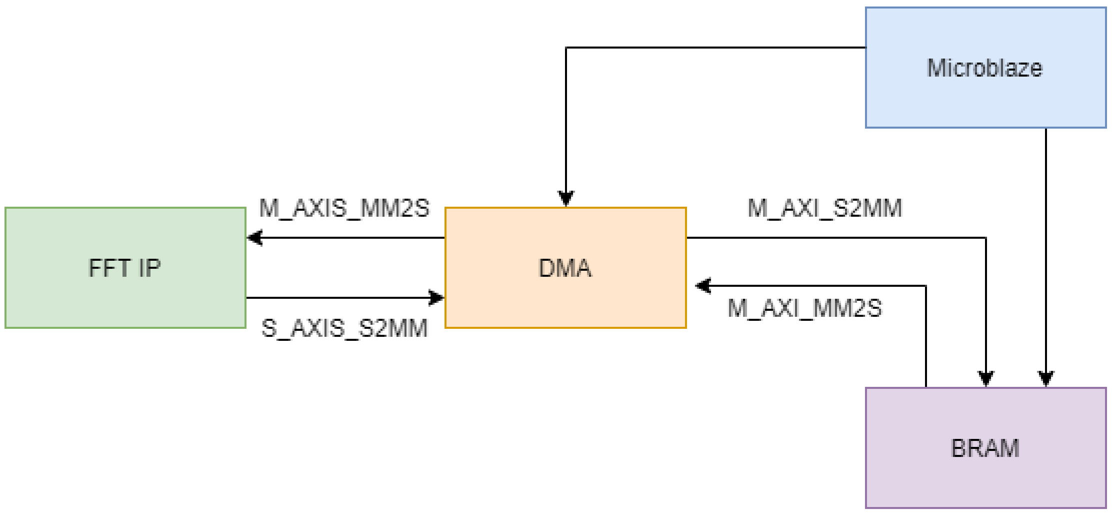
Lab 12 Handout

Introduction

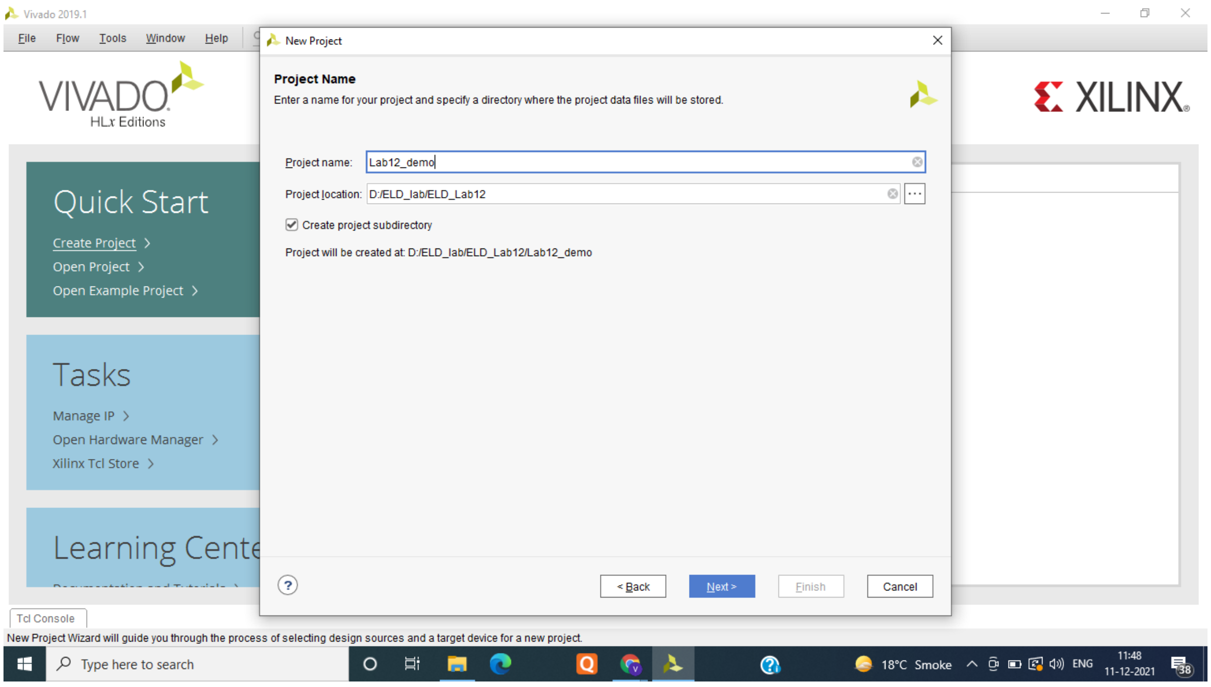
In this lab, we will be using Microblaze processor. The processor will store some data inside the BRAM and configure the DMA to perform data transfer between BRAM and FFT IP. The DMA will read the data from the BRAM via the port M_AXI_MM2S and sends it to the FFT IP via the port M_AXIS_MM2S. The FFT IP after the processing initiates the transfer, the DMA receives the processed data from FFT IP via the S_AXIS_S2MM port. The DMA writes the processed data into the BRAM via the port M_AXI_S2MM.

Block Diagram



Steps:

1) Open Vivado 2019.1. Create new project.



New Project

Project Name

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

Project name: Lab12_demo

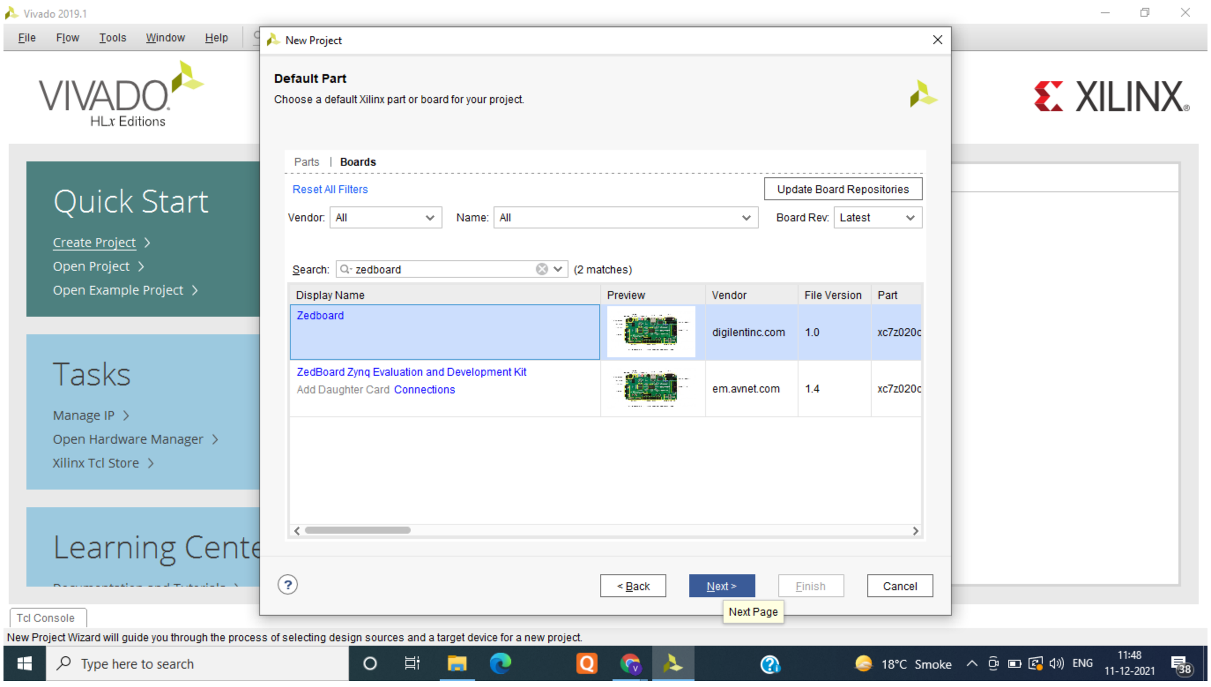
Project location: D:/ELD_lab/ELD_Lab12

Create project subdirectory

Project will be created at: D:/ELD_lab/ELD_Lab12/Lab12_demo

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

2) Click on Next.
3) Select the Zedboard from the boards tab.



New Project

Default Part

Choose a default Xilinx part or board for your project.

Vendor: All Name: All Board Rev: Latest

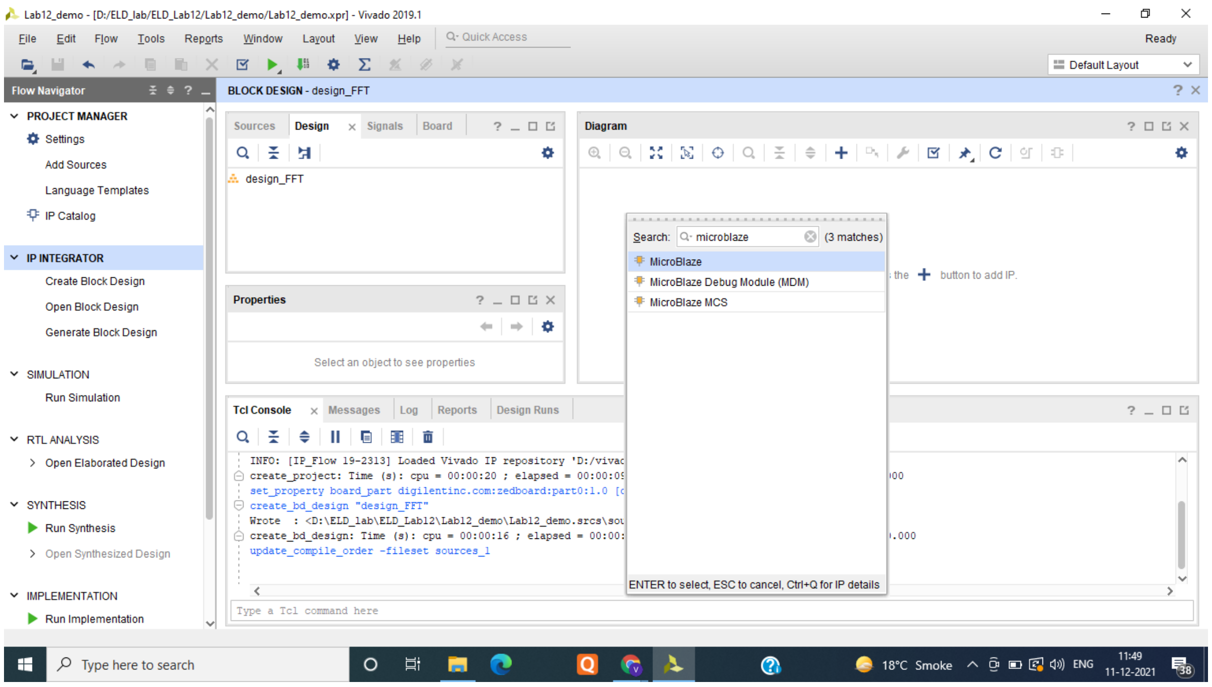
Search: zedboard (2 matches)

Display Name	Preview	Vendor	File Version	Part
Zedboard		digilentinc.com	1.0	xc7z020c
ZedBoard Zynq Evaluation and Development Kit Add Daughter Card Connections		em.avnet.com	1.4	xc7z020c

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

4) Click on next and finish to complete creating the new project.
5) In the project, click on create block design and name it.

6) Click on **+** sign on the block design window to add new IP. Search for Microblaze and add it to the block design.


Lab12_demo - [D:/ELD_lab/ELD_Lab12/Lab12_demo/Lab12_demo.xpr] - Vivado 2019.1

BLOCK DESIGN - design_FFT

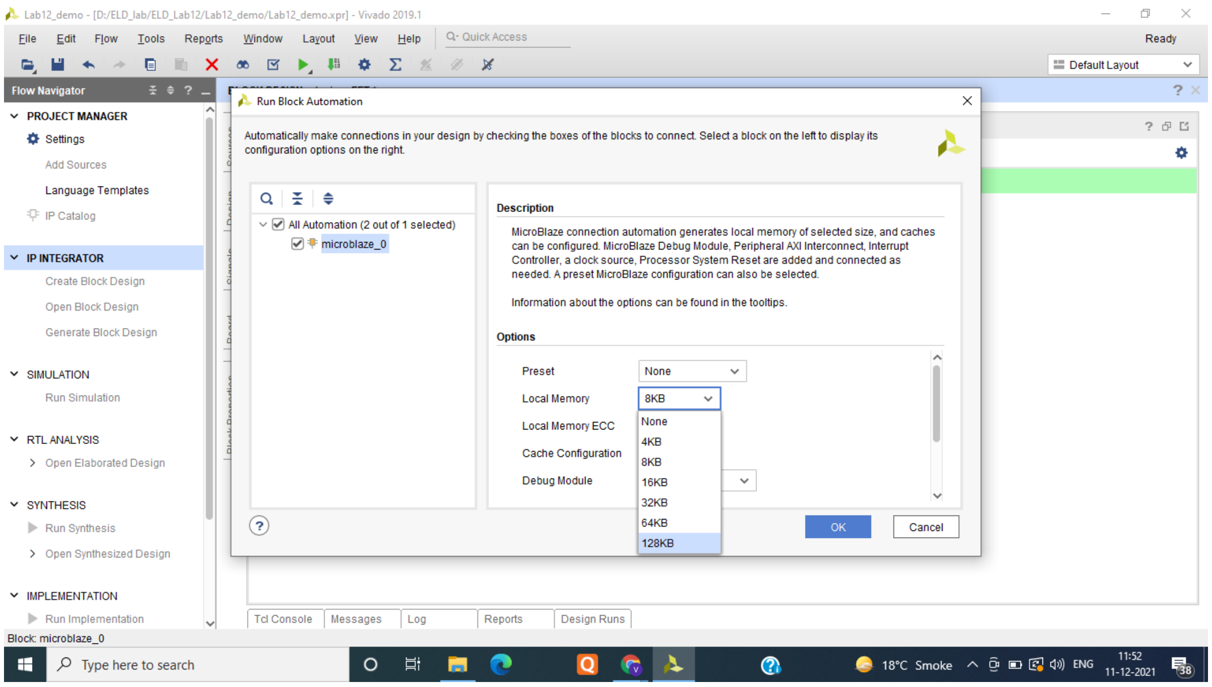
Search: microblaze (3 matches)

- MicroBlaze
- MicroBlaze Debug Module (MDM)
- MicroBlaze MCS

ENTER to select, ESC to cancel, Ctrl+Q for IP details

```
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'D:/vivac
create_project: Time (s): cpu = 00:00:20 ; elapsed = 00:00:0
set_property board_part digilentinc.com:zedboard:part0:1.0 [c
create_bd_design "design_FFT"
Wrote  : <D:\ELD_lab\ELD_Lab12\Lab12_demo\Lab12_demo.srcs\sou
create_bd_design: Time (s): cpu = 00:00:16 ; elapsed = 00:00:
update_compile_order -fileset sources_1
```

7) After adding Microblaze to the block design, click on Run Block Automation. Choose the Local memory to be **128KB** and the Clock Connection as **New External Port (100MHz)**.Now click on OK.


Lab12_demo - [D:/ELD_lab/ELD_Lab12/Lab12_demo/Lab12_demo.xpr] - Vivado 2019.1

Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

- All Automation (2 out of 1 selected)
  - microblaze_0

Description

MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI Interconnect, Interrupt Controller, a clock source, Processor System Reset are added and connected as needed. A preset MicroBlaze configuration can also be selected.

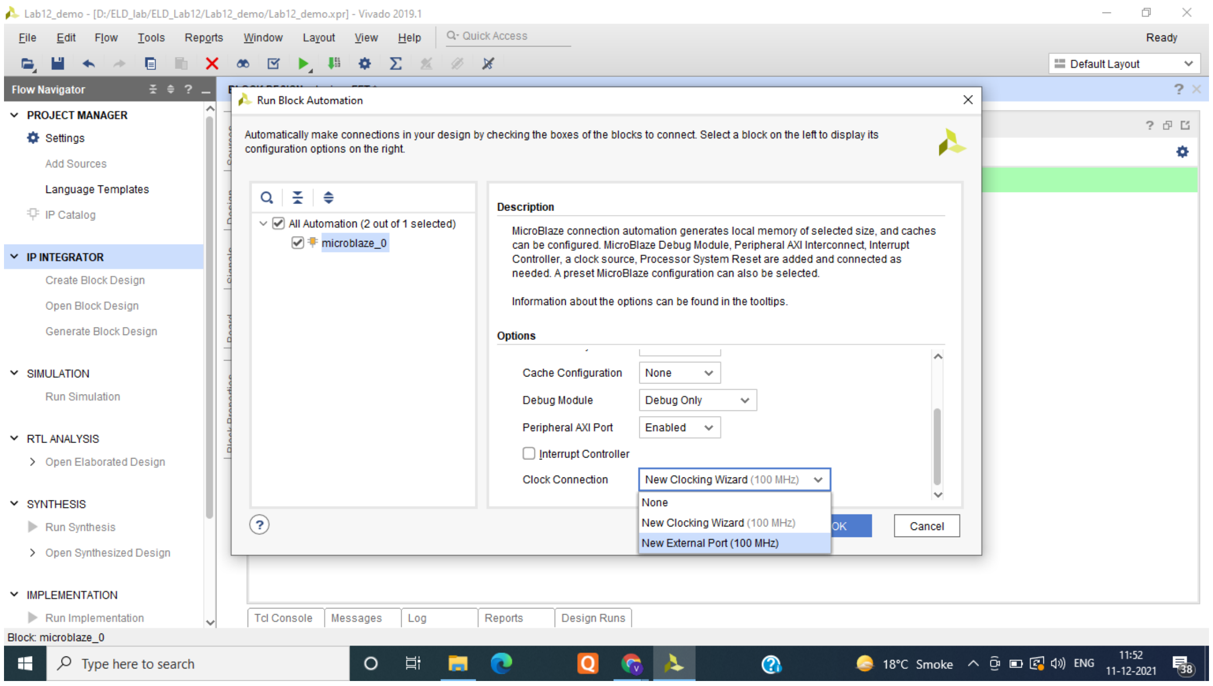
Information about the options can be found in the tooltips.

Options

Preset	None
Local Memory	8KB
Local Memory ECC	
Cache Configuration	
Debug Module	

Local Memory list: None, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB

Block: microblaze_0



Lab12_demo - [D:/ELD_lab/ELD_Lab12/Lab12_demo/Lab12_demo.xpr] - Vivado 2019.1

Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

- [✓] All Automation (2 out of 1 selected)
  - [✓] microblaze_0

Description

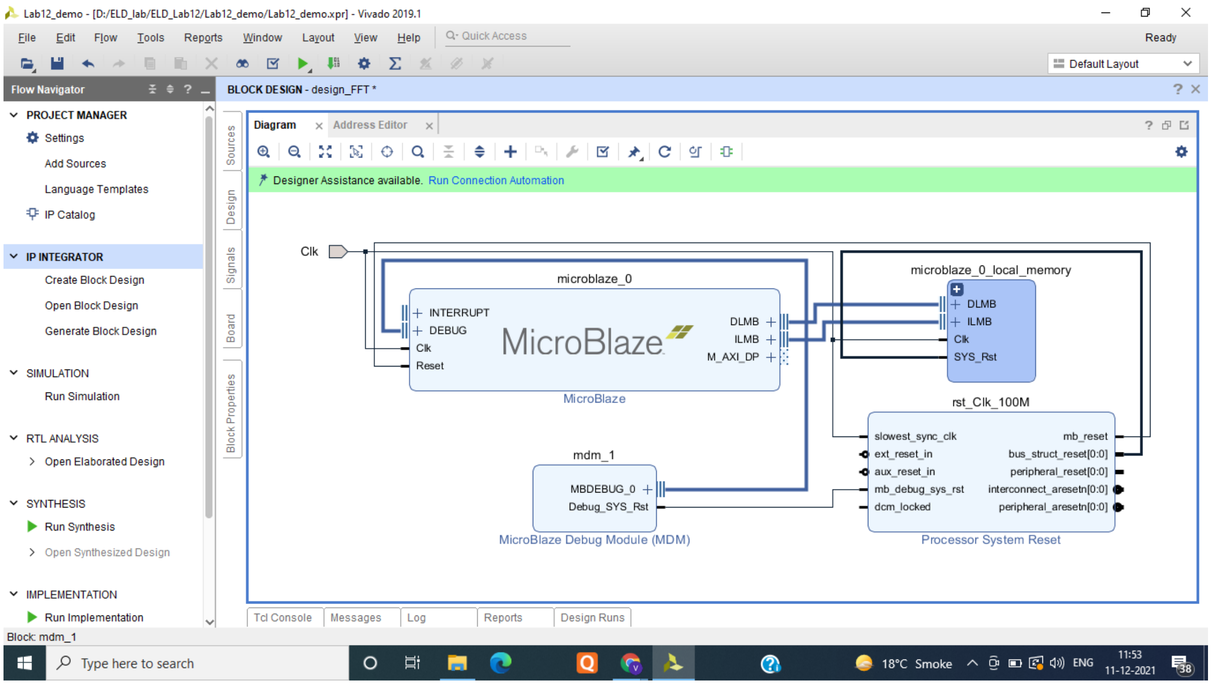
MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI Interconnect, Interrupt Controller, a clock source, Processor System Reset are added and connected as needed. A preset MicroBlaze configuration can also be selected.

Information about the options can be found in the tooltips.

Options

Cache Configuration	None
Debug Module	Debug Only
Peripheral AXI Port	Enabled
[ ] Interrupt Controller	
Clock Connection	New Clocking Wizard (100 MHz)

8) The block design now should look like this in the figure below. The local memory and Processor System Reset has been added.

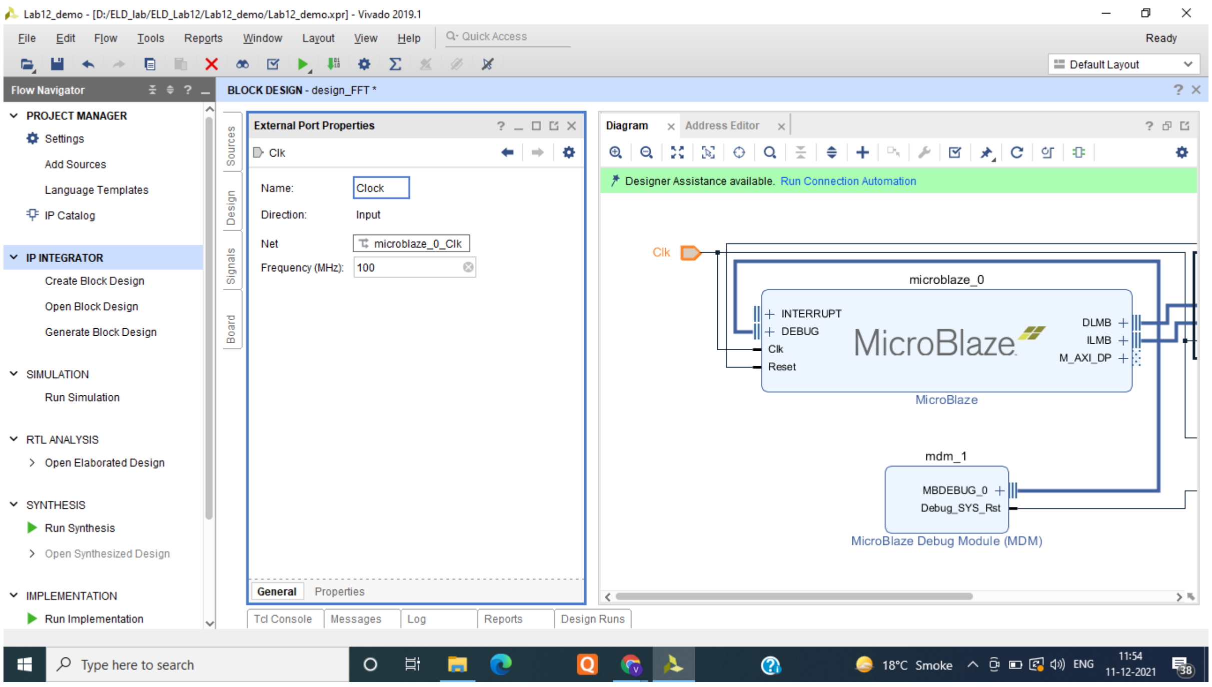


Lab12_demo - [D:/ELD_lab/ELD_Lab12/Lab12_demo/Lab12_demo.xpr] - Vivado 2019.1

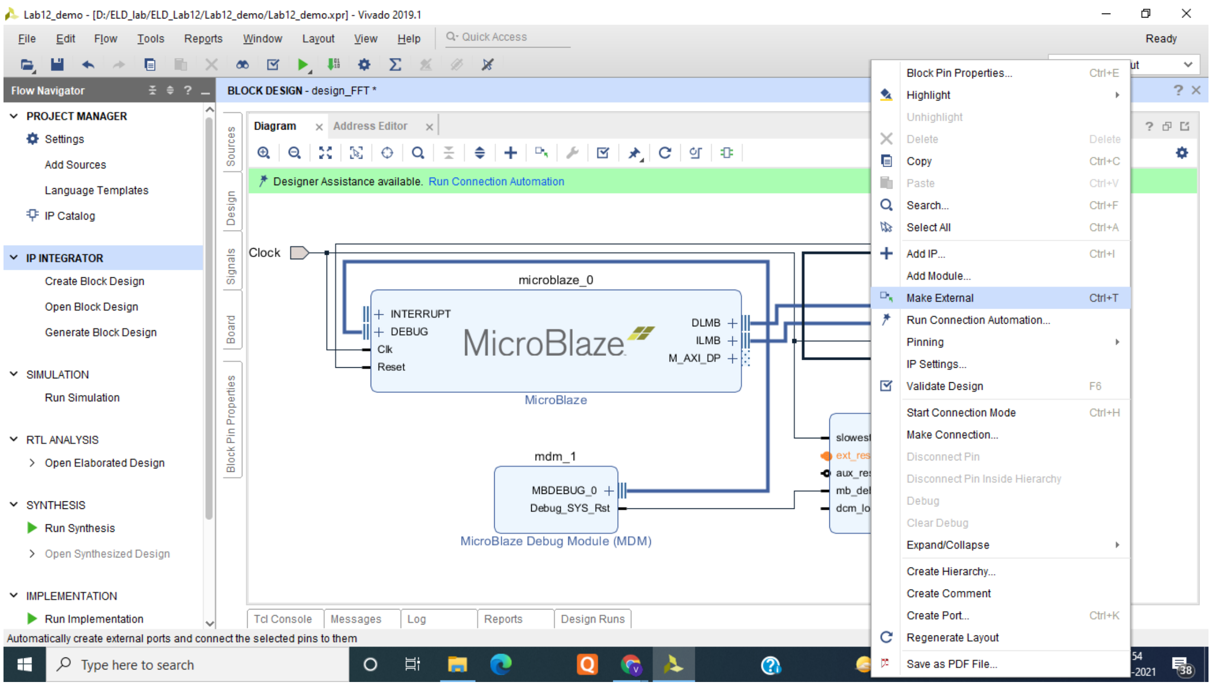
BLOCK DESIGN - design_FFT *

Designer Assistance available. Run Connection Automation

9) Select the Clk port and change its name to **Clock** (Same name should be used in the test bench) in External Port Properties.

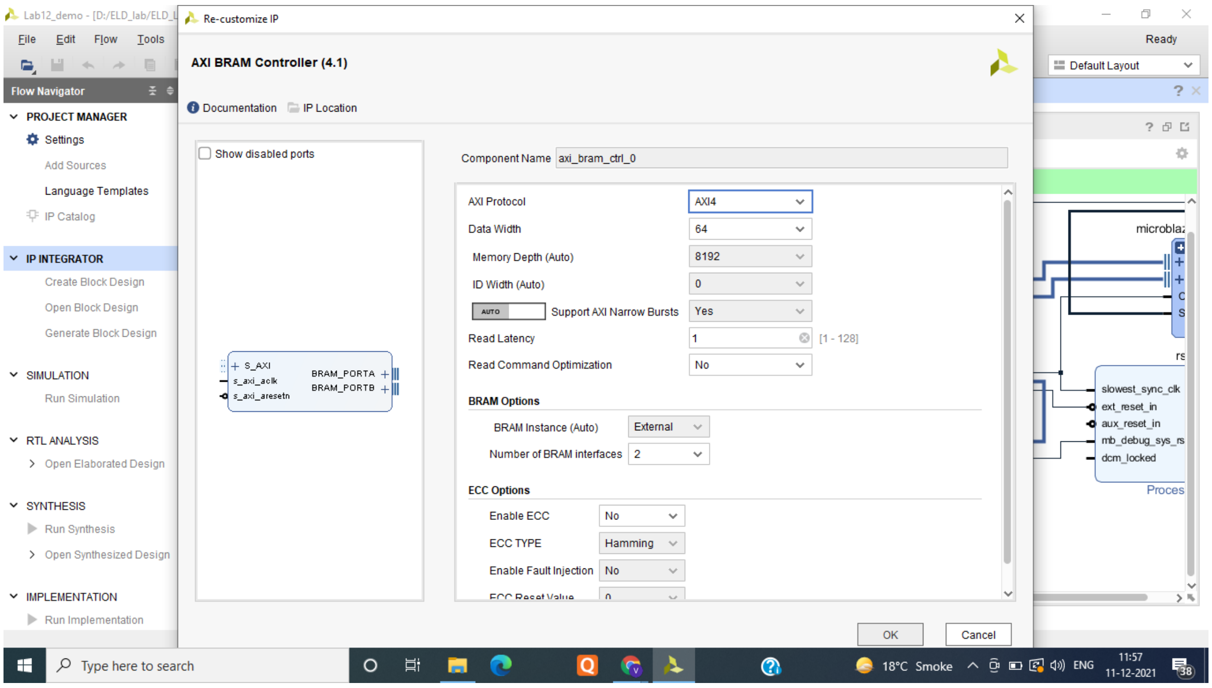


10) In the Processor System Reset, right click on the port **ext_reset_in** and select **Make External.** This will create an external port for the reset.



11) Now, change the port name ext_reset_in_0 to **Reset** in the similar way as we did for Clock.

12) Add **AXI BRAM Controller** to the block design and double click on it to customize it. Choose the **Data Width** to be **64 bits** and click on OK.


Re-customize IP

AXI BRAM Controller (4.1)

Documentation   IP Location

Show disabled ports

Component Name: axi_bram_ctrl_0

AXI Protocol	AXI4
Data Width	64
Memory Depth (Auto)	8192
ID Width (Auto)	0
Support AXI Narrow Bursts	Yes
Read Latency	1 [1 - 128]
Read Command Optimization	No

BRAM Options

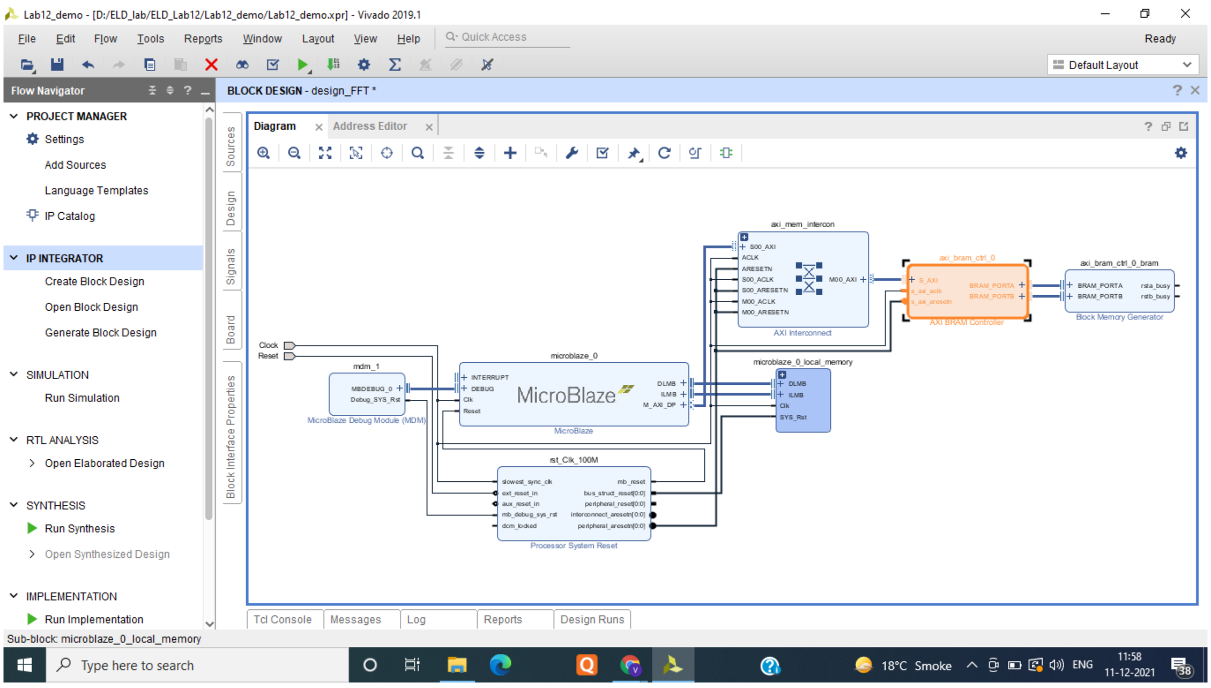
BRAM Instance (Auto)	External
Number of BRAM interfaces	2

ECC Options

Enable ECC	No
ECC TYPE	Hamming
Enable Fault Injection	No
ECC Reset Value	0

OK   Cancel

13) Click on Run Connection Automation, in the Run Connection Automation window select **All Automation** and click on OK. The Block memory generator has been added. The block diagram will now look like the figure below.



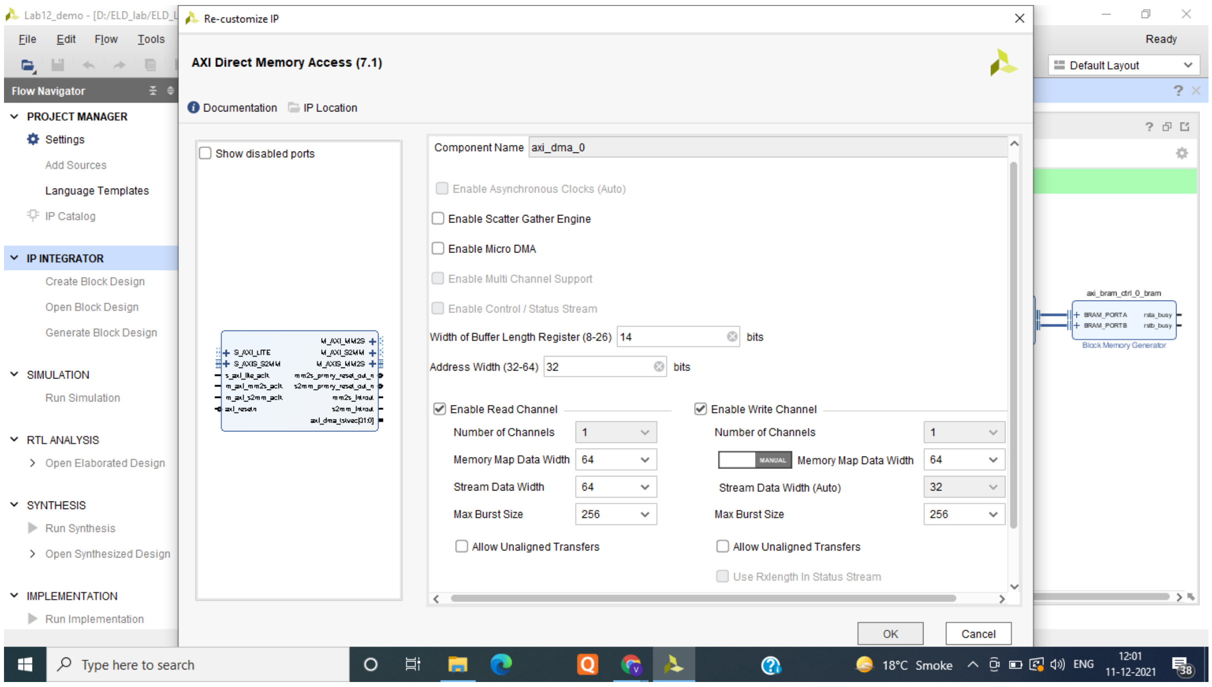
14) Add the **AXI Direct Memory Access** to the block design and double click on it to re-customize it. Make the following changes.

a) Unselect the Enable Scatter Gather Engine. This special mode of DMA is used to read from non-contiguous memory locations.

b) In the Enable Read Channel, increase the Memory Map Data Width and Stream Data Width to 64. Select Max Burst Size of 256.

c) In the Enable Write Channel, increase the Memory Map Data Width to 64. Select Max Burst Size of 256.

Click on OK



Re-customize IP

AXI Direct Memory Access (7.1)

Component Name: axi_dma_0

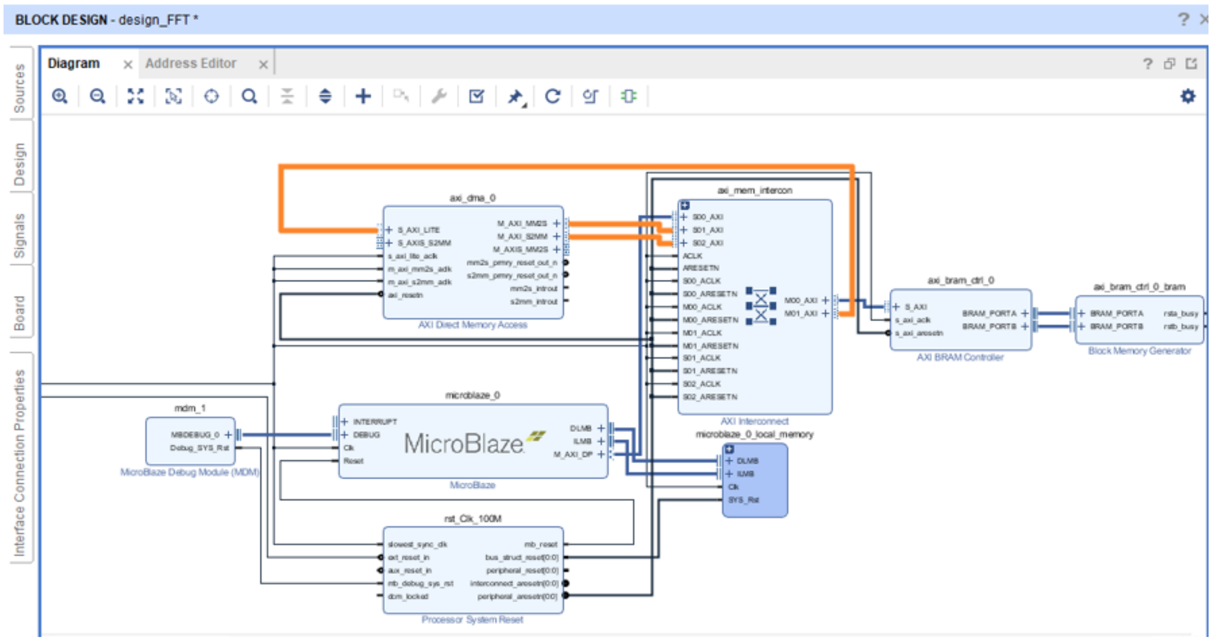
- [ ] Show disabled ports
- [ ] Enable Asynchronous Clocks (Auto)
- [ ] Enable Scatter Gather Engine
- [ ] Enable Micro DMA
- [ ] Enable Multi Channel Support
- [ ] Enable Control / Status Stream

Width of Buffer Length Register (8-26): 14 bits

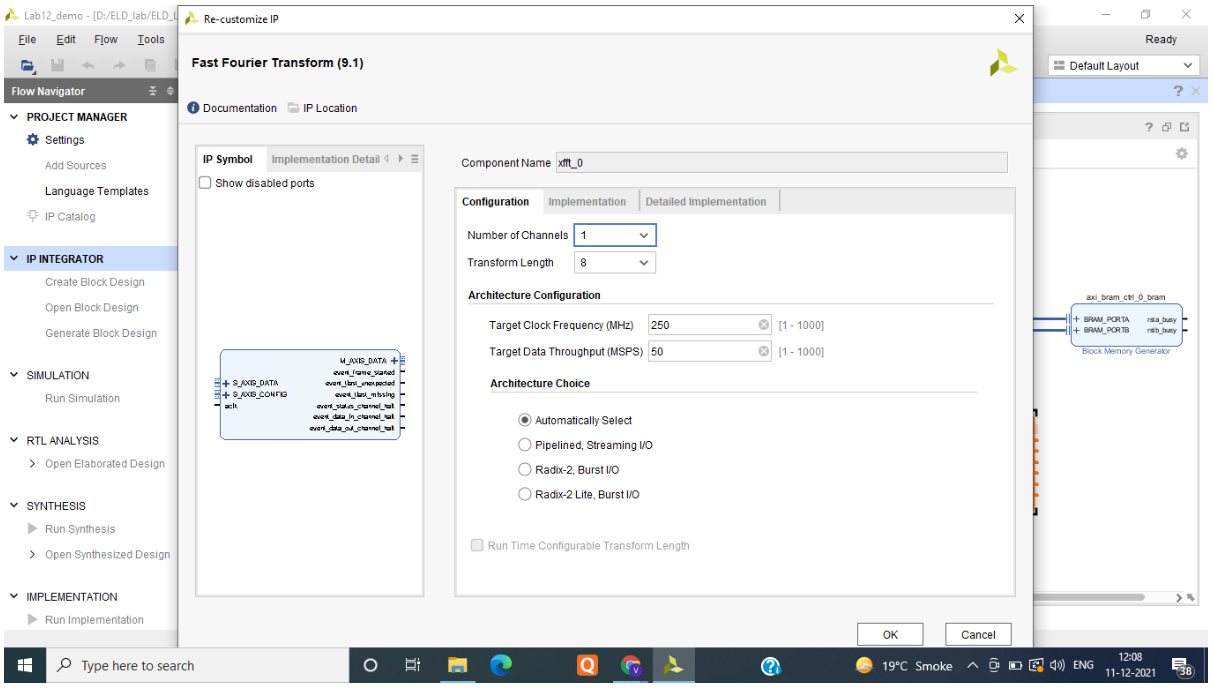
Address Width (32-64): 32 bits

[✓] Enable Read Channel		[✓] Enable Write Channel	
Number of Channels	1	Number of Channels	1
Memory Map Data Width	64	Memory Map Data Width	64
Stream Data Width	64	Stream Data Width (Auto)	32
Max Burst Size	256	Max Burst Size	256
[ ] Allow Unaligned Transfers		[ ] Allow Unaligned Transfers	
		[ ] Use Rxlength In Status Stream	

15) Click on Run Connection Automation, in the Run Connection Automation window select **All Automation** and click on OK. Two slave ports are added to the AXI interconnect. The AXI DMA will perform read and write operations from the Block Memory Generator via AXI interconnect. The DMA port **M_AXI_MM2S** is for reading the data and the port **M_AXI_S2MM** is for writing the data into the Block Memory Generator. The DMA gets configured by the Microblaze through the **S_AXI_LITE** port.



16) Add the **Fast Fourier Transform (FFT)** IP and double click on it to re-customize. Under the configuration tab select the **Transform length** as **8** (8-point FFT).


Fast Fourier Transform (9.1)

Component Name xfft_0

Configuration

Number of Channels 1

Transform Length 8

Architecture Configuration

Target Clock Frequency (MHz) 250 [1 - 1000]

Target Data Throughput (MSPS) 50 [1 - 1000]

Architecture Choice

(●) Automatically Select

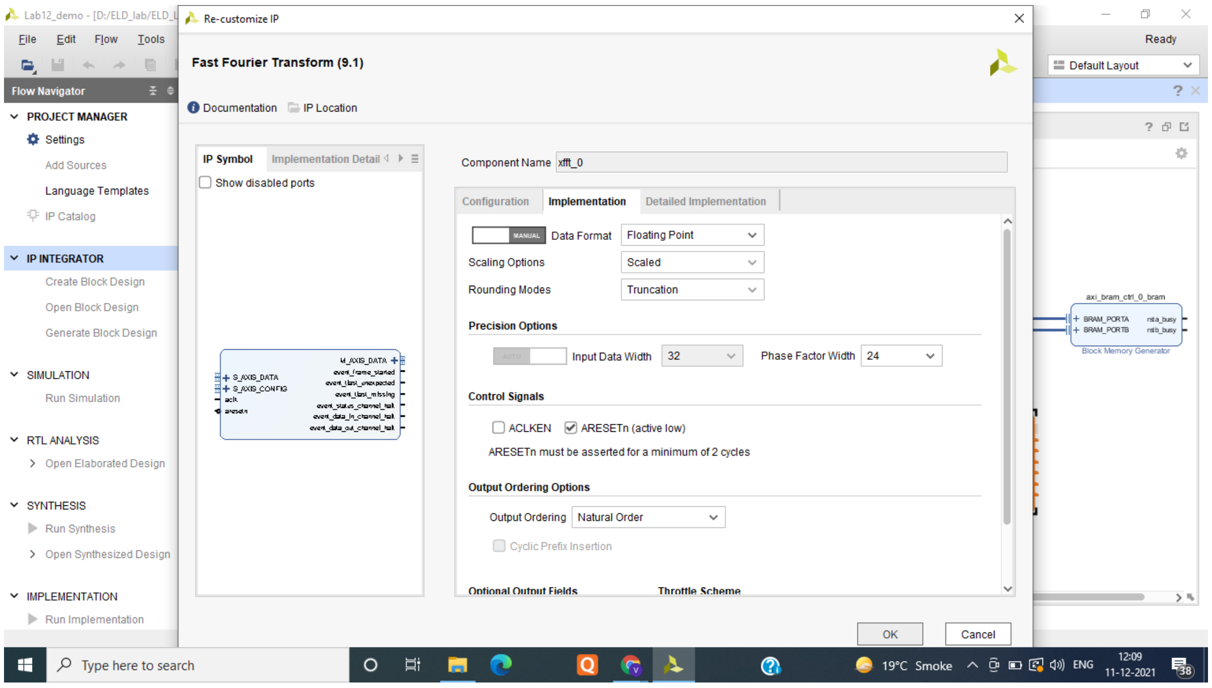
( ) Pipelined, Streaming I/O

( ) Radix-2, Burst I/O

( ) Radix-2 Lite, Burst I/O

[ ] Run Time Configurable Transform Length

17) Under the implementation tab, change the Data Format to Floating Point. Add the control signal ARESETn. Select the Output Ordering as Natural Order. Click on OK.


Fast Fourier Transform (9.1)

Component Name xfft_0

Implementation

Data Format Floating Point

Scaling Options Scaled

Rounding Modes Truncation

Precision Options

Input Data Width 32 Phase Factor Width 24

Control Signals

[ ] ACLKEN [✓] ARESETn (active low)

ARESETn must be asserted for a minimum of 2 cycles

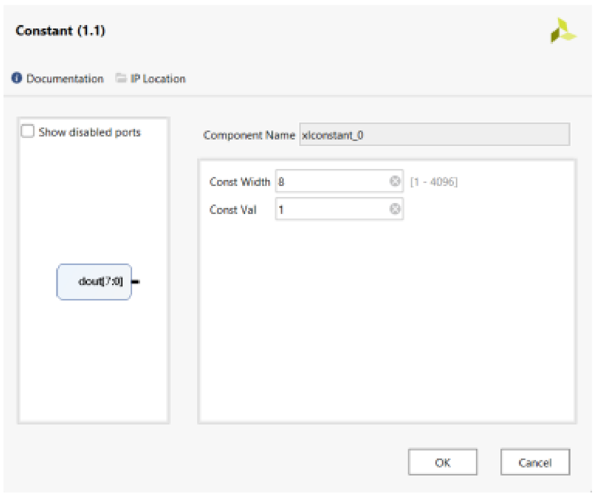
Output Ordering Options

Output Ordering Natural Order

[ ] Cyclic Prefix Insertion

18) FFT IP connections

a) Manually connect the aclk and aresetn of FFT IP to the common Clock port and Reset port respectively.
b) For Configuration of the FFT IP, the S_axis_config_tdata and S_axis_config_tvalid should be manually connected to constants of value 1.
c) S_axis_config_tdata [7:0] is connected to a constant of width 8 and value 1

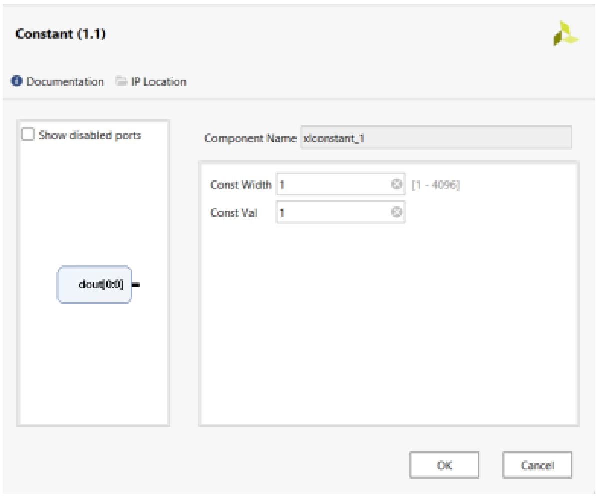

Constant (1.1)

Component Name xlconstant_0

Const Width 8 [1 - 4096]

Const Val 1

d) S_axis_config_tvalid is connected to a constant of width 1 and value 1


Constant (1.1)

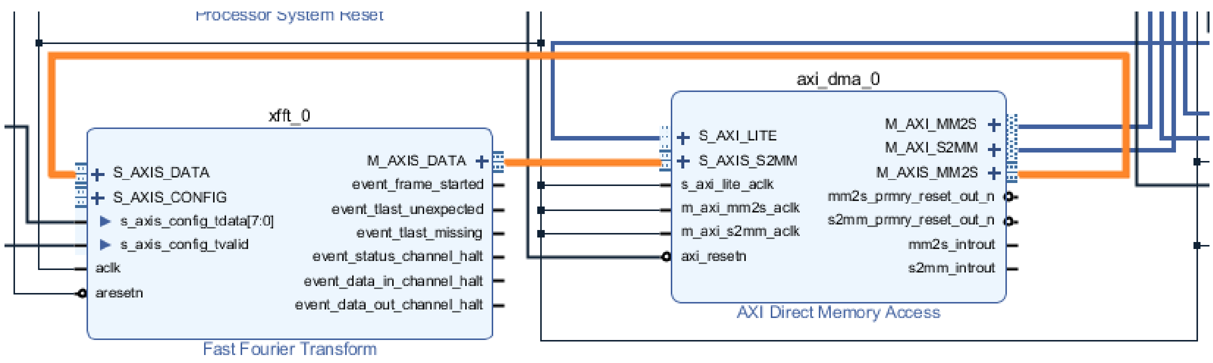
Component Name xlconstant_1

Const Width 1 [1 - 4096]

Const Val 1

e) Manually connect the M_AXIS_MM2S port of the DMA IP to the S_AXIS_DATA port of the FFT IP. The DMA IP transfers the data read from the BRAM to the FFT IP.

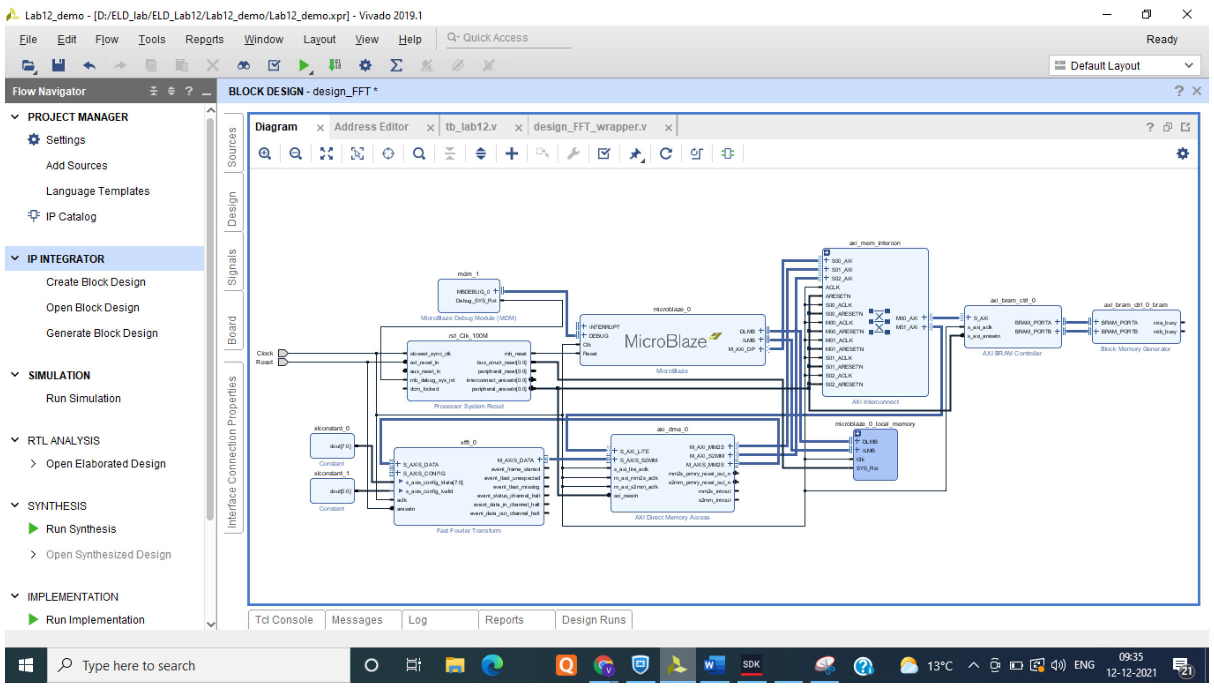
f) Manually connect the S_AXIS_S2MM port of the DMA IP to the M_AXIS_DATA port of the FFT IP. The FFT IP transfers the data after processing to the DMA.



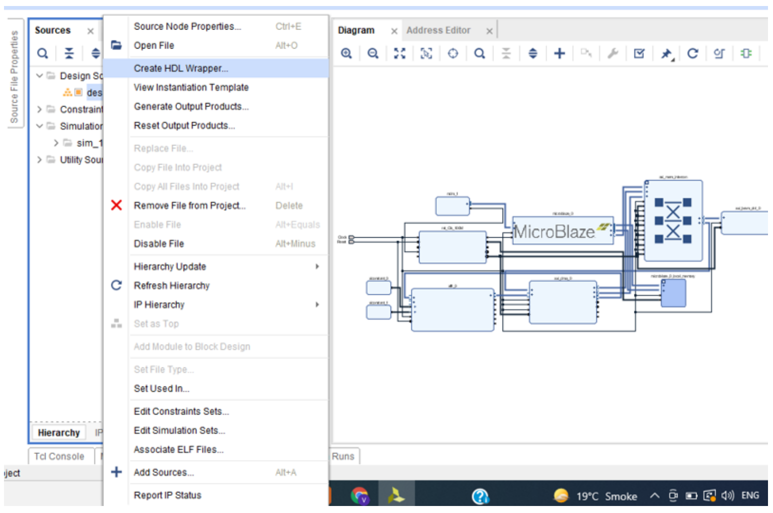
19) The block design is now complete.

20) Validate the design to check for any missing or incorrect connections.

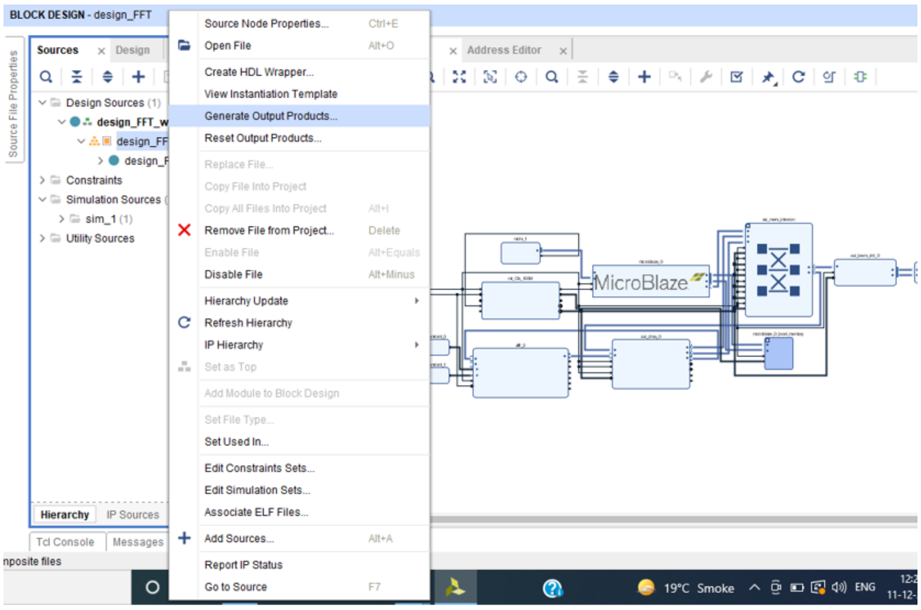
21) The final block design will look like the below figure.



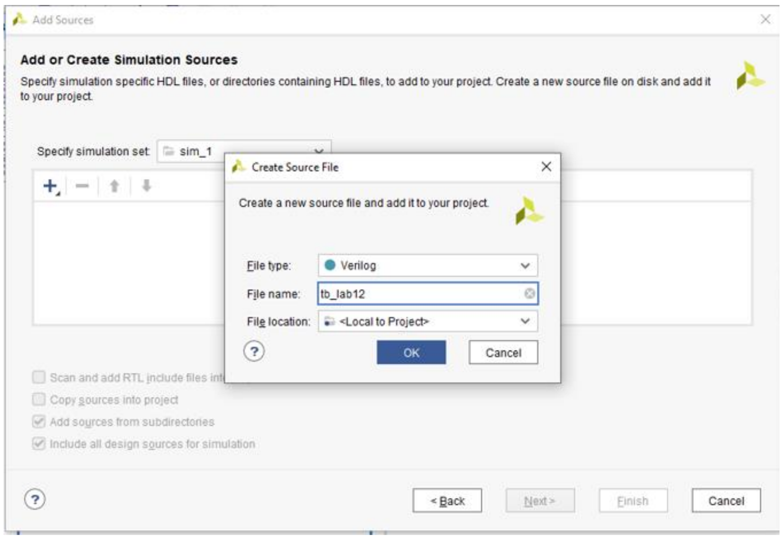
22) In the sources tab, right-click on your block design file and select Create HDL Wrapper. Click on OK. This step will generate the corresponding Verilog files of your design.



23) In the sources tab, right-click on your block design file and select Generate Output Products. Click on Generate.



24) Create a simulation source for writing the testbench.


Add Sources

Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

Specify simulation set: sim_1

Create Source File

Create a new source file and add it to your project.

File type: Verilog

File name: tb_lab12

File location: <Local to Project>

OK Cancel

Scan and add RTL include files into project

Copy sources into project

Add sources from subdirectories

Include all design sources for simulation

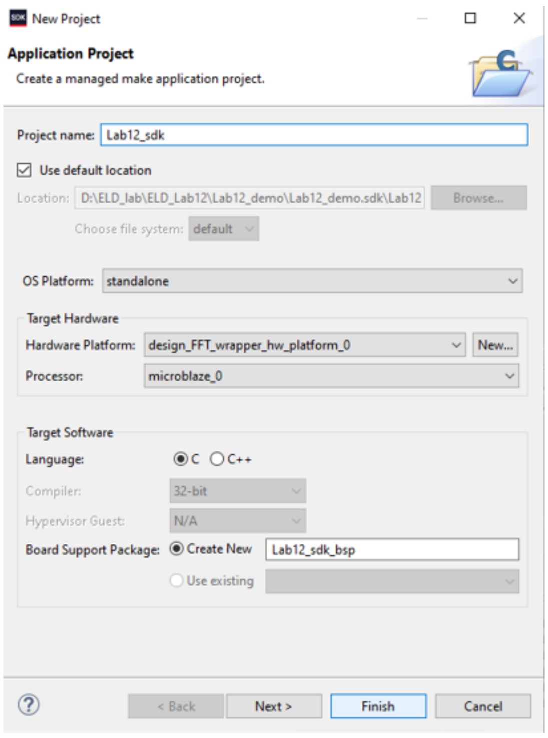
< Back Next > Finish Cancel

25) Write the test bench as shown below.

```
module tb_lab12(

    );
    reg Clock,Reset;

    initial begin
    Clock = 0;
    Reset = 0;
    #100 Reset = 1;
    end

    always
    #5 Clock = ~Clock;

   design_FFT_wrapper dut(.Clock(Clock),.Reset(Reset));


endmodule
```

26) Export Hardware and Launch SDK.

27) In the SDK, create new application project (File->New-> Application Project). Click on Finish.


New Project

Application Project

Create a managed make application project.

Project name: Lab12_sdk

[✓] Use default location

Location: D:\ELD_lab\ELD_Lab12\Lab12_demo\Lab12_demo.sdk\Lab12

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: design_FFT_wrapper_hw_platform_0

Processor: microblaze_0

Target Software

Language: [✓ C] C++

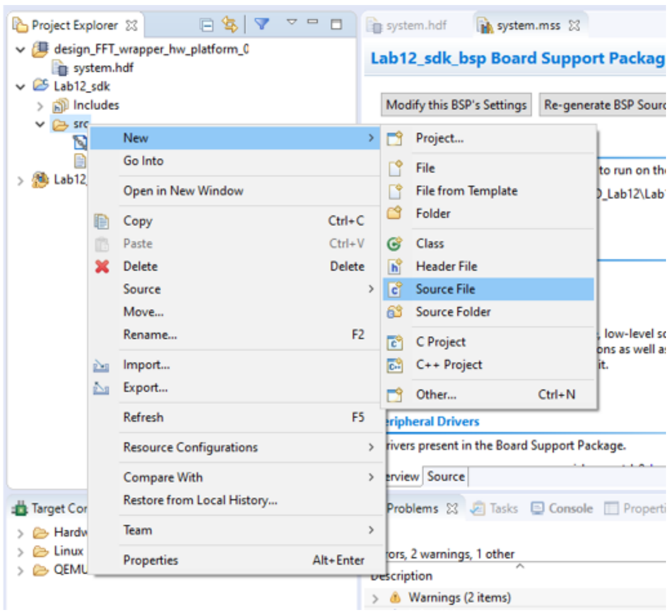
Compiler: 32-bit

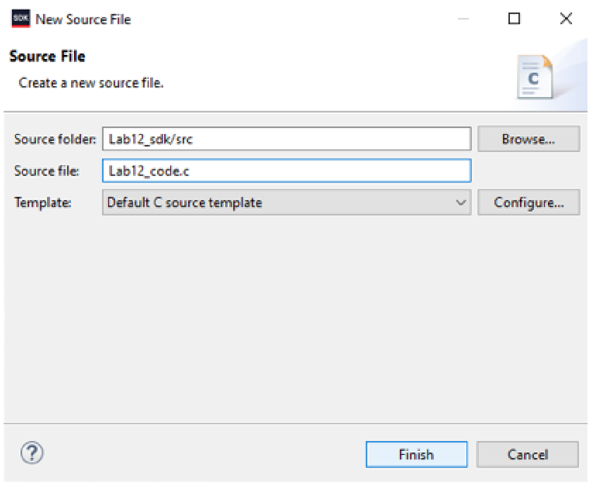
Hypervisor Guest: N/A

Board Support Package: [✓ Create New] Lab12_sdk_bsp

Use existing

28) Add new source file to the project created as shown below and name the file with .c extension.




New Source File

Source File

Create a new source file.

Source folder: Lab12_sdk/src

Source file: Lab12_code.c

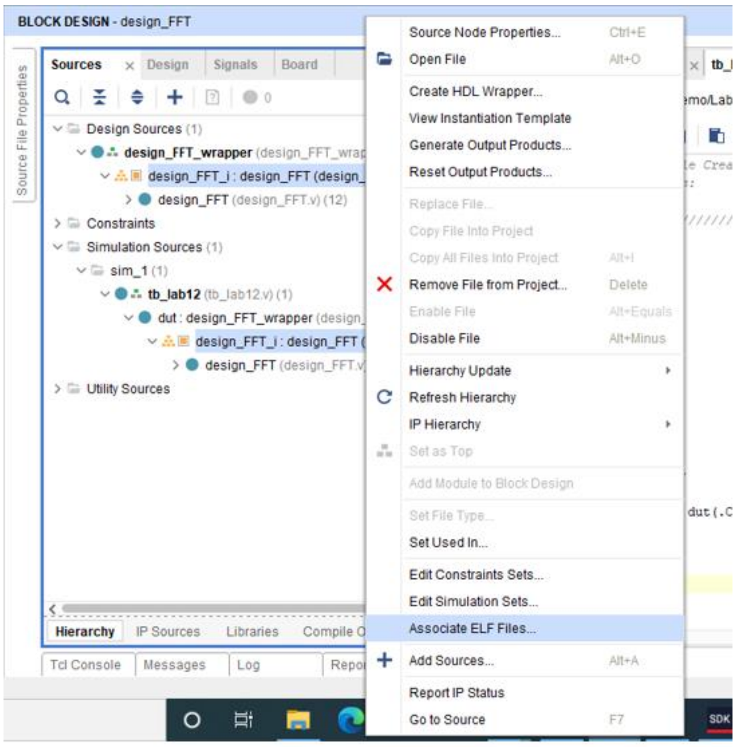
Template: Default C source template

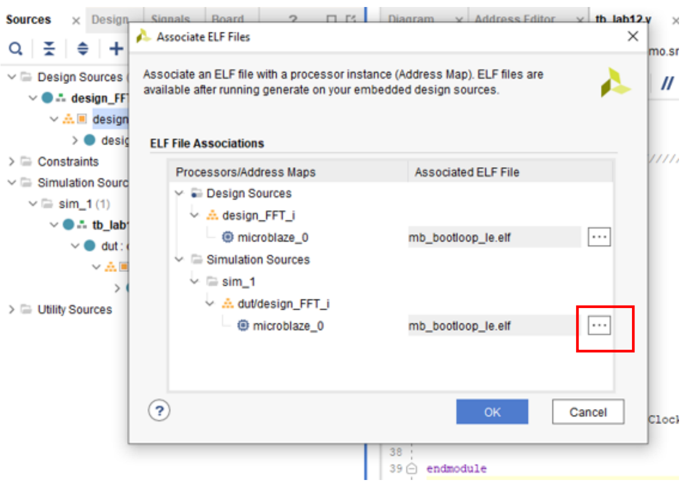
29) Open the .c file from the src folder and add the following code. Save it and the ELF file is generated.

```c
#include "xparameters.h"
#include "xaxidma.h"
#include "xio.h"
#include "complex.h"
```

```
#define N 8
int main()
{
	// First real then imag : Each pair of data is a pair of real and
imaginary.
	const float FFT_input[2*N] =
{0x41300000,0x41b80000,0x42000000,0x41200000,0x42b60000,0x42bc0000,0x41700000,0x42
8a0000,0x423c0000,0x42c00000,0x42300000,0x41400000,0x42c00000,0x41880000,0x4244000
0,0x42680000};

	int status;

	int init_addr = XPAR_BRAM_0_BASEADDR;
	// Writing data into the BRAM
	for(int i=0;i<2*N;i++)
	{
		XIo_Out32(init_addr,FFT_input[i]);
		init_addr = init_addr + 4; //Incrementing the address by 4 bytes
since we are writing 32 bits data.
	}
	XAxiDma my_dma;
	XAxiDma_Config *my_dma_config;

	//Copying of the configuration information given the base address of the
DMA
	my_dma_config = XAxiDma_LookupConfigBaseAddr(XPAR_AXI_DMA_0_BASEADDR);

	//Configuration initialization
	XAxiDma_CfgInitialize(&my_dma,my_dma_config);
	// IP to DMA transfer configuration
	XAxiDma_SimpleTransfer(&my_dma,XPAR_BRAM_0_BASEADDR+sizeof(float)*N*2,sizeo
f(float )*N*2,XAXIDMA_DEVICE_TO_DMA);
	// DMA to IP transfer configuration
	XAxiDma_SimpleTransfer(&my_dma,XPAR_BRAM_0_BASEADDR,sizeof(float)*N*2,XAXID
MA_DMA_TO_DEVICE);


	// Checking whether the DMA has completed the transfers or not


	status = XAxiDma_ReadReg(XPAR_AXI_DMA_0_BASEADDR,0x4) & XAXIDMA_HALTED_MASK
	;
    while(status!=1)
    {
	status = XAxiDma_ReadReg(XPAR_AXI_DMA_0_BASEADDR,0x4) & XAXIDMA_HALTED_MASK
	;
    }

    status = XAxiDma_ReadReg(XPAR_AXI_DMA_0_BASEADDR,0x34) & XAXIDMA_HALTED_MASK ;
    while(status!=1)
	{
		status = XAxiDma_ReadReg(XPAR_AXI_DMA_0_BASEADDR,0x34) &
XAXIDMA_HALTED_MASK ;
	}

	return 0;



}
```

30) Open Vivado. Under the simulation sources, right click on the design and click on Associate ELF files. Click on the 3 dots to add the ELF file.



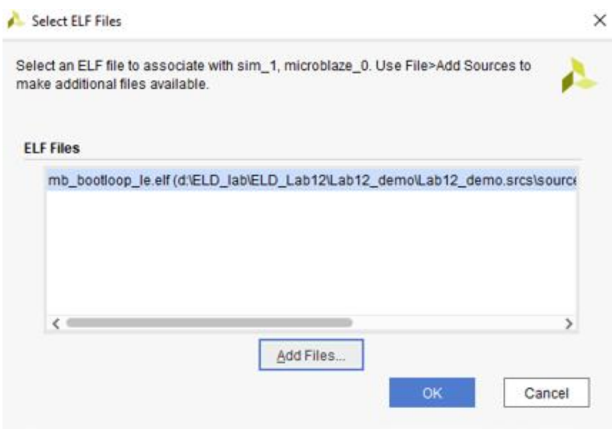


Associate an ELF file with a processor instance (Address Map). ELF files are available after running generate on your embedded design sources.

ELF File Associations

Processors/Address Maps	Associated ELF File
Design Sources	
design_FFT_i	
microblaze_0	mb_bootloop_le.elf
Simulation Sources	
sim_1	
dut/design_FFT_i	
microblaze_0	mb_bootloop_le.elf

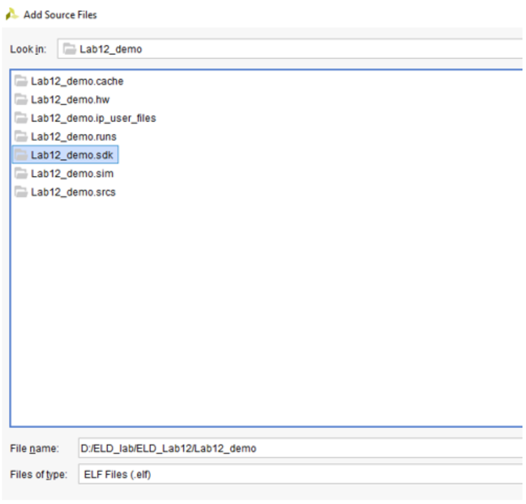
31) The path to the elf file -> lab_folder-> lab_folder.sdk->sdk_folder->Debug->.elf



Select an ELF file to associate with sim_1, microblaze_0. Use File>Add Sources to make additional files available.

ELF Files

mb_bootloop_le.elf (d:\ELD_lab\ELD_Lab12\Lab12_demo\Lab12_demo.srcs\source

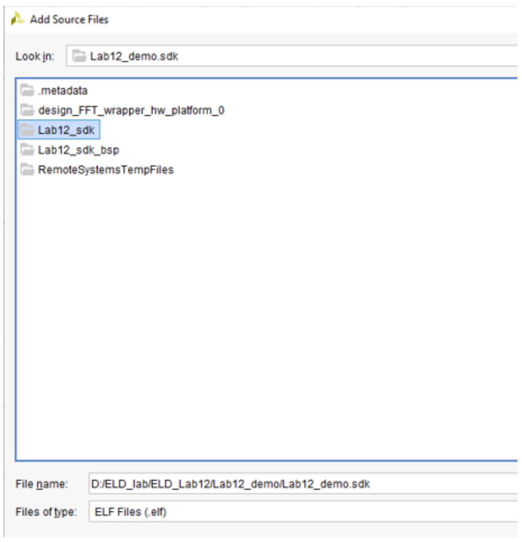


Look in: Lab12_demo

Lab12_demo.cache
Lab12_demo.hw
Lab12_demo.ip_user_files
Lab12_demo.runs
Lab12_demo.sdk
Lab12_demo.sim
Lab12_demo.srcs

File name: D:/ELD_lab/ELD_Lab12/Lab12_demo

Files of type: ELF Files (.elf)

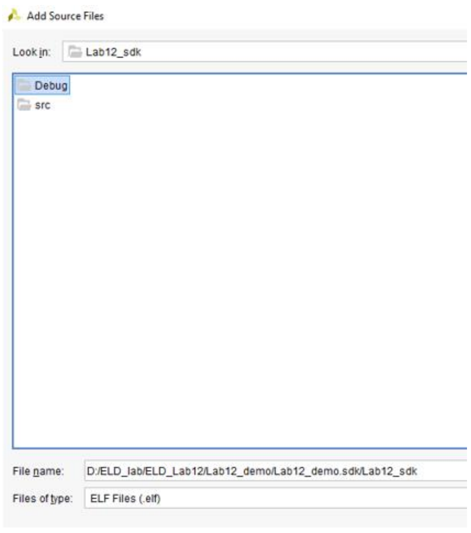

Add Source Files

Look in: Lab12_demo.sdk

.metadata
design_FFT_wrapper_hw_platform_0
Lab12_sdk
Lab12_sdk_bsp
RemoteSystemsTempFiles

File name: D:/ELD_lab/ELD_Lab12/Lab12_demo/Lab12_demo.sdk

Files of type: ELF Files (.elf)

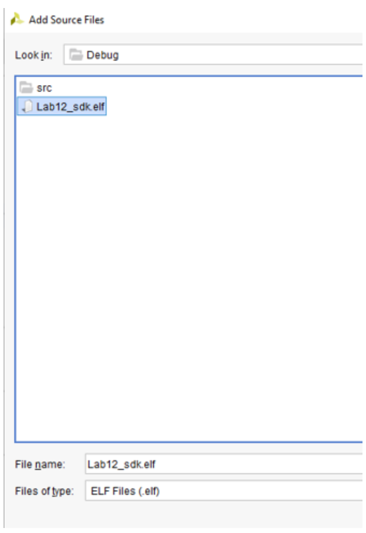

Add Source Files

Look in: Lab12_sdk

Debug
src

File name: D:/ELD_lab/ELD_Lab12/Lab12_demo/Lab12_demo.sdk/Lab12_sdk

Files of type: ELF Files (.elf)


Add Source Files

Look in: Debug

src
Lab12_sdk.elf

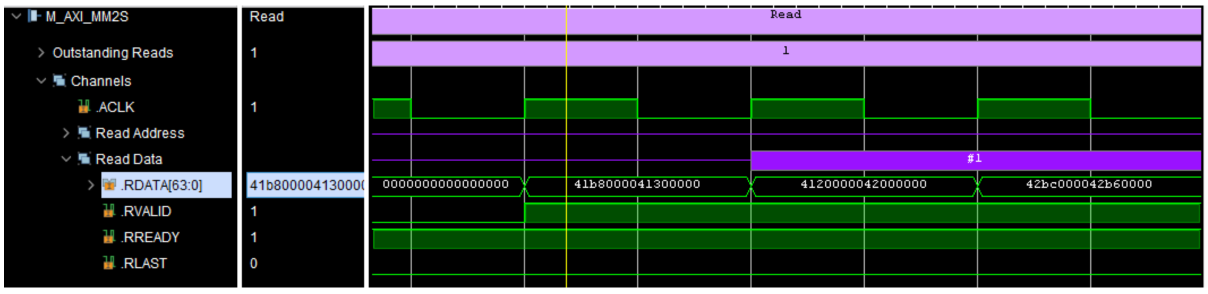
File name: Lab12_sdk.elf

Files of type: ELF Files (.elf)

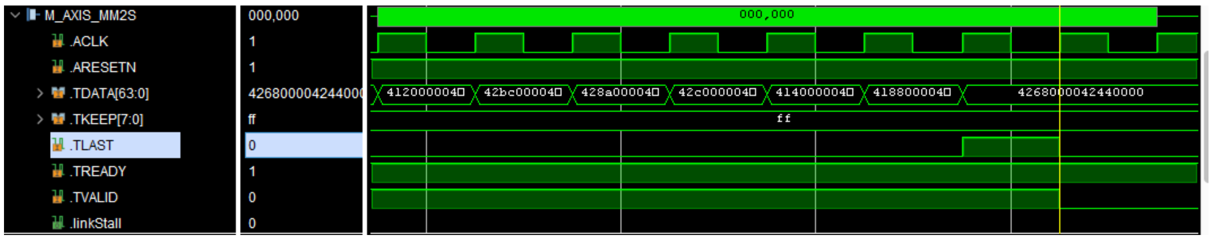
32) Now, run simulation to observe the waveforms.

AXI transactions waveforms

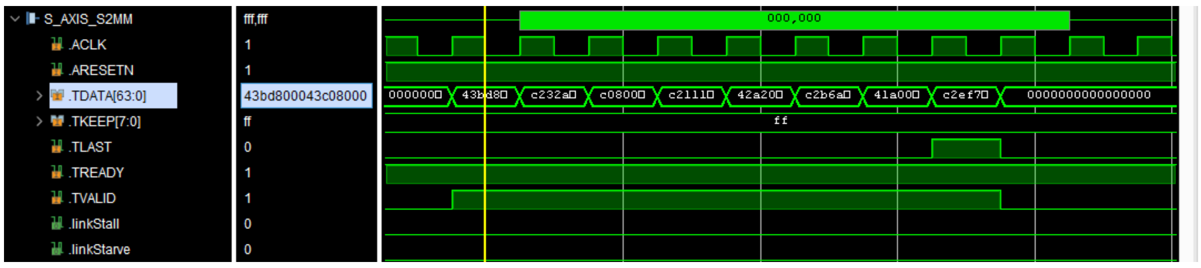
BRAM to DMA data transfer



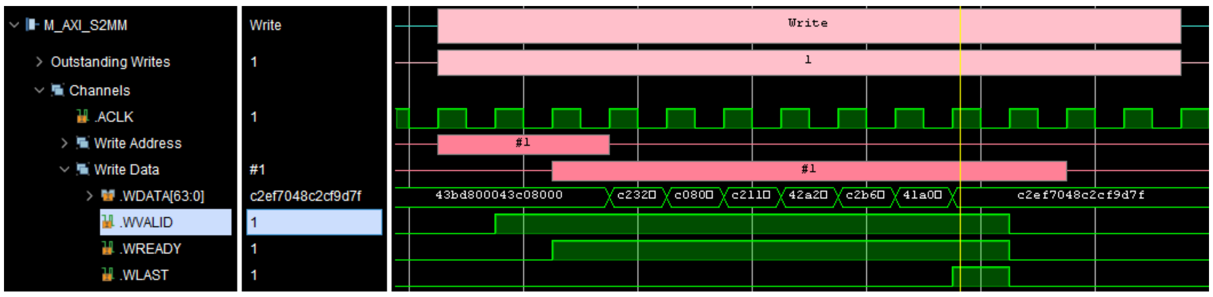
DMA to FFT IP data transfer



FFT IP to DMA transfer after processing



DMA to BRAM data transfer (writing the processed data into BRAM)



M_AXI_S2MM
Write
Outstanding Writes
1
Channels
ACLK
1
Write Address
Write Data
#1
WDATA[63:0]
c2ef7048c2cf9d7f
WVALID
1
WREADY
1
WLAST
1
Write
1
#1
#1
43bd800043c08000
c2320
c0800
c2110
42a20
c2b60
41a00
c2ef7048c2cf9d7f