***'OK'***



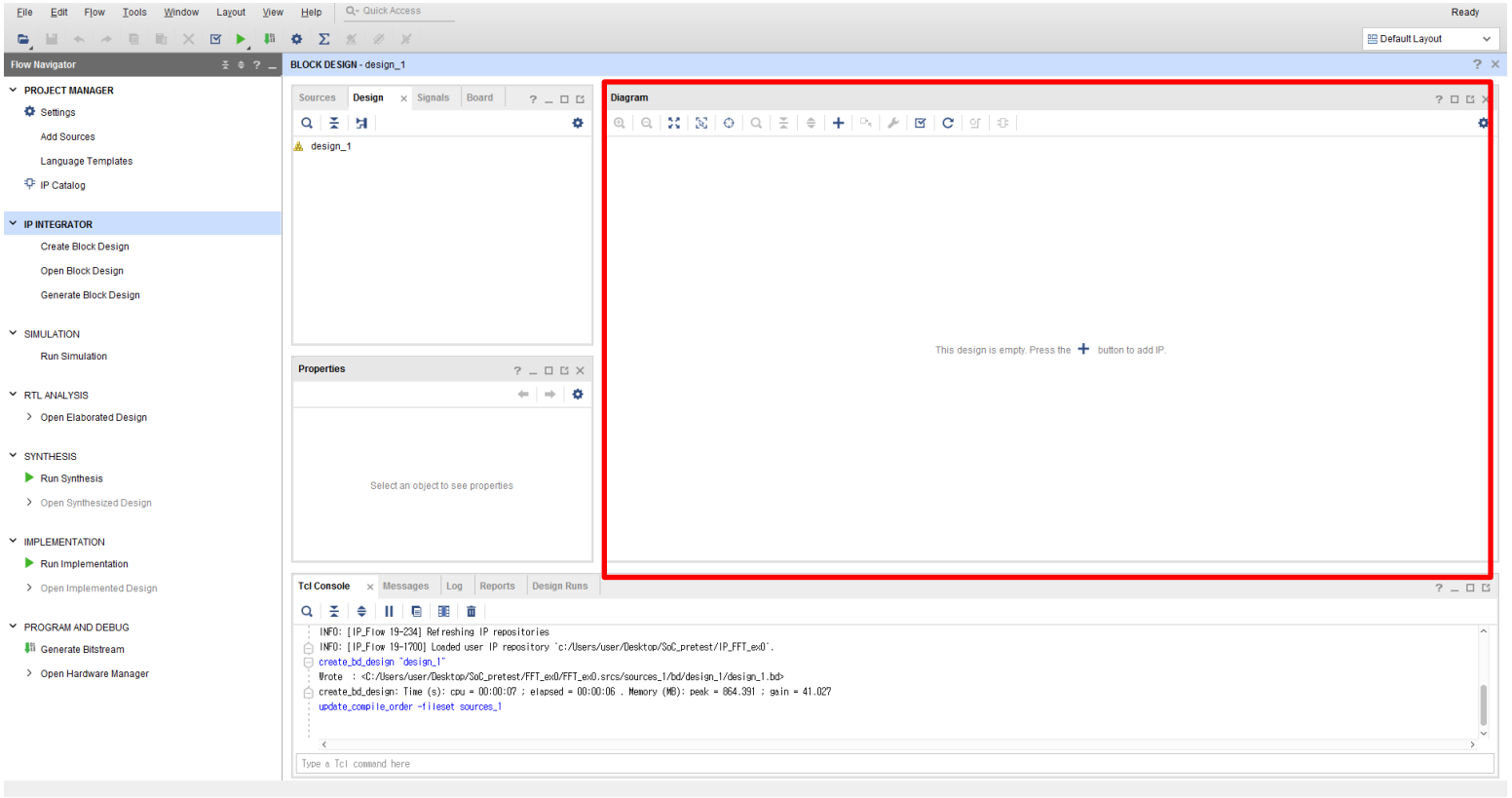
Creating Block Designs

- Create Block Design
  - Click ***'Create Block Design'***
  - Type a name of the block design and then click ***'OK'***



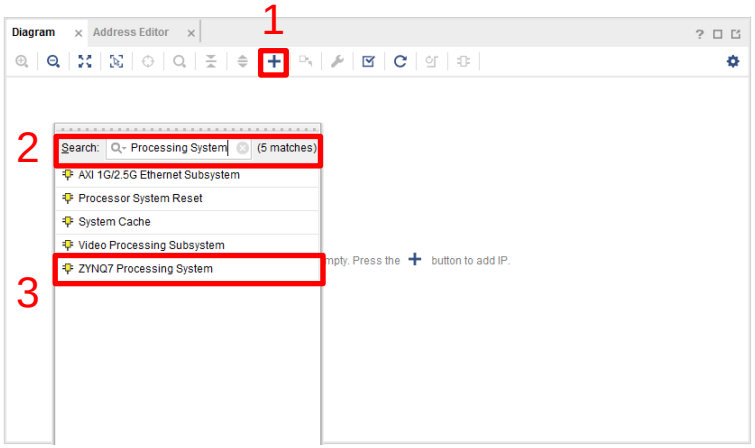
Creating Block Designs

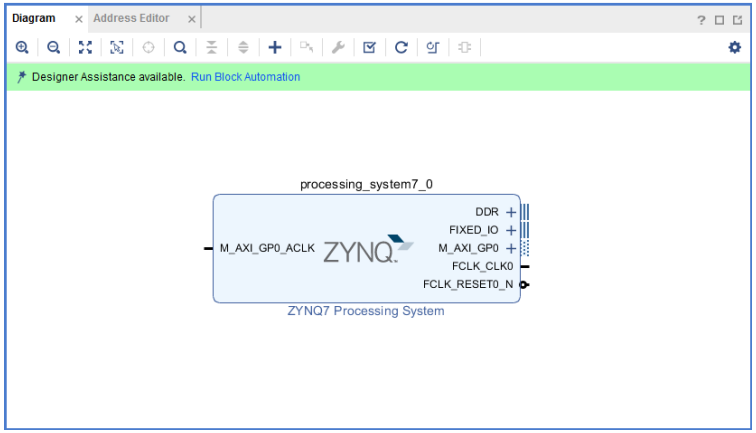
- Create Block Design (Cont'd)
  - You should be able to see the Diagram tap below



Creating Block Designs

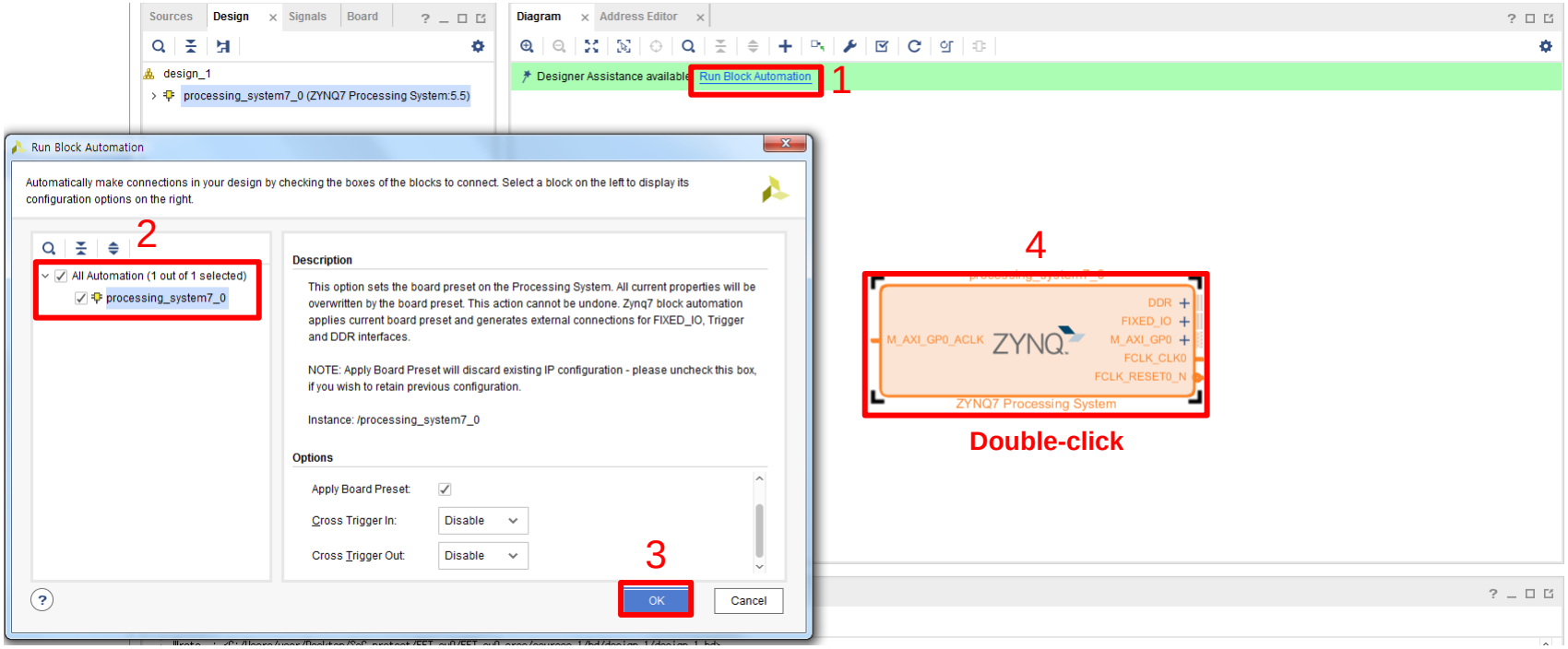
- Add Processing System
  - Click the ***'Add IP'*** icon and type "***Processing System"*** in the Search field
  - Double-click ***'ZYNQ7 Processing System'***





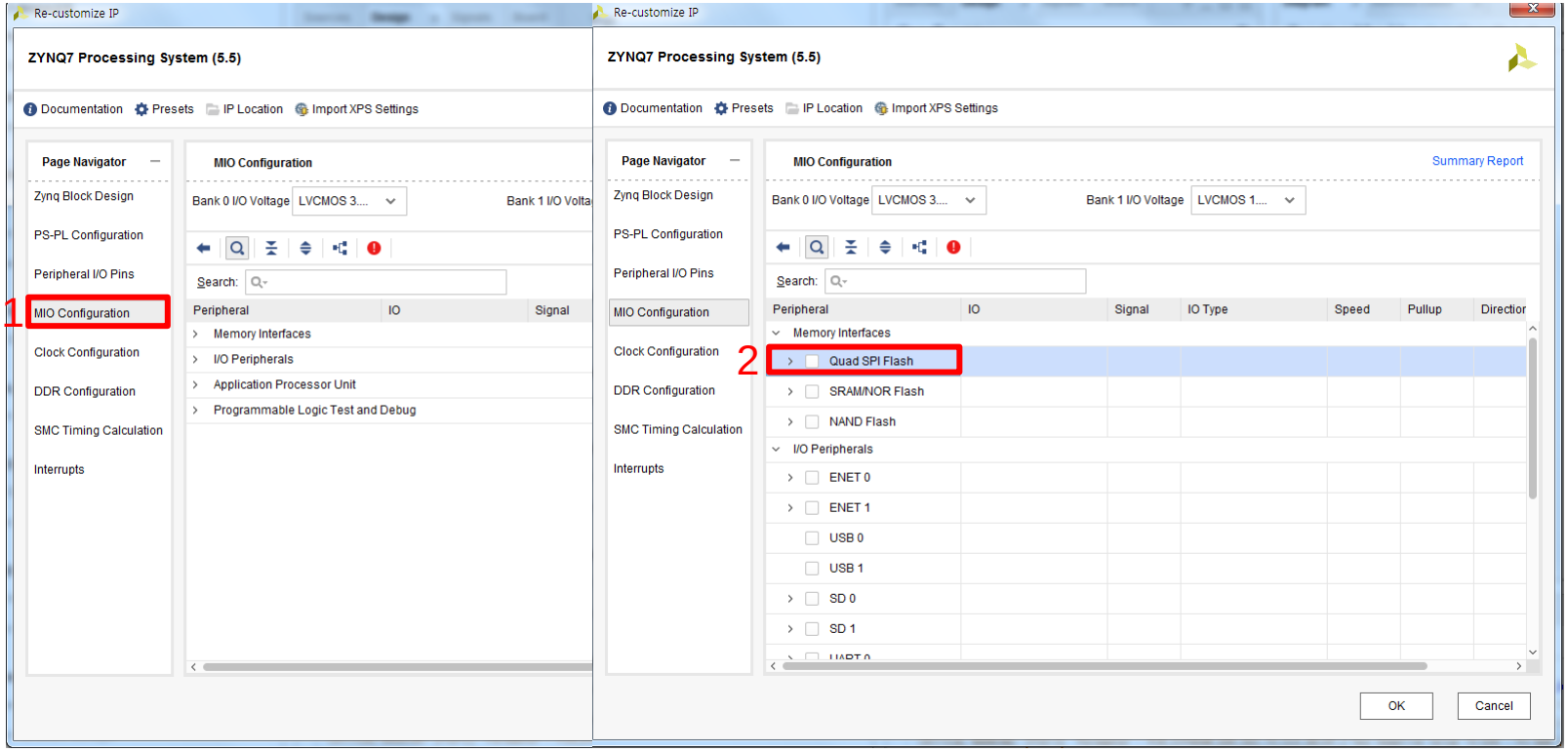
Creating Block Designs

- Make external connection
  - Click ***'Run Block Automation'***
  - Select the ***'processing_system7_0'*** and then click ***'OK'***
  - Double-click the Processing System block



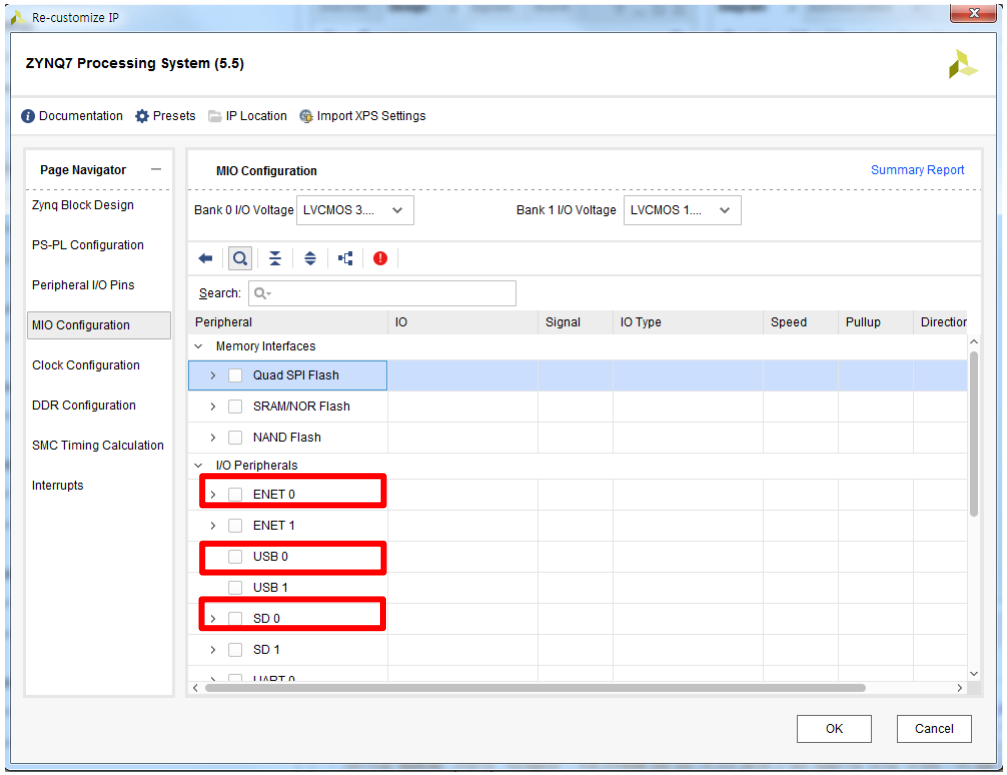
Creating Block Designs

- Re-customize Processing System
  - Click ***'MIO Configuration'*** and unfold ***'Memory Interfaces'***
  - Uncheck ***'Quad SPI Flash'***



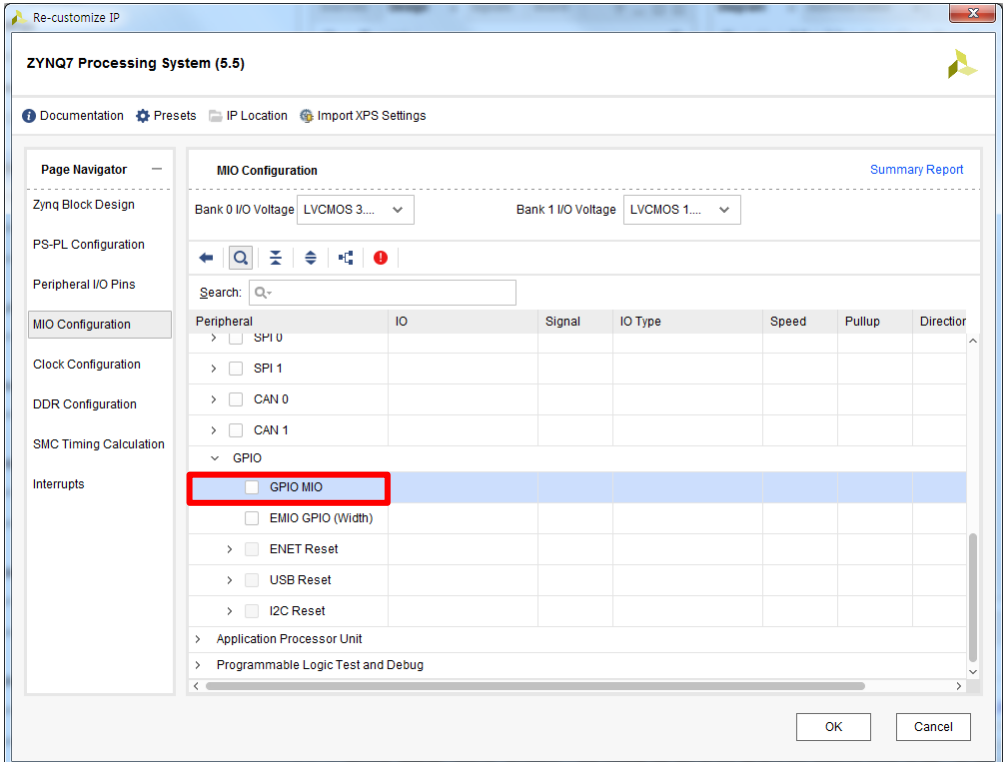
Creating Block Designs

- Re-customize Processing System (cont'd)
  - Unfold ***'I/O peripherals'***
  - Uncheck ***'ENET 0'***, ***'USB 0'*** and ***'SD 0'***



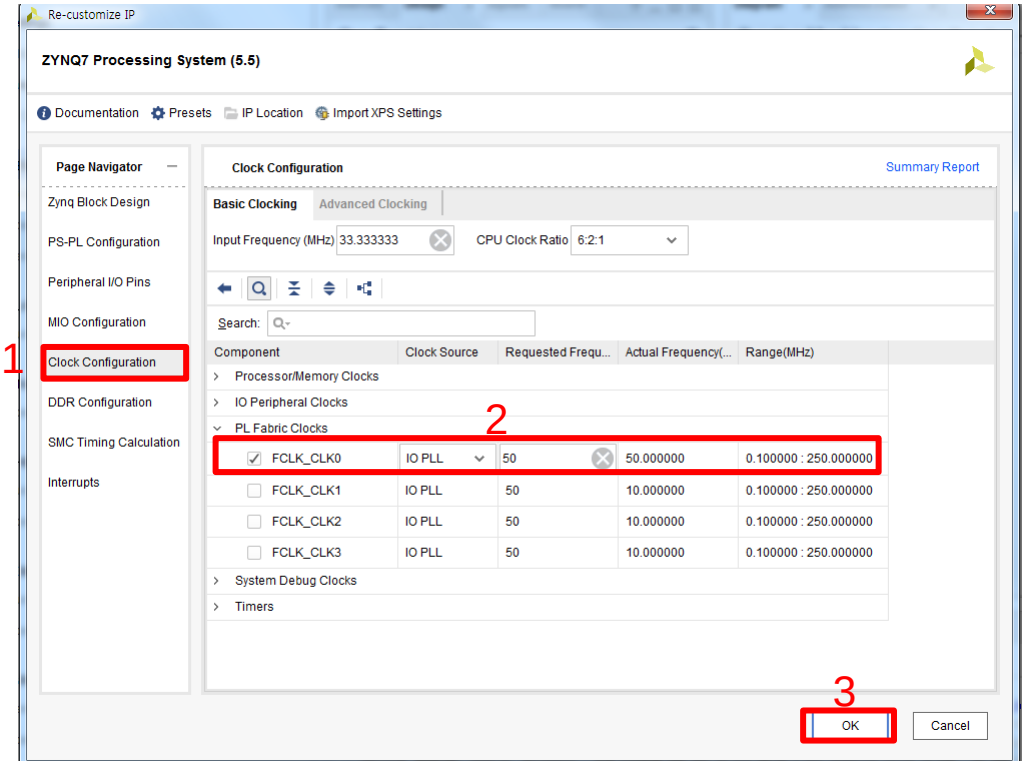
Creating Block Designs

- Re-customize Processing System (cont'd)
  - Unfold ***'GPIO'*** and uncheck ***'GPIO MIO'***



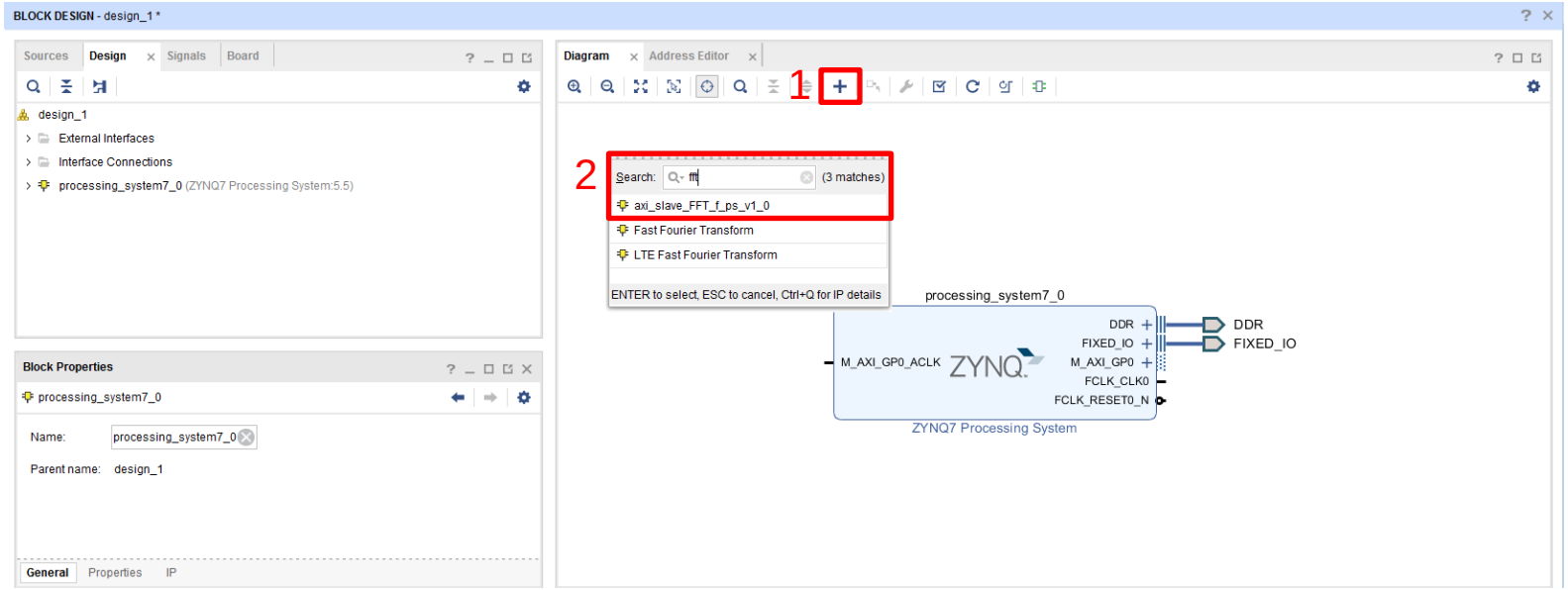
Creating Block Designs

- Re-customize Processing System (cont'd)
  - Click ***'Clock Configuration'*** and unfold ***'PL Fabric Clocks'***
  - Set ***'FCLK_CLK0'*** to ***'50.00000'*** (MHz)
  - Click ***'OK'***



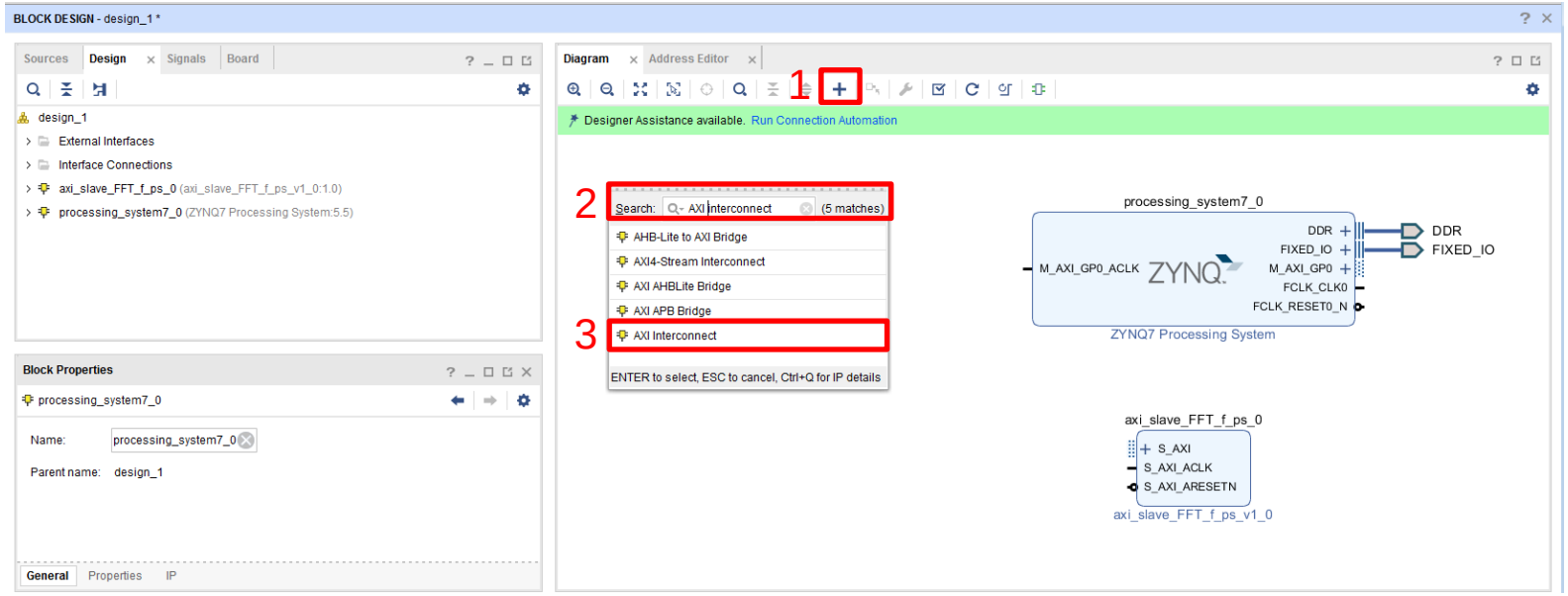
Creating Block Designs

- Add FFT IP
  - Click the ***'Add IP'*** icon, type ***"fft"*** in the Search field
  - Double-click the ***'axi_slave_FFT_f_ps_v1_0'***



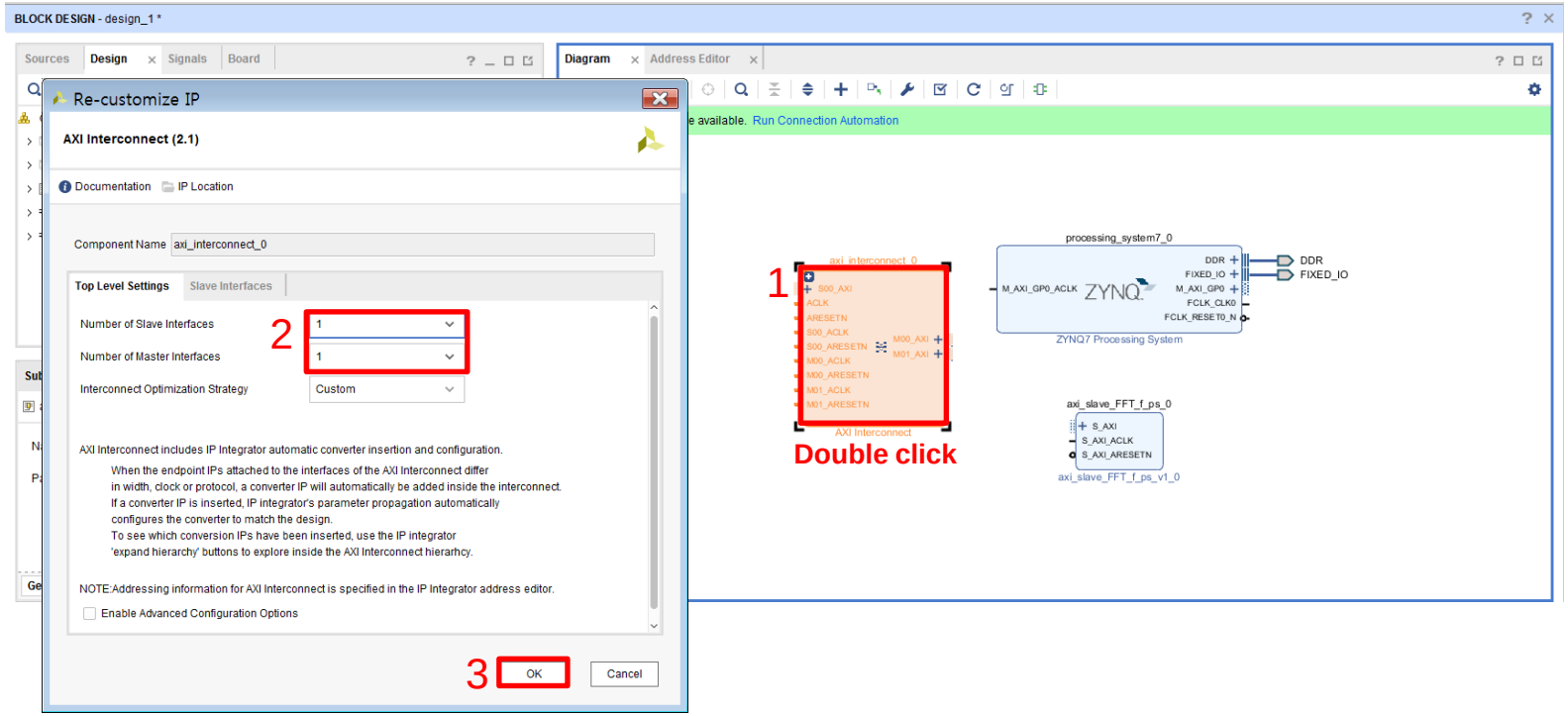
Creating Block Designs

- Add AXI Interconnect IP
  - Click the ***'Add IP'*** icon, type ***"AXI interconnect"*** in the Search field
  - Double-click the ***'AXI Interconnect'***



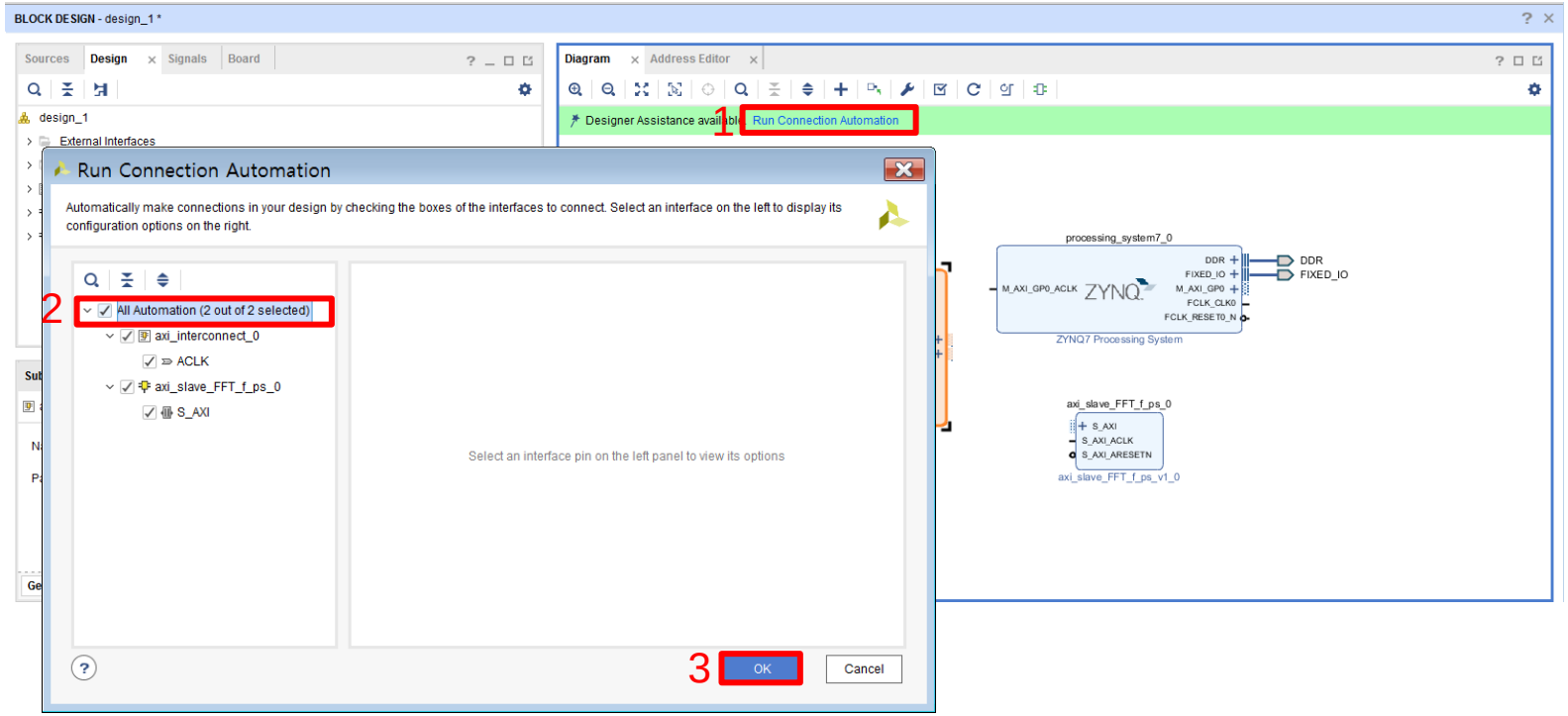
Creating Block Designs

- Add AXI Interconnect IP (Cont'd)
  - Double-click the ***'AXI Interconnect'*** block
  - Choose ***'Number of Slave, Master Interfaces'***
  - Click ***'OK'***



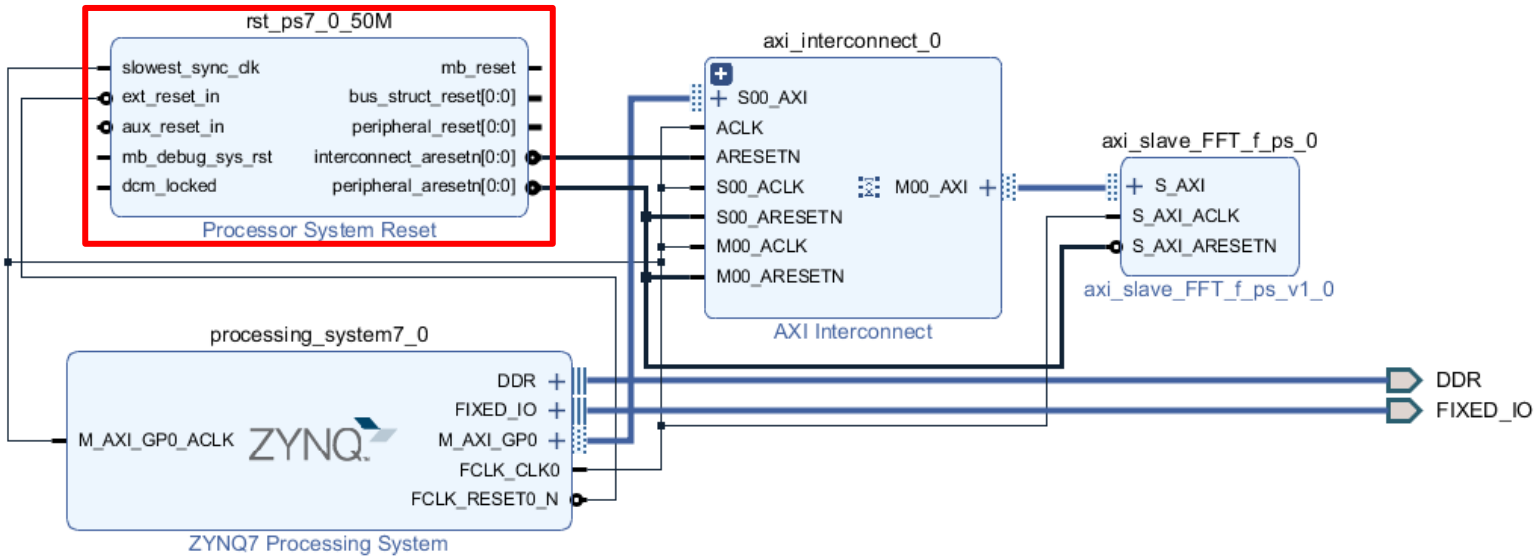
Creating Block Designs

- Run Connection Automation
  - Click the ***'Run Connection Automation'***
  - Select the ***'All Automation'*** and then click ***'OK***



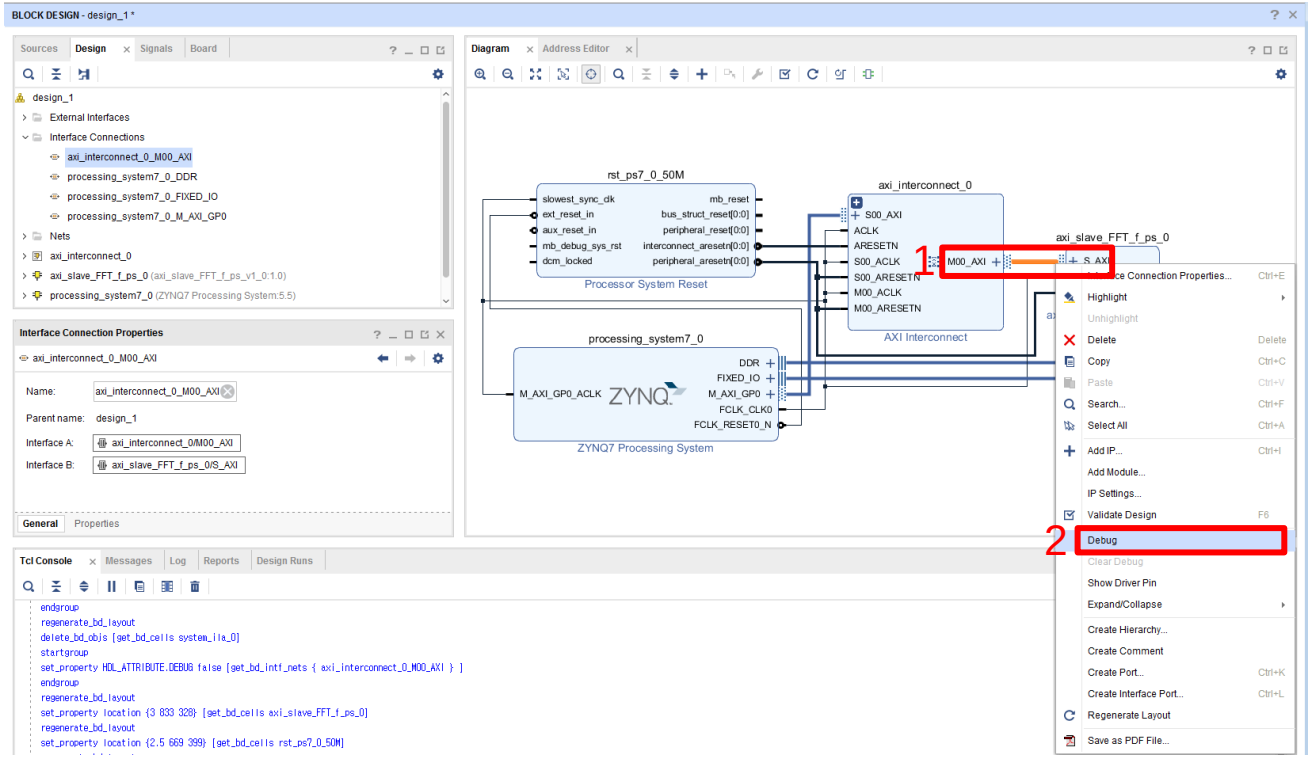
Creating Block Designs

- Run Connection Automation (cont'd)
  - Notice that additional block, '***Processor System Reset'*** have automatically been added to design



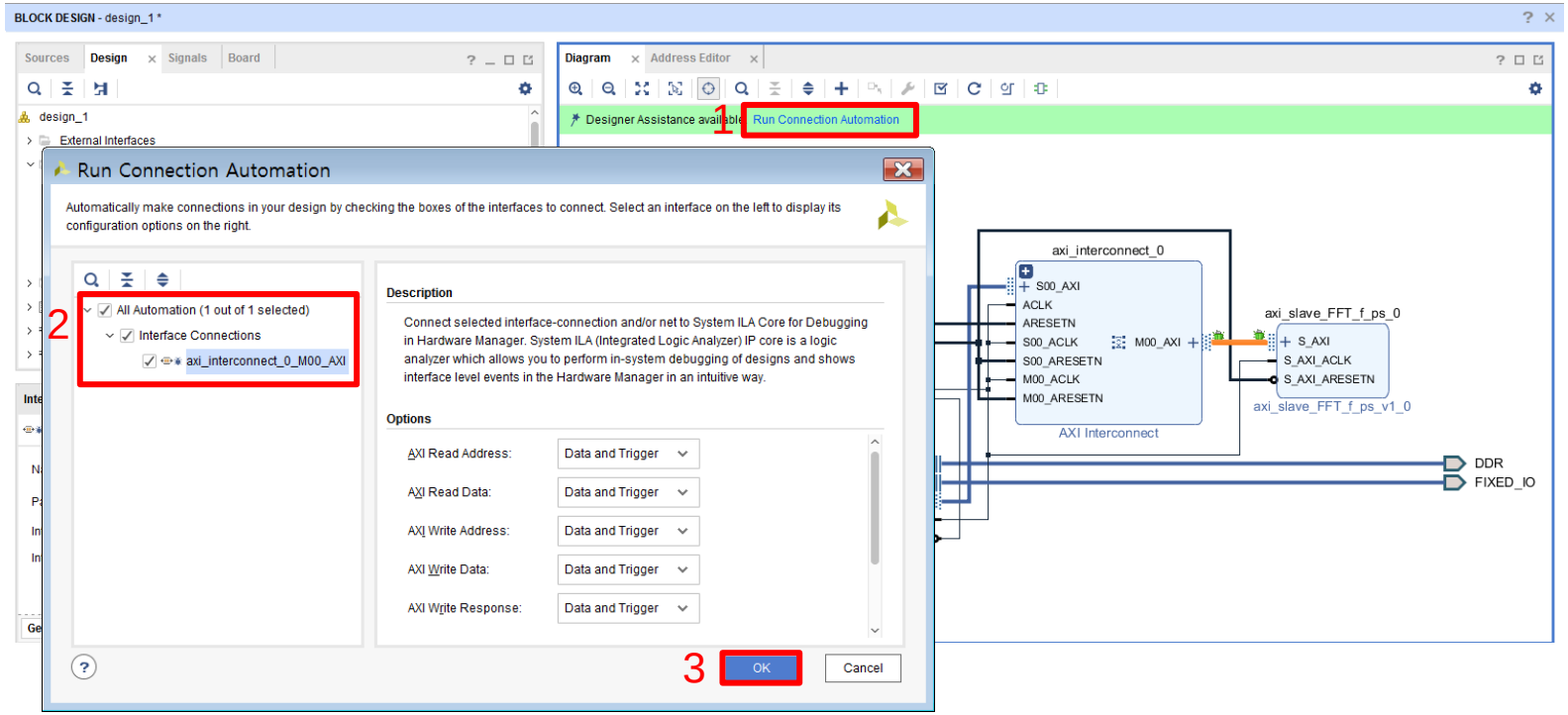
Creating Block Designs

- Mark Debug
  - Click the connection between ***'M00_AXI'*** and ***'S_AXI'***
  - Right-click and choose ***'Debug'***



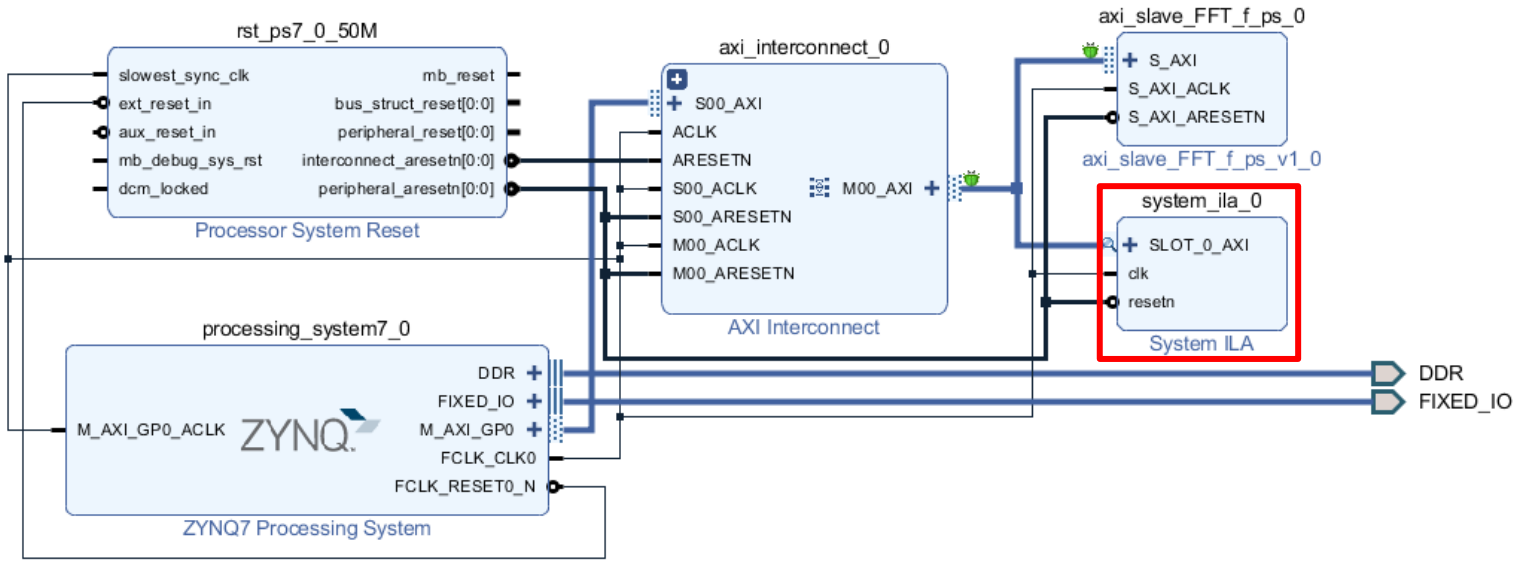
Creating Block Designs

- Mark Debug (cont'd)
  - Click the ***'Run Connection Automation'***
  - Select the ***'All Automation'*** and then click ***'OK***



Creating Block Designs

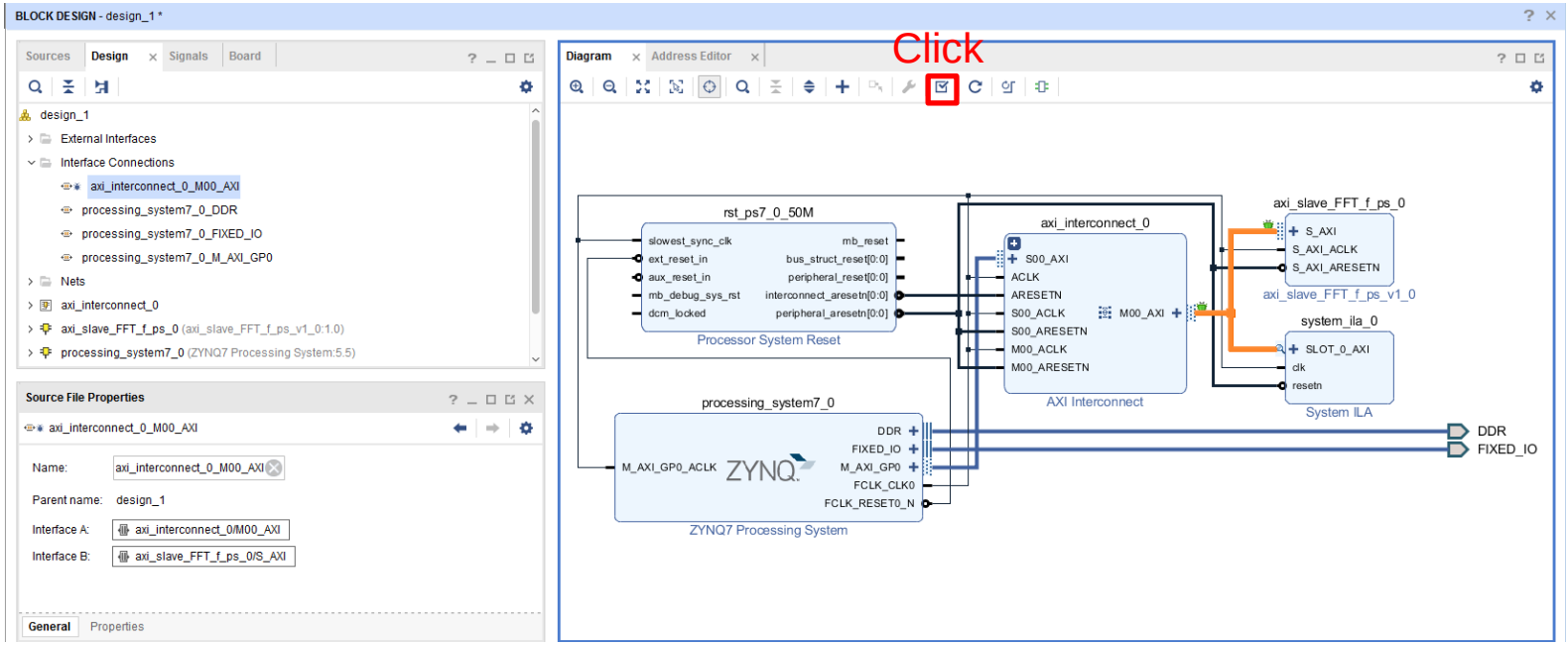
- Mark Debug (cont'd)
  - Notice that additional block, '***System ILA'*** have automatically been added to design



Creating Block Designs

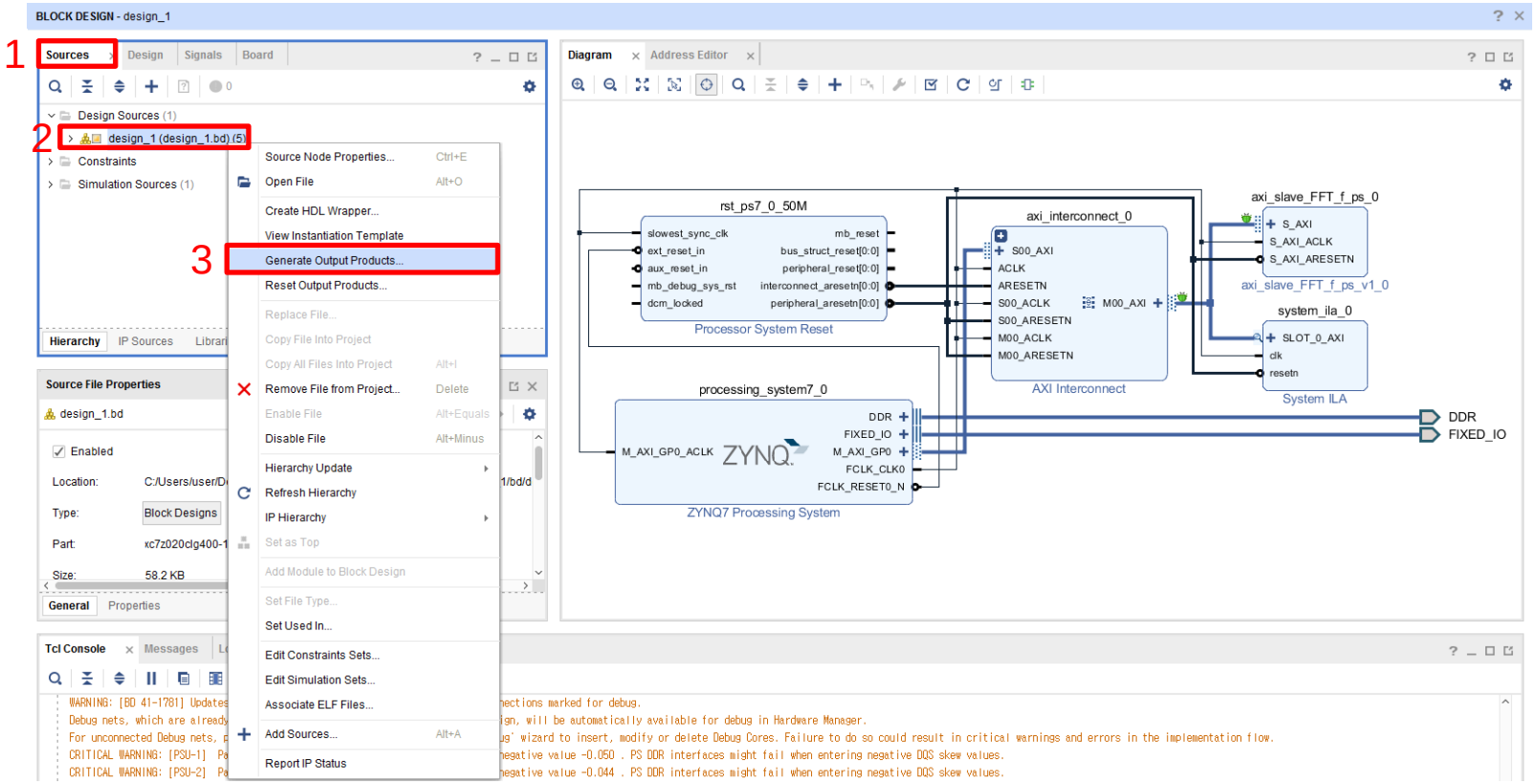
❑ Validate Design

- Choose the ***'Diagram'*** tap and click the ***'Validate Design'*** icon



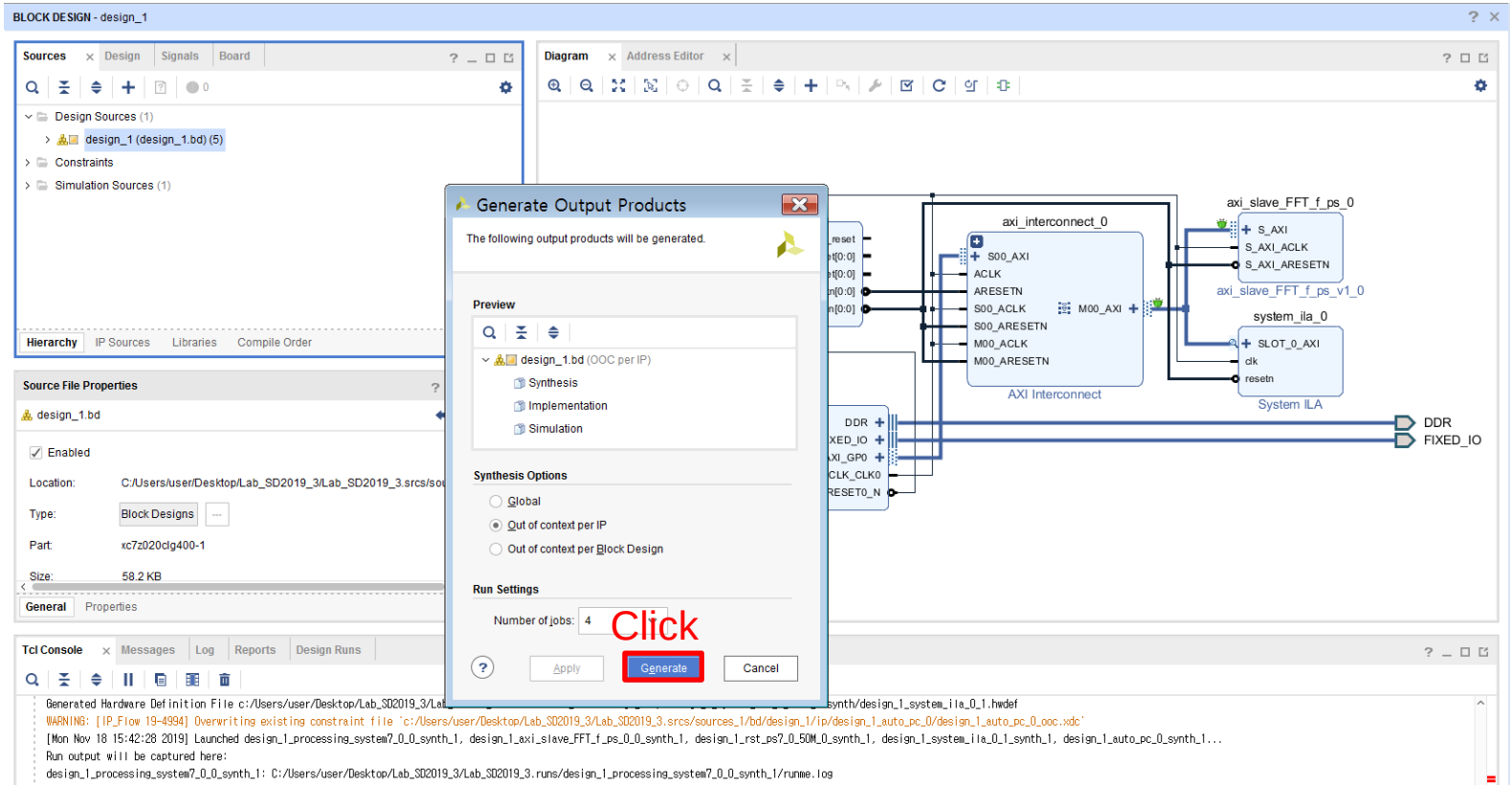
Creating Block Designs

- Generate Output Products
  - Choose the ***'Sources'*** tap and then right-click ***'design_1'***
  - Click ***'Generate Output Products'***



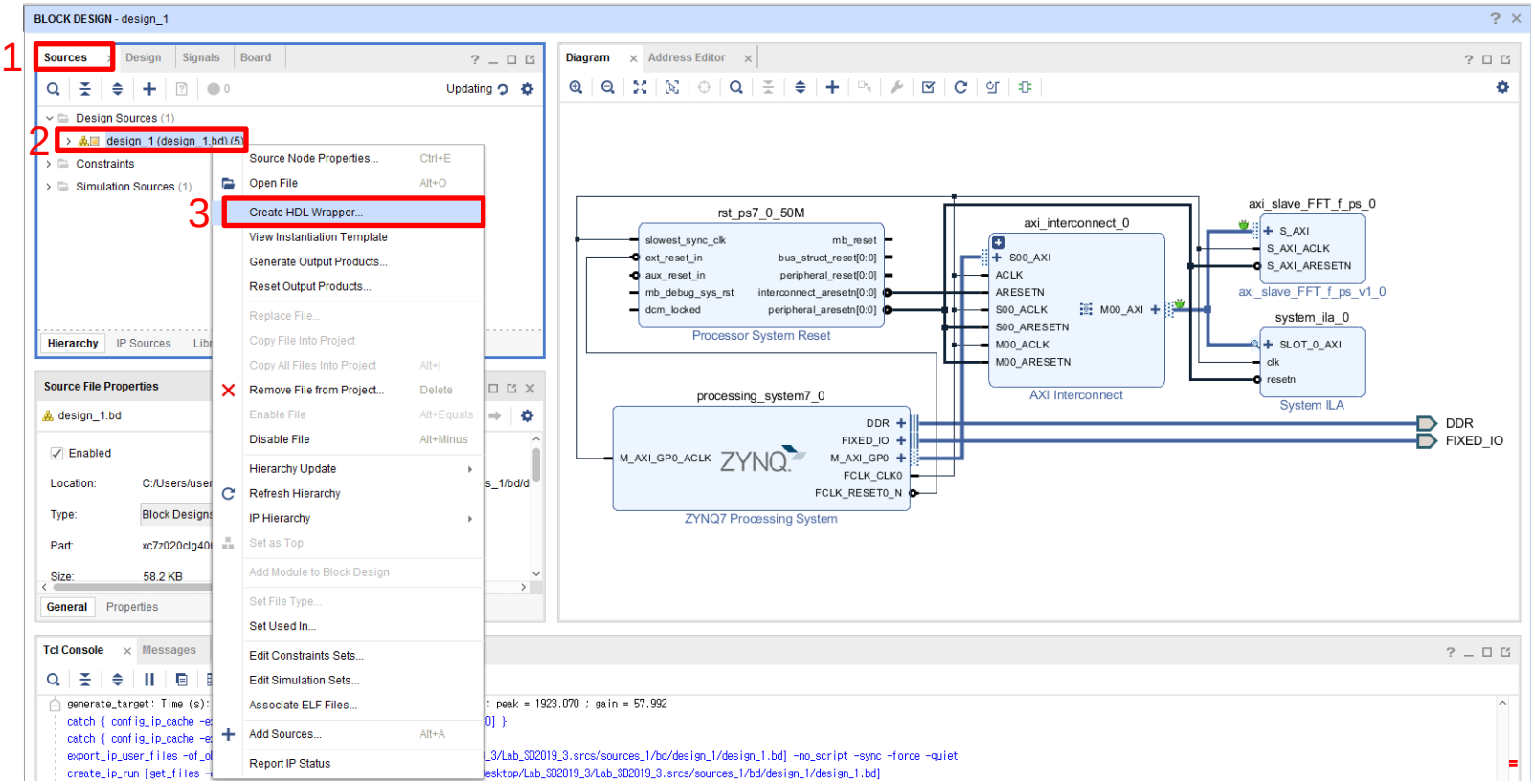
Creating Block Designs

- Generate Output Products (Cont'd)
  - Click '***Generate***'
  - You may wait for a few seconds



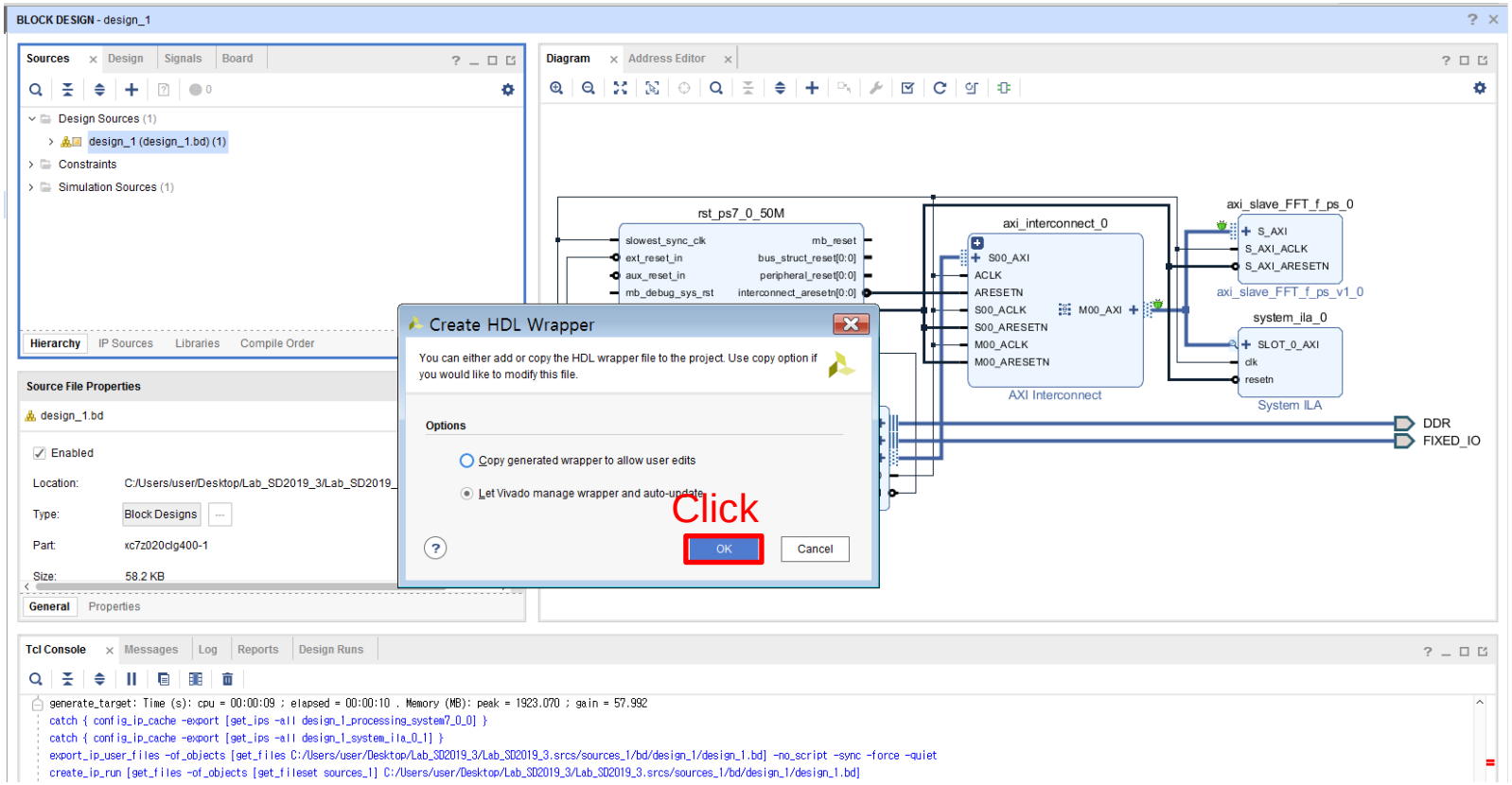
Creating Block Designs

- Create HDL Wrapper
  - Choose the ***'Sources'*** tap and then right-click ***'design_1'***
  - Click ***'Create HDL Wrapper'***



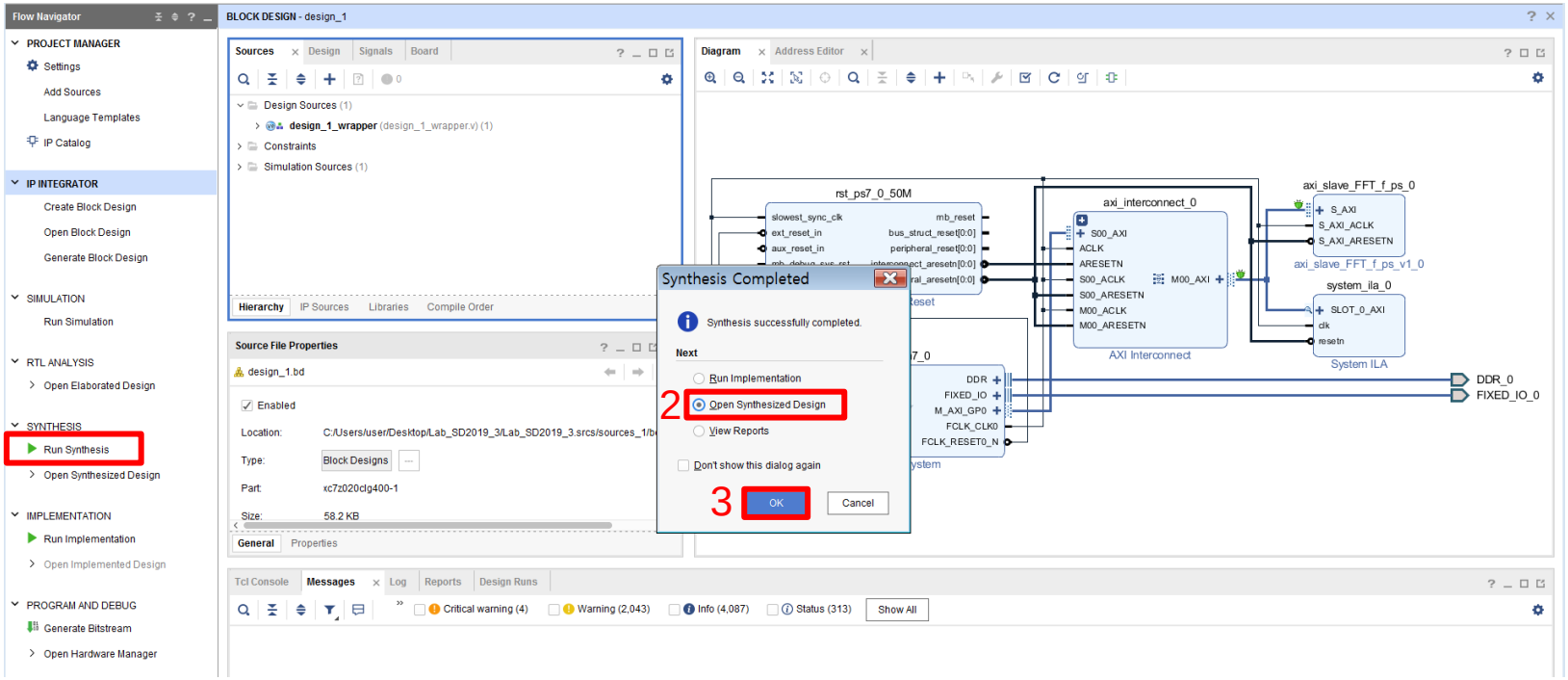
Creating Block Designs

- Create HDL Wrapper (Cont'd)
  - Click '***OK***'
  - You may wait for a few seconds



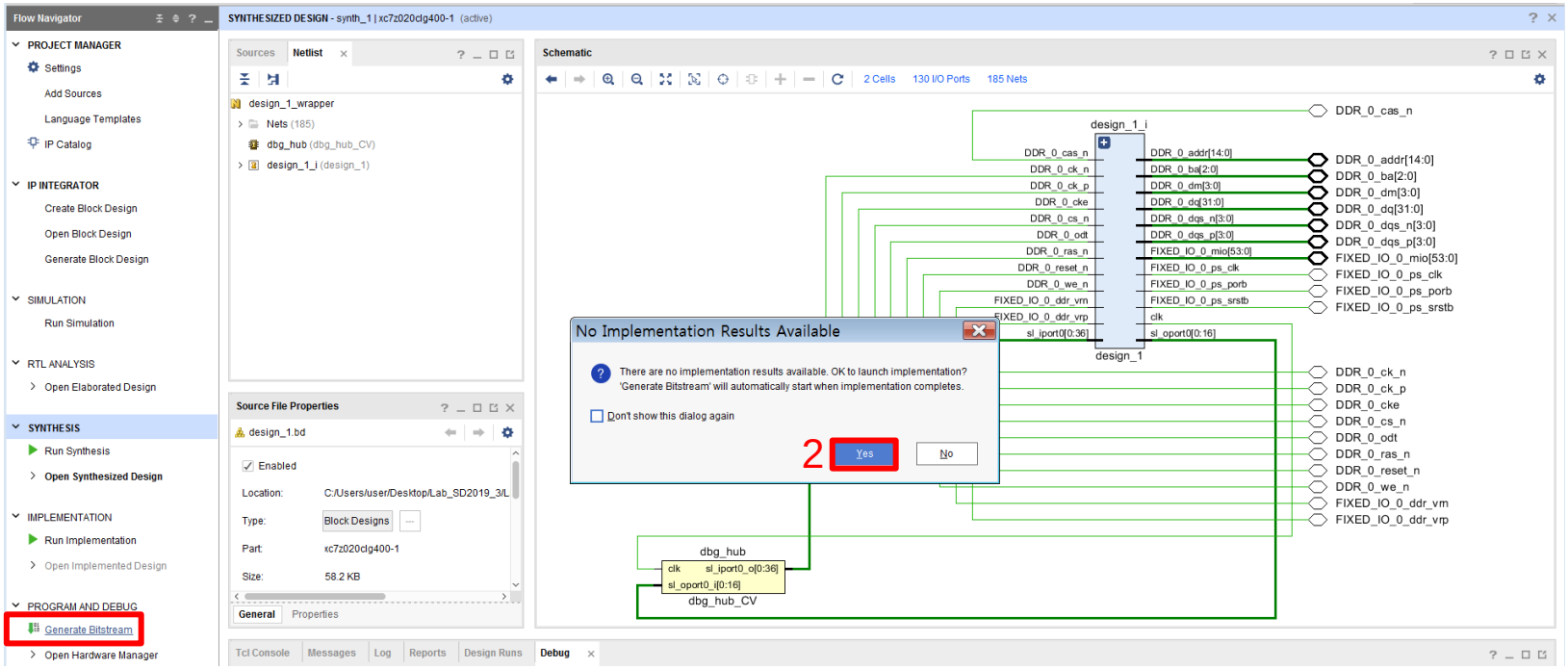
Creating Block Designs

- Run Synthesis
  - Select ***'Synthesis' > 'Run Synthesis'*** in ***'Flow Navigator'***
  - Once the Synthesis ends, choose ***'Open Synthesis Design'*** and then click ***'OK'***



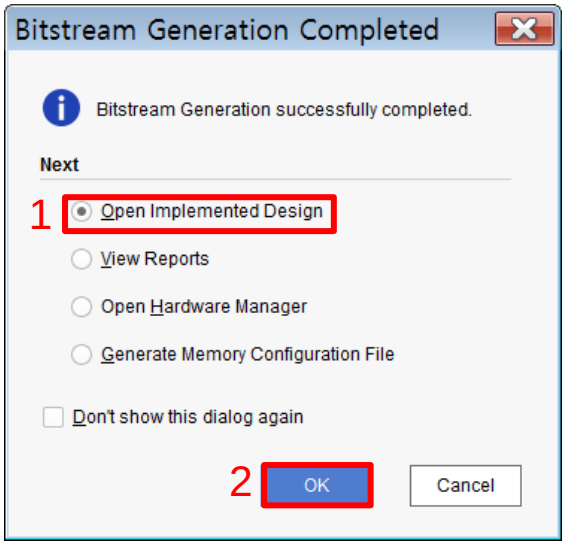
Generating Bitstream

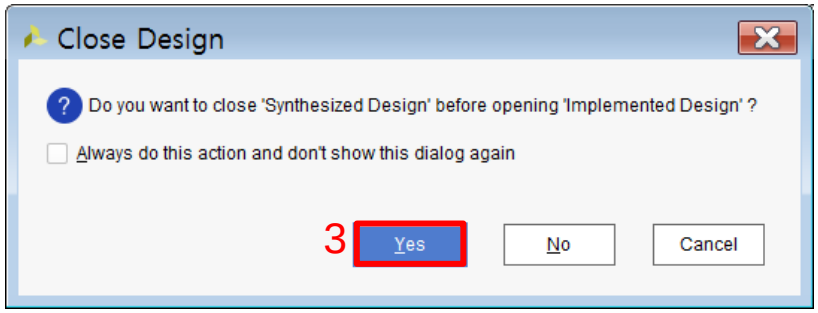
- Generate Bitstream
  - Select ***'Program And Debug > Generate Bitstream'*** in ***'Flow Navigator'***
  - You may wait for a few seconds



Generating Bitstream

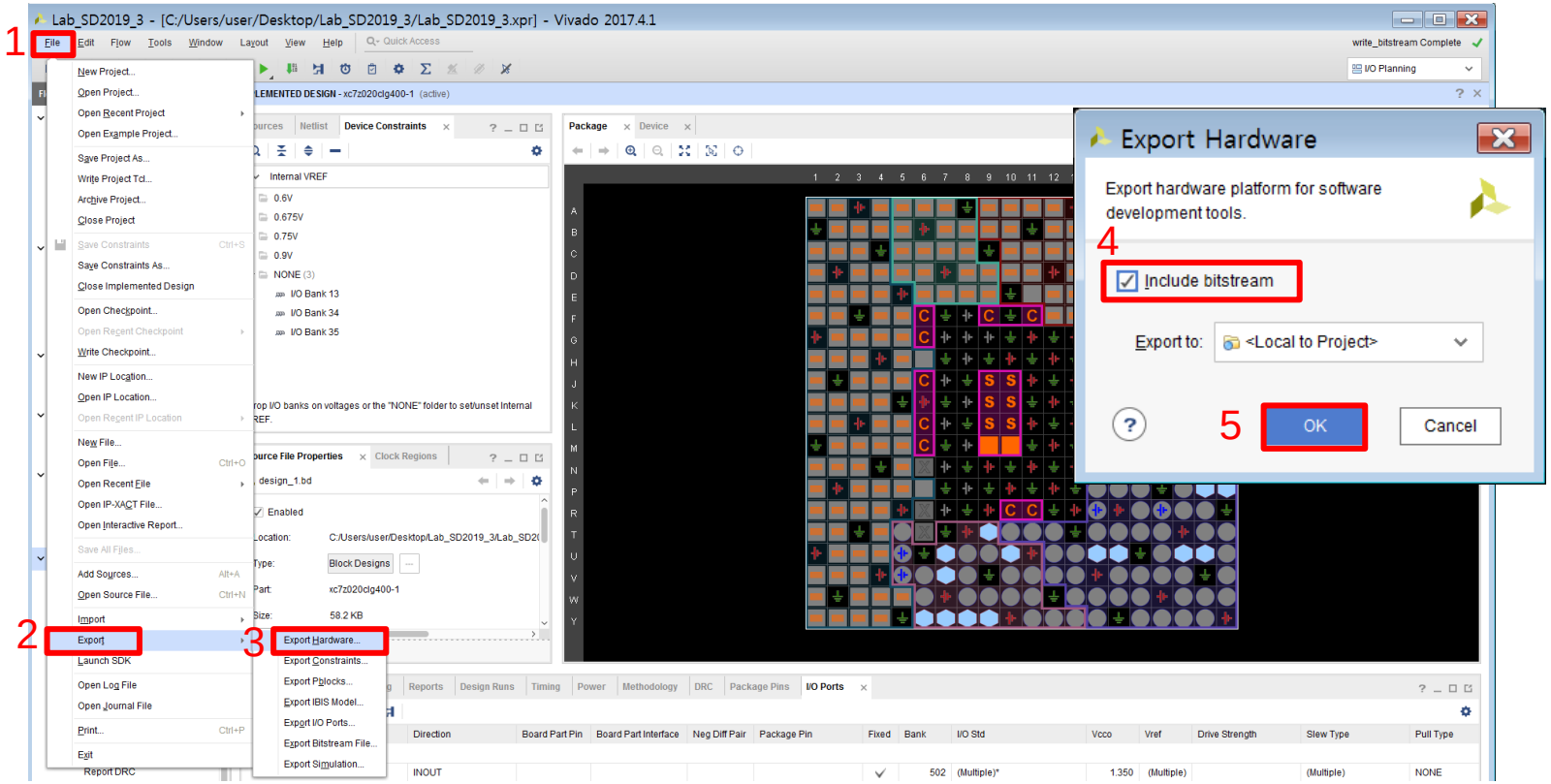
- Generate Bitstream (Cont'd)
  - Once the Bitstream Generation ends, choose ***'Open Implemented Design'***
  - Click ***'OK > Yes'***





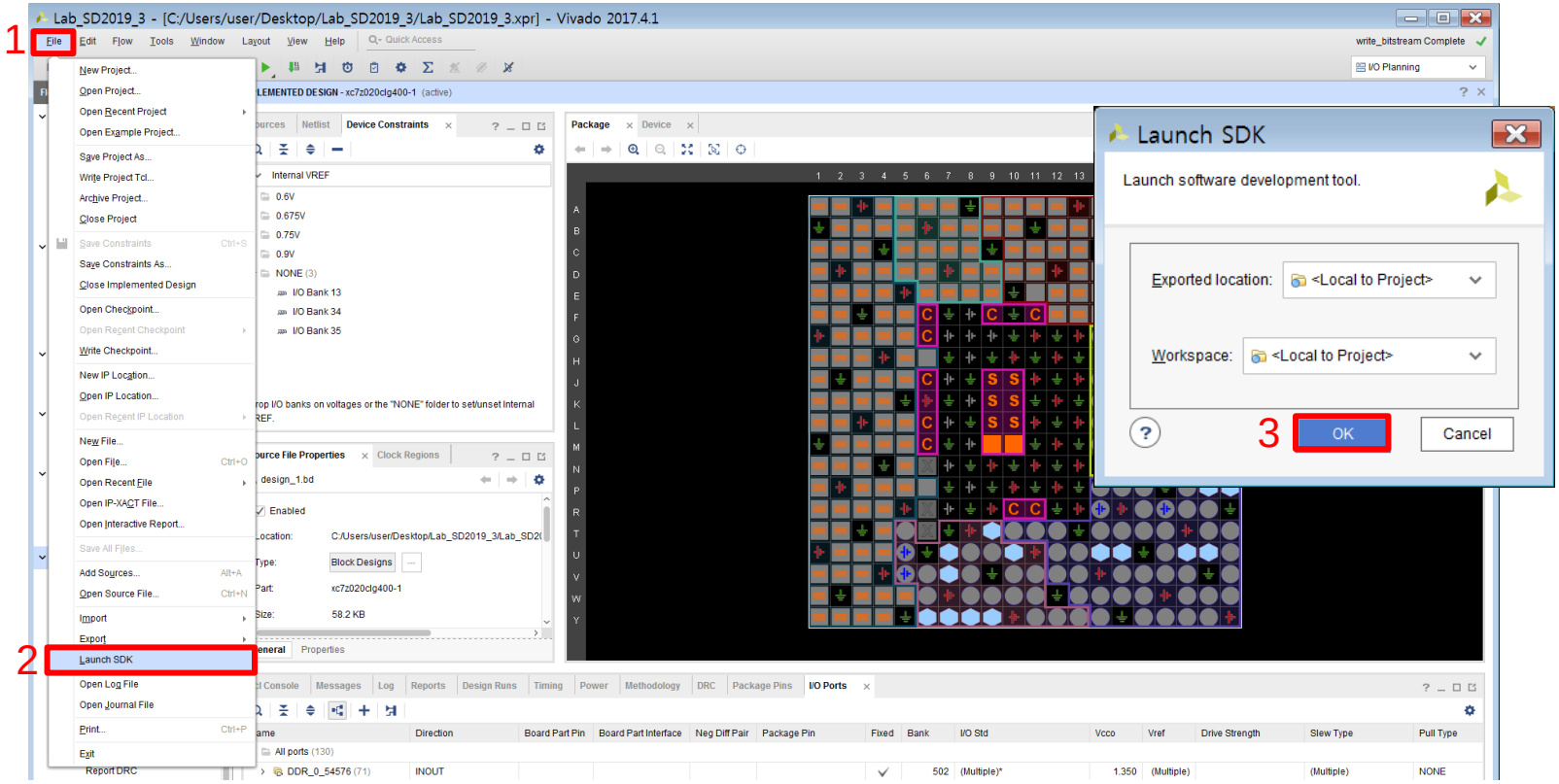
Generating Bitstream

- Export Hardware
  - Click ***'Export > Export Hardware'*** in ***'File'*** menu
  - Select ***'Include bitstream'*** and then click ***'OK'***



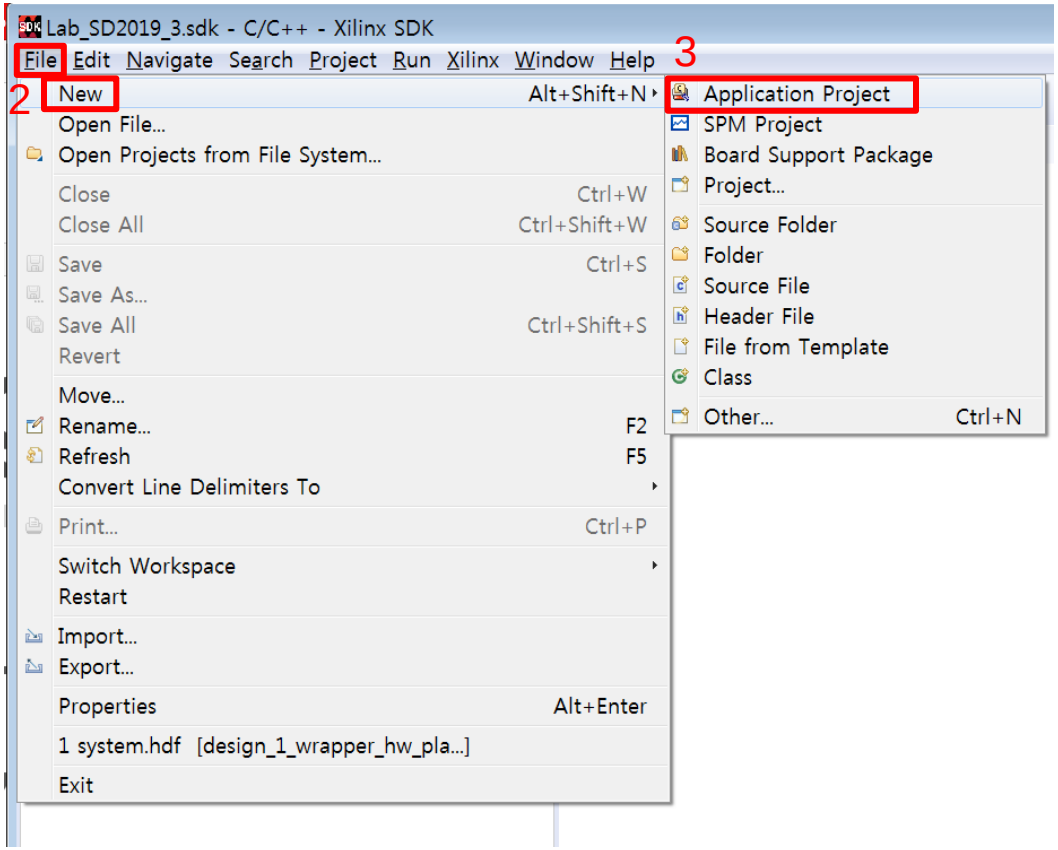
Generating Bitstream

- Launch SDK
  - Click ***'Launch SDK'*** in ***'File'*** menu
  - Click ***'OK'***



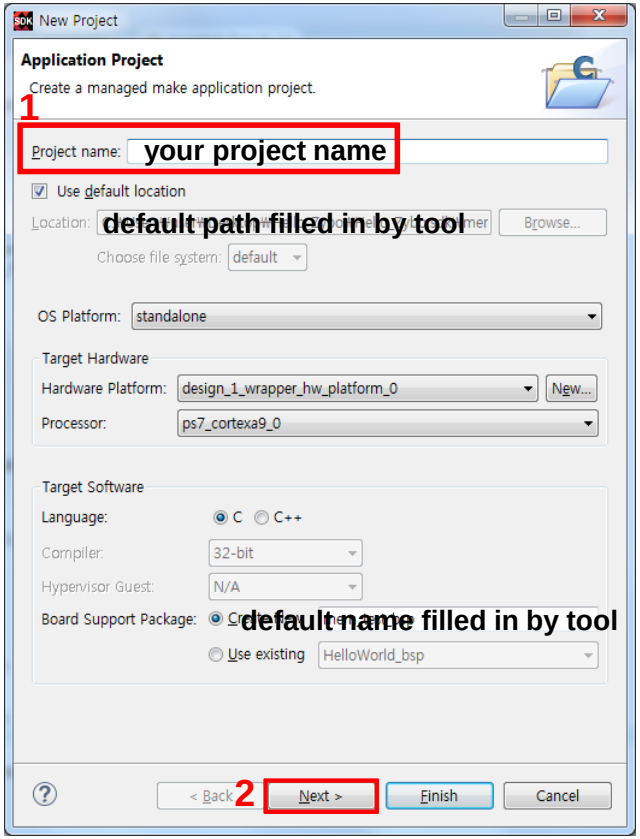
Running C Applications

- Create a C application project
  - Click ***'File' > 'New' > 'Application Project'***



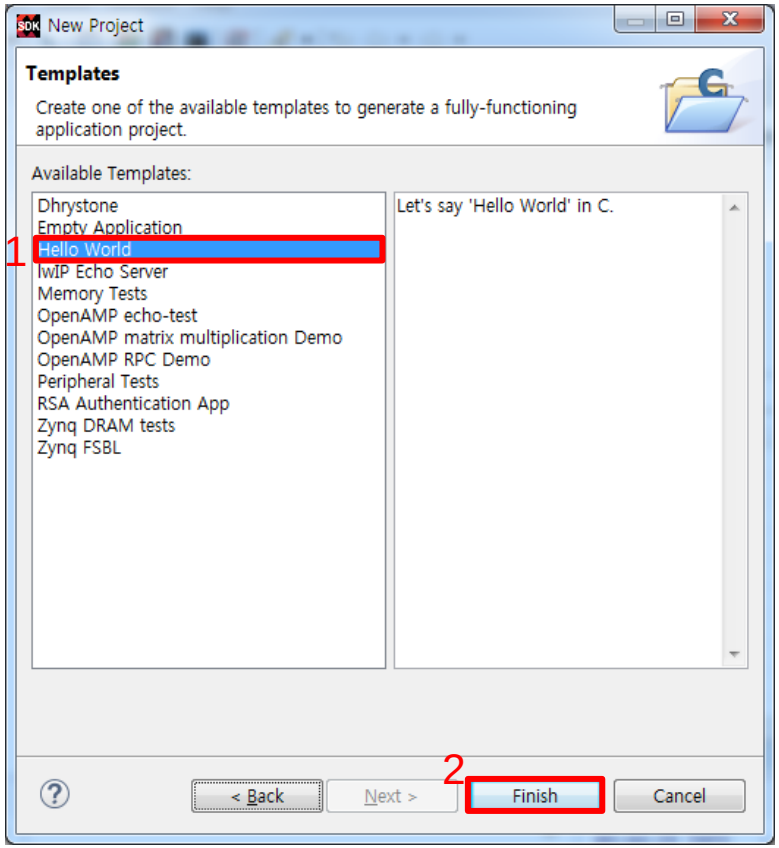
Running C Applications

- Create a C application project (cont'd)
  - Type the project name
  - Click ***'Next'***



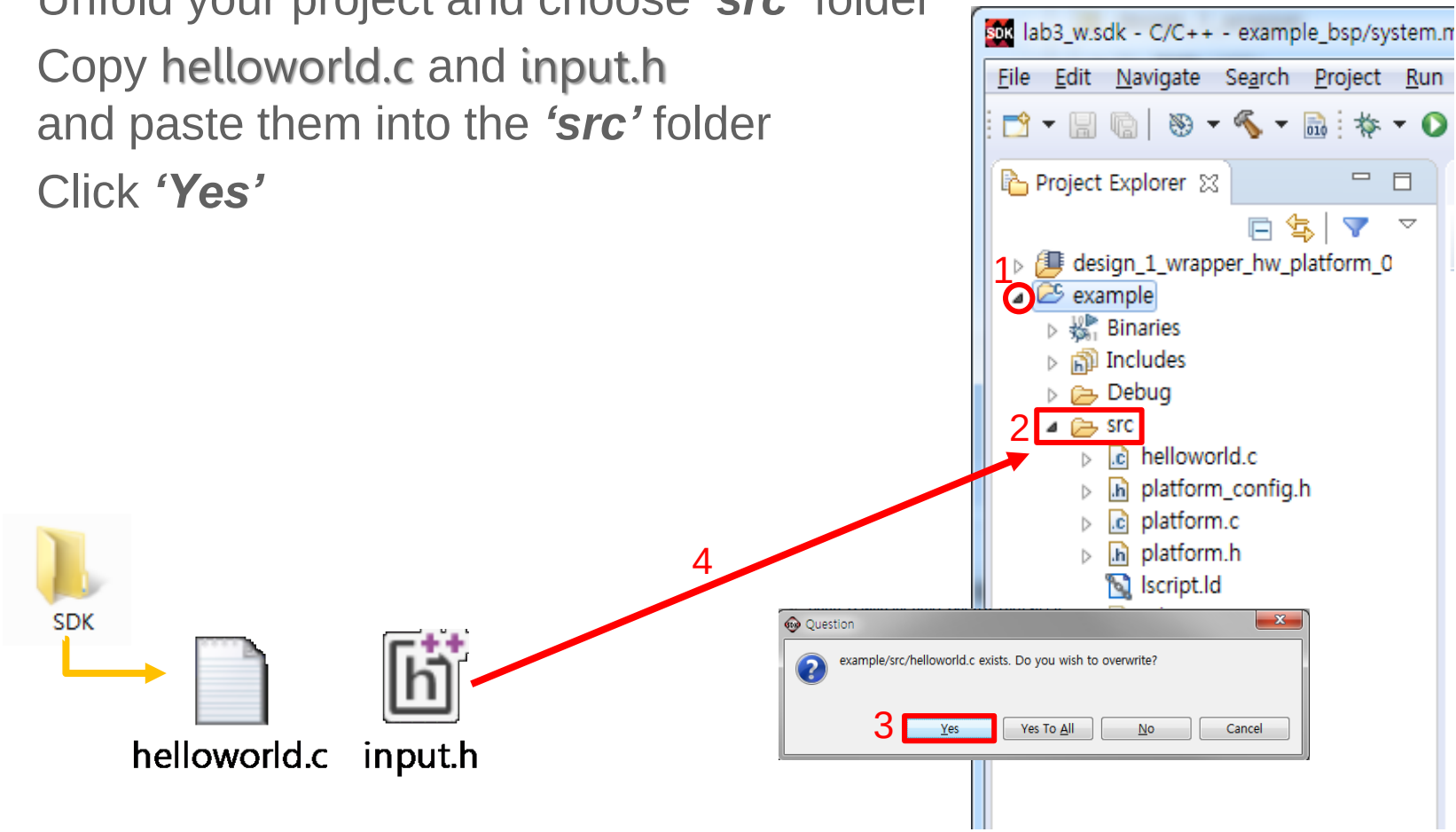
Running C Applications

- Create a C application project (cont'd)
  - Choose ***'Hello World'*** and then click ***'Finish'***



Running C Applications

- Add source files
  - Unfold your project and choose ***'src'*** folder
  - Copy helloworld.c and input.h and paste them into the ***'src'*** folder
  - Click ***'Yes'***



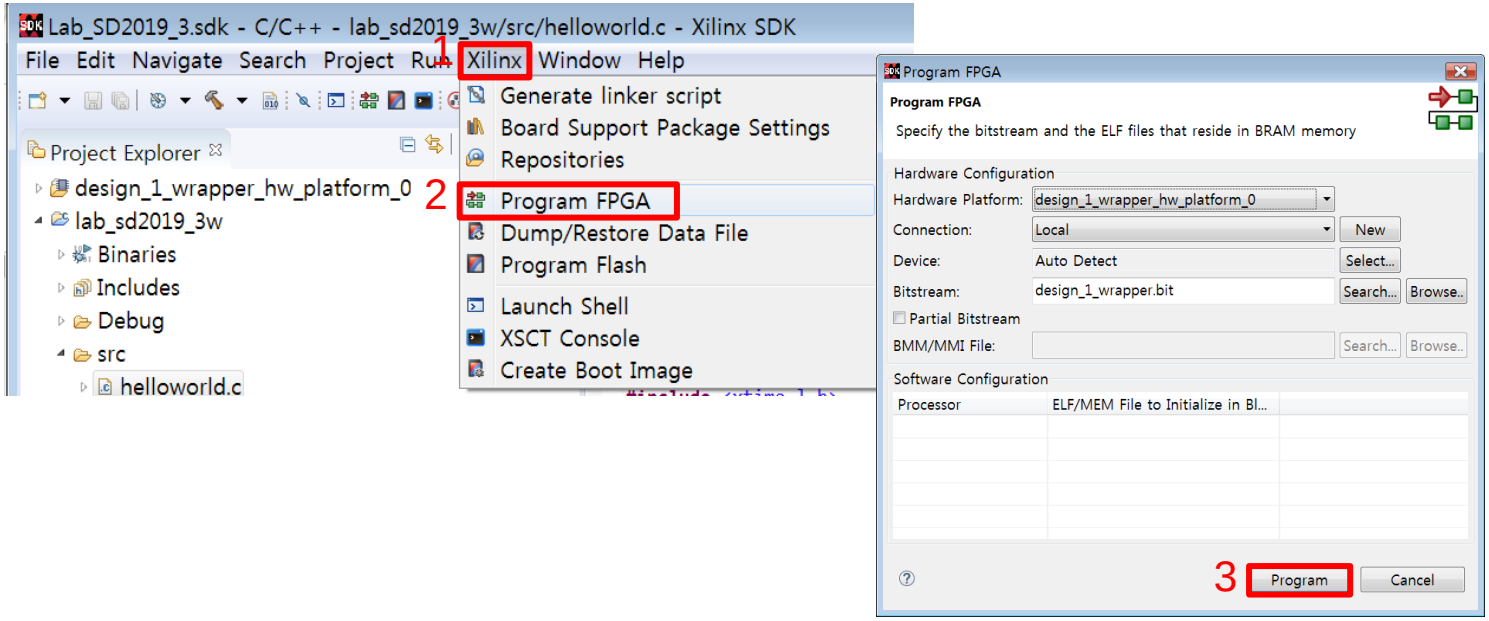
Running C Applications

- Review the function '***main()***'
  - ① Assigns input data into an array
  - ② Sends input data to and get output data from a HW block
  - ③ Converts output data to binary
  - ④ Prints binary output data

```
Xil_Out32(IP_FOR_PS_BASE,0x7FFFFFFF);

for(i = 0; i < 134; i++)                                    ①
{
    tmp  = inReal[i]<<16;
    tmp1 = (0x0000FFFF & inImag[i]);
    input_array[i] = tmp + tmp1;
}

for(i = 0; i < 64; i++)                                     ②
{
    Xil_Out32(IP_FOR_PS_BASE, input_array[i]);
    output_array[i] = Xil_In32(IP_FOR_PS_BASE + 4*i);
}

for(i = 0; i < 134; i++)                                    ③
{
    for(j=0;j<32;j++)
        if ((output_array[i]>>(31-j))&0x00000001)
            o[i][j] = '1';
        else
            o[i][j] = '0';
}

for(i = 0; i < 134; i++)                                    ④
{
    xil_printf("%3d: ",i);
    for(j=0;j<16;j++)
        xil_printf("%c",o[i][j]);
    xil_printf(" ");
    for(j=16;j<32;j++)
        xil_printf("%c",o[i][j]);
    xil_printf("\n");
}
```

Running C Applications

- Program FPGA
  - Choose the ***'Xilinx'*** menu and then click ***'Program FPGA'***
  - Click ***'Program'***

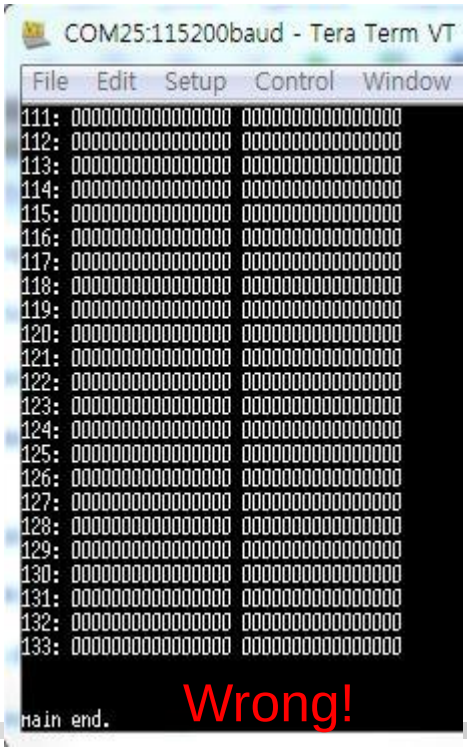


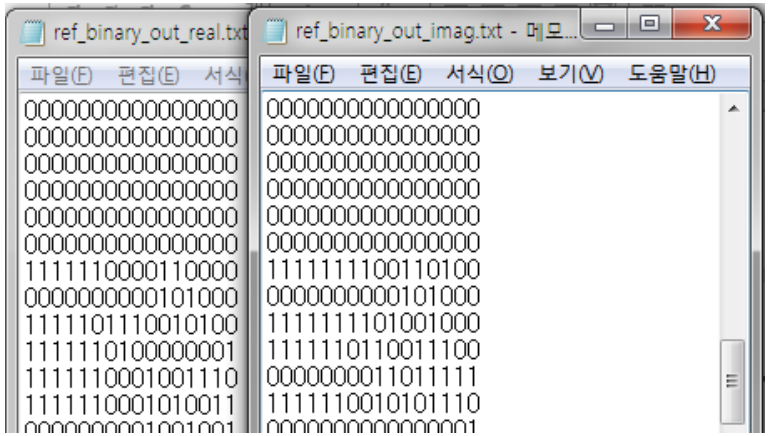
Running C Applications

- Repeat the previous steps
  - Follow pp. 31~34 of the following lab workbook: **Lab_EC_0w.pdf**

Running C Applications

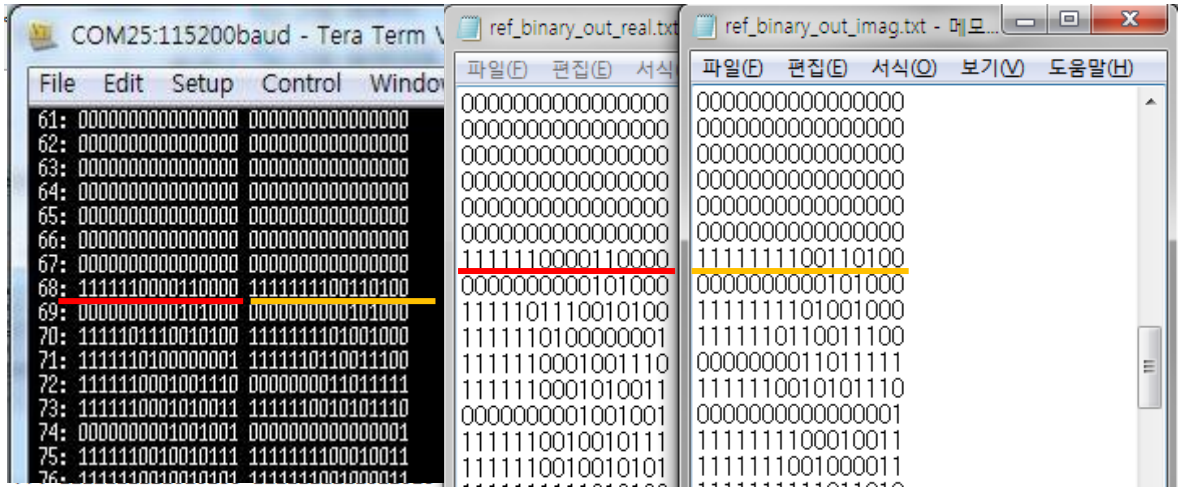
- Run the application
  - Check the output of the application on ***'Tera Term'***
    - ✓ The reference output is provided in ***'ref_binary_out_real.txt'*** and ***'ref_binary_out_imag.txt'*** that are located in the ***'references'*** folder.





Running C Applications

- Modify the function '***main()***'
  - Check the loop counts.
  - The output of the application should match the reference output as follows.

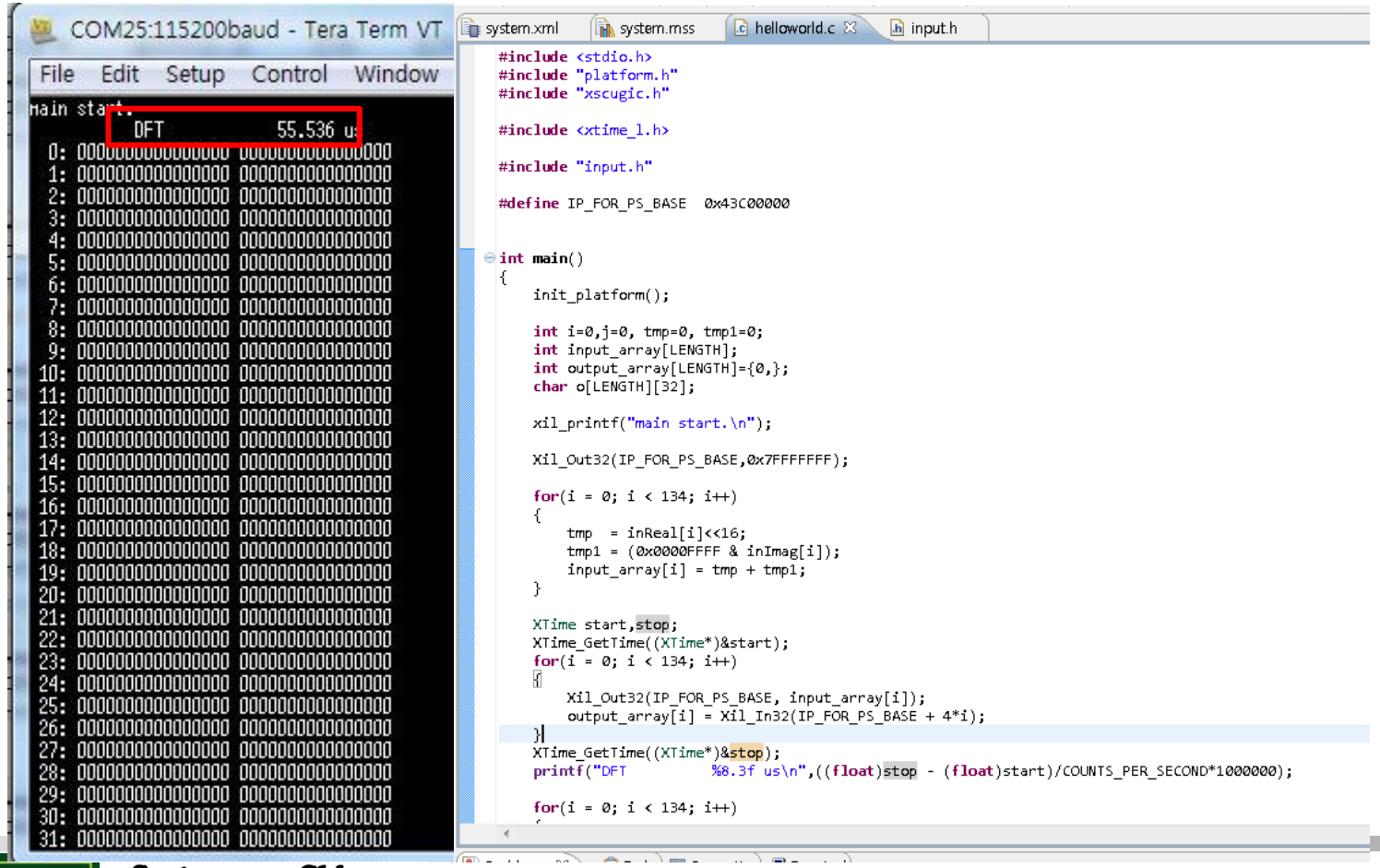


Correct!

```
for(i = 0; i < 134; i++)
{
    tmp  = inReal[i]<<16;
    tmp1 = (0x0000FFFF & inImag[i]);
    input_array[i] = tmp + tmp1;
}

for(i = 0; i < 64; i++)
{
    Xil_Out32(IP_FOR_PS_BASE, input_array[i]);
    output_array[i] = Xil_In32(IP_FOR_PS_BASE + 4*i);
}

for(i = 0; i < 134; i++)
{
    for(j=0;j<32;j++)
        if ((output_array[i]>>(31-j))&0x00000001)
            o[i][j] = '1';
        else
            o[i][j] = '0';
}


for(i = 0; i < 134; i++)
{
    xil_printf("%3d: ",i);
    for(j=0;j<16;j++)
        xil_printf("%c",o[i][j]);
    xil_printf(" ");
    for(j=16;j<32;j++)
        xil_printf("%c",o[i][j]);
    xil_printf("\n");
}
```

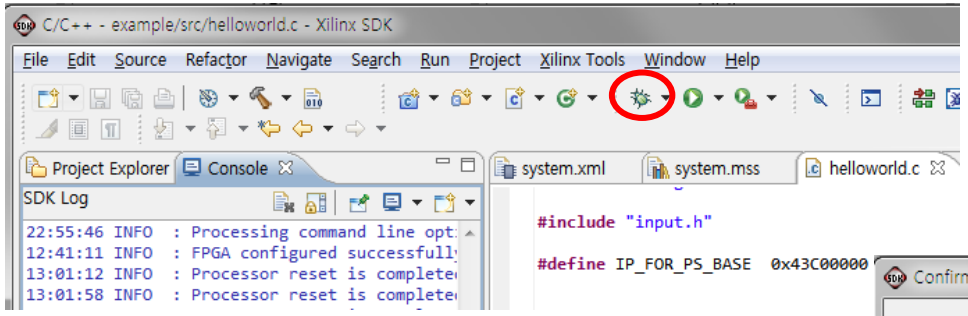
Running C Applications

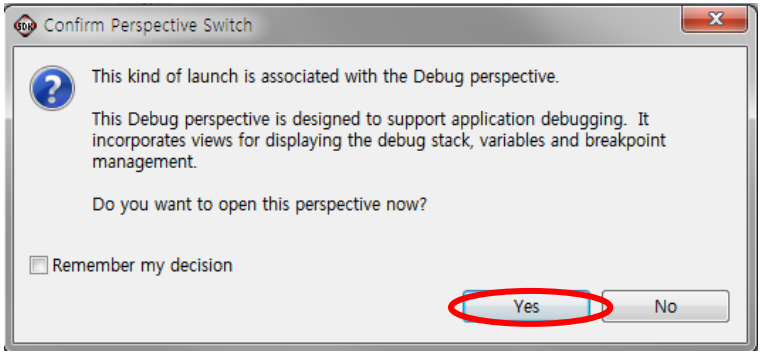
- Measure the execution time
  - Modify the ‘main’ function as below.



Debugging Designs in ILA

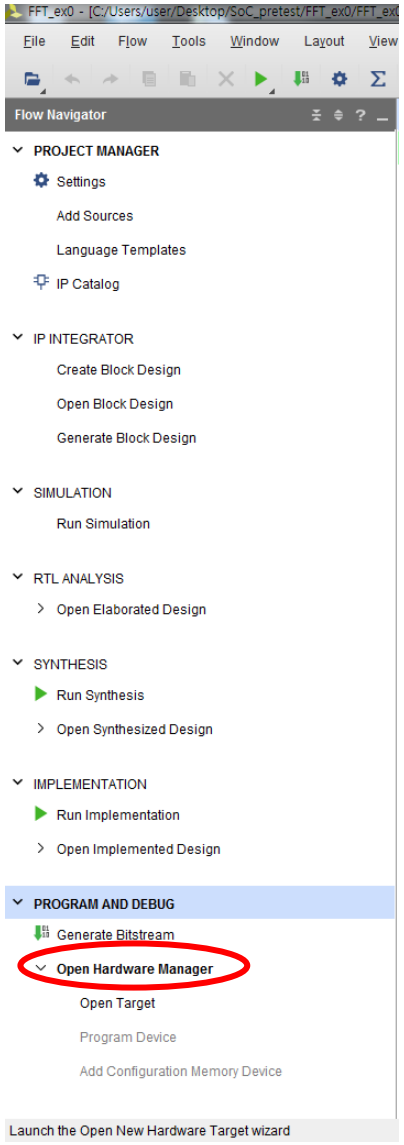
- Run Debug
  - Click the ***'Debug'*** icon and then click ***'Yes'***.





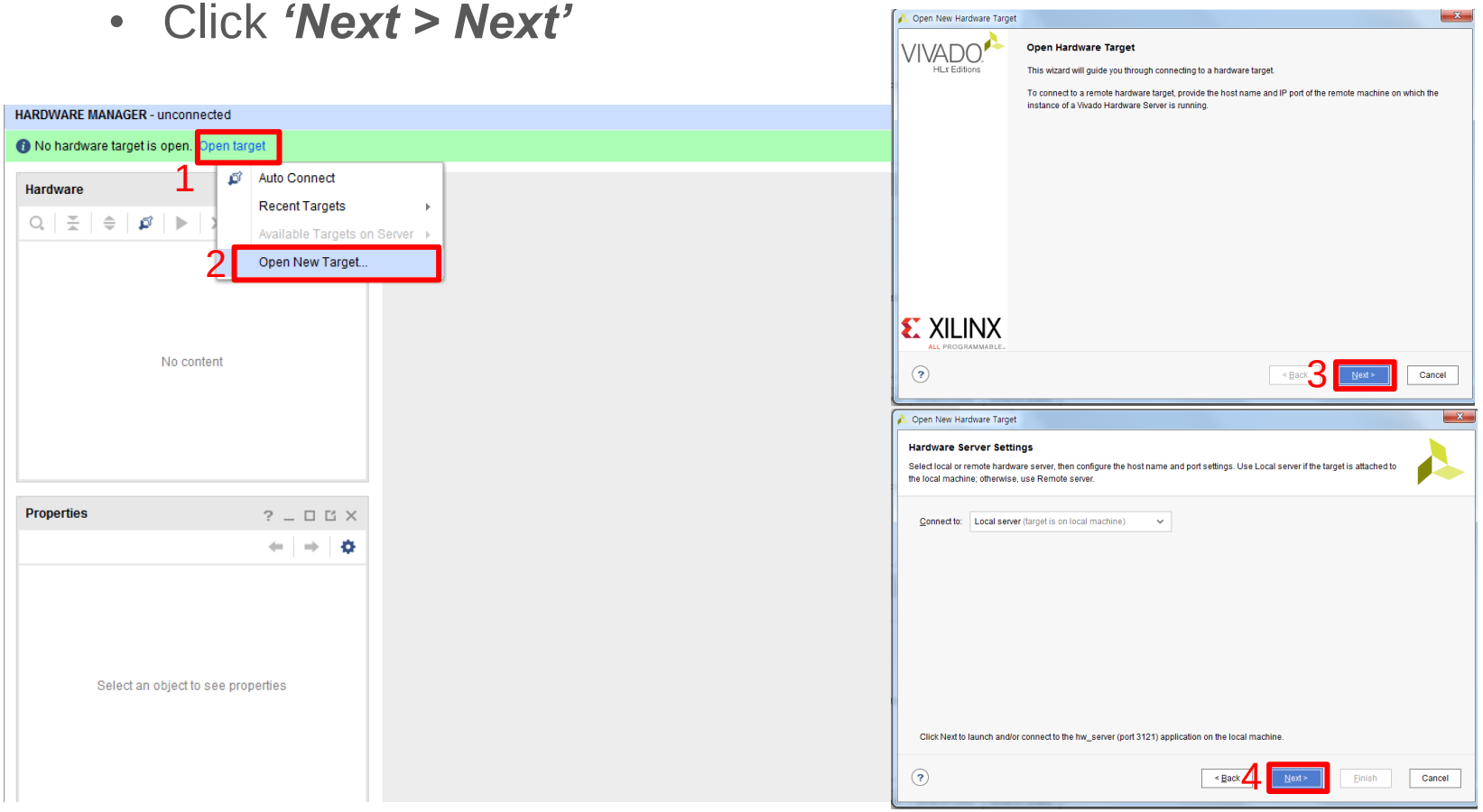
Debugging Designs in ILA

- Open Hardware Manager
  - Go back to the Vivado project that includes the Block Design.
  - Make sure that both Block Design and Implemented Design are open.
  - Click ***'Open Hardware Manager'***



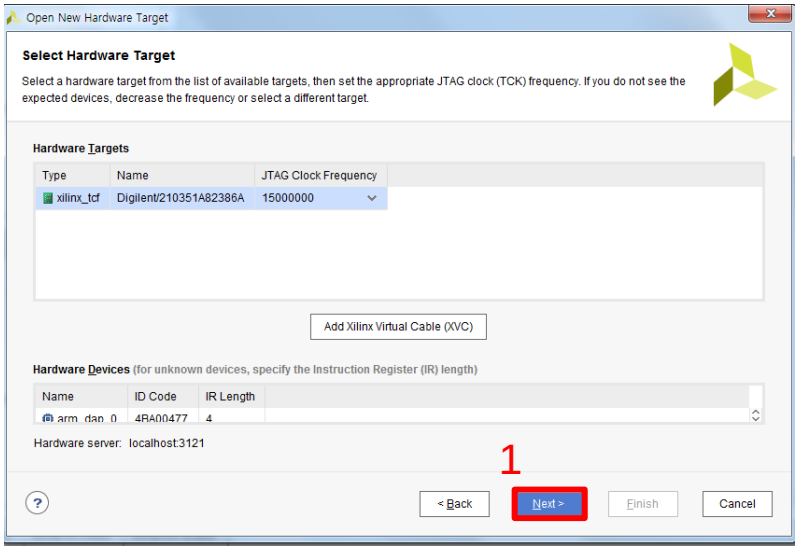
Debugging Designs in ILA

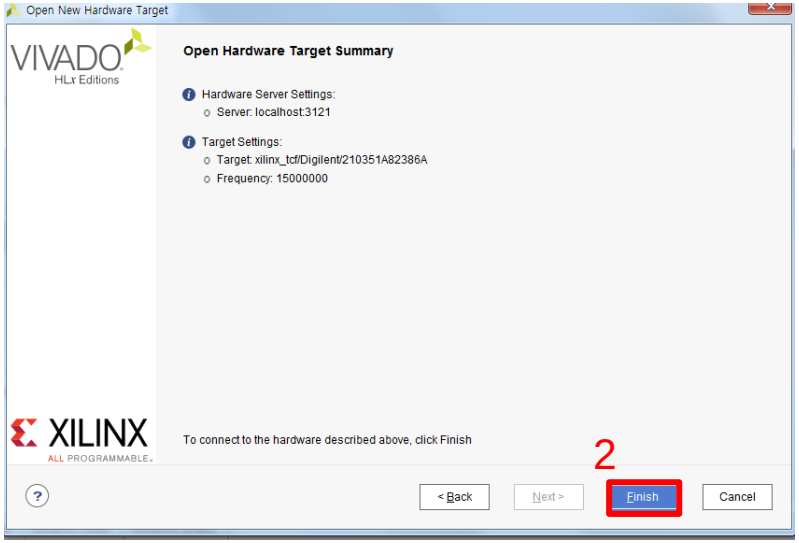
- Open Hardware Manager (cont'd)
  - Click ***'Open target > Open New Target'***
  - Click ***'Next > Next'***



Debugging Designs in ILA

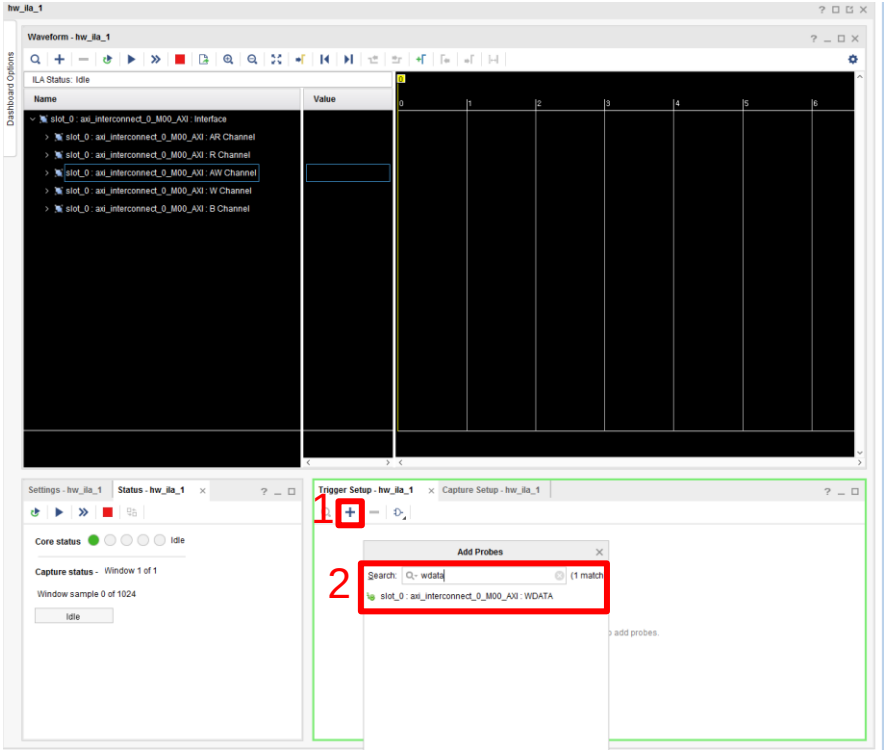
- Open Hardware Manager (cont'd)
  - Click ***'Next > Finish'***





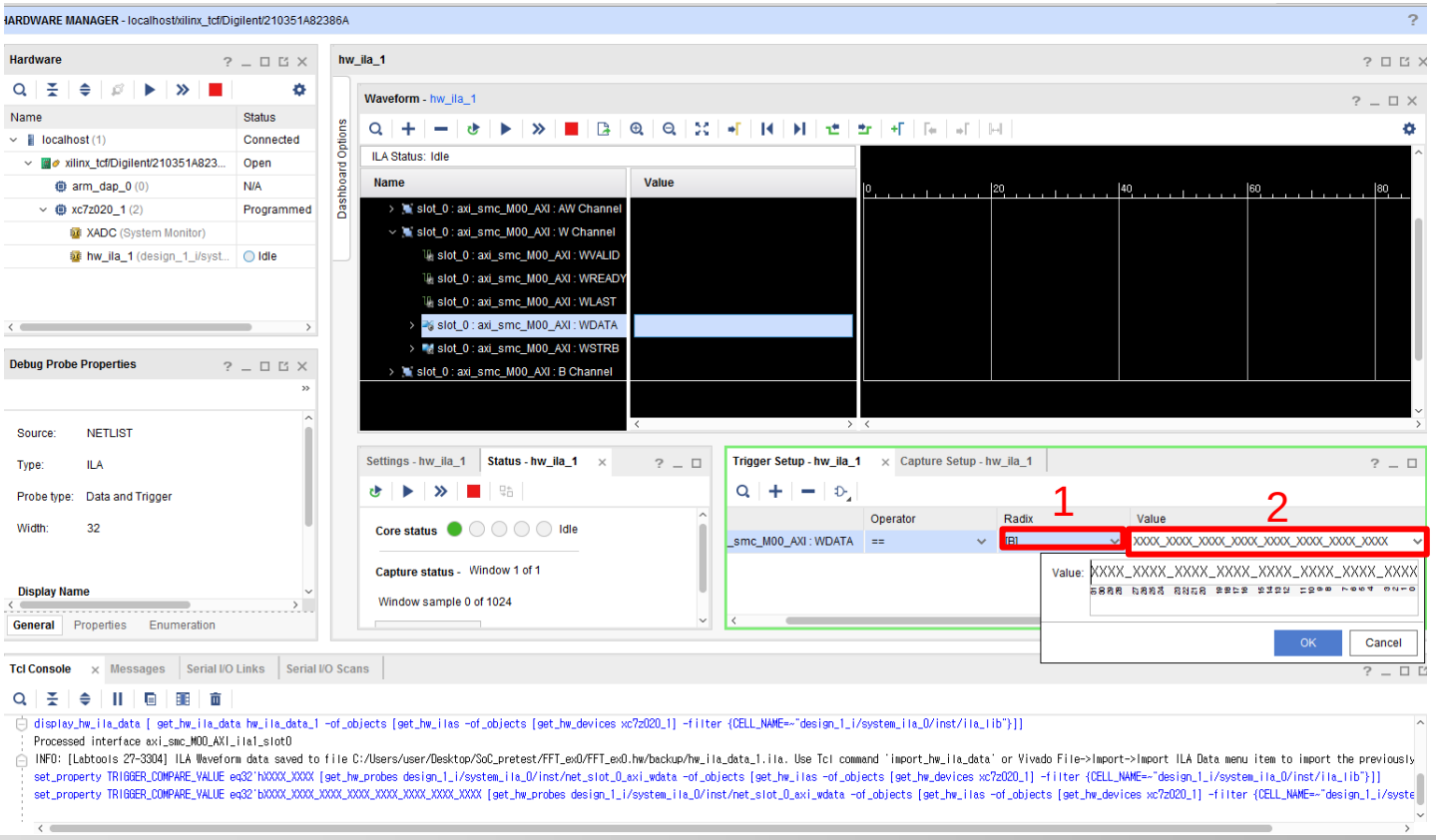
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA)
  - In the ***'Trigger Setup'*** window, click the ***'Add Probe'*** icon, type ***"WDATA"*** in the Search field
  - Double-click the ***'axi_interconnect_0_M00_AXI:WDATA'***



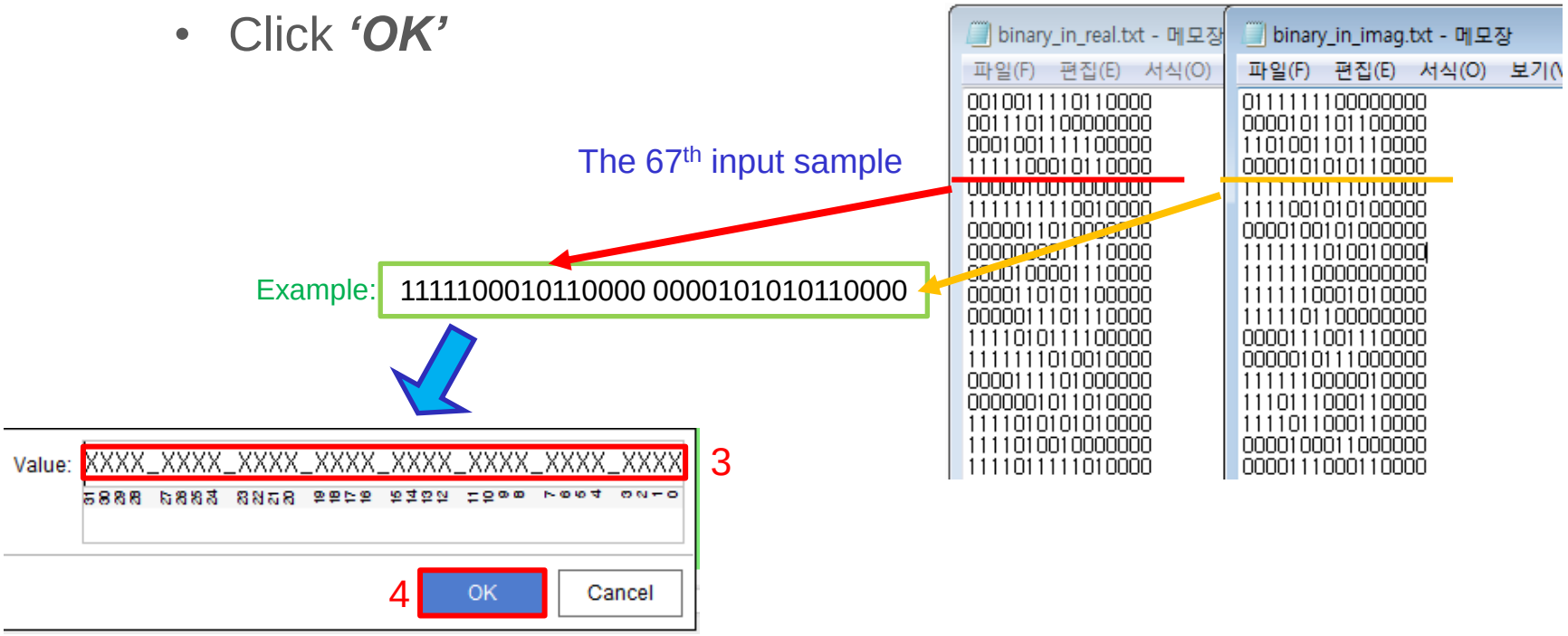
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Choose ***'[B] (Binary)'*** in the '***Radix'*** field
  - Click '***Value'***



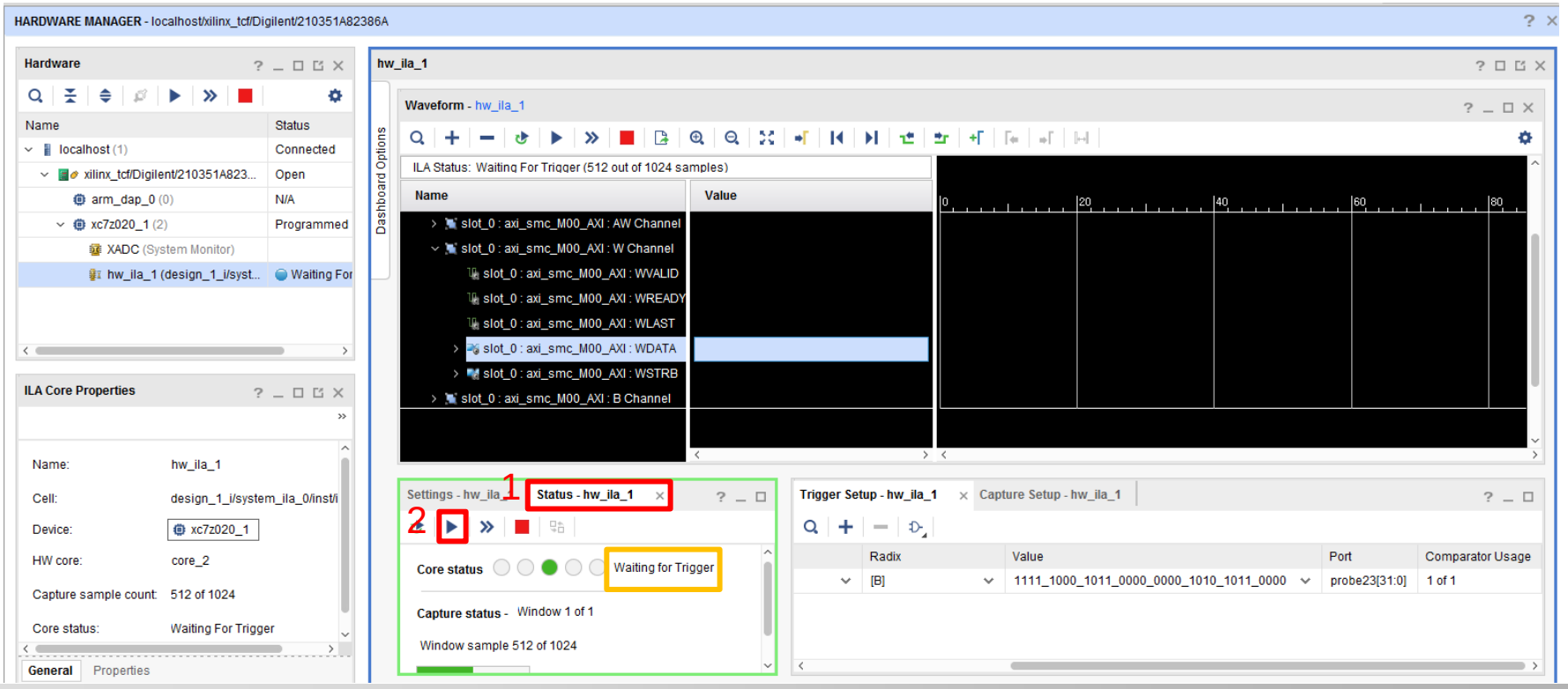
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Enter the input values (both the real and imaginary values) for triggering in the ***'Value'*** field.
    - ✓ The binary input is provided in ***'binary_in_real.txt'*** and ***'binary_in_imag.txt'*** in the **inputs** folder (or in ***'input.h'***)
  - Click ***'OK'***



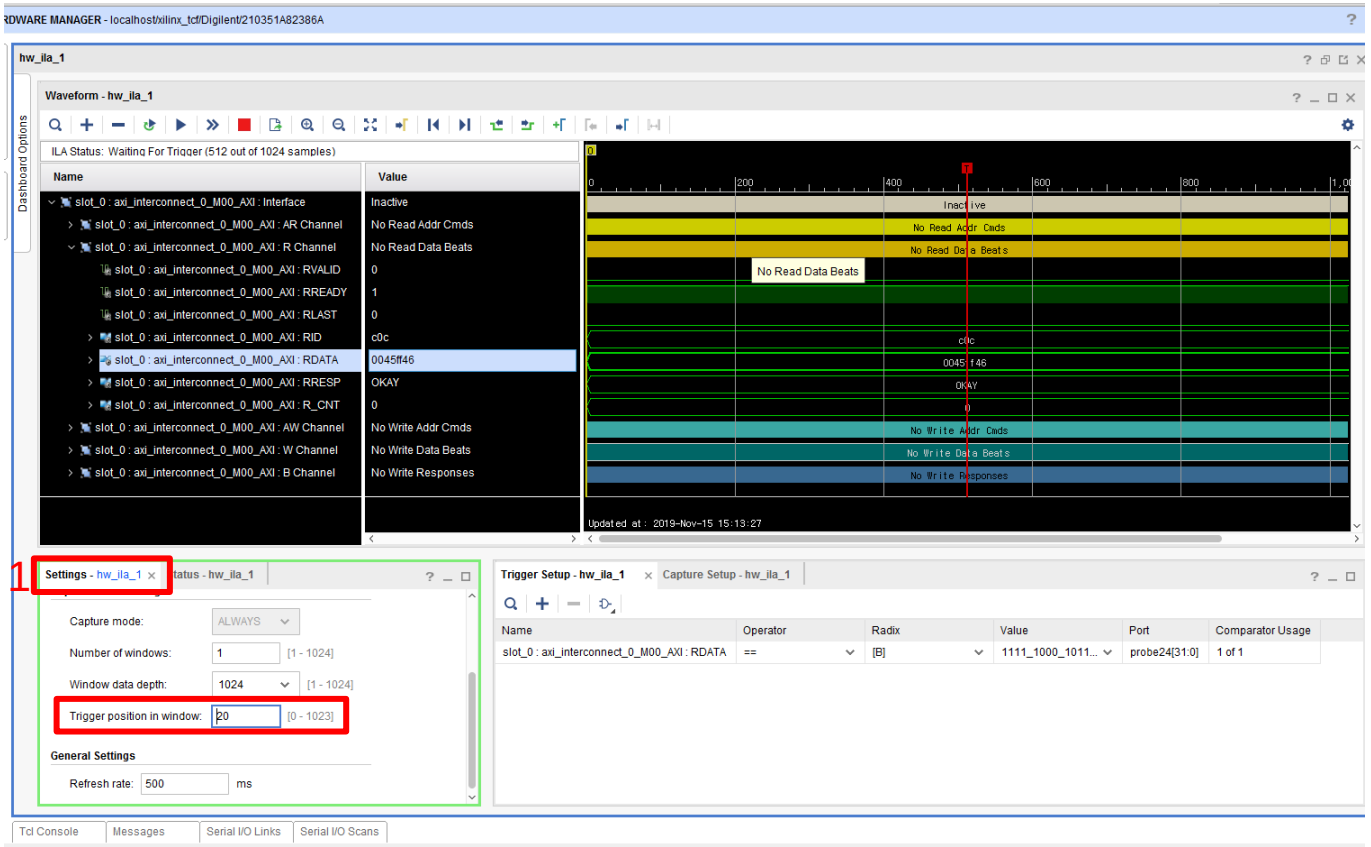
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Choose the ***'Status'*** tap and the click the ***'Run trigger for this ILA core'*** icon
  - Make sure that the status is ***'Waiting For Trigger'***.



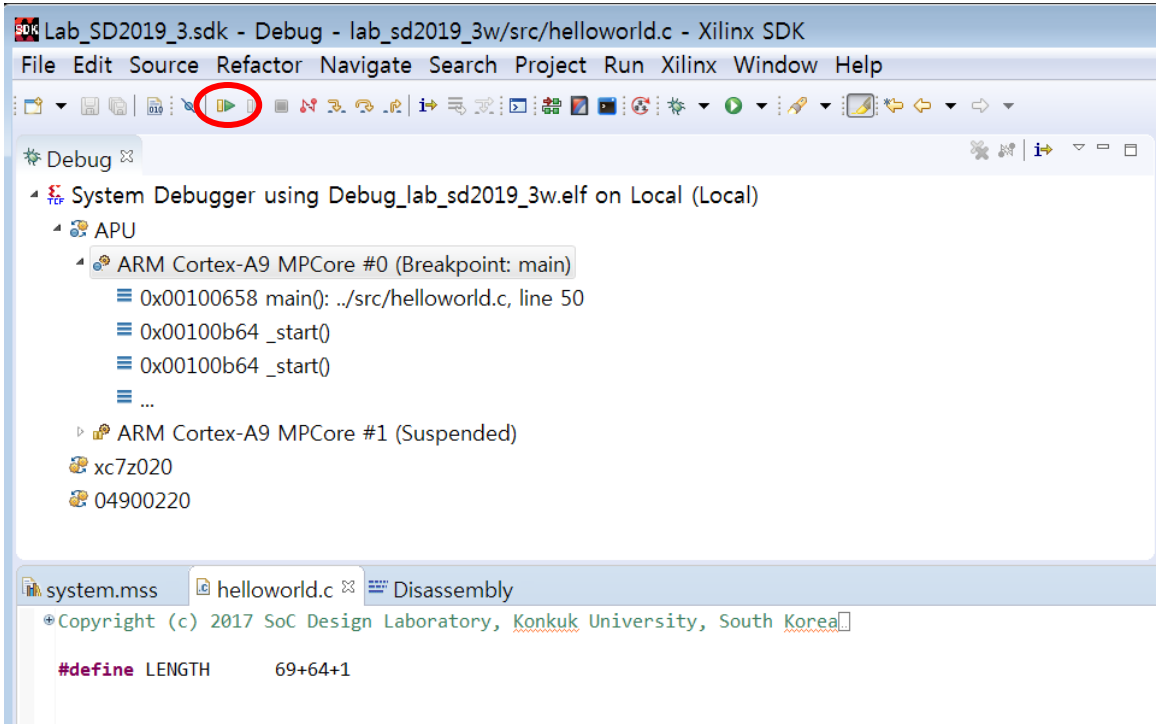
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Choose the ***'Setting'*** tap and the type ***'Trigger position in window'***



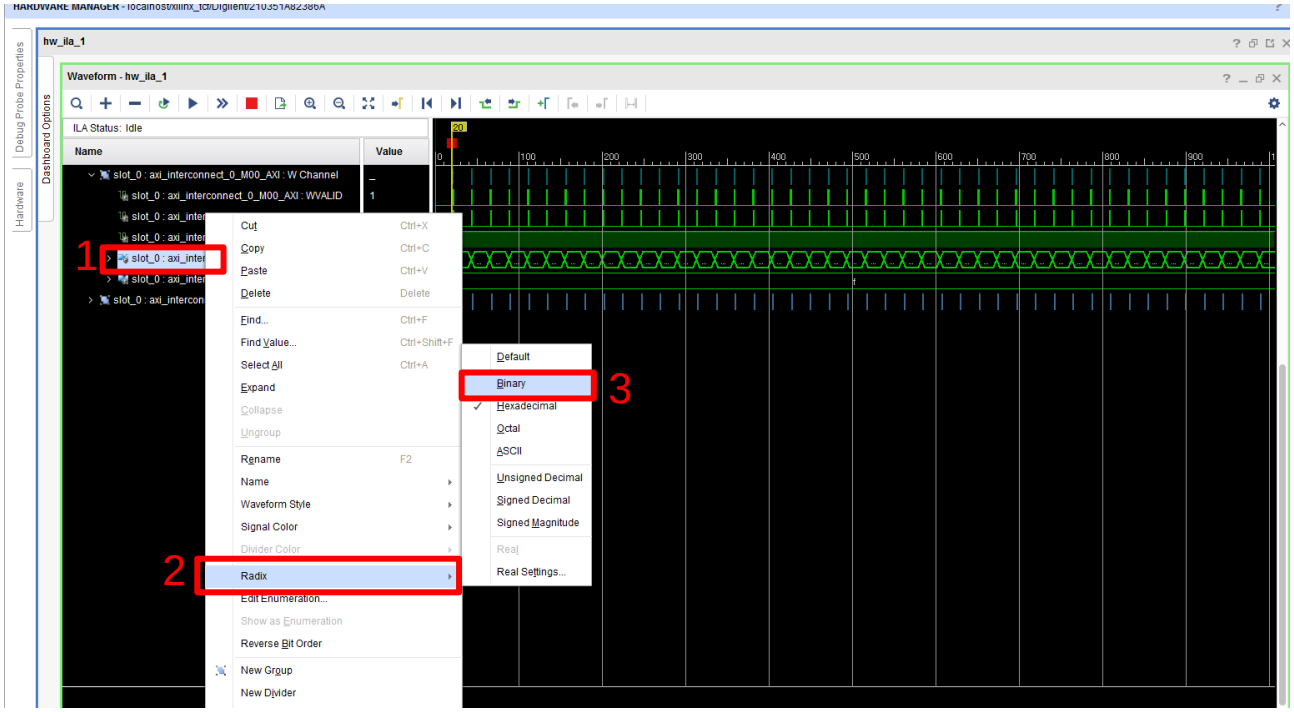
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Go back to SDK and then click the ***'Resume'*** icon
  - Check the output of the application on ***'Tera Term'***



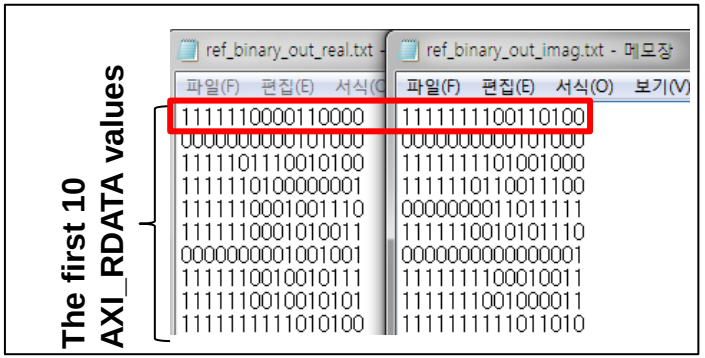
Debugging Designs in ILA

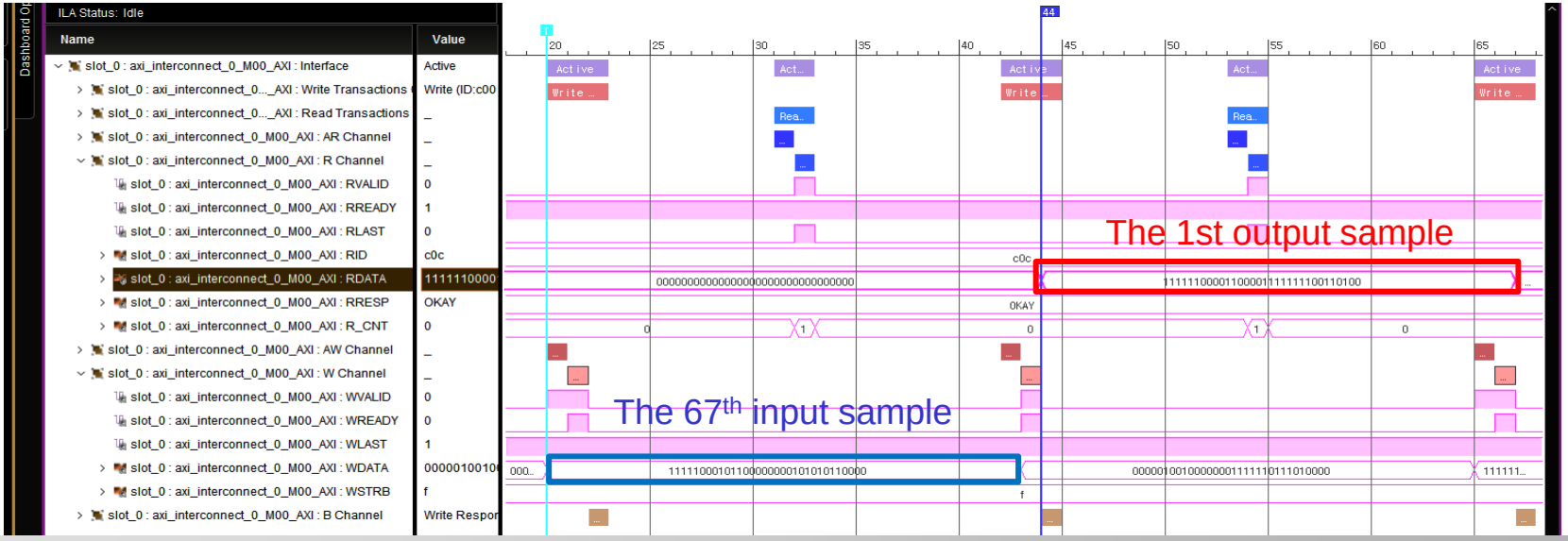
- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Select ***'W Channel > WDATA'*** and ***'R Channel > RDATA'*** and then right-click ***'Radix*** > ***Binary'***
  - Zoom in/out by scrolling up/down while pressing the Ctrl key.



Debugging Designs in ILA

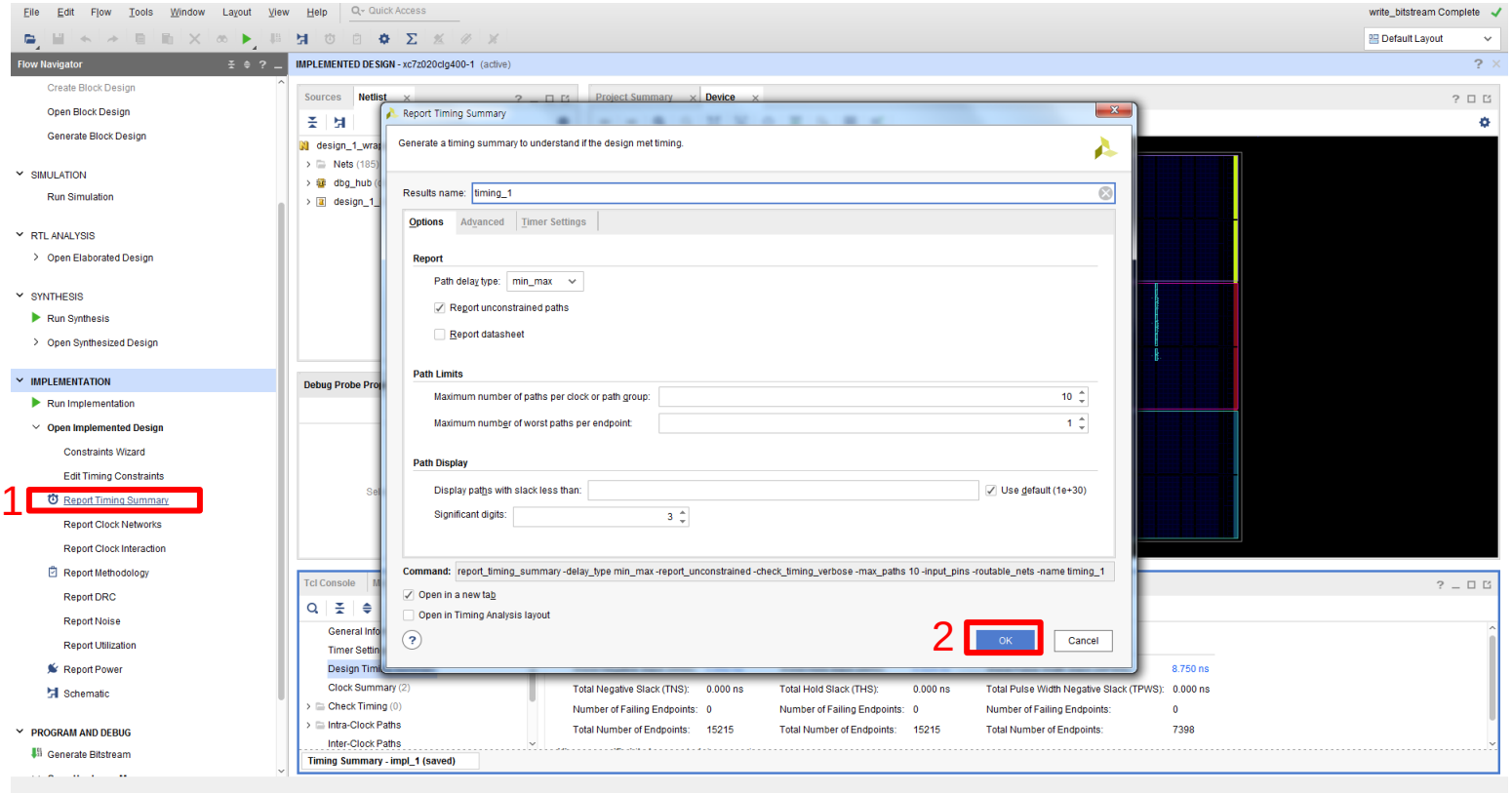
- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Check the first few values of ***'RDATA'*** at the **rising** edges of ***'RVALID'***
  - Compare them with the reference outputs.





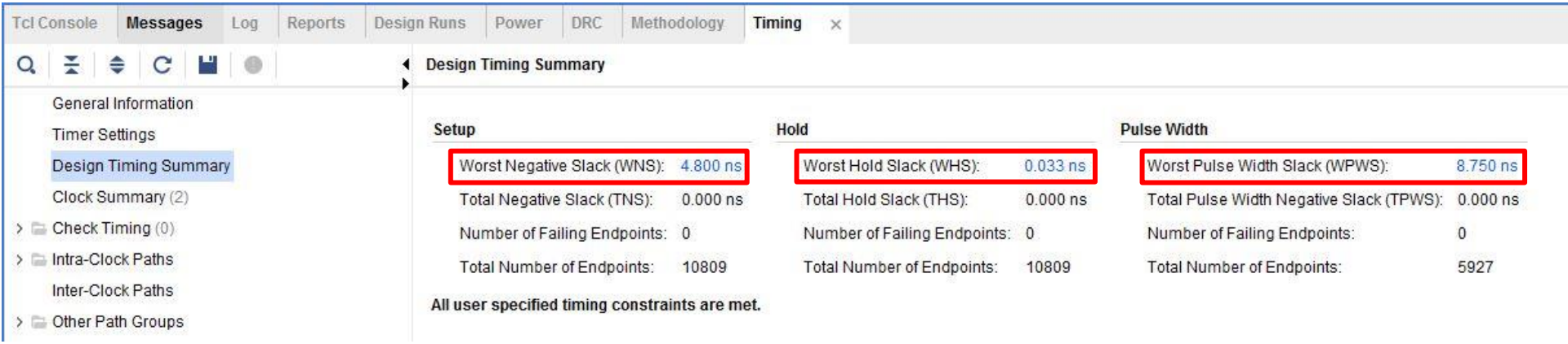
Debugging Designs in ILA

- Measure the execution time
  - Click ***'Report Timing Summary'*** and then click ***'OK'***.



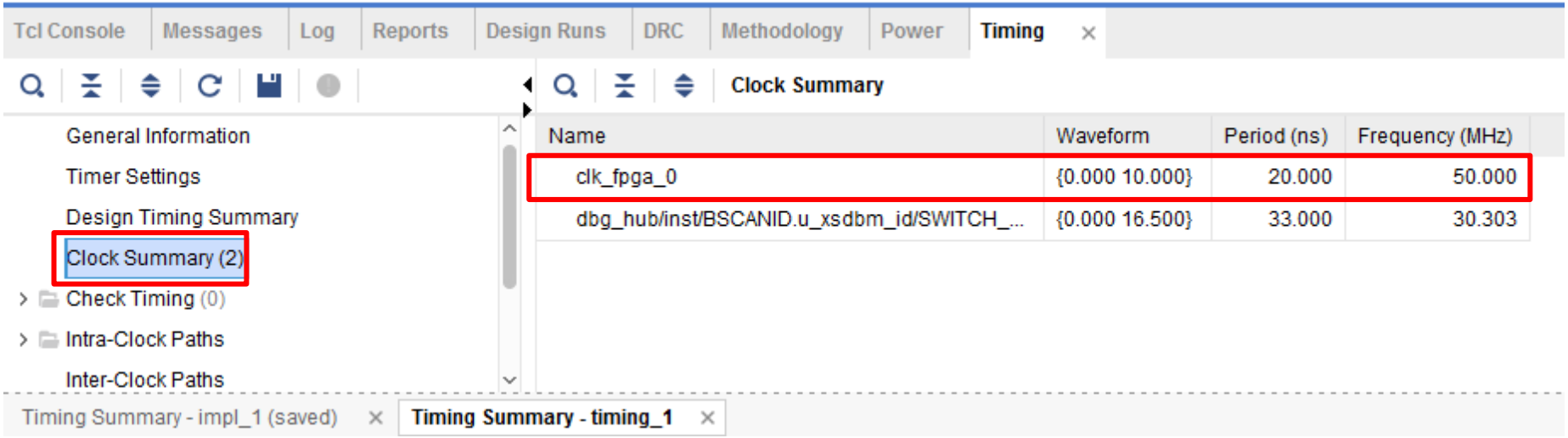
Debugging Designs in ILA

- Measure the execution time (cont'd)
  - Choose ***'Timing'*** tap and the click ***'Design Timing Summary'***
  - Check that the slack is positive.



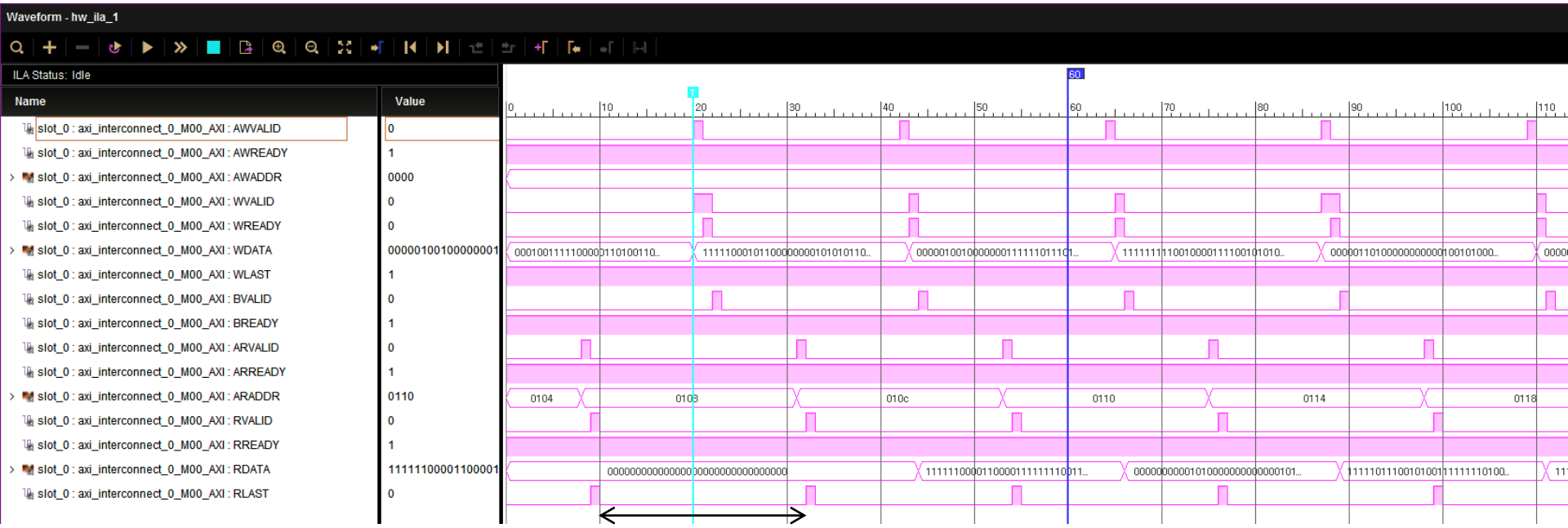
Debugging Designs in ILA

- Measure the execution time (cont'd)
  - Click ***'Clock Summary'*** and check the clock period.



Debugging Designs in ILA

- Measure the execution time (cont'd)
  - Check if it is consistent with the execution time measured by **Xtime**



20 nsec/cycle x ~20 cycles/sample
= 400 nsec/sample

Debugging Designs in ILA

- Review the source files
  - Top module: **axi_slave_FFT_f_ps.v**
  - FIFO (slave): **axi_slave_fifo_sync.v**

Assignment

- Repeat the previous steps for the following designs
  - **128-pt** FFT with **interpolated** inputs
    - ✓ The same WL settings: [1,15] input, [8,8] output
    - ✓ Reordering in **SW** (not in HW)

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**

References

- Vivado Design Suite User Guide
  https://www.xilinx.com/support/documentation-navigation/design-hubs.html