Lab 4: DMA-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 design in ILA

Creating IP Projects

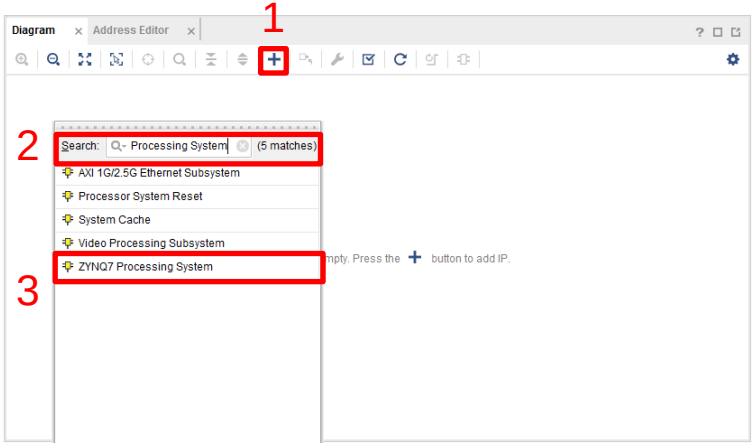
- Repeat the previous steps for '**Creating IP Projects**'
  - Get Started
  - Create a New Vivado Project
  - Enter Project Name
    - ✓ A **new** project name and a **new** project location should be used
  - Choose Project Type
  - Add Sources
    - ✓ HDL source files in the ***'IP'*** folder
  - Add Existing IP & Constraints
  - Choose Default Part
  - Check New Project Summary

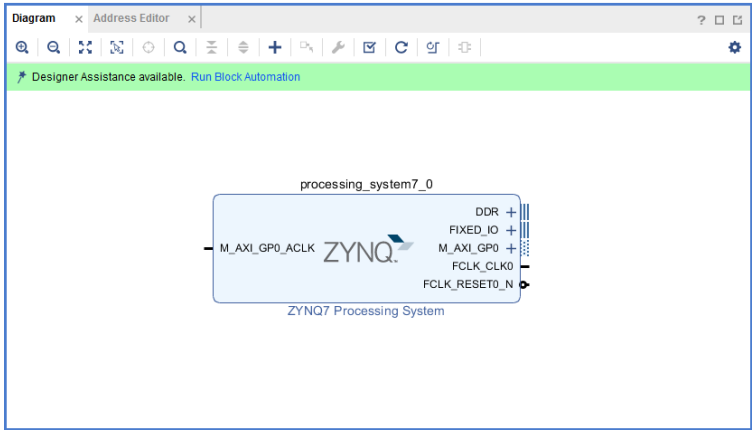
Creating Block Designs

- Repeat the previous steps for '**Creating IP Projects**'
  - 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
  - Add IP repository
  - Create Block Design

Creating Block Designs

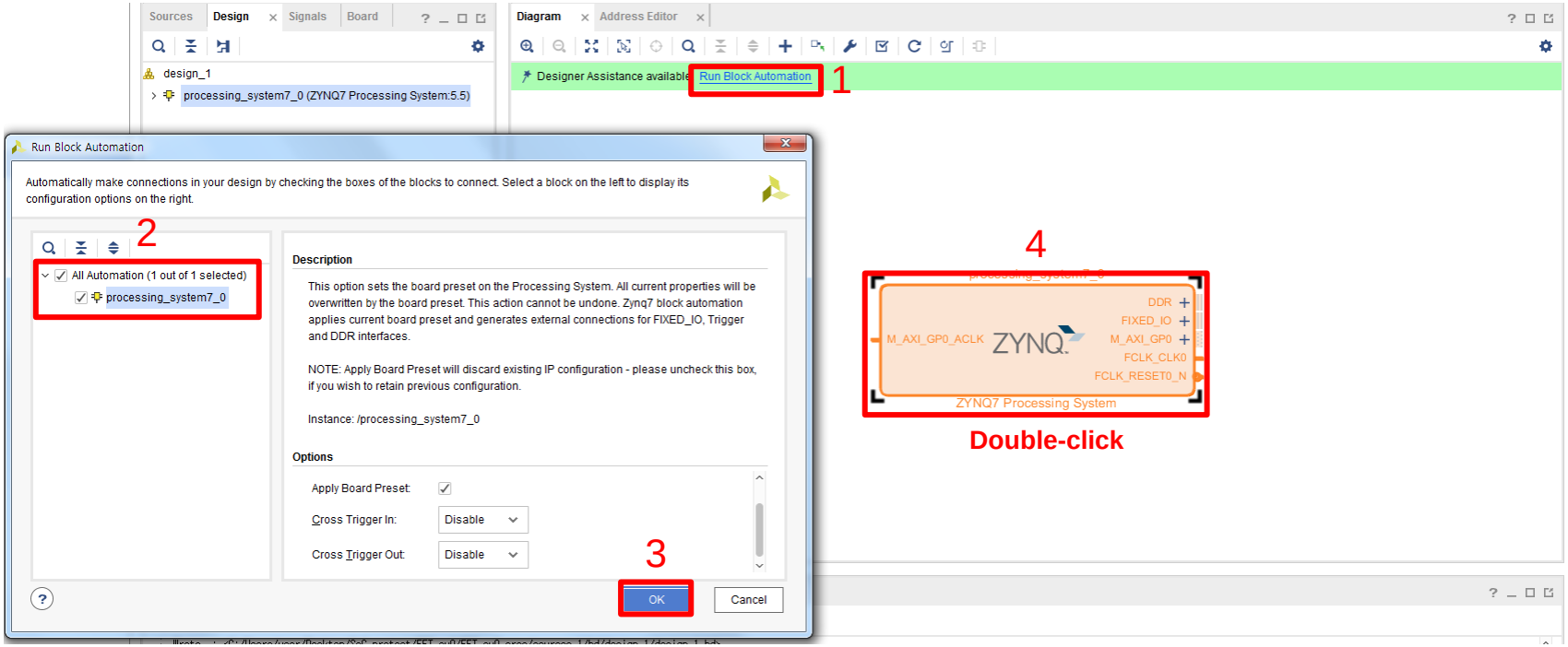
- Add Processing System
  - Click the ***'Add IP'*** icon and type "***Processing System"*** in the Search field
  - Double-click ***'ZYNQ7 Processing System'***





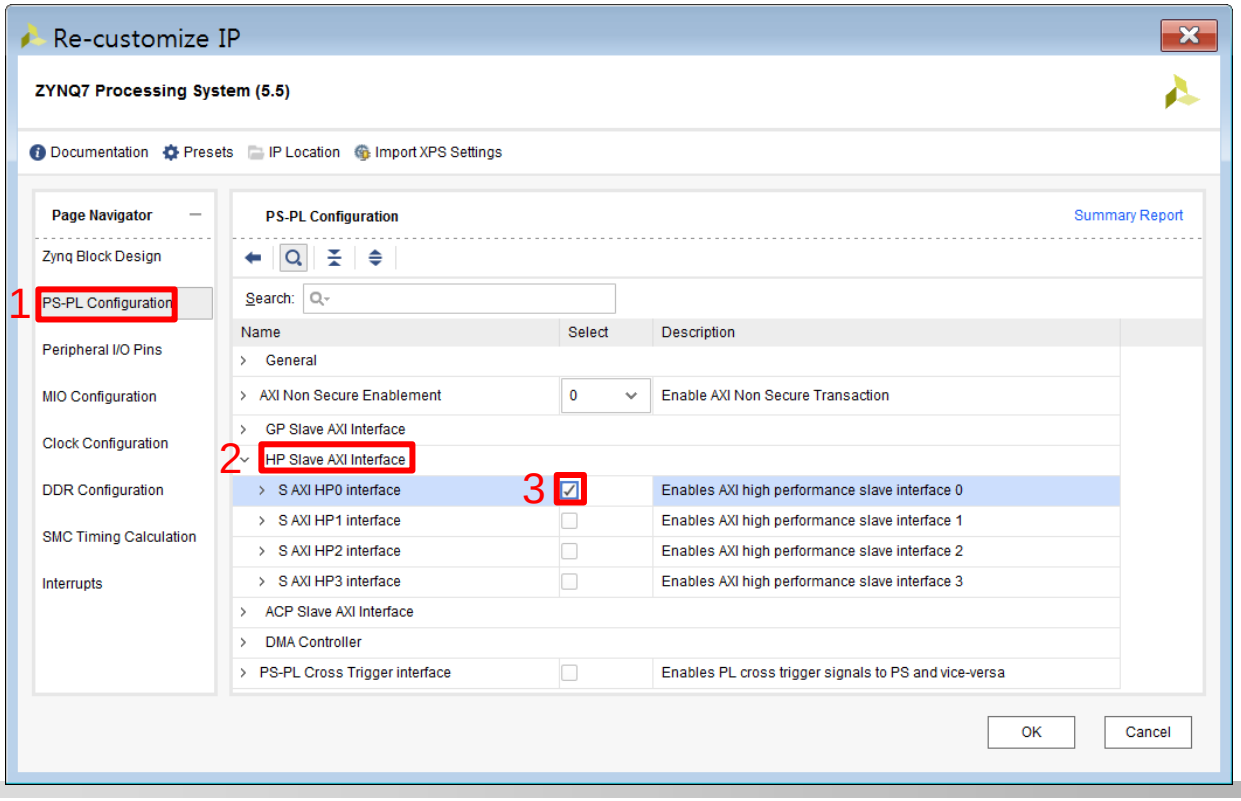
Creating Block Designs

- Make external connections
  - Click ***'Run Block Automation' > '/processing_system7_0'***
  - Double-click the Processing System block



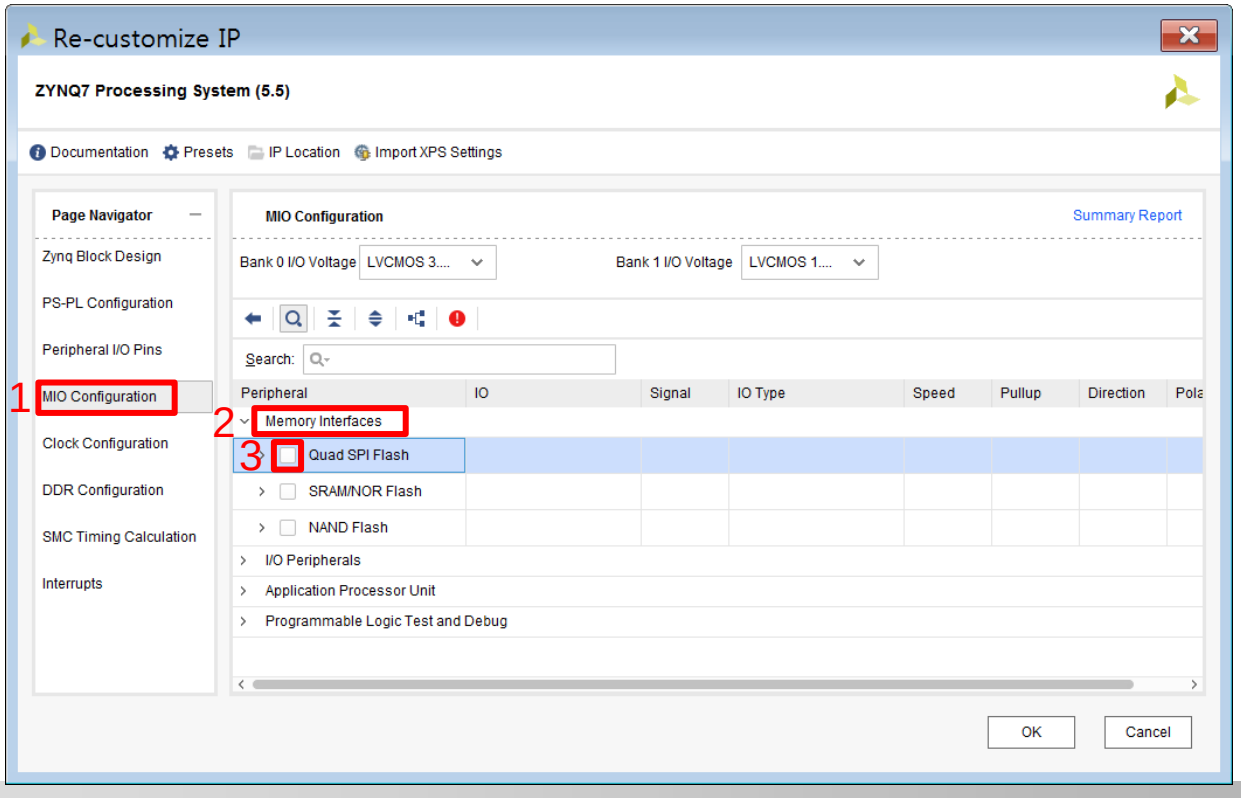
Creating Block Designs

- Re-customize Processing System
  - Click ***'PS-PL Configuration'*** and then unfold ***'HP Slave Interface'***
  - Check ***'S AXI HP0 Interface'***



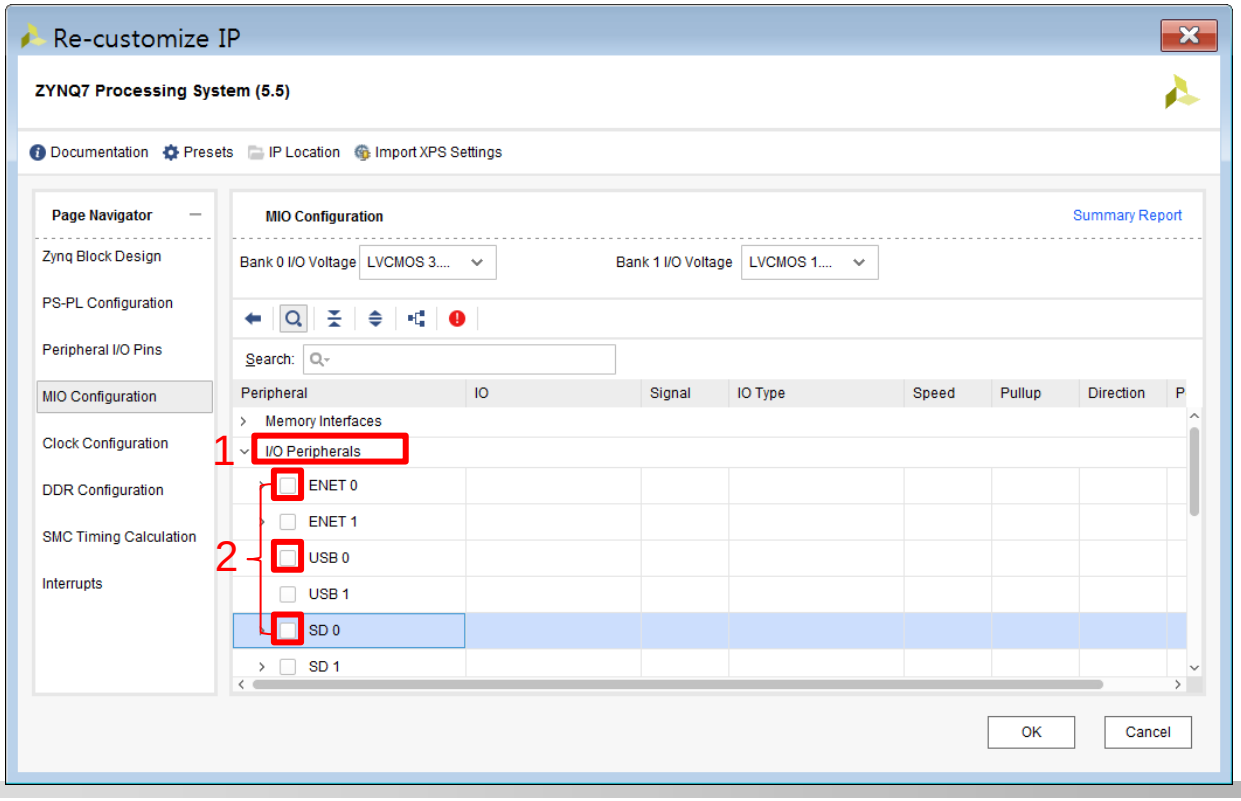
Creating Block Designs

- Re-customize Processing System (Cont'd)
  - Click ***'MIO Configuration'*** and then unfold ***'Memory Interfaces'***
  - Uncheck ***'Quad SPI Flash'***



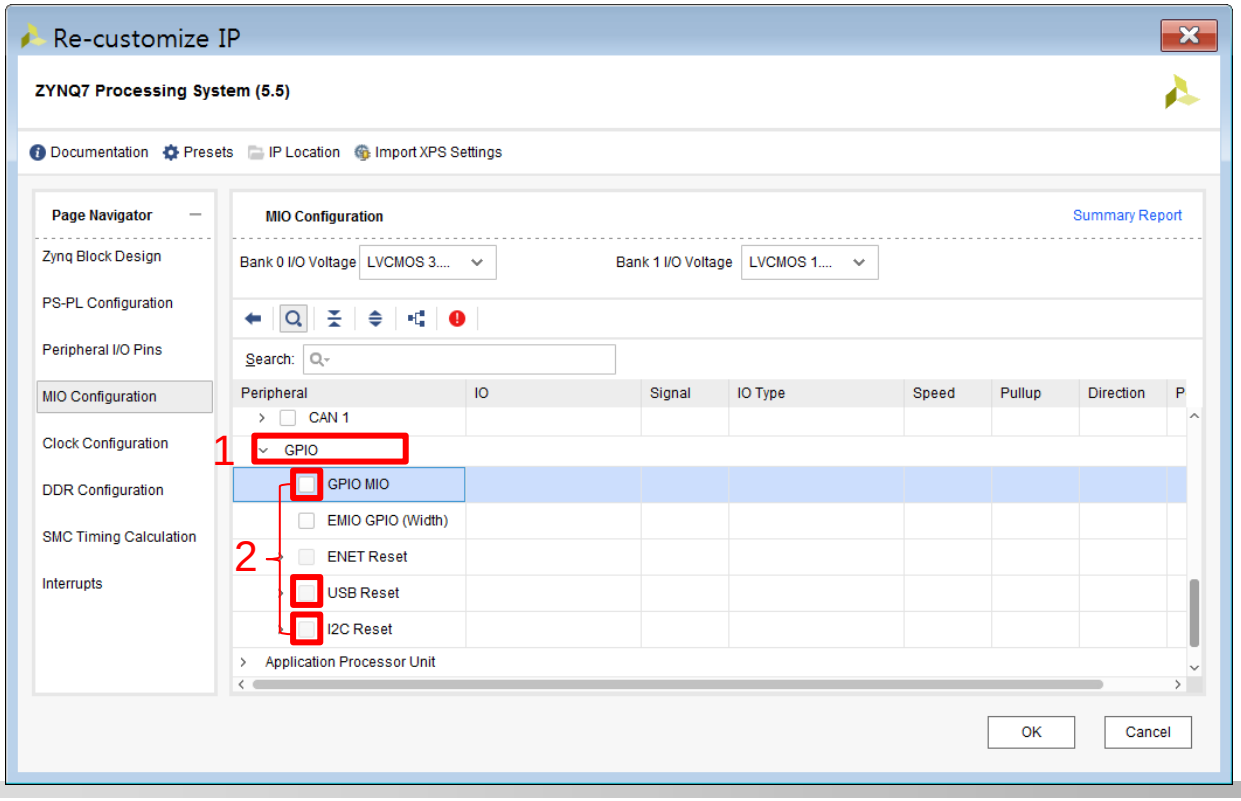
Creating Block Designs

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



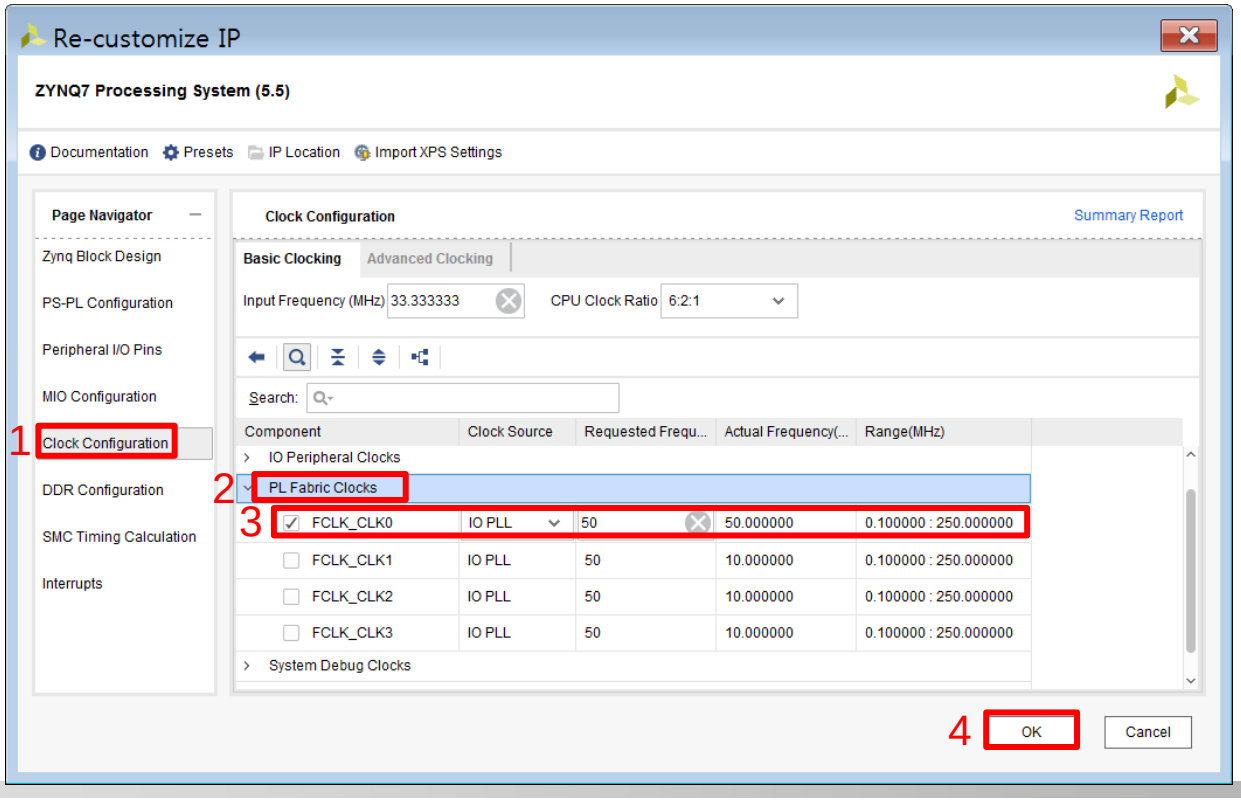
Creating Block Designs

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



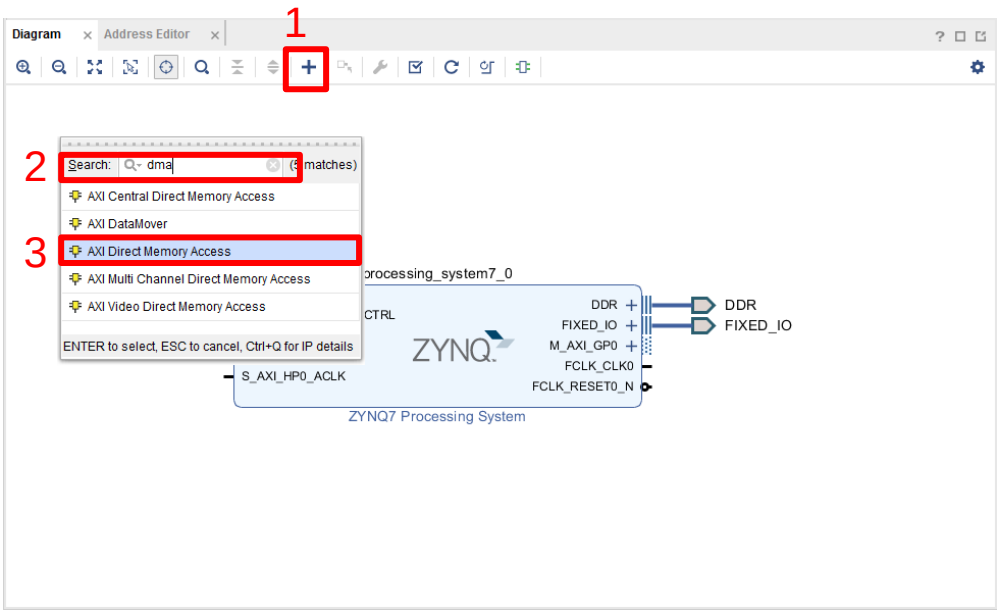
Creating Block Designs

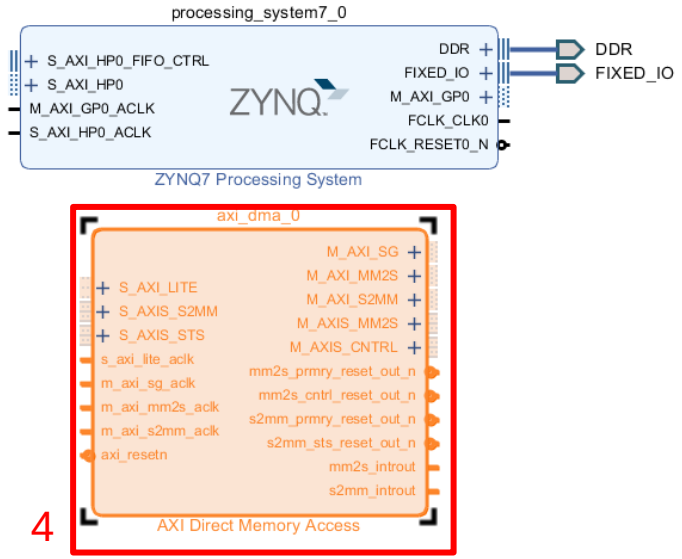
- Re-customize Processing System (Cont'd)
  - Click ***'Clock Configuration'*** and unfold ***'PL Fabric Clocks'***
  - Set ***'FCLK_CLK0'*** to ***'50.00000'*** (MHz)
  - Click ***'OK'***



Creating Block Designs

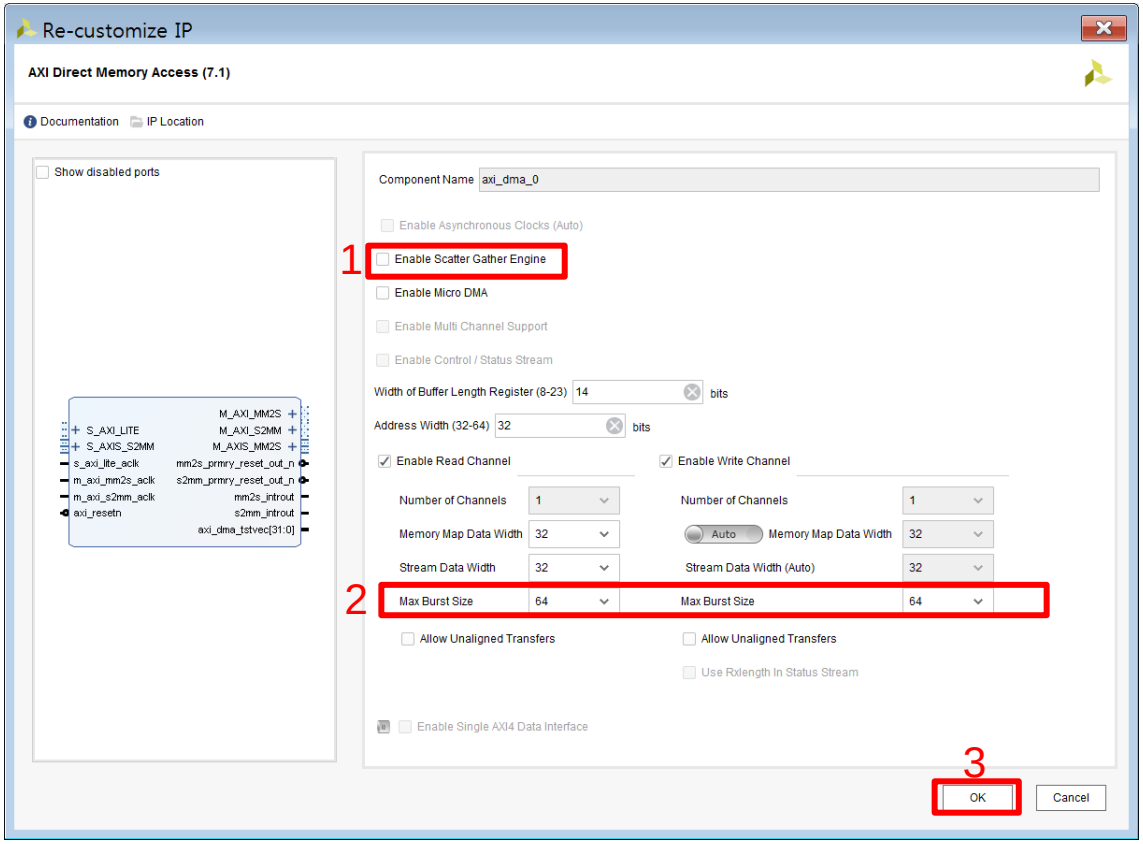
- Add DMA controller
  - Click the ***'Add IP'*** icon and type ***"dma"*** in the Search field
  - Double-click ***'AXI Direct Memory Access'***
  - Double-click the ***AXI Direct Memory Access*** block





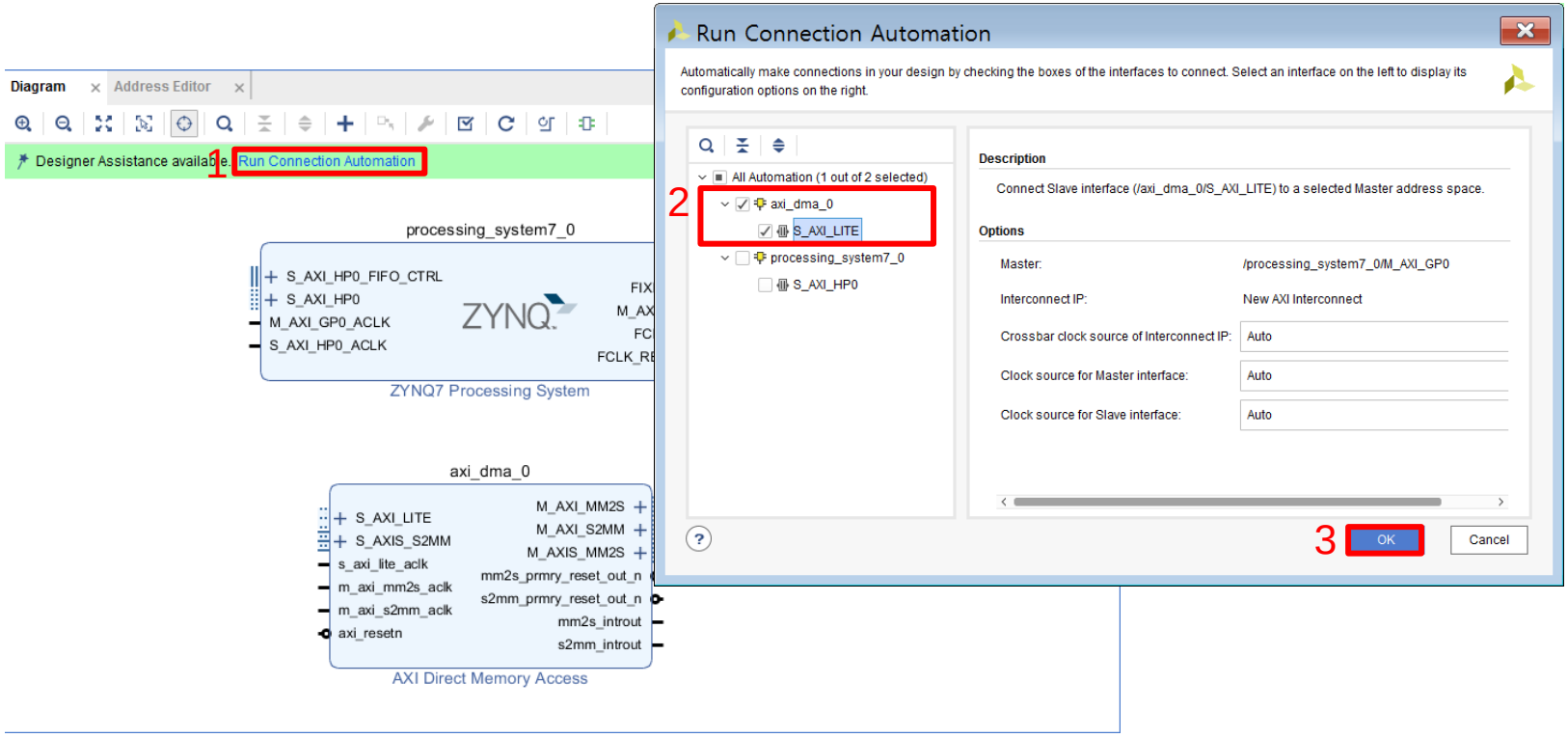
Creating Block Designs

- Add DMA controller (Cont'd)
  - Uncheck ***'Enable Scatter Gather Engine'***
  - Set ***'Max Burst Size'*** to ***'64'***
  - Click ***'OK'***



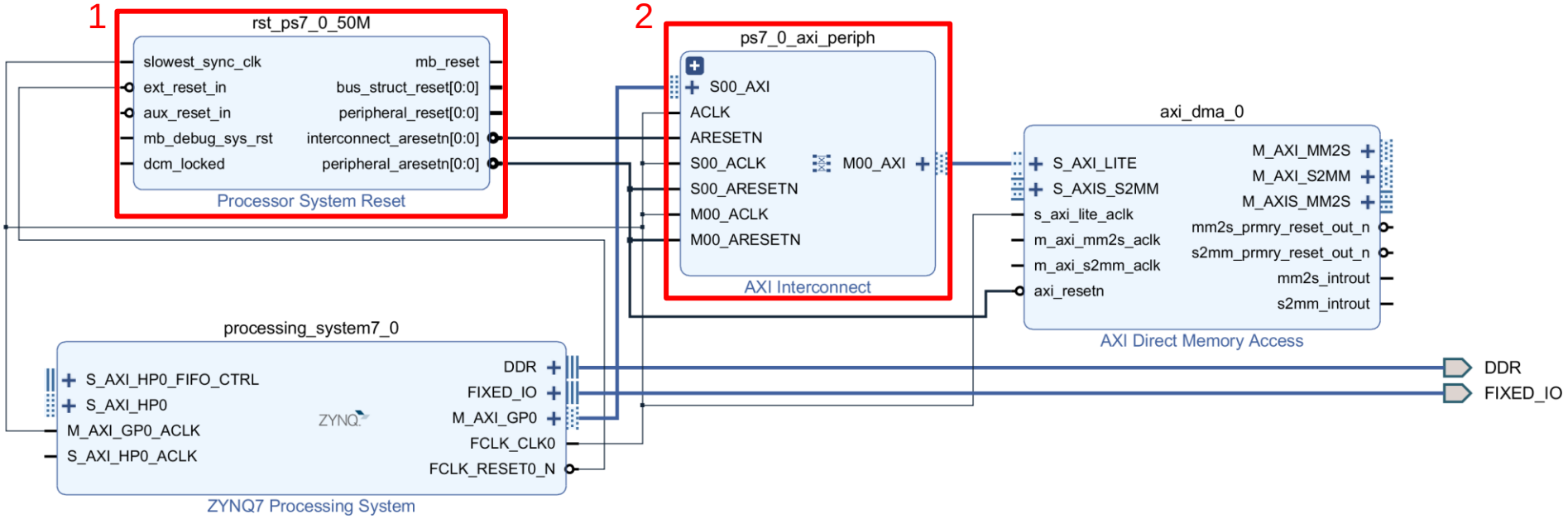
Creating Block Designs

- Run Connection Automation
  - Click the ***'Run Connection Automation'***
  - Check the ***'axi_dma_0/S_AXI_LITE'*** and then click ***'OK'***



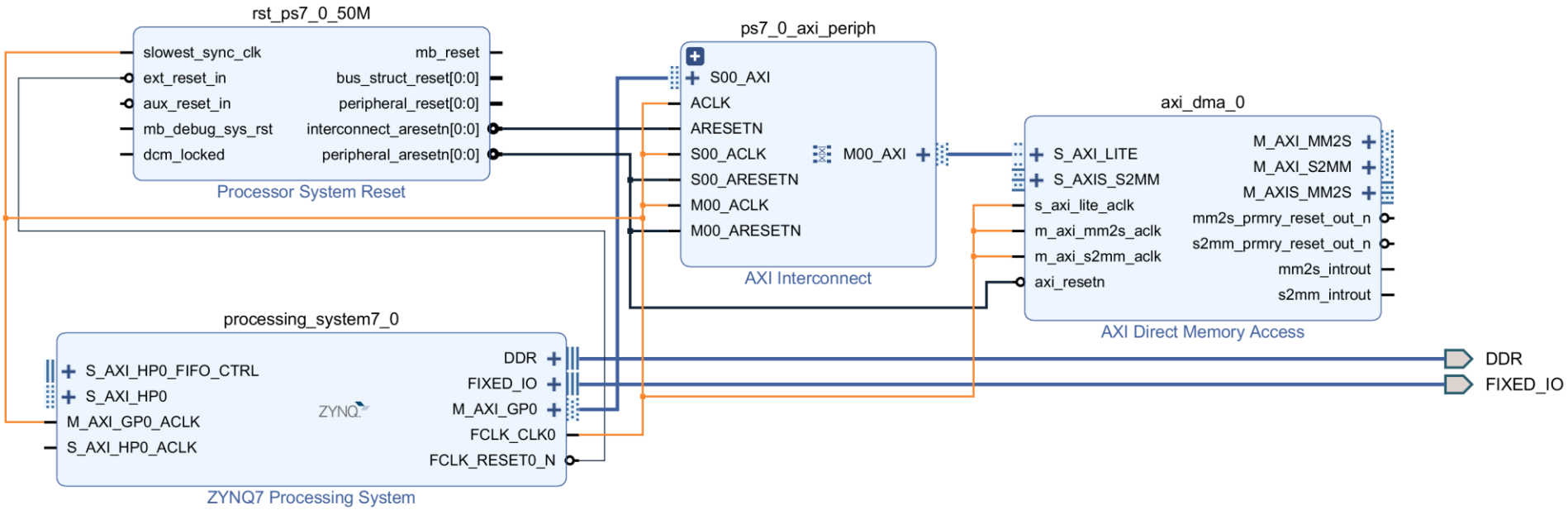
Creating Block Designs

- Run Connection Automation (Cont'd)
  - Notice that two additional blocks, ***Processor System Reset*** and ***AXI Interconnect***, have automatically been added.



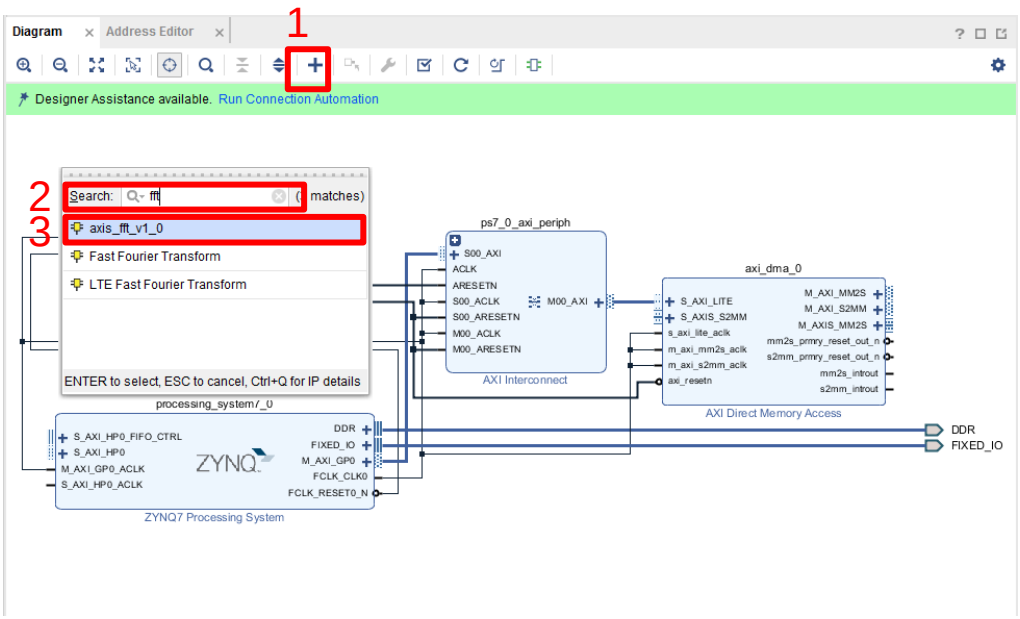
Creating Block Designs

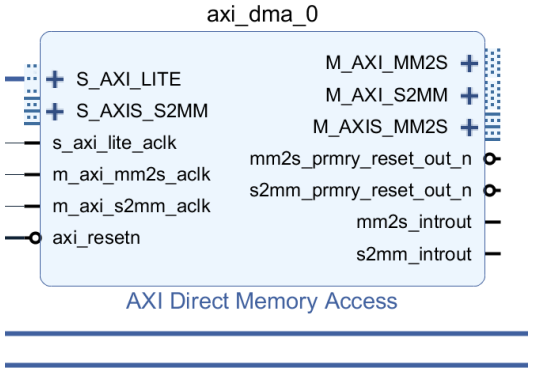
- Run Connection Automation (Cont'd)
  - Connect ***'FCLK_CLK0'*** with ***'m_axi_mm2s_aclk'*** and ***'m_axi_s2mm_aclk'***

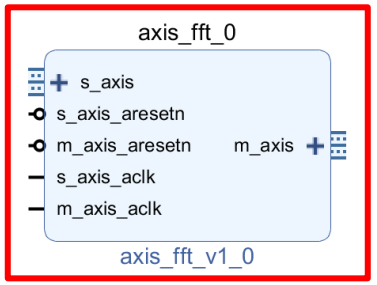


Creating Block Designs

- Add AXIS FFT IP
  - Click the ***'Add IP'*** icon and type ***"fft"*** in the Search field
  - Double-click ***'axis_fft_v1_0'***

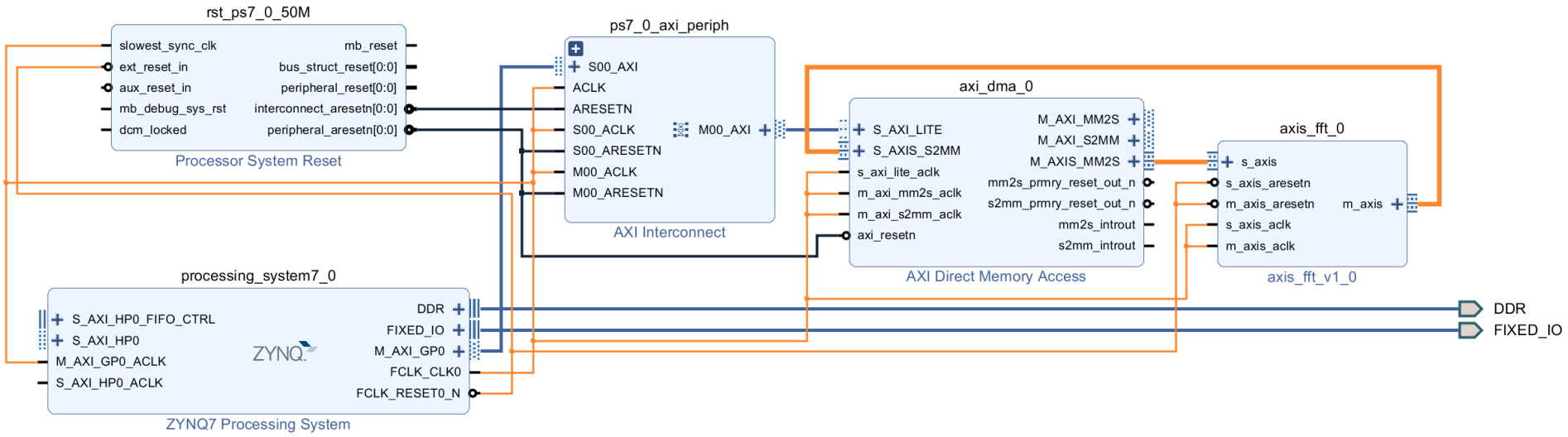






Creating Block Designs

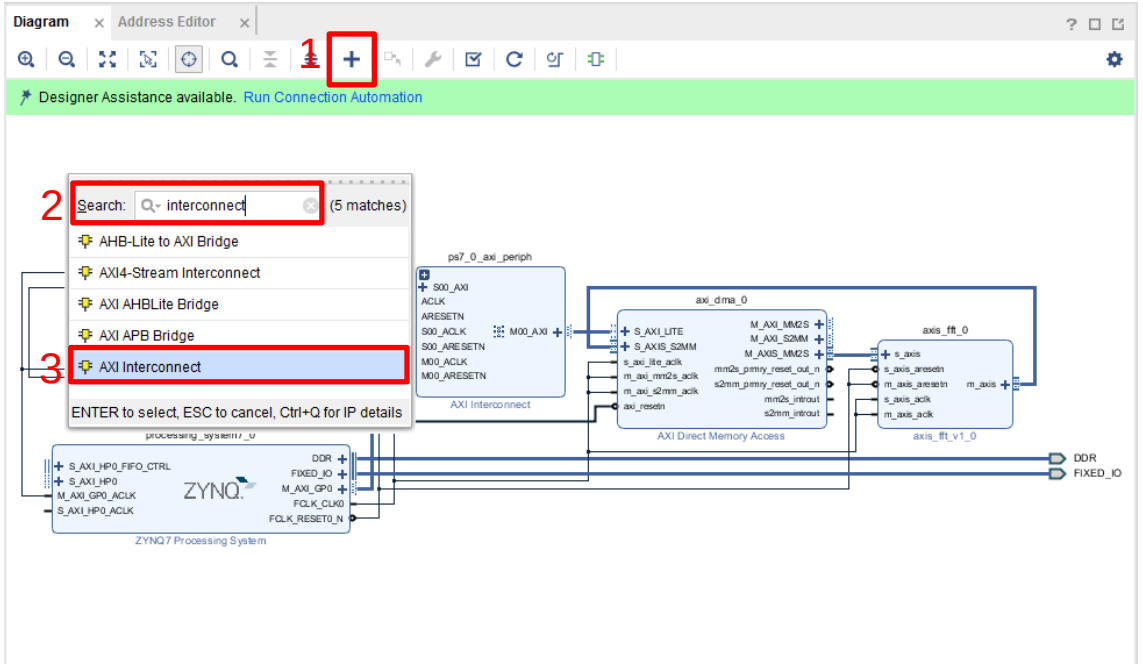
- Make connections
  - Connect ***'M_AXIS_MM2S'*** with ***'s_axis'***
  - Connect ***'S_AXIS_S2MM'*** with ***'m_axis'***
  - Connect ***'FCLK_CLK0'*** with ***'s_axis_aclk', 'm_axis_aclk'***
  - Connect ***'FCLK_RESET0_N'*** with ***'s_axi_aresetn'*** and ***'m_axi_aresetn'***

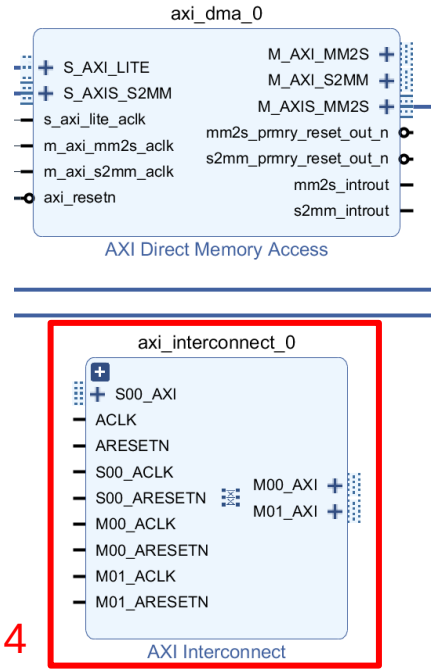


Creating Block Designs

❑ Add AXI Interconnect

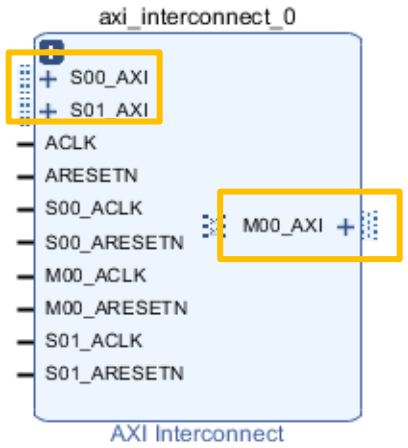
- Click the ***'Add IP'*** icon and type ***"interconnect"*** in the Search field
- Double-click ***'AXI Interconnect'***
- Double-click the '***AXI Interconnect'*** block

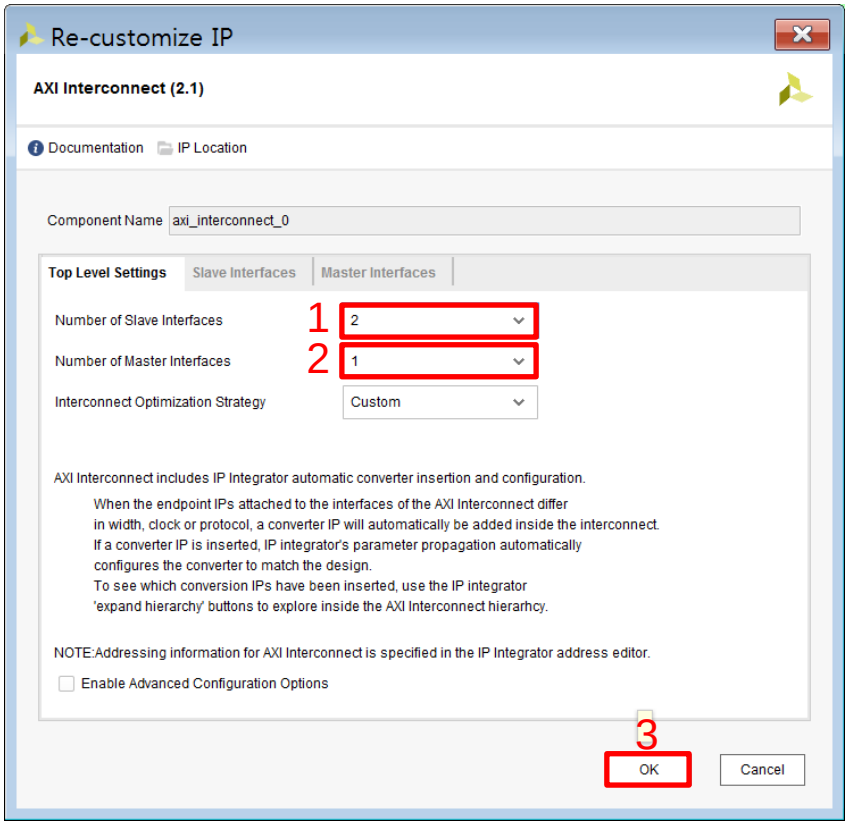




Creating Block Designs

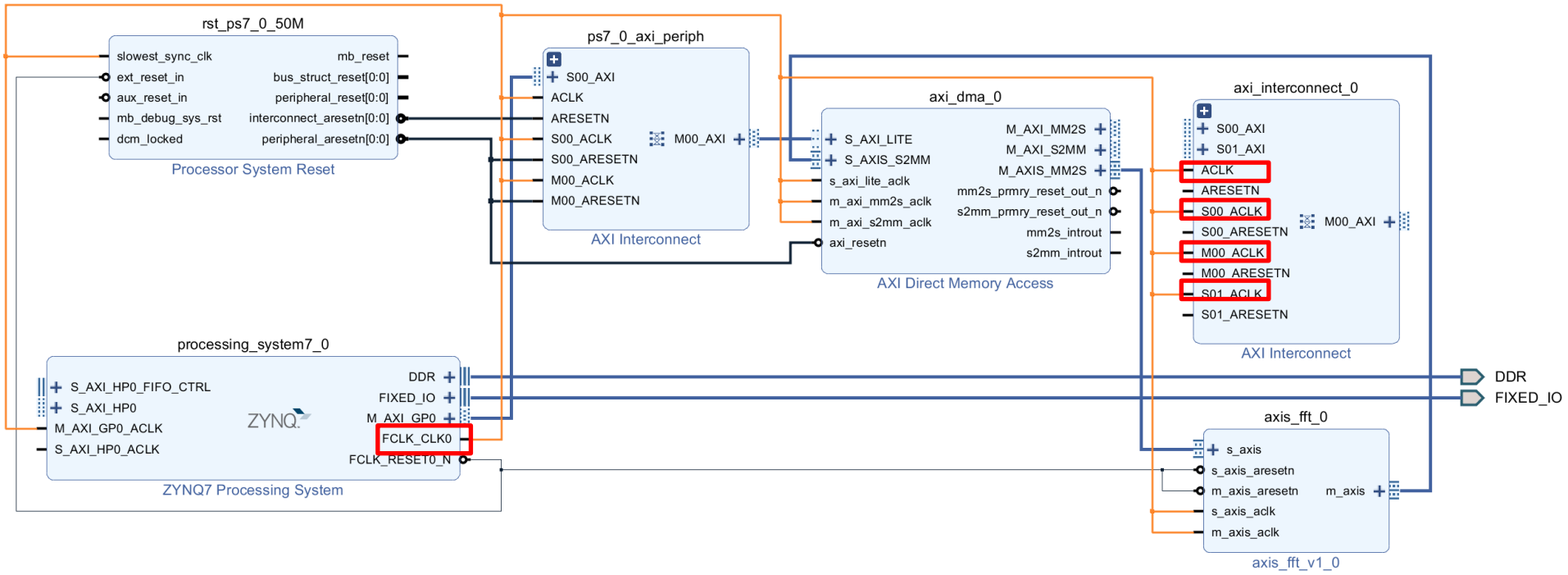
- Add AXI Interconnect (Cont'd)
  - Click ***'Number of Slave Interfaces'***, select ***'2'***
  - Click ***'Number of Master Interfaces'***, select ***'1'***
  - Click ***'OK'***





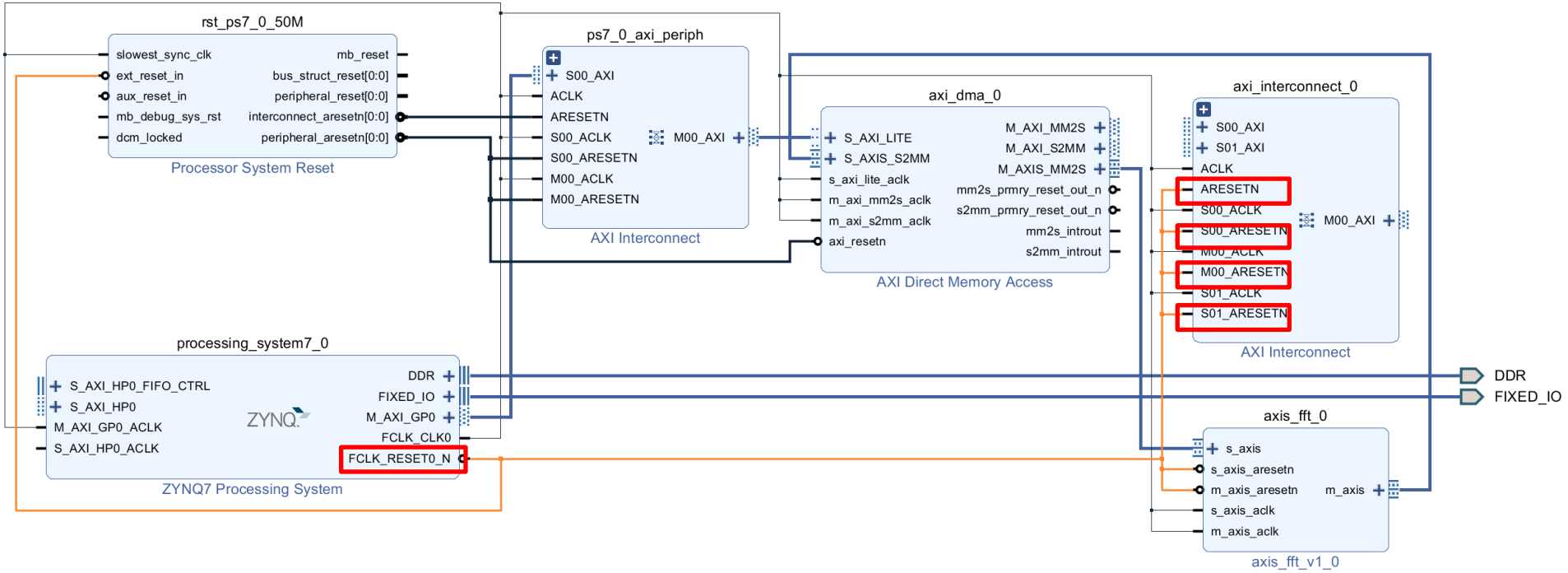
Creating Block Designs

- Make connections
  - Connect ***'FCLK_CLK0'*** with ***'ACLK', 'S00_ACLK', 'M00_ACLK',*** and ***'S01_ACLK'***



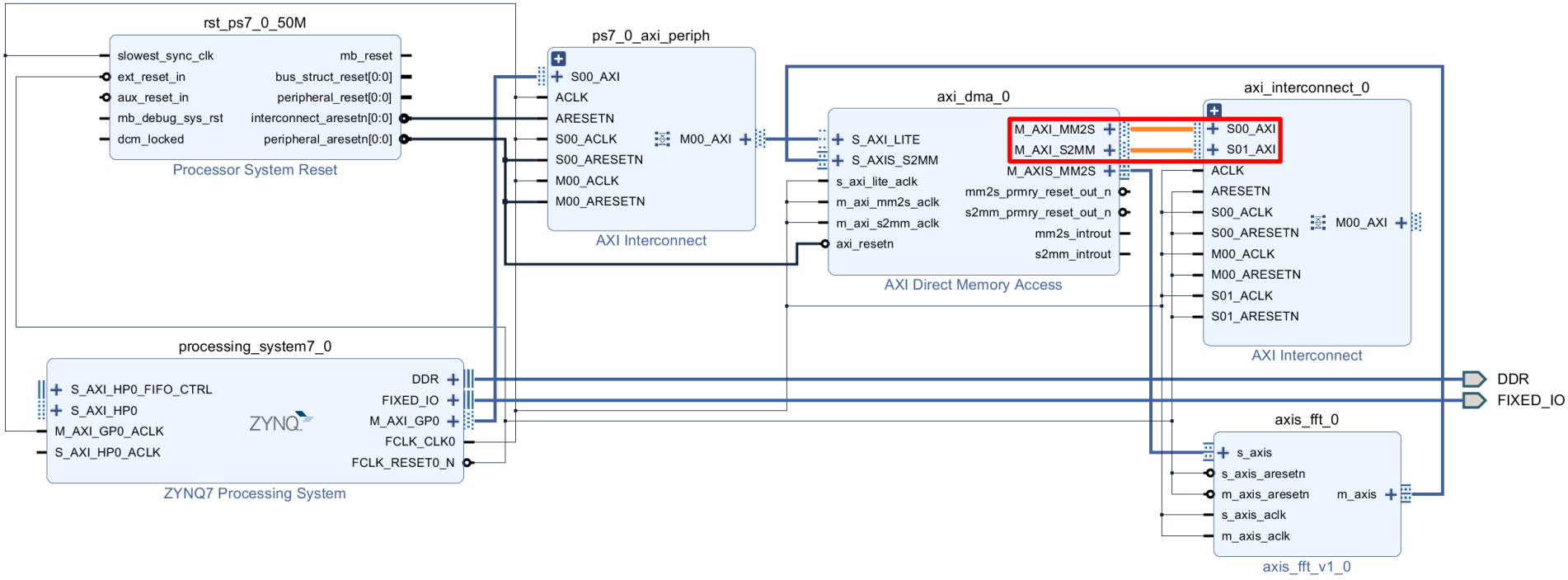
Creating Block Designs

- Make connections (Cont'd)
  - Connect ***'FCLK_RESET0_N'*** with ***'ARESETN'***, ***'S00_ARESETN'***, ***'M00_ARESETN'***, and ***'S01_ARESETN'***



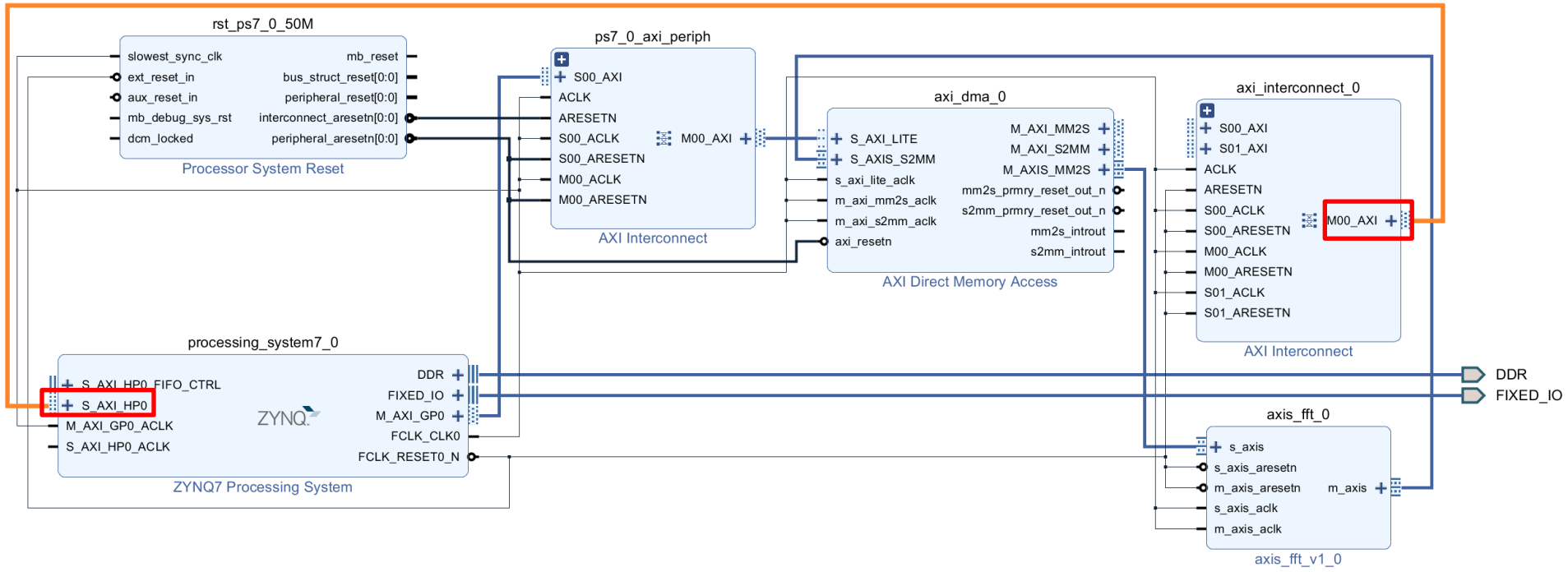
Creating Block Designs

- Make connections (Cont'd)
  - Connect ***'M_AXI_MM2S'*** and ***'M_AXI_S2MM'*** with ***'S00_AXI'*** and ***'S01_AXI'***, respectively.



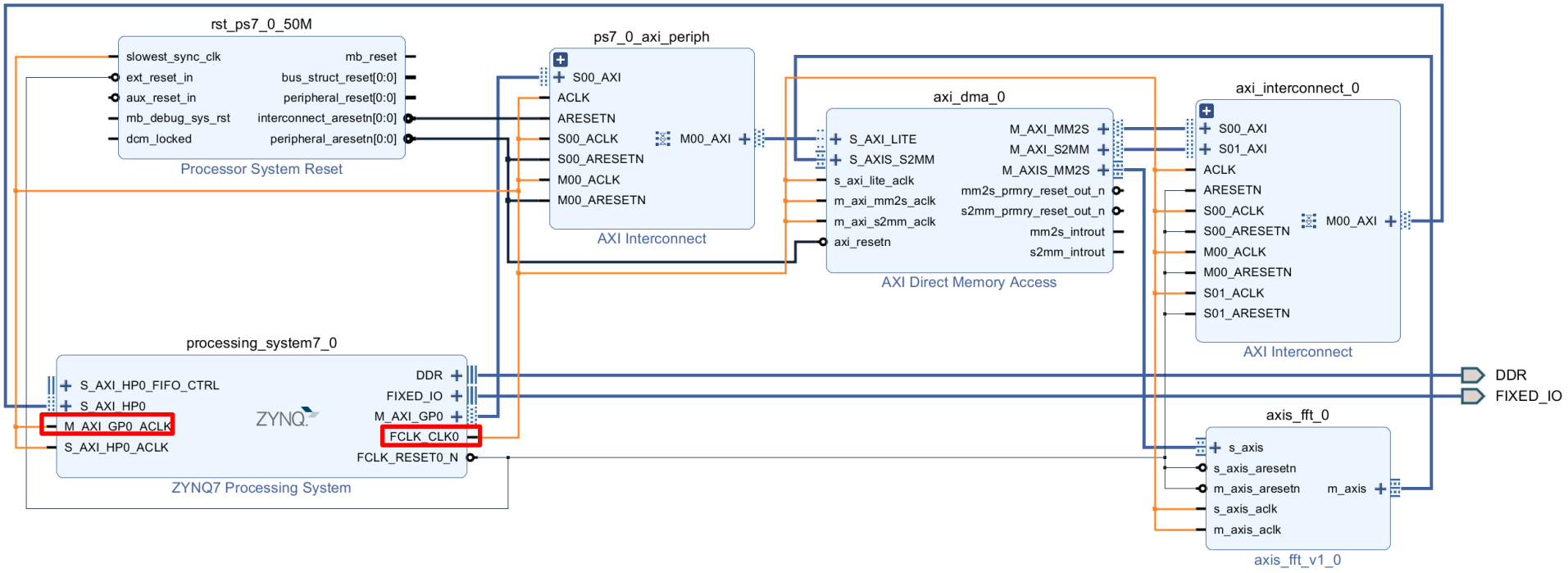
Creating Block Designs

- Make connections (Cont'd)
  - Connect ***'M00_AXI'*** with ***'S_AXI_HP0'***



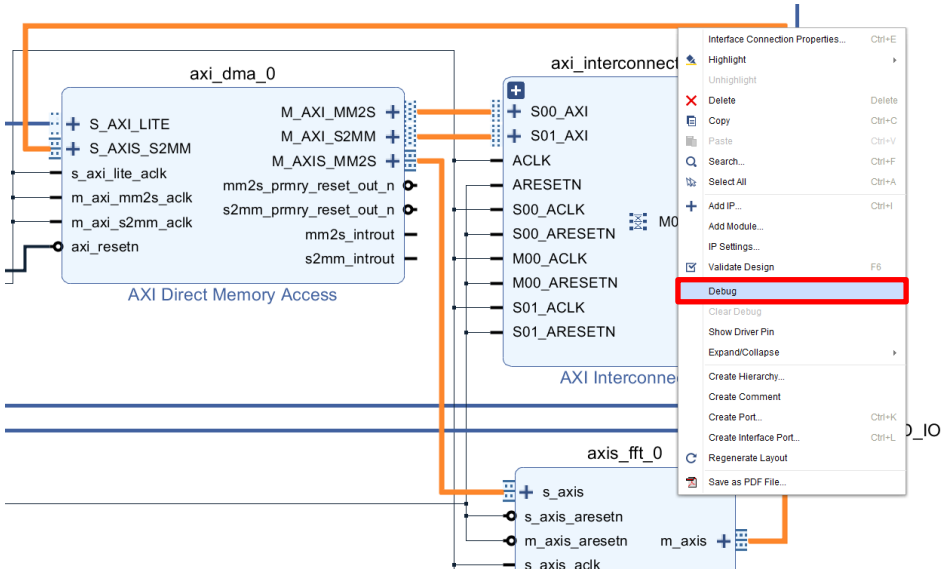
Creating Block Designs

- Connect AXI HP port (Cont'd)
  - Connect ***'FCLK_CLK0'*** with ***'S_AXI_HP0_ACLK'***



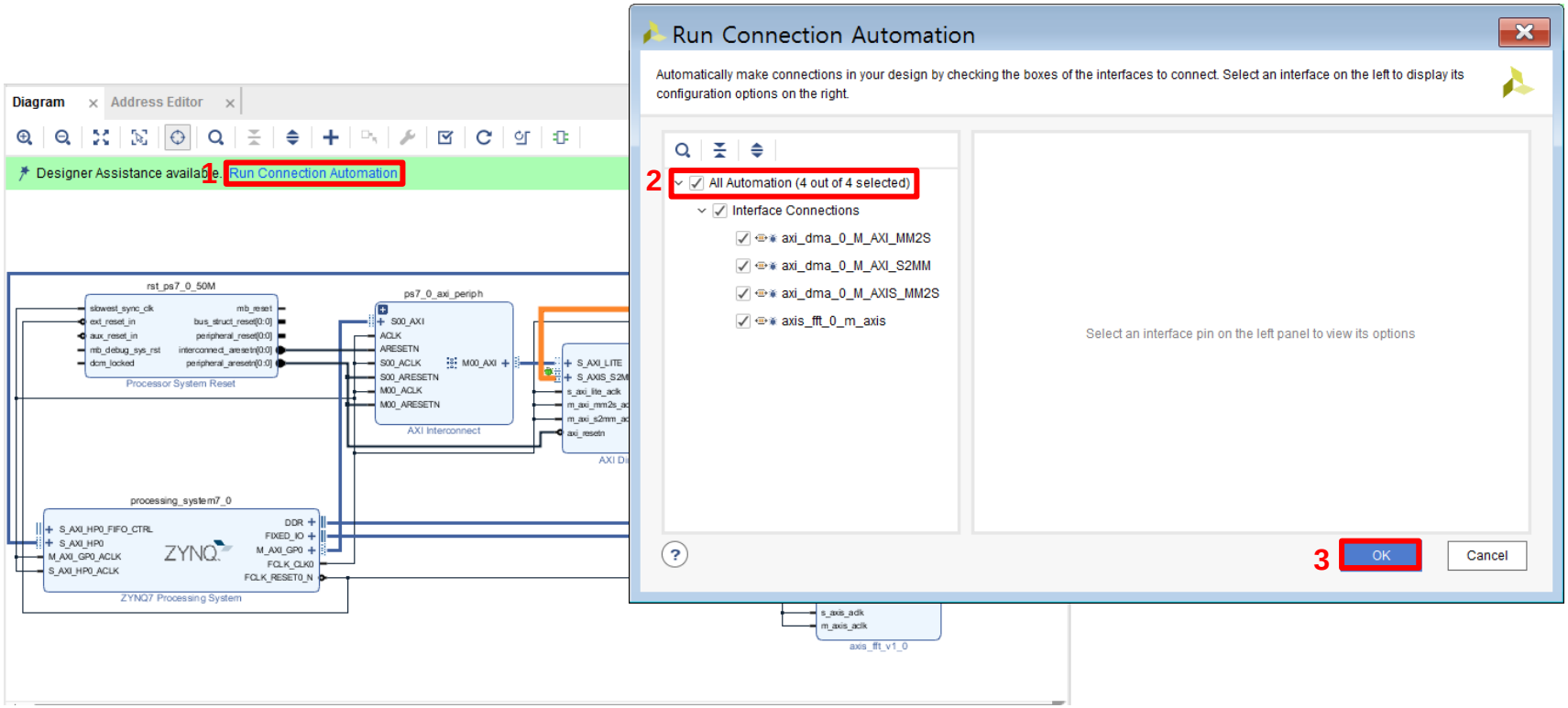
Creating Block Designs

- Mark Debug
  - Click the connection between ***'M_AXI_MM2S'*** and ***'S00_AXI'***
  - Click the connection between ***'M_AXI_S2MM'*** and ***'S01_AXI'***
  - Click the connection between ***'M_AXIS_S2MM'*** and ***'s_axis'***
  - Click the connection between ***'S_AXIS_MM2S'*** and ***'m_axis'***
  - Right-click and choose ***'Debug'***



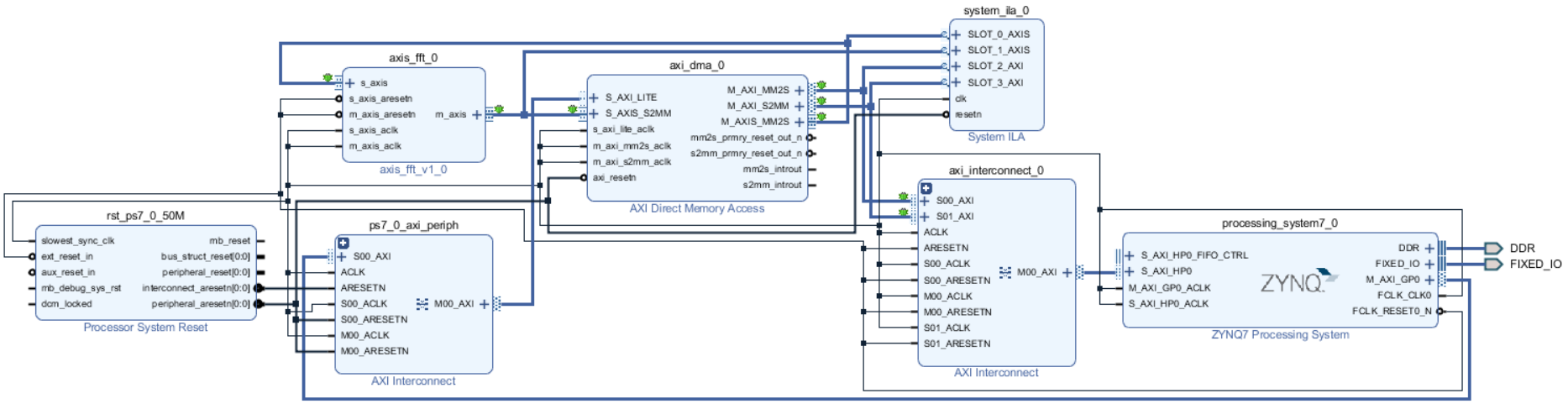
Creating Block Designs

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



Creating Block Designs

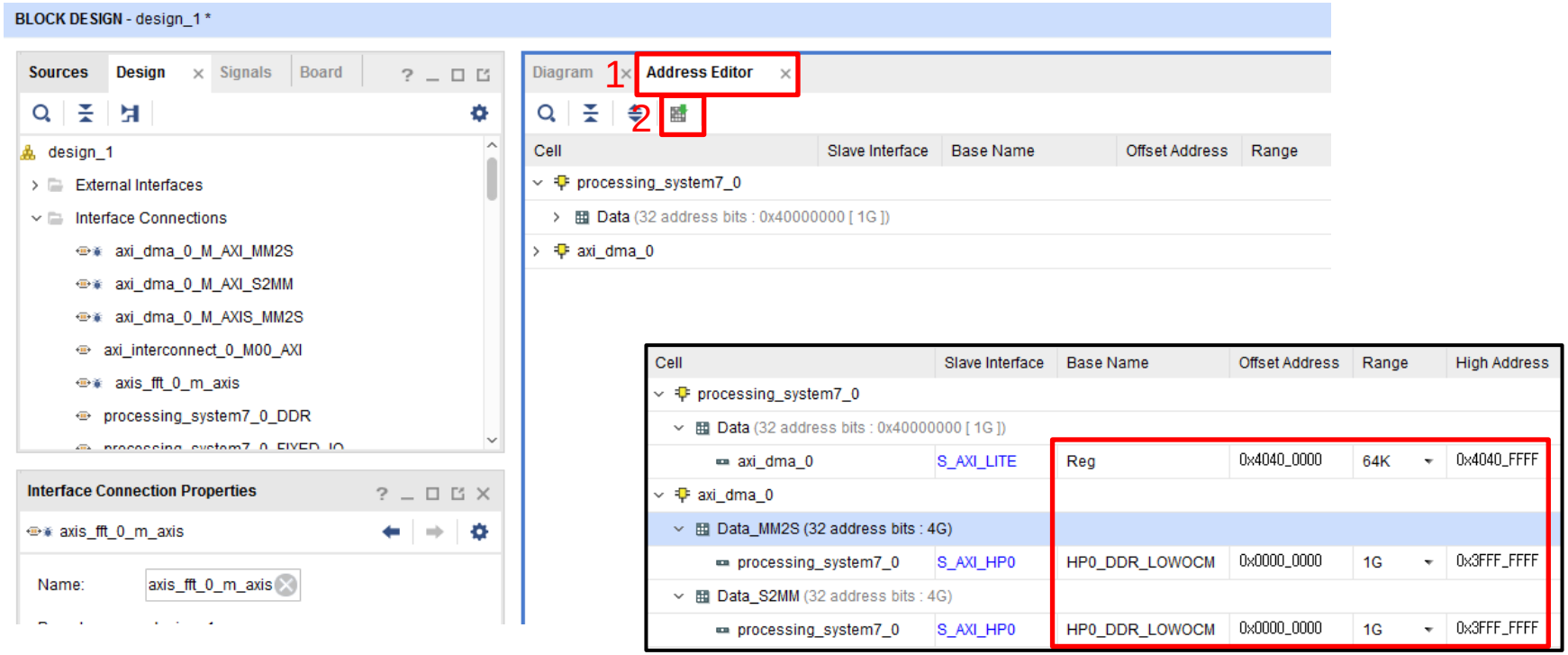
- Mark Debug (Cont'd)
  - The complete block diagram is given below.



Creating Block Designs

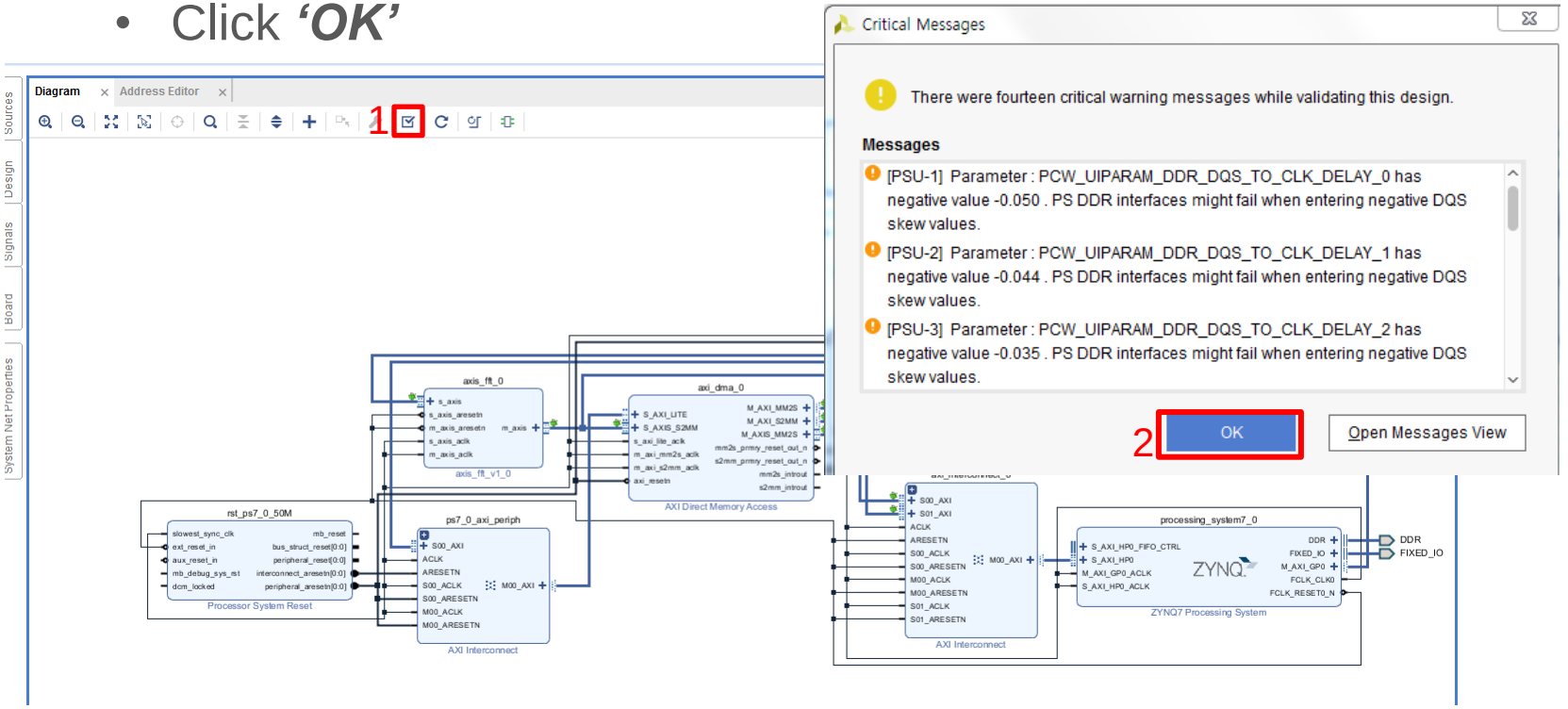
❑ Assign Address

- Choose the ***'Address Editor'*** tap and then click the ***'Auto Assign Address'*** icon
- The addresses will be automatically assigned.



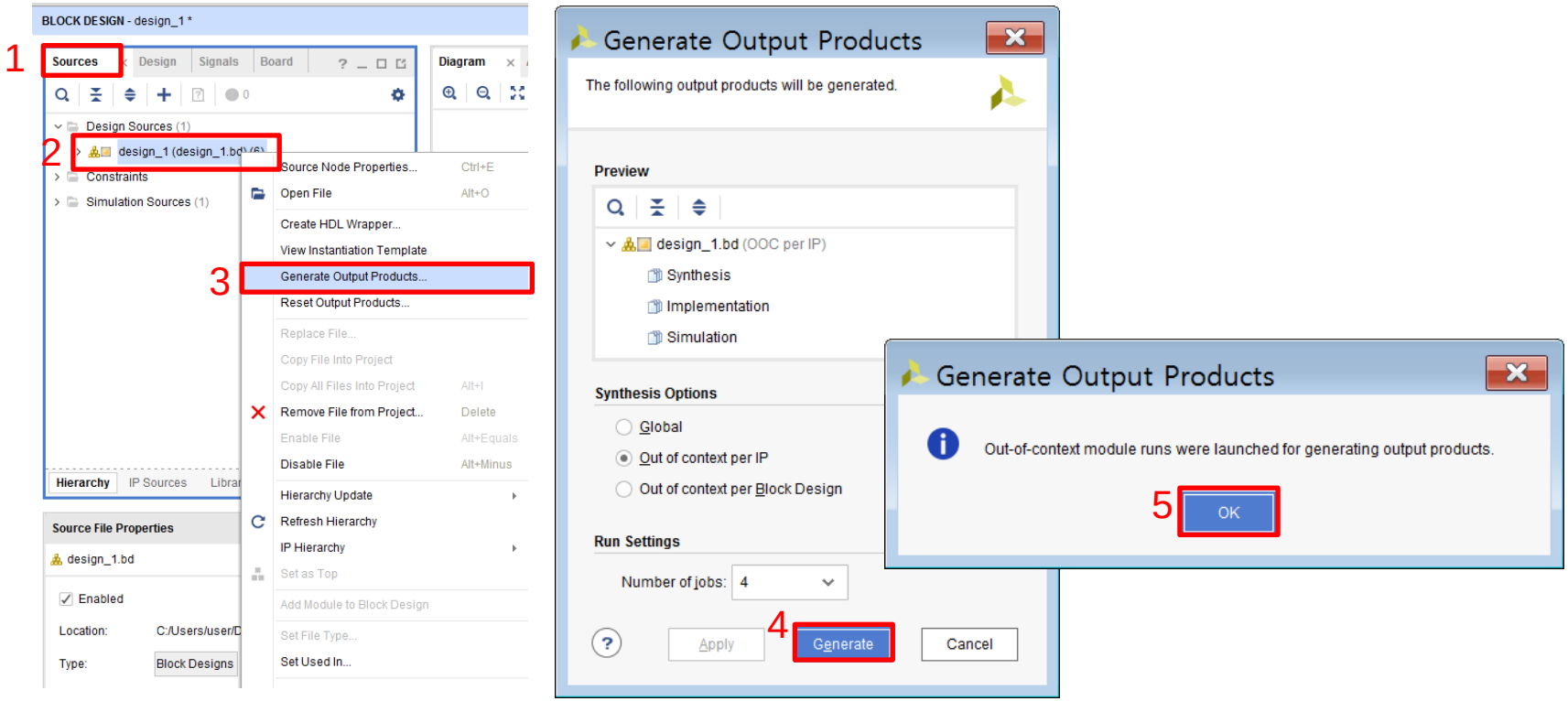
Creating Block Designs

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



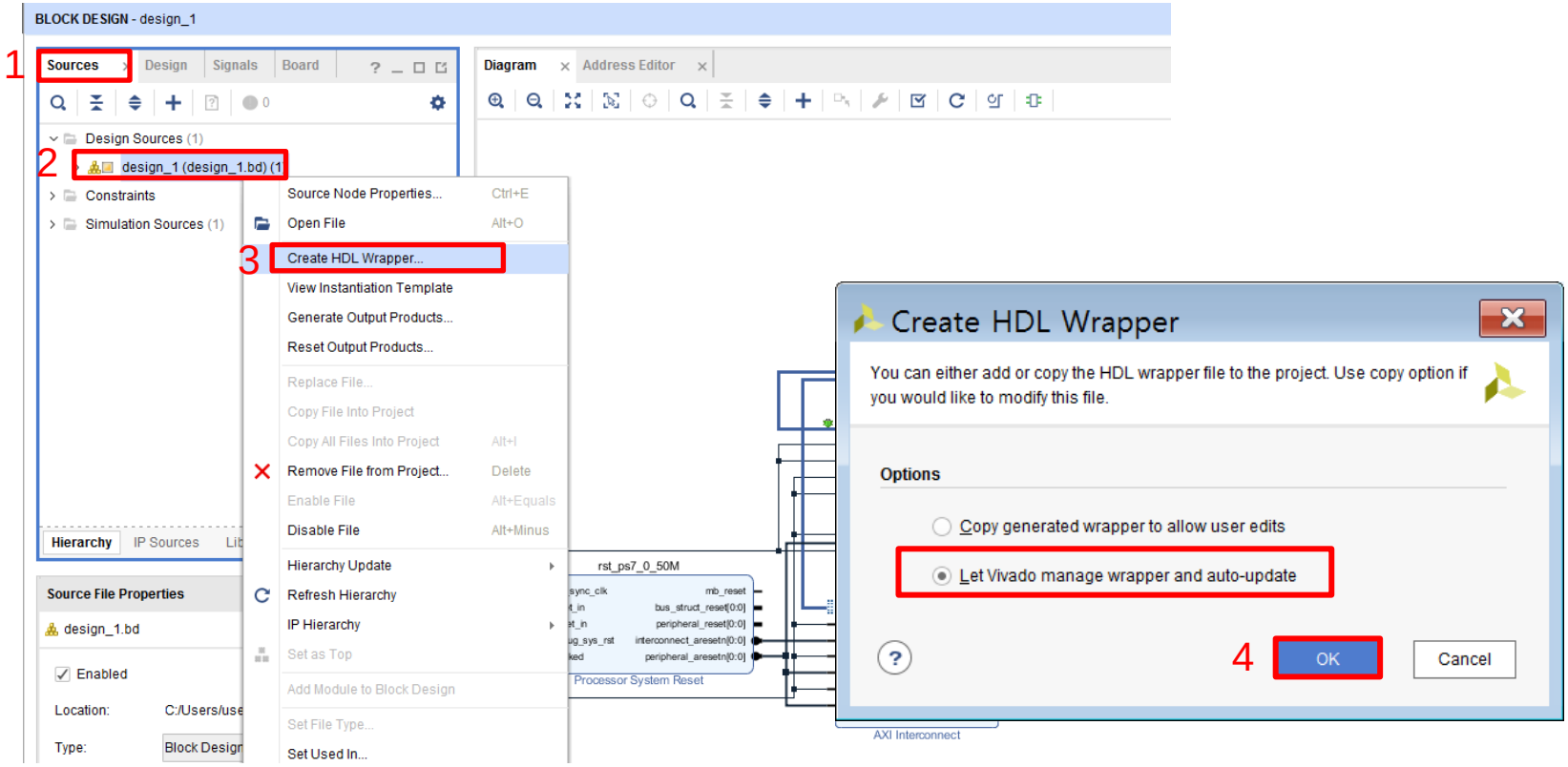
Creating Block Designs

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



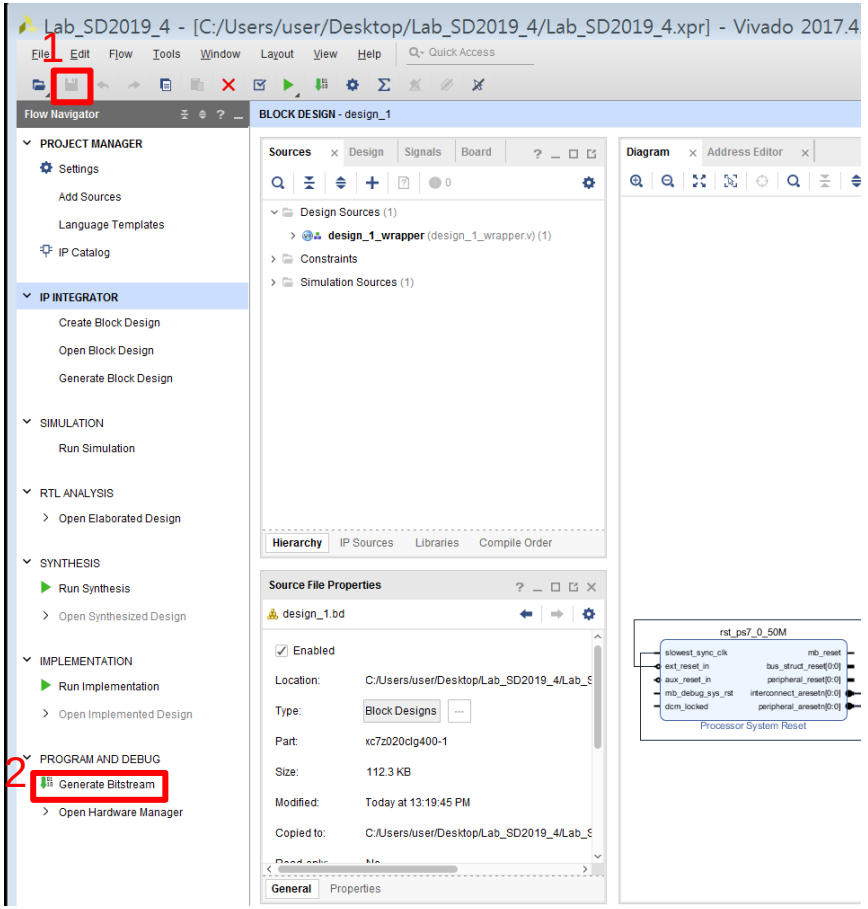
Creating Block Designs

- Create HDL Wrapper
  - Select the ***'Sources'*** tap and then right-click ***'design_1'***
  - Click ***'Create HDL Wrapper > OK'***



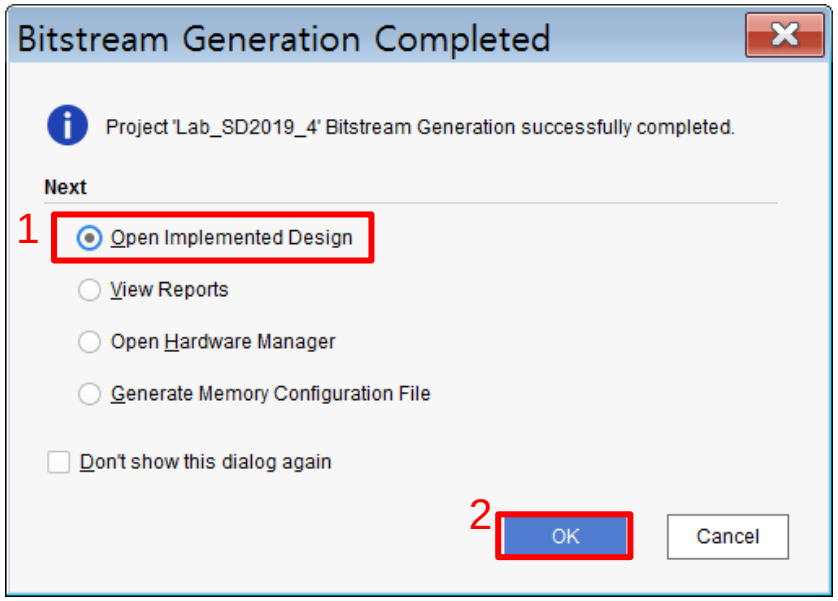
Generating Bitstream

- Generate Bitstream
  - Click the ***'Save Block Design'*** icon and then click ***'Generate Bitstream'***



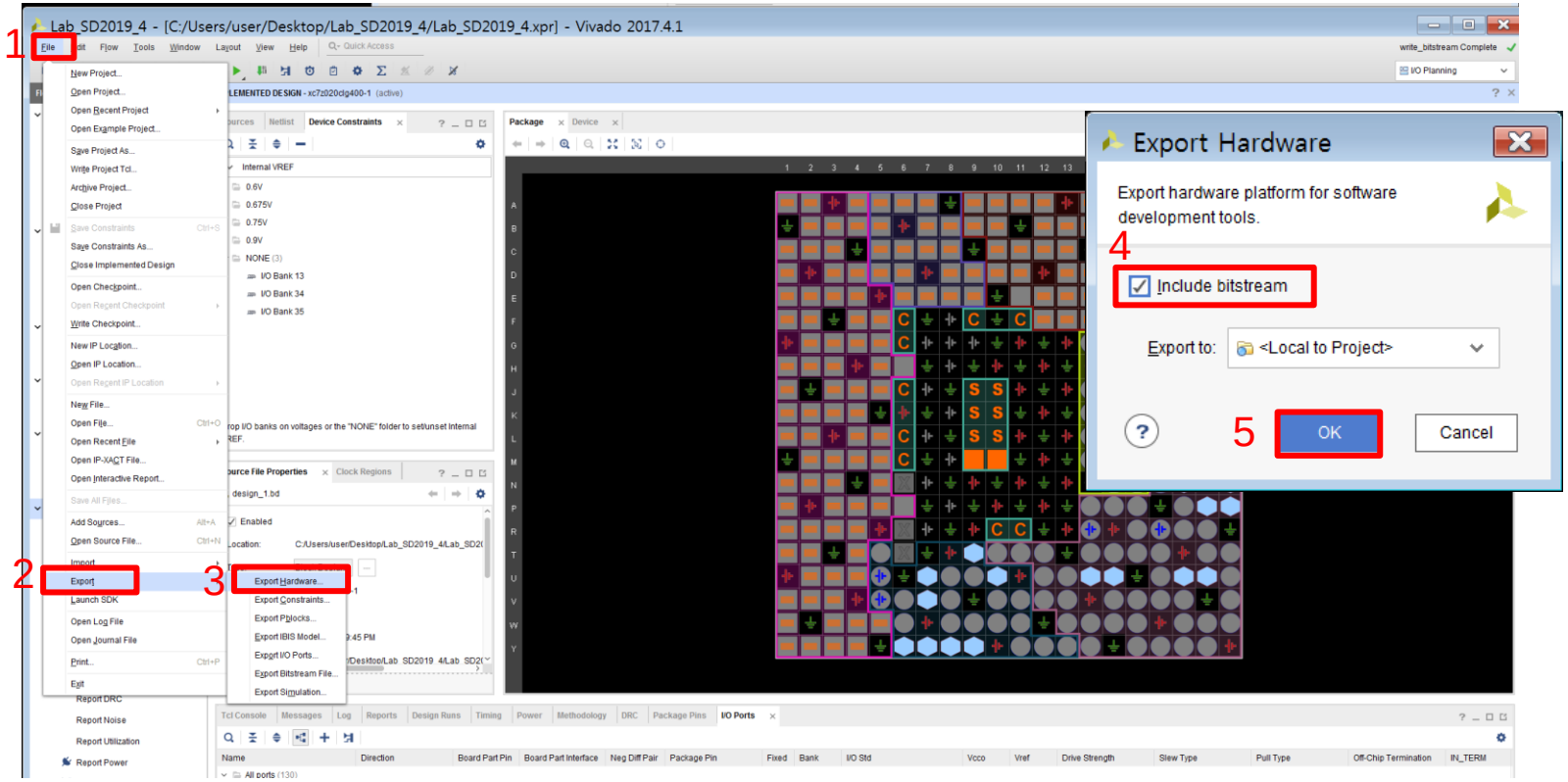
Generating Bitstream

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



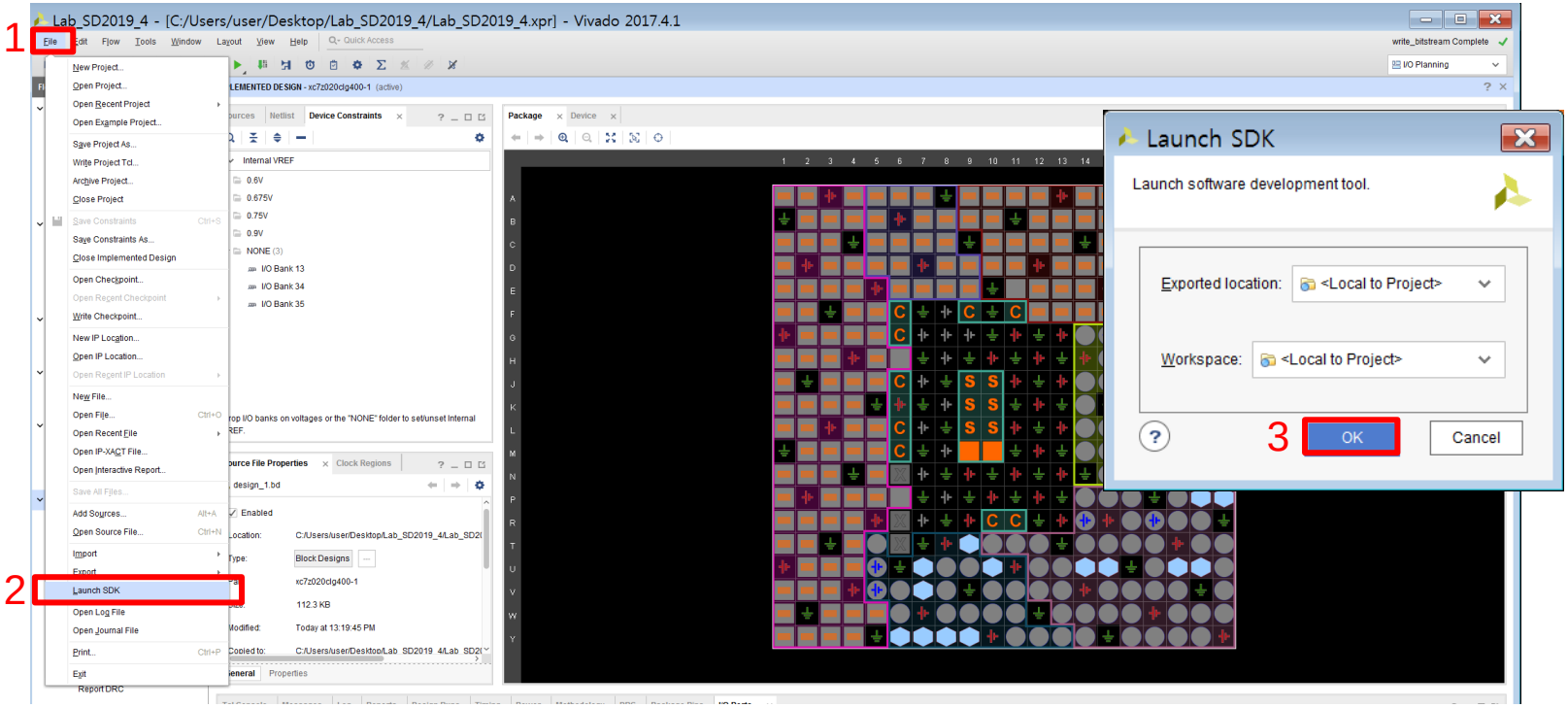
Generating Bitstream

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



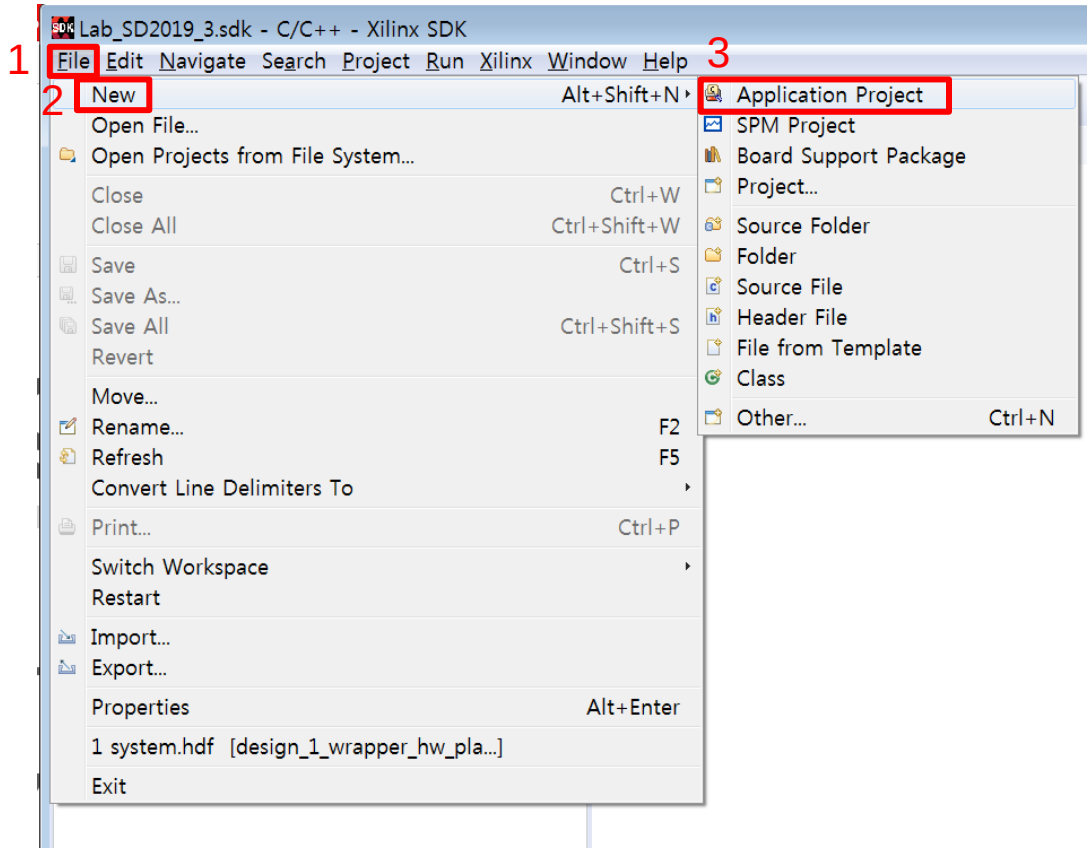
Generating Bitstream

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



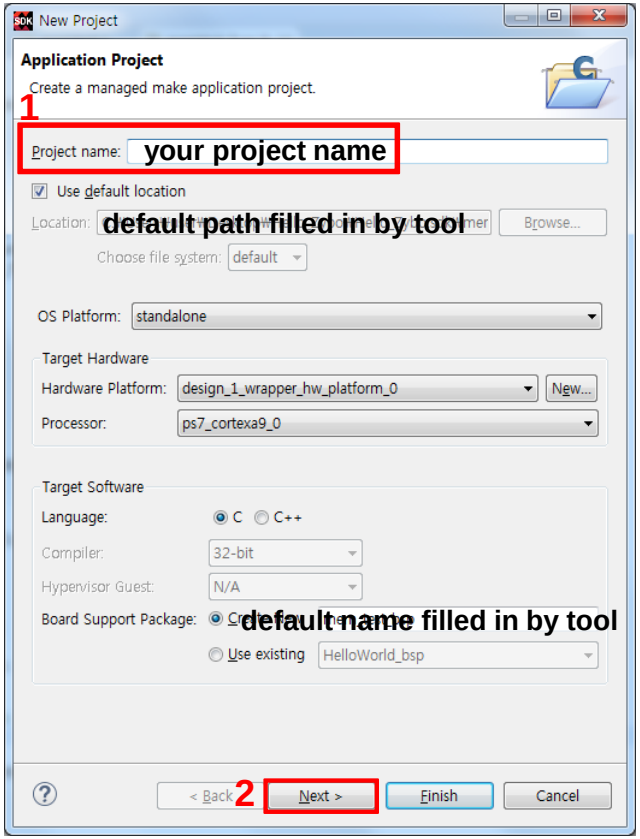
Running C Applications

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



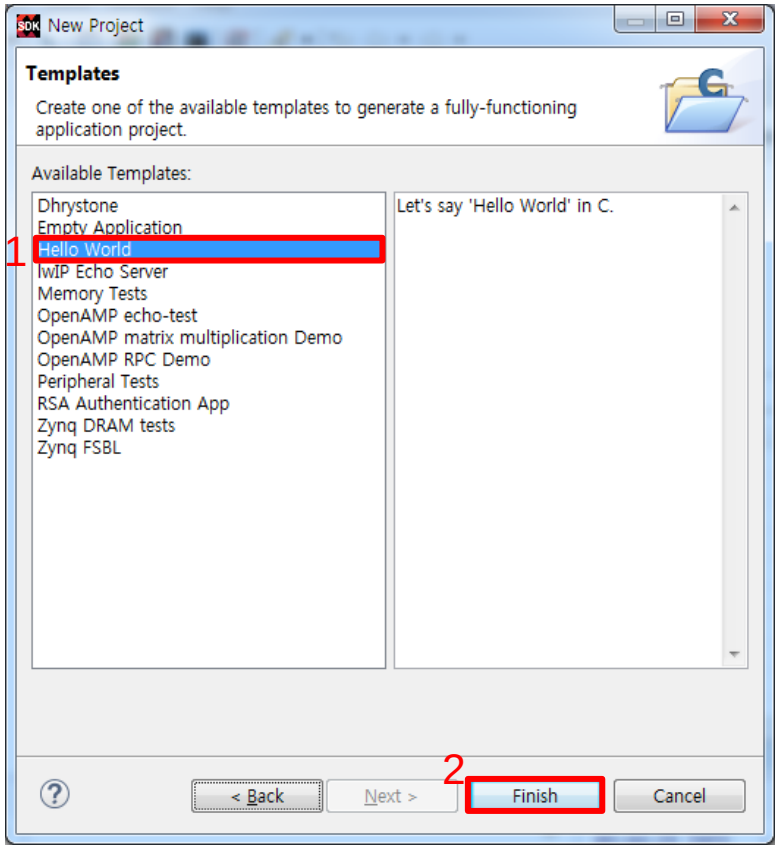
Running C Applications

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



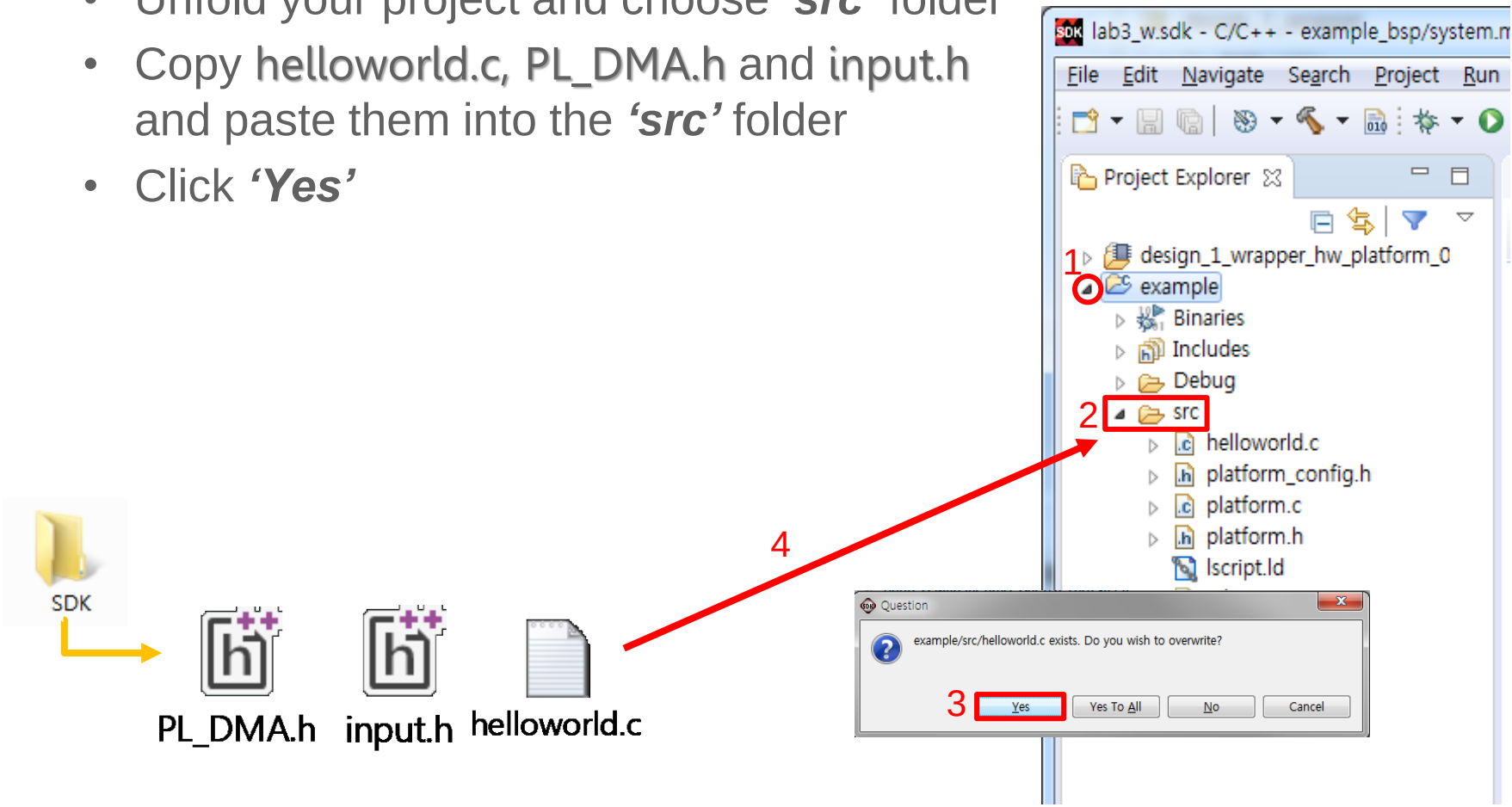
Running C Applications

- 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, PL_DMA.h and input.h and paste them into the ***'src'*** folder
  - Click ***'Yes'***



Running C Applications

- Review the function main()
  - ① Initialize input/output base address
  - ② Assign input data into an array
  - ③ Set DMA control registers
  - ④ Assign output data into an array
  - ⑤ Convert output data to binary

①
```
// Initialize
for (i = 0; i < LENGTH; i++){
    Xil_Out32(OUTPUT_BASE+i*4, 0);
    Xil_Out32(INPUT_BASE+i*4, 0);
}
```

②
```
// Set input data to array
for(i = 0; i < LENGTH; i++){
    tmp  = inReal[i]<<16;
    tmp1 = (0x0000FFFF & inImag[i]);
    input_array[i] = tmp + tmp1;
}

// Put input data to DMA source
Xil_Out32(INPUT_BASE,0x7FFFFFFF);
for(i = 0; i < LENGTH; i++)
    Xil_Out32(INPUT_BASE + 4*(i+1),input_array[i]);
```

③
```
// DMA basic setting
DMA_preset();

// DMA transfer control
DMA_transfer(INPUT_BASE, OUTPUT_BASE, LENGTH);
```

④
```
// Get output data from DMA destination
for(i = 0; i < LENGTH; i++)
    output_array[i] = Xil_In32(OUTPUT_BASE + 4*i);
```

⑤
```
// Console output
for(i = 0; i < LENGTH; 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 < LENGTH; 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");
}
```

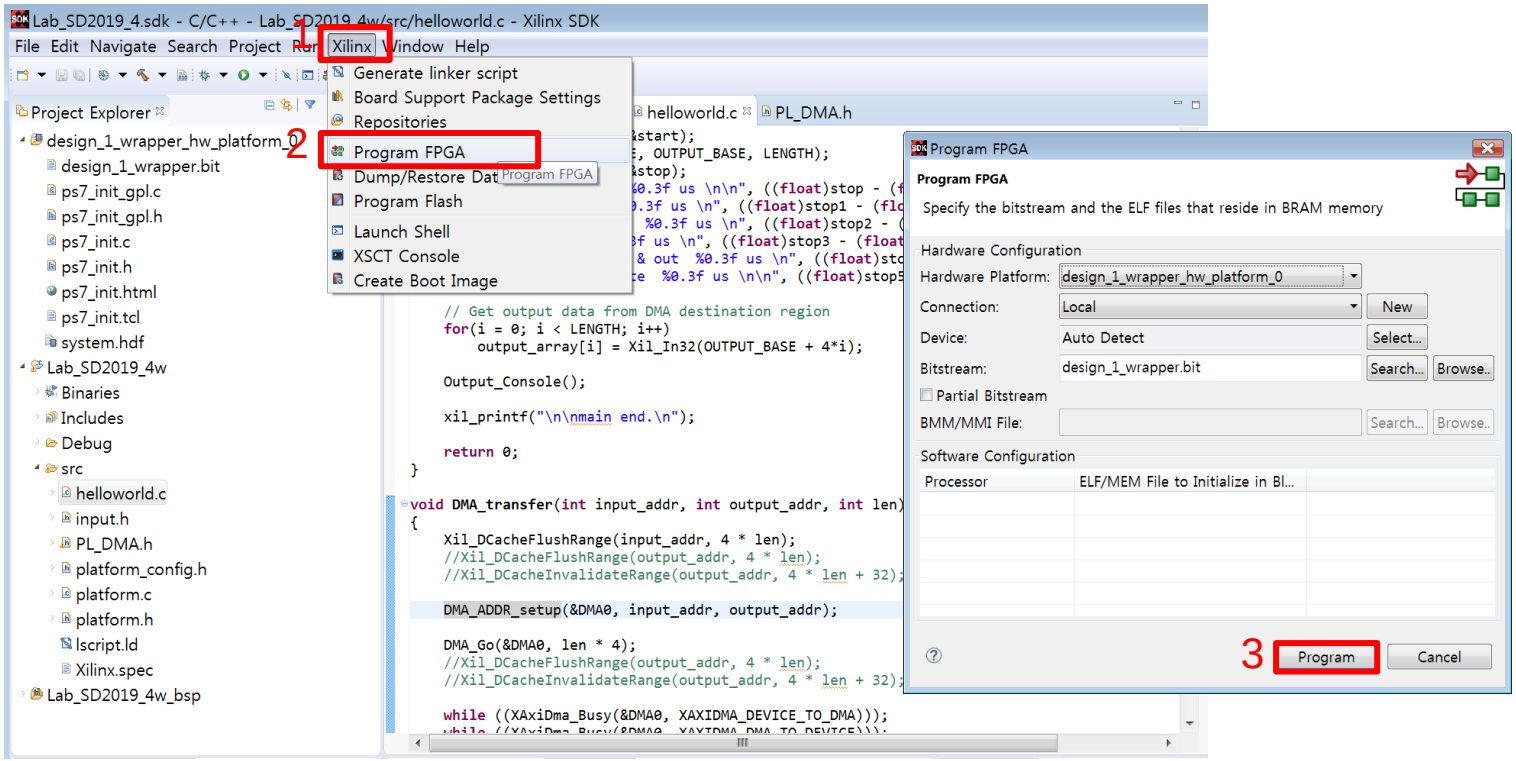
Running C Applications

- Review the function DMA_transfer()
  - ① Flush the source region of the D-cache
  - ② Set the (start) addresses of the source & destination regions
  - ③ Set the burst length (i.e., the number of beats per burst)
  - ④ Check the DMA status register to see if the transfer ends

```
void DMA_transfer(int input_addr, int output_addr, int len)
{
    Xil_DCacheFlushRange(input_addr, 4 * len);                          ①

    DMA_ADDR_setup(&DMA0, input_addr, output_addr);                     ②

    DMA_Go(&DMA0, len * 4);                                             ③

    XTime_GetTime((XTime*)&start);
    while ((XAxiDma_Busy(&DMA0, XAXIDMA_DEVICE_TO_DMA)));               ④
    XTime_GetTime((XTime*)&stop);
     printf("Accelerator in & out  %0.3f us \n", ((float)stop - (float)start)/COUNTS_PER_SECOND*100000);
}
```

Running C Applications

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

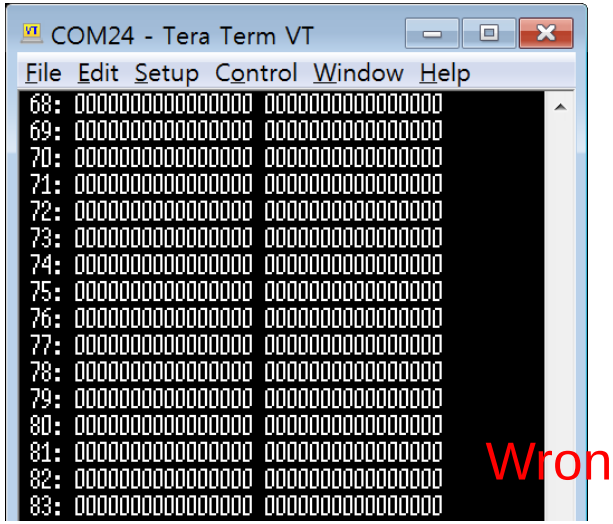


Running C Applications

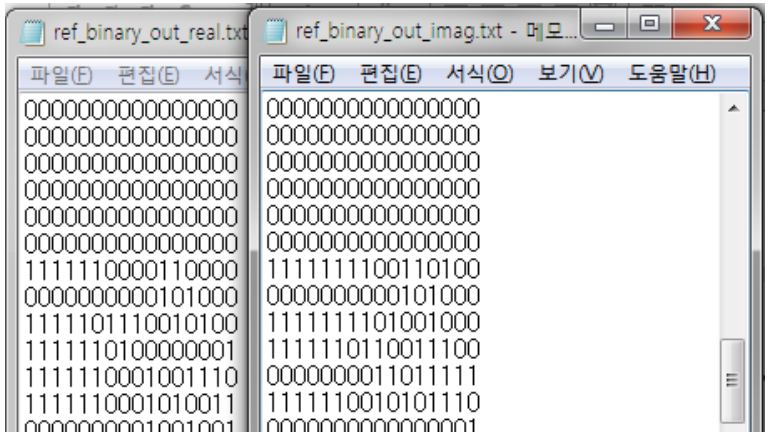
- Repeat the previous steps
  - Follow pp. 31~34 of the following lab workbook: **Lab_EC_0w.pdf**

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.



Wrong!



Running C Applications

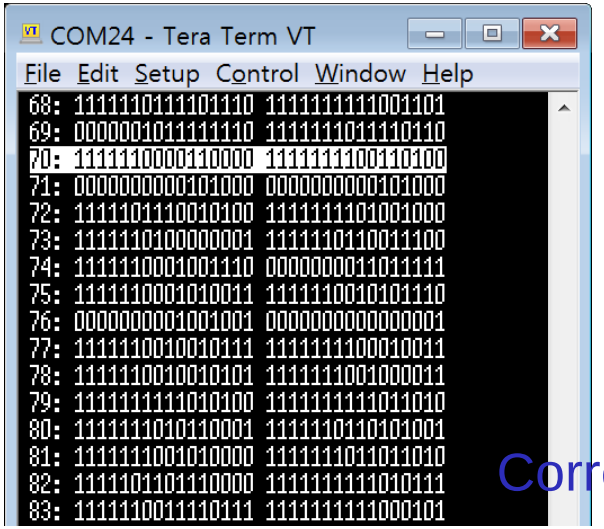
- Modify the function main()
  - Check which memory (i.e., either cache or main memory) holds the accelerator output
  - Figure out which of the following C statements should be **uncommented** to correct the console output

*helloworld.c | PL_DMA.h | xil_cache.c

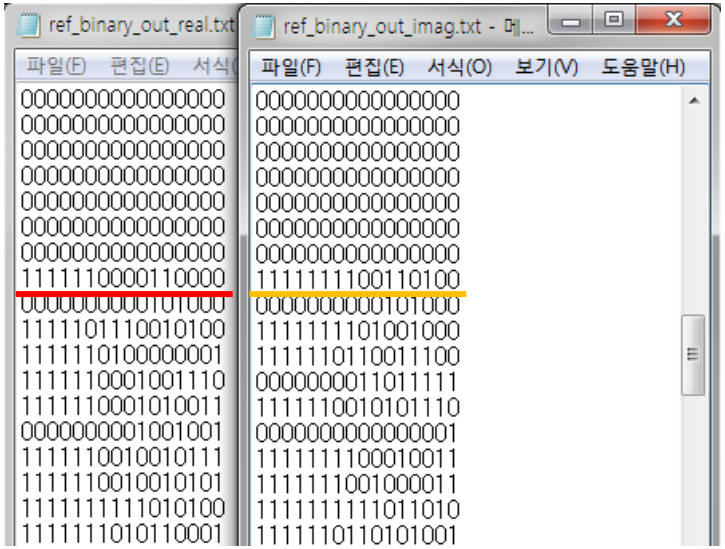
```
void DMA_transfer(int input_addr, int output_addr, int len)
{
    Xil_DCacheFlushRange(input_addr, 4 * len);
    //Xil_DCacheFlushRange(output_addr, 4 * len);          (1)
    //Xil_DCacheInvalidateRange(output_addr, 4 * len + 32);  (2)

    DMA_ADDR_setup(&DMA0, input_addr, output_addr);

    DMA_Go(&DMA0, len * 4);
    //Xil_DCacheFlushRange(output_addr, 4 * len);          (3)
    //Xil_DCacheInvalidateRange(output_addr, 4 * len + 32);  (4)

    while ((XAxiDma_Busy(&DMA0, XAXIDMA_DEVICE_TO_DMA)));
    while ((XAxiDma_Busy(&DMA0, XAXIDMA_DMA_TO_DEVICE)));
    //Xil_DCacheFlushRange(output_addr, 4 * len);          (5)
    //Xil_DCacheInvalidateRange(output_addr, 4 * len + 32);  (6)
}
```

Running C Applications

- Modify the function main()
  - The output on ***'Tera Term'*** should match the reference output as follows.

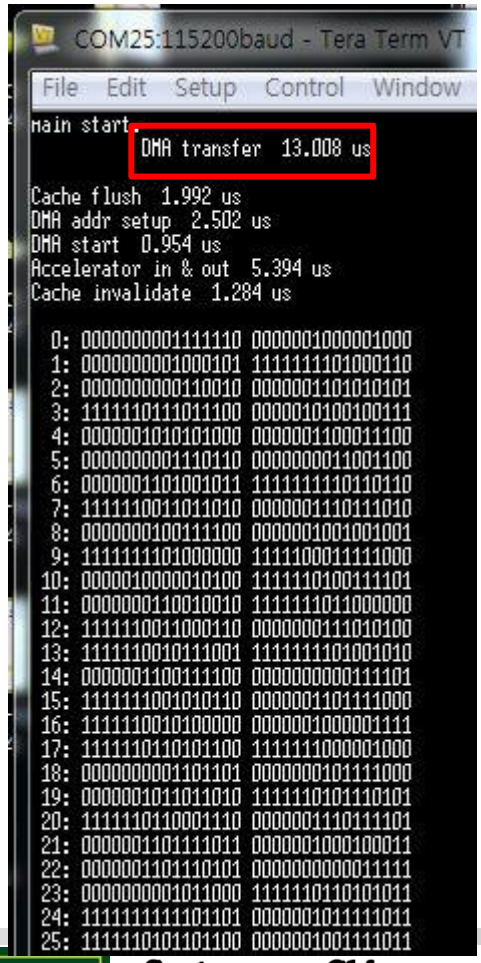


Correct!



Running C Applications

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



helloworld.c | PL_DMA.h

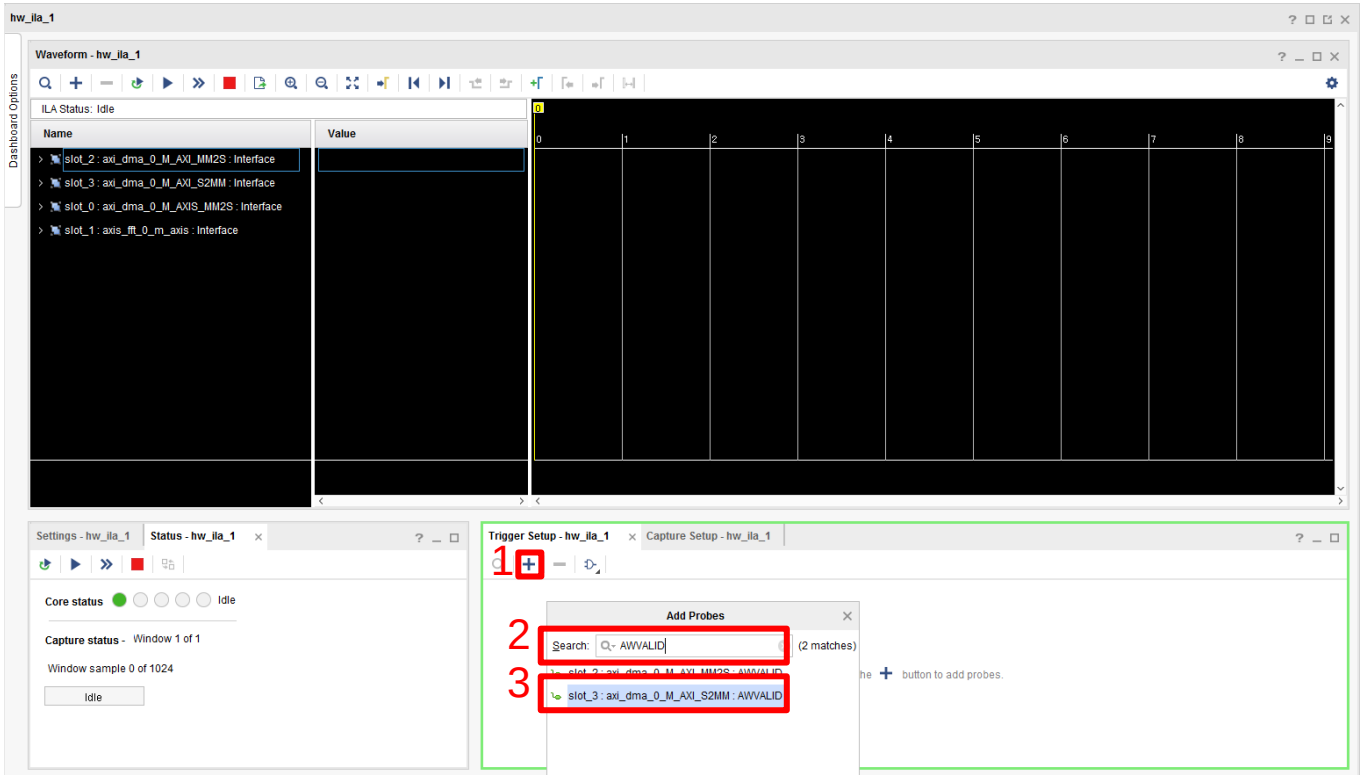
```
    DMA_preset();

    // DMA transfer control
    XTime_GetTime((XTime*)&start);
    DMA_transfer(INPUT_BASE, OUTPUT_BASE, LENGTH);
    XTime_GetTime((XTime*)&stop);
    printf("DMA transfer  %0.3f us \n\n", ((float)stop - (float)start)/COUNTS_PER_SECOND*1000000);
    printf("Cache flush  %0.3f us \n", ((float)stop1 - (float)start1)/COUNTS_PER_SECOND*1000000);
    printf("DMA addr setup  %0.3f us \n", ((float)stop2 - (float)start2)/COUNTS_PER_SECOND*1000000);
    printf("DMA start  %0.3f us \n", ((float)stop3 - (float)start3)/COUNTS_PER_SECOND*1000000);
    printf("Accelerator in & out  %0.3f us \n", ((float)stop4 - (float)start4)/COUNTS_PER_SECOND*1000000);
    printf("Cache invalidate  %0.3f us \n\n", ((float)stop5 - (float)start5)/COUNTS_PER_SECOND*1000000);

    // Get output data from DMA destination region
    for(i = 0; i < LENGTH; i++)
        output_array[i] = Xil_In32(OUTPUT_BASE + 4*i);

    Output_Console();

    xil_printf("\n\nmain end.\n");

    return 0;
}

void DMA_transfer(int input_addr, int output_addr, int len)
{
    XTime_GetTime((XTime*)&start1);
    Xil_DCacheFlushRange(input_addr, 4 * len);
    XTime_GetTime((XTime*)&stop1);

    XTime_GetTime((XTime*)&start2);
    DMA_ADDR_setup(&DMA0, input_addr, output_addr);
    XTime_GetTime((XTime*)&stop2);

    XTime_GetTime((XTime*)&start3);
    DMA_Go(&DMA0, len * 4);
    XTime_GetTime((XTime*)&stop3);

    XTime_GetTime((XTime*)&start4);
    while ((XAxiDma_Busy(&DMA0, XAXIDMA_DEVICE_TO_DMA)));
    //while ((XAxiDma_Busy(&DMA0, XAXIDMA_DMA_TO_DEVICE)));
    XTime_GetTime((XTime*)&stop4);

    XTime_GetTime((XTime*)&start5);
    Xil_DCacheInvalidateRange(output_addr, 4 * len);
    XTime_GetTime((XTime*)&stop5);
}
```

Debugging Designs in ILA

- Repeat the previous steps
  - Follow pp. 58~61 of the following lab workbook: **Lab_EC_3w.pdf**

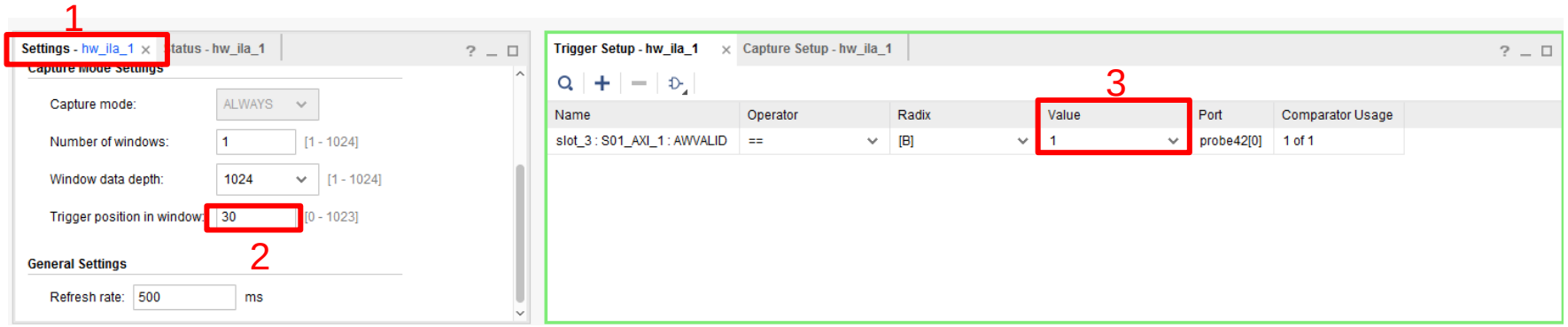
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA)
  - In the ***'Trigger Setup'*** window, click the ***'Add Probe'*** icon, type ***"AWVALID"*** in the Search field
  - Double-click the ***'axi_dma_0_M_AXI_S2MM:AWVALID'***



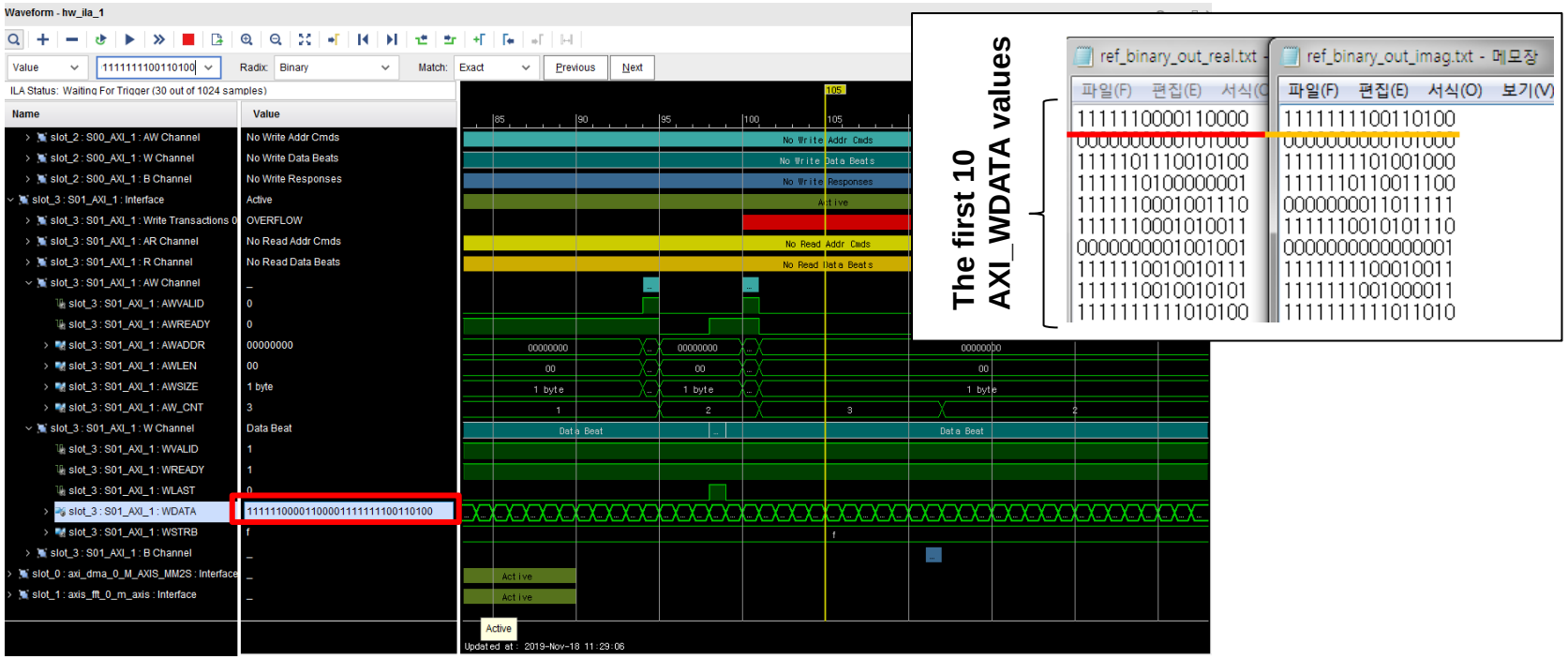
Debugging Designs in ILA

- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Choose ***'[B] (Binary)'*** in the '***Radix'*** field
  - Click '***Value'*** and then select ***'1(logical one)'***
  - Choose the ***'Setting'*** tap and the type ***'Trigger position in window'***



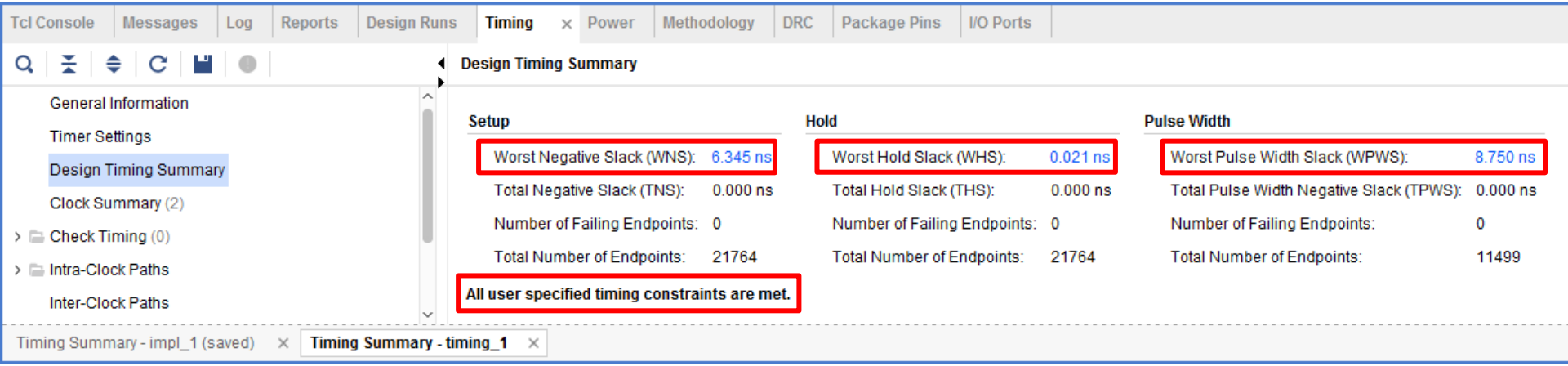
Debugging Designs in ILA

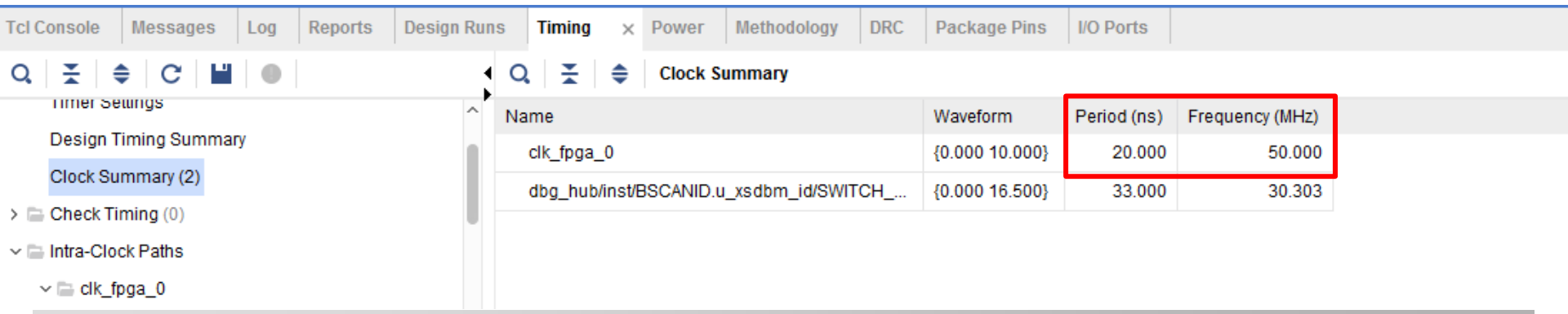
- Run Integrated Logic Analyzer (ILA) (cont'd)
  - Check the first few values of ***'AXI_WDATA'*** at the **rising** edges of ***'AXI_WVALID'***
  - Compare them with the reference outputs.



Debugging Designs in ILA

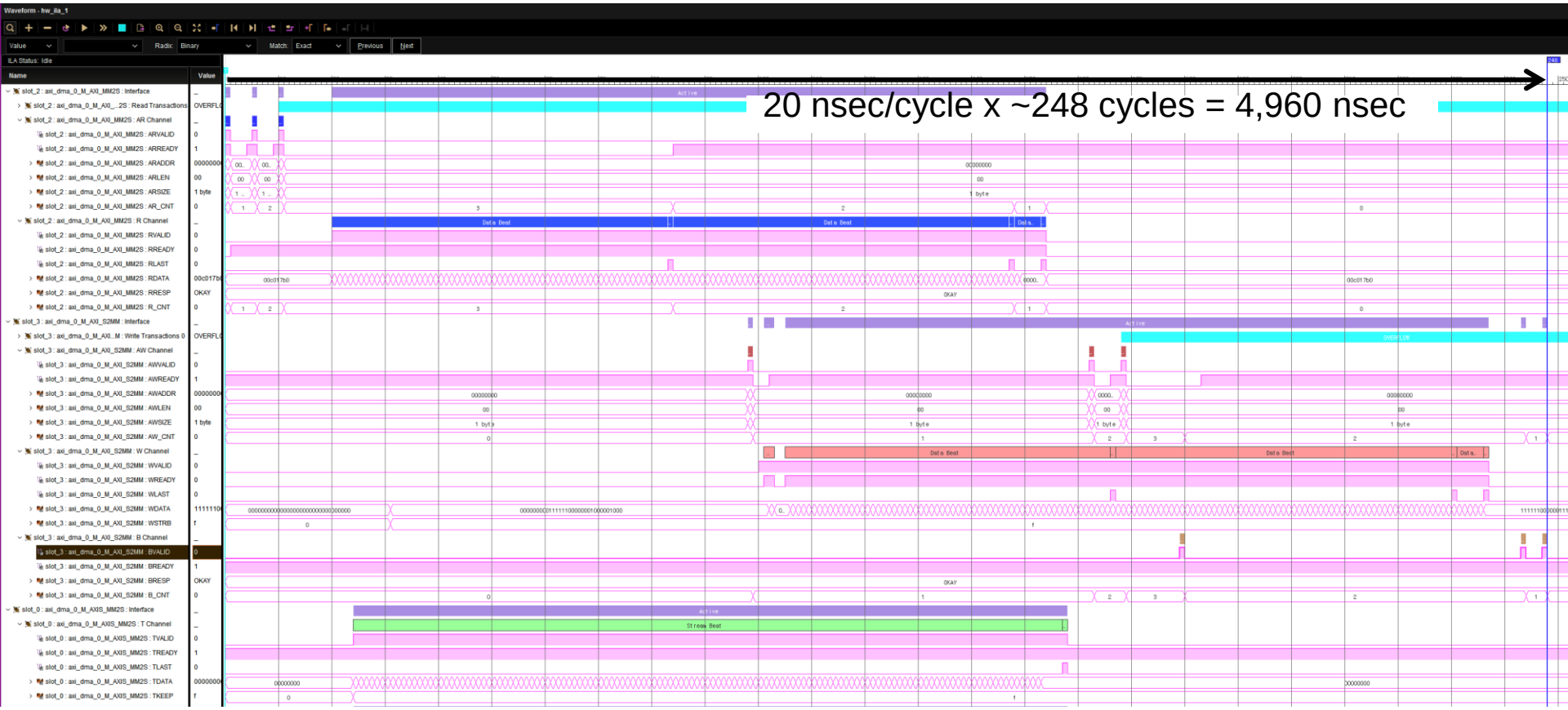
- Measure the execution time
  - Follow pp. 70~72 of the following lab workbook: **Lab_EC_3w.pdf**





Debugging Designs in ILA

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



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

- "UG585 – Zynq-7000 All Programmable SoC Technical Reference Manual", version 2017.10
- "UG939 – Vivado Design Suite Tutorial: Designing with IP", version 2017.3
- "UG1165 – Zynq-7000 All Programmable SoC: Embedded Design Tutorial, A Hands-On Guide to Effective Embedded System Design", version 2015.4
- "AXI DMA v7.1: LogiCORE IP Product Guide", version 2015.11
- For more information, refer to the following site: https://www.xilinx.com/support.html#documentation