HLS_CORDIC_LAB

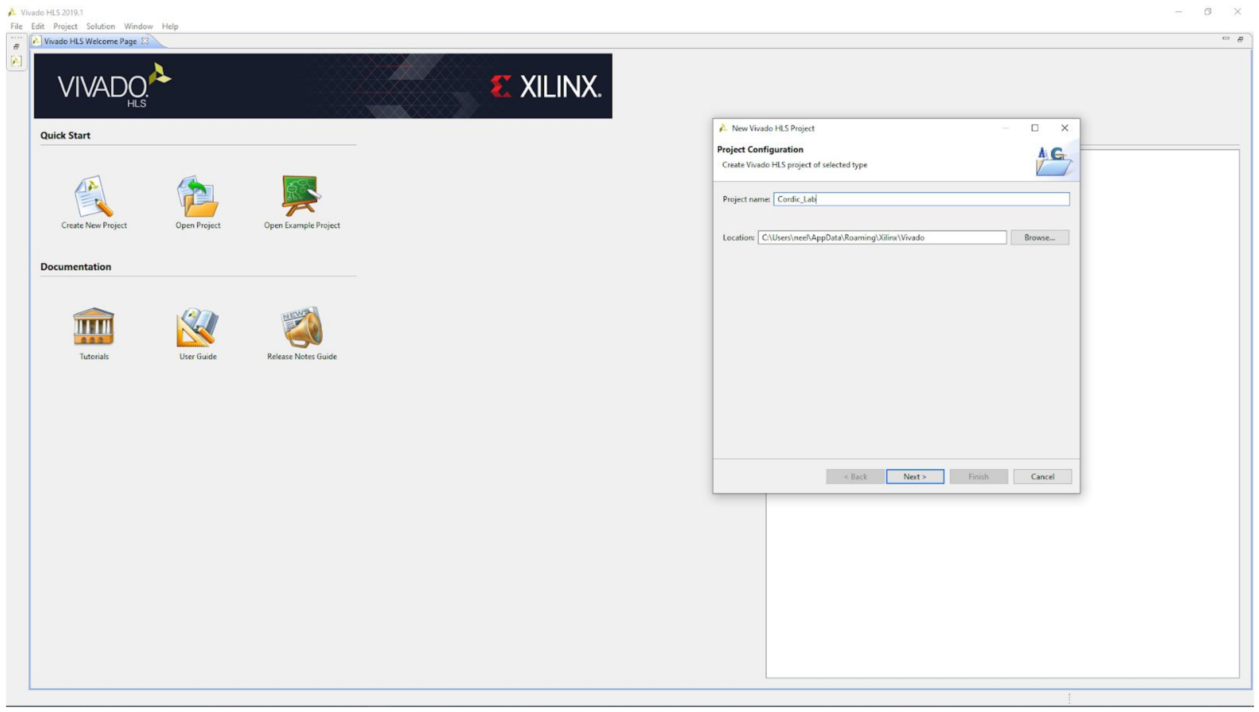
In this lab, we are going to calculate the sin function using the CORDIC algorithm. Then using HLS IP generated we are going to integrate the IP with ARM of Zync.

TASKS TO BE DONE

1) Create a custom AXI-4 Lite HLS IP to calculate the sine function.
   To learn more about the CORDIC algorithm refer to :
   https://drive.google.com/open?id=1JoHJQw4WbgkegzhvXt212qGi_rE8ozzB&authuser=sumit%40iiitd.ac.in&usp=drive_fs
   To learn more about HLS refer to:
   Vivado HLS Course Training
2) Test the functionality using Testbench in Vivado HLS.
3) Create a block design with Zynq PS and the designed IP in Vivado.
4) Compute the Q-Function using PS and using PL and compare the results and timings in Vivado SDK.

Part-1

1) Create a new Vivado HLS project.

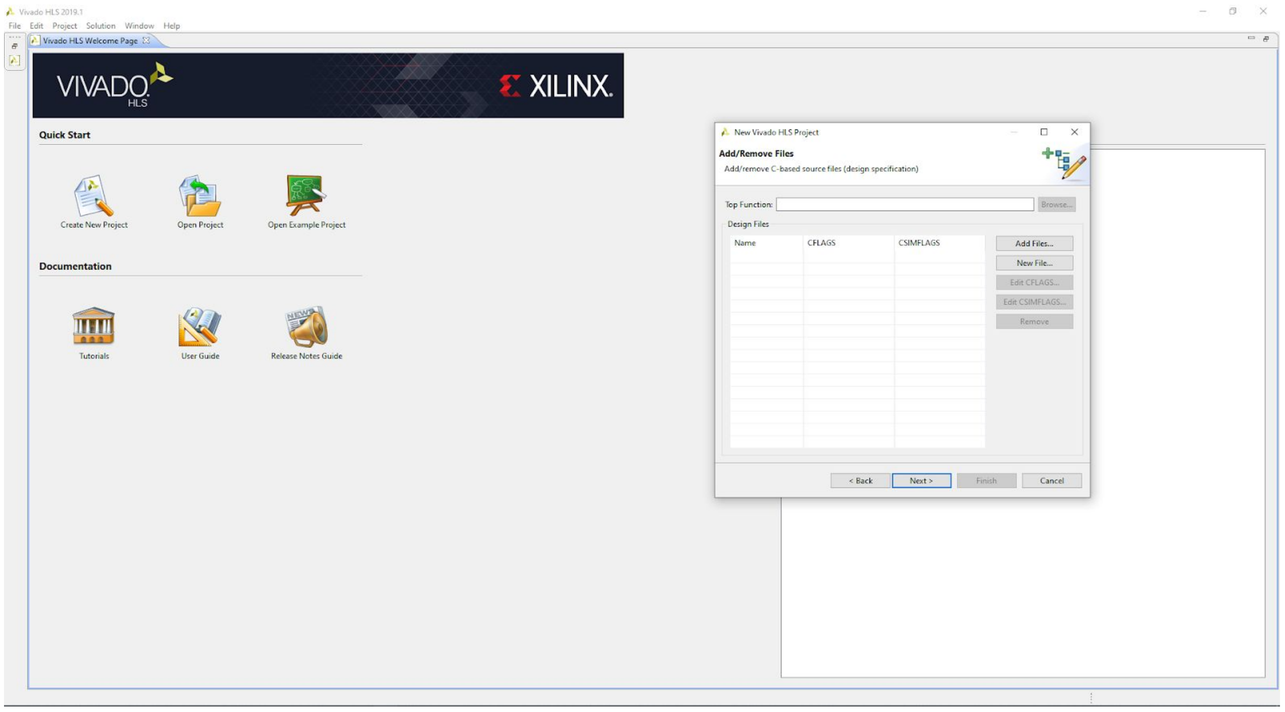


Project Configuration
Create Vivado HLS project of selected type

Project name: Cordic_Lab

Location: C:\Users\neel\AppData\Roaming\Xilinx\Vivado

2) There is no need to specify the Top function, Design files, or the Testbench files now. We can do this later as well.



New Vivado HLS Project

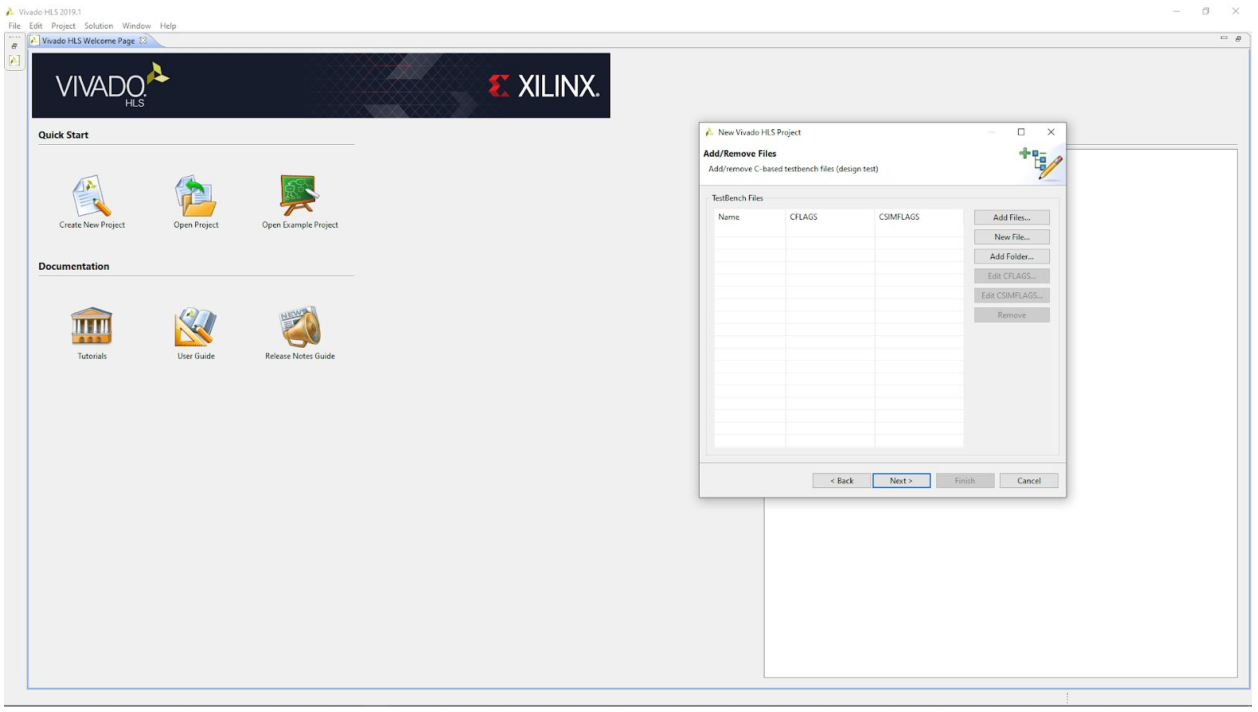
Add/Remove Files

Add/remove C-based source files (design specification)

Top Function:

Design Files

Name	CFLAGS	CSIMFLAGS



New Vivado HLS Project

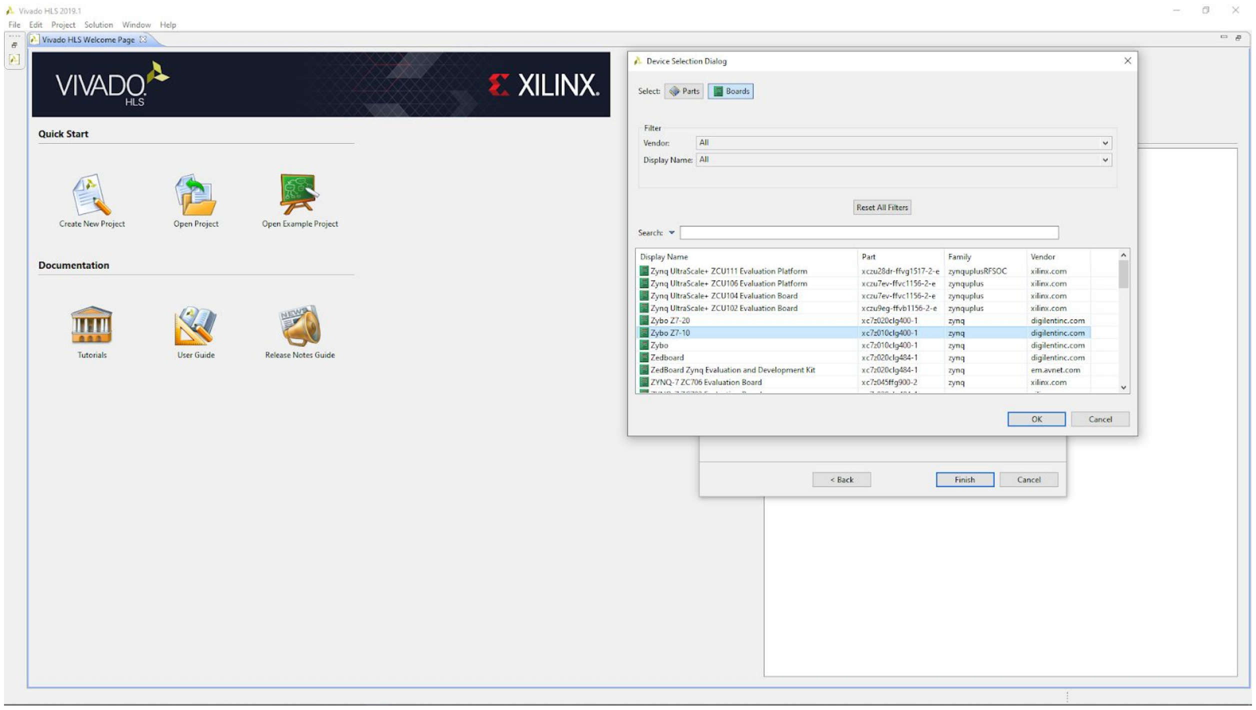
Add/Remove Files

Add/remove C-based testbench files (design test)

TestBench Files

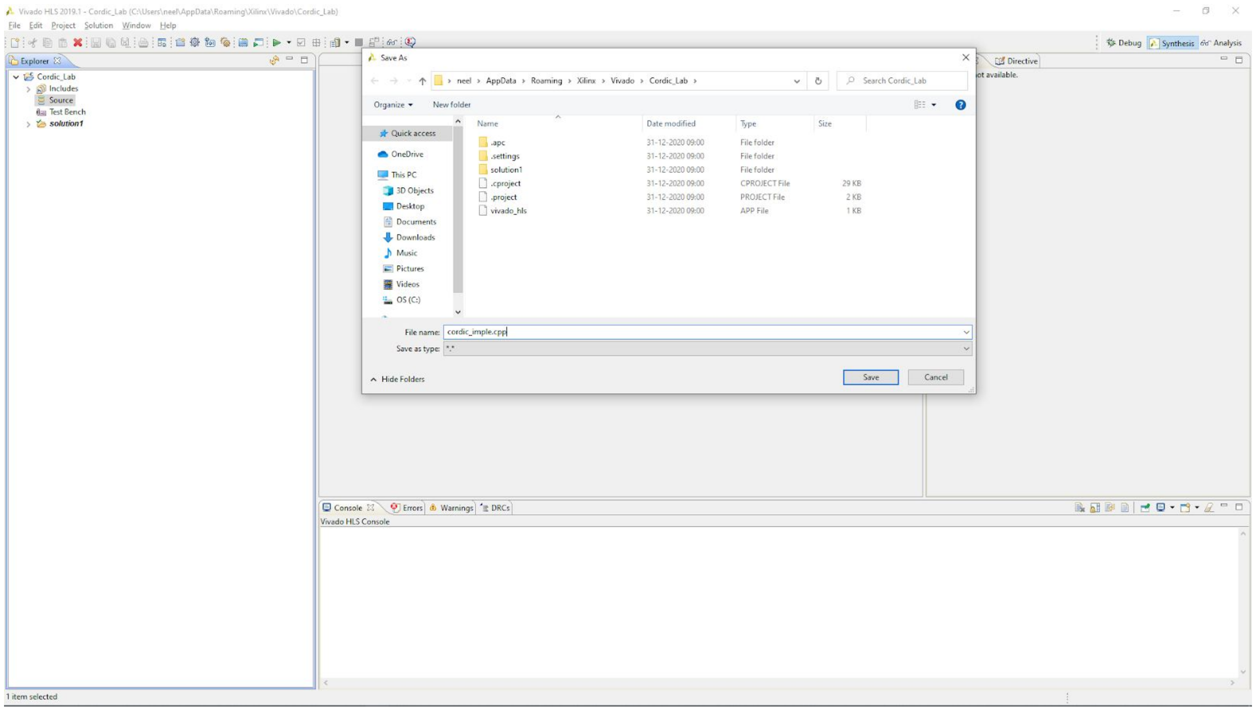
Name	CFLAGS	CSIMFLAGS

3) Select the Zybo Z7-10 board[Used in IIIT labs] or the respective board that you have at your disposal.



Display Name	Part	Family	Vendor
Zynq UltraScale+ ZCU111 Evaluation Platform	xczu28dr-ffvg1517-2-e	zynquplusRFSOC	xilinx.com
Zynq UltraScale+ ZCU106 Evaluation Platform	xczu7ev-ffvc1156-2-e	zynquplus	xilinx.com
Zynq UltraScale+ ZCU104 Evaluation Board	xczu7ev-ffvc1156-2-e	zynquplus	xilinx.com
Zynq UltraScale+ ZCU102 Evaluation Board	xczu9eg-ffvb1156-2-e	zynquplus	xilinx.com
Zybo Z7-20	xc7z020clg400-1	zynq	digilentinc.com
Zybo Z7-10	xc7z010clg400-1	zynq	digilentinc.com
Zybo	xc7z010clg400-1	zynq	digilentinc.com
Zedboard	xc7z020clg484-1	zynq	digilentinc.com
ZedBoard Zynq Evaluation and Development Kit	xc7z020clg484-1	zynq	em.avnet.com
ZYNQ-7 ZC706 Evaluation Board	xc7z045ffg900-2	zynq	xilinx.com

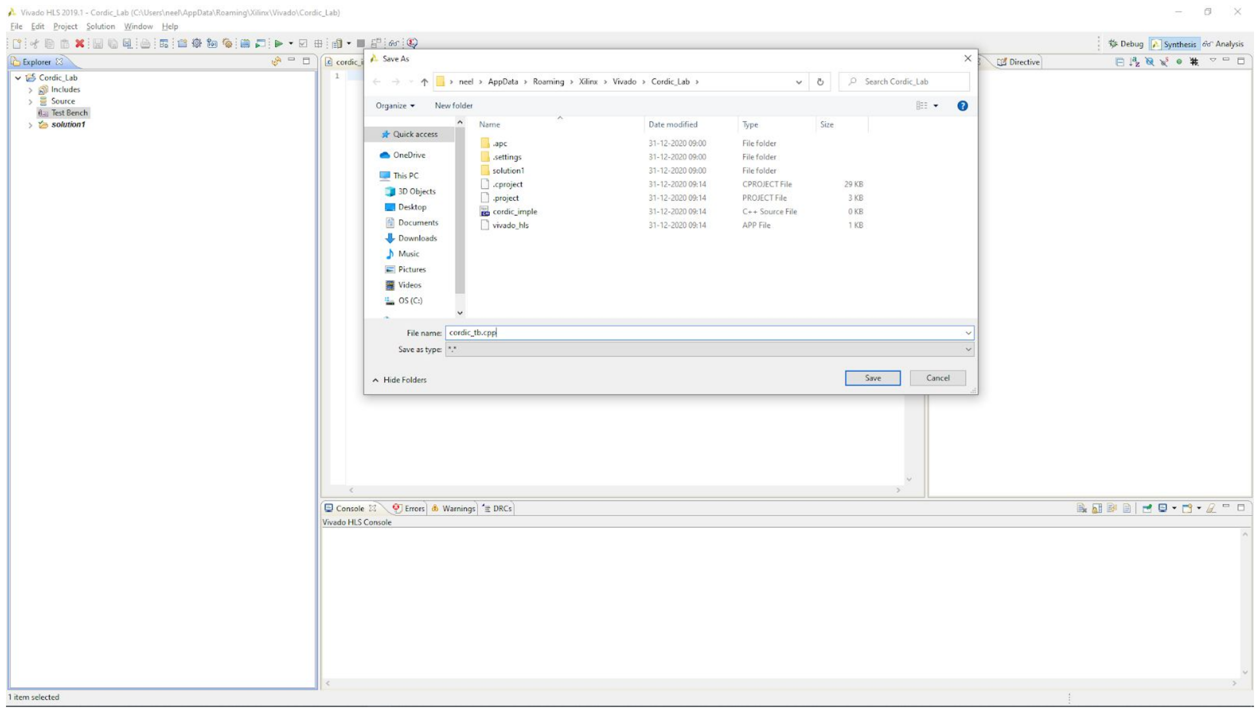
4) Now create a new design file by right-clicking on the source folder.



Name	Date modified	Type	Size
.apc	31-12-2020 09:00	File folder	
.settings	31-12-2020 09:00	File folder	
solution1	31-12-2020 09:00	File folder	
.cproject	31-12-2020 09:00	CPROJECT File	29 KB
.project	31-12-2020 09:00	PROJECT File	2 KB
vivado_hls	31-12-2020 09:00	APP File	1 KB

File name: cordic_imple.cpp
Save as type: *.*

5) Similarly, create a new testbench file by right-clicking o the testbench folder.



Name	Date modified	Type	Size
.apc	31-12-2020 09:00	File folder	
.settings	31-12-2020 09:00	File folder	
solution1	31-12-2020 09:00	File folder	
.cproject	31-12-2020 09:14	CPROJECT File	29 KB
.project	31-12-2020 09:14	PROJECT File	3 KB
cordic_imple	31-12-2020 09:14	C++ Source File	0 KB
vivado_hls	31-12-2020 09:14	APP File	1 KB

File name: cordic_tb.cpp

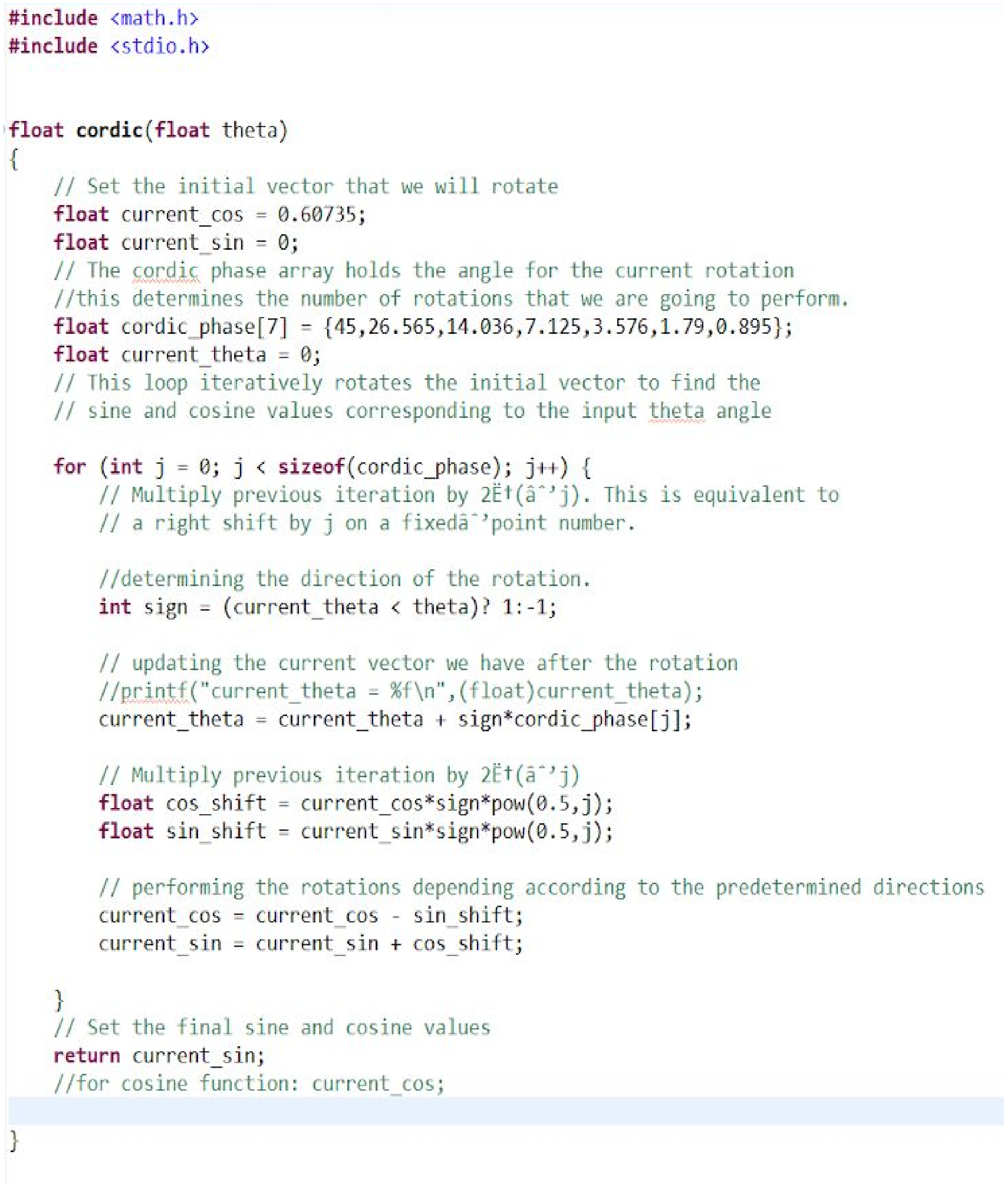
Save as type: *.*

Note: we will define the top-function before synthesis when the verify that our CORDIC implementation is working properly.

6) Now we will write the CORDIC algorithm in our design file. Below given are two codes for calculating the sine function, one with optimization a one without.

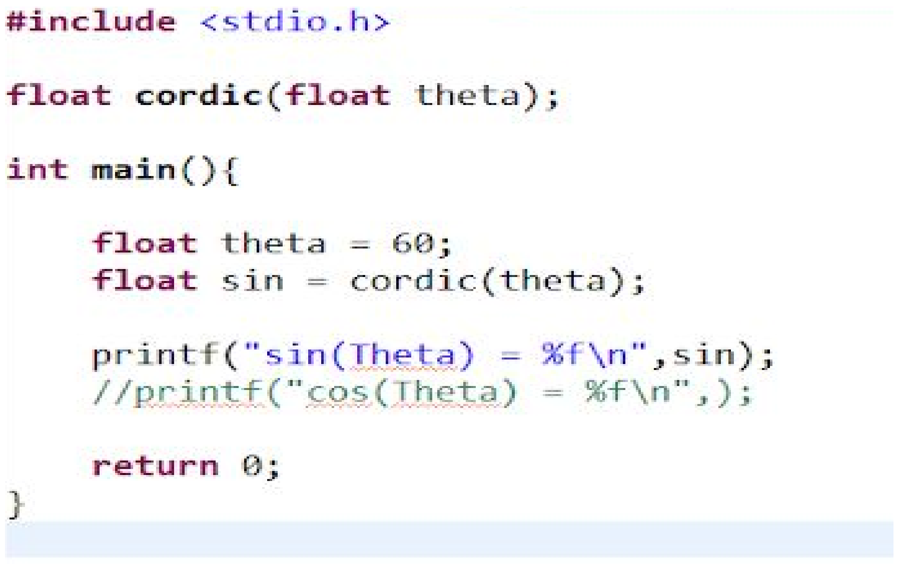
Without optimization

Design file :



```c
#include <math.h>
#include <stdio.h>


float cordic(float theta)
{
    // Set the initial vector that we will rotate
    float current_cos = 0.60735;
    float current_sin = 0;
    // The cordic phase array holds the angle for the current rotation
    //this determines the number of rotations that we are going to perform.
    float cordic_phase[7] = {45,26.565,14.036,7.125,3.576,1.79,0.895};
    float current_theta = 0;
    // This loop iteratively rotates the initial vector to find the
    // sine and cosine values corresponding to the input theta angle

    for (int j = 0; j < sizeof(cordic_phase); j++) {
        // Multiply previous iteration by 2Ë†(âˆ’j). This is equivalent to
        // a right shift by j on a fixedâˆ’point number.

        //determining the direction of the rotation.
        int sign = (current_theta < theta)? 1:-1;

        // updating the current vector we have after the rotation
        //printf("current_theta = %f\n",(float)current_theta);
        current_theta = current_theta + sign*cordic_phase[j];

        // Multiply previous iteration by 2Ë†(âˆ’j)
        float cos_shift = current_cos*sign*pow(0.5,j);
        float sin_shift = current_sin*sign*pow(0.5,j);

        // performing the rotations depending according to the predetermined directions
        current_cos = current_cos - sin_shift;
        current_sin = current_sin + cos_shift;

    }
    // Set the final sine and cosine values
    return current_sin;
    //for cosine function: current_cos;

}
```

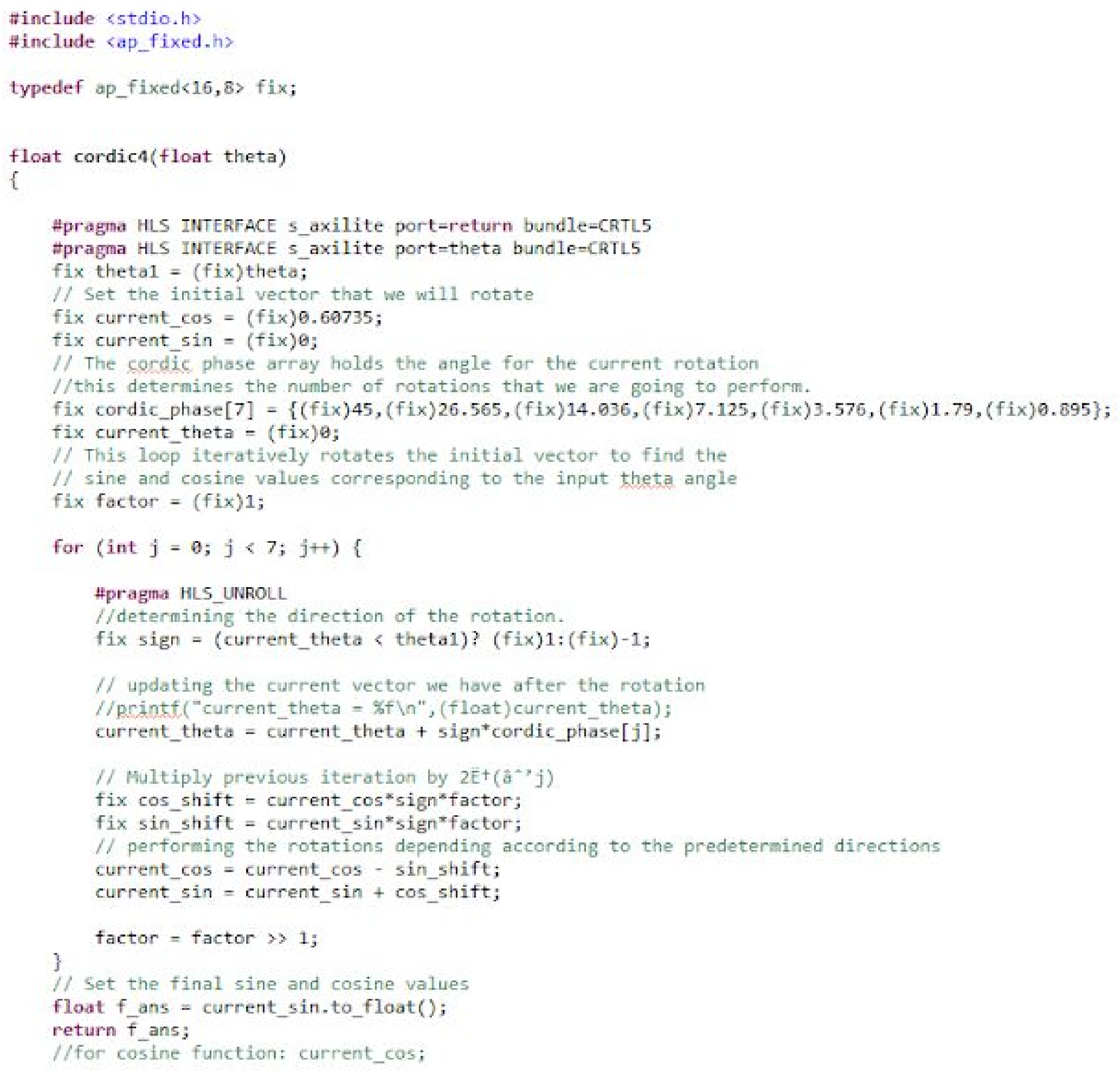
Testbench file :



```
#include <stdio.h>

float cordic(float theta);

int main(){

    float theta = 60;
    float sin = cordic(theta);

    printf("sin(Theta) = %f\n",sin);
    //printf("cos(Theta) = %f\n",);

    return 0;
}
```

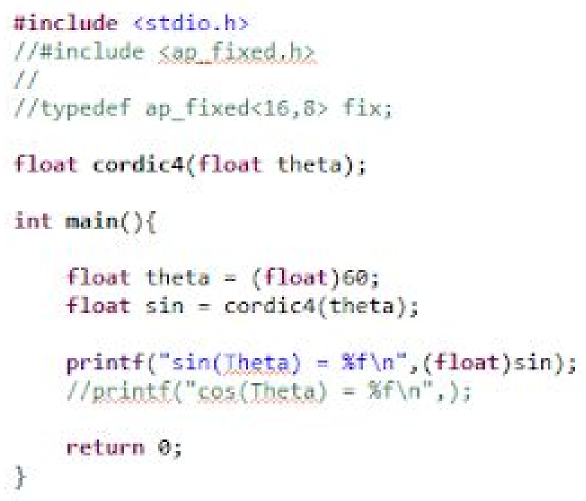
With optimization

Design file :



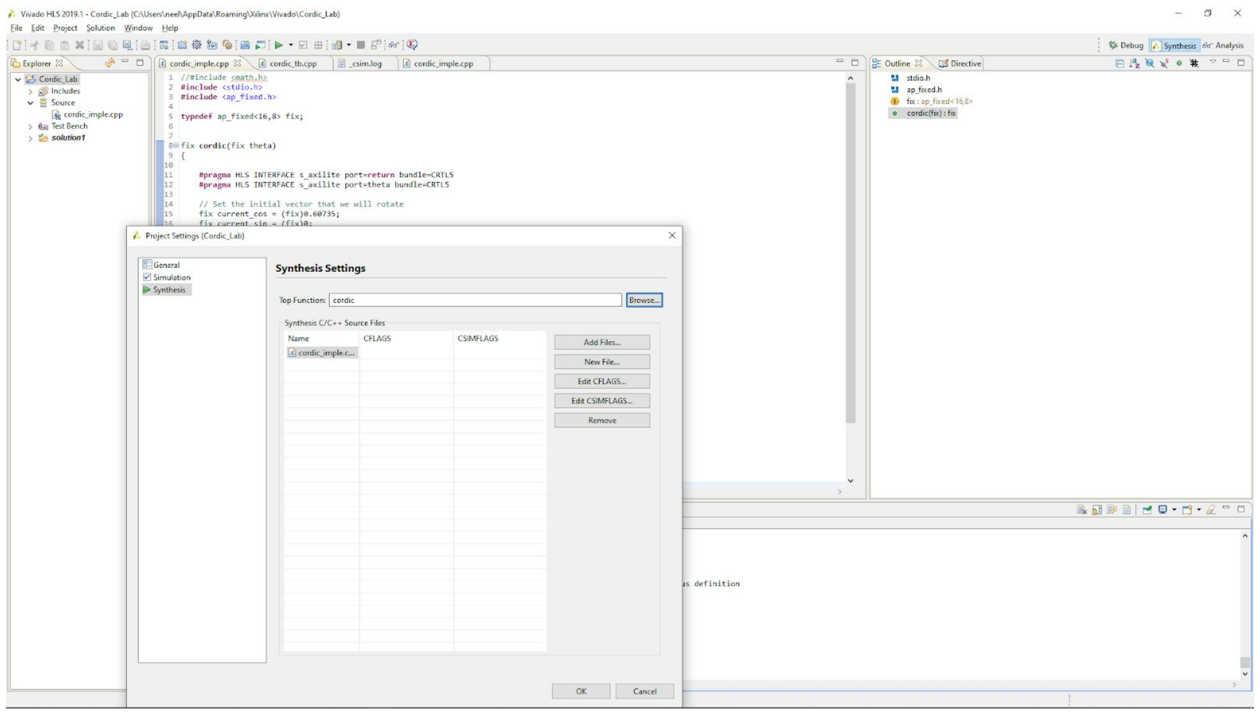
```
#include <stdio.h>
#include <ap_fixed.h>

typedef ap_fixed<16,8> fix;


float cordic4(float theta)
{

    #pragma HLS INTERFACE s_axilite port=return bundle=CRTL5
    #pragma HLS INTERFACE s_axilite port=theta bundle=CRTL5
    fix theta1 = (fix)theta;
    // Set the initial vector that we will rotate
    fix current_cos = (fix)0.60735;
    fix current_sin = (fix)0;
    // The cordic phase array holds the angle for the current rotation
    //this determines the number of rotations that we are going to perform.
    fix cordic_phase[7] = {(fix)45,(fix)26.565,(fix)14.036,(fix)7.125,(fix)3.576,(fix)1.79,(fix)0.895};
    fix current_theta = (fix)0;
    // This loop iteratively rotates the initial vector to find the
    // sine and cosine values corresponding to the input theta angle
    fix factor = (fix)1;

    for (int j = 0; j < 7; j++) {

        #pragma HLS_UNROLL
        //determining the direction of the rotation.
        fix sign = (current_theta < theta1)? (fix)1:(fix)-1;

        // updating the current vector we have after the rotation
        //printf("current_theta = %f\n",(float)current_theta);
        current_theta = current_theta + sign*cordic_phase[j];

        // Multiply previous iteration by 2Ë†(â^'j)
        fix cos_shift = current_cos*sign*factor;
        fix sin_shift = current_sin*sign*factor;
        // performing the rotations depending according to the predetermined directions
        current_cos = current_cos - sin_shift;
        current_sin = current_sin + cos_shift;

        factor = factor >> 1;
    }
    // Set the final sine and cosine values
    float f_ans = current_sin.to_float();
    return f_ans;
    //for cosine function: current_cos;
```

Testbench file:



```
#include <stdio.h>
//#include <ap_fixed.h>
//
//typedef ap_fixed<16,8> fix;

float cordic4(float theta);

int main(){

    float theta = (float)60;
    float sin = cordic4(theta);

    printf("sin(Theta) = %f\n",(float)sin);
    //printf("cos(Theta) = %f\n",);

    return 0;
}
```

7) Now go to project>>project settings>>synthesis and name your top_function. You should write the name of the function that you want to be synthesized and in this case, it is “cordic”.



```
//#include <math.h>
#include <stdio.h>
#include <ap_fixed.h>

typedef ap_fixed<16,8> fix;


fix cordic(fix theta)
{

    #pragma HLS INTERFACE s_axilite port=return bundle=CRTLS
    #pragma HLS INTERFACE s_axilite port=theta bundle=CRTLS

    // Set the initial vector that we will rotate
    fix current_cos = (fix)0.60735;
    fix current_sin = (fix)0;
```

Synthesis Settings

Top Function: cordic

Synthesis C/C++ Source Files

Name	CFLAGS	CSIMFLAGS
cordic_imple.c...		

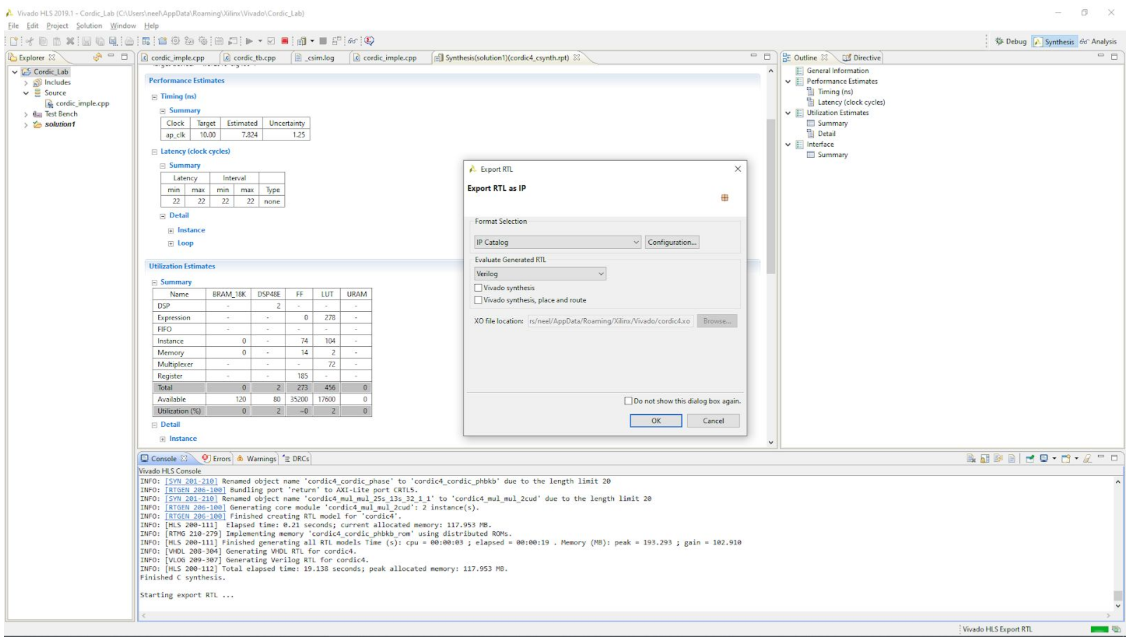
8) Synthesize your code and note the latency and Hardware utilization of your code.

NOTE: try and synthesize both un-optimized code and optimized code to check for the difference in latency and utilization estimates, you should see a massive improvement from our optimized code.

PART-2

NOTE: from here we will only be working with the optimized code.

9) Click on export RTL as IP and export the project.


Vivado HLS 2019.1 - Cordic_Lab (C:\Users\neel\AppData\Roaming\Xilinx\Vivado\Cordic_Lab)
Explorer
Outline
Directive
Debug
Synthesis
Analysis
Console
Errors
Warnings
DRCs

Synthesis(solution1)(cordic4_csynth.rpt)

Performance Estimates

Timing (ns) — Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.824	1.25

Latency (clock cycles) — Summary

Latency		Interval		
min	max	min	max	Type
22	22	22	22	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	2	-	-	-
Expression	-	-	0	278	-
FIFO	-	-	-	-	-
Instance	0	-	74	104	-
Memory	0	-	14	2	-
Multiplexer	-	-	-	72	-
Register	-	-	185	-	-
Total	0	2	273	456	0
Available	120	80	35200	17600	0
Utilization (%)	0	2	~0	2	0

Export RTL

Export RTL as IP

Format Selection: IP Catalog

Evaluate Generated RTL: Verilog

☐ Vivado synthesis

☐ Vivado synthesis, place and route

XO file location: rs/neel/AppData/Roaming/Xilinx/Vivado/cordic4.xo

☐ Do not show this dialog box again.

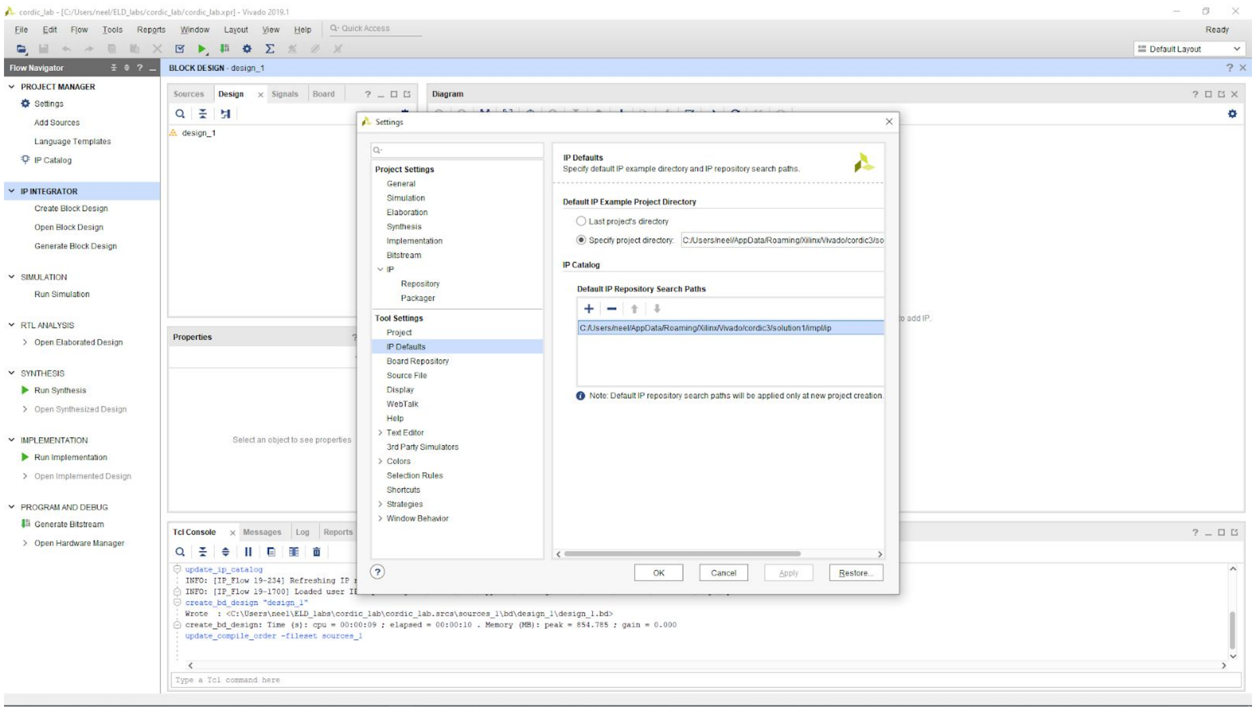
Vivado HLS Console

```
INFO: [SYN 201-210] Renamed object name 'cordic4_cordic_phase' to 'cordic4_cordic_phbkb' due to the length limit 20
INFO: [RTGEN 206-100] Bundling port 'return' to AXI-Lite port CRTLS.
INFO: [SYN 201-210] Renamed object name 'cordic4_mul_mul_25s_13s_32_1_1' to 'cordic4_mul_mul_2cud' due to the length limit 20
INFO: [RTGEN 206-100] Generating core module 'cordic4_mul_mul_2cud': 2 instance(s).
INFO: [RTGEN 206-100] Finished creating RTL model for 'cordic4'.
INFO: [HLS 200-111]  Elapsed time: 0.21 seconds; current allocated memory: 117.953 MB.
INFO: [RTMG 210-279] Implementing memory 'cordic4_cordic_phbkb_rom' using distributed ROMs.
INFO: [HLS 200-111] Finished generating all RTL models Time (s): cpu = 00:00:03 ; elapsed = 00:00:19 . Memory (MB): peak = 193.293 ; gain = 102.910
INFO: [VHDL 208-304] Generating VHDL RTL for cordic4.
INFO: [VLOG 209-307] Generating Verilog RTL for cordic4.
INFO: [HLS 200-112] Total elapsed time: 19.138 seconds; peak allocated memory: 117.953 MB.
Finished C synthesis.

Starting export RTL ...
```

10) **Open vivado** and create a new project. Don't specify the Design sources and select Zybo Z7-10 or the board that you have at your disposal.

11) Click on **create block design.**

12) We need to specify the path from where vivado is going to receive the exported RTL. For that go to settings under a project manager.



IP Defaults

Specify default IP example directory and IP repository search paths.

Default IP Example Project Directory

○ Last project's directory

● Specify project directory: C:/Users/neel/AppData/Roaming/Xilinx/Vivado/cordic3/so

IP Catalog

Default IP Repository Search Paths

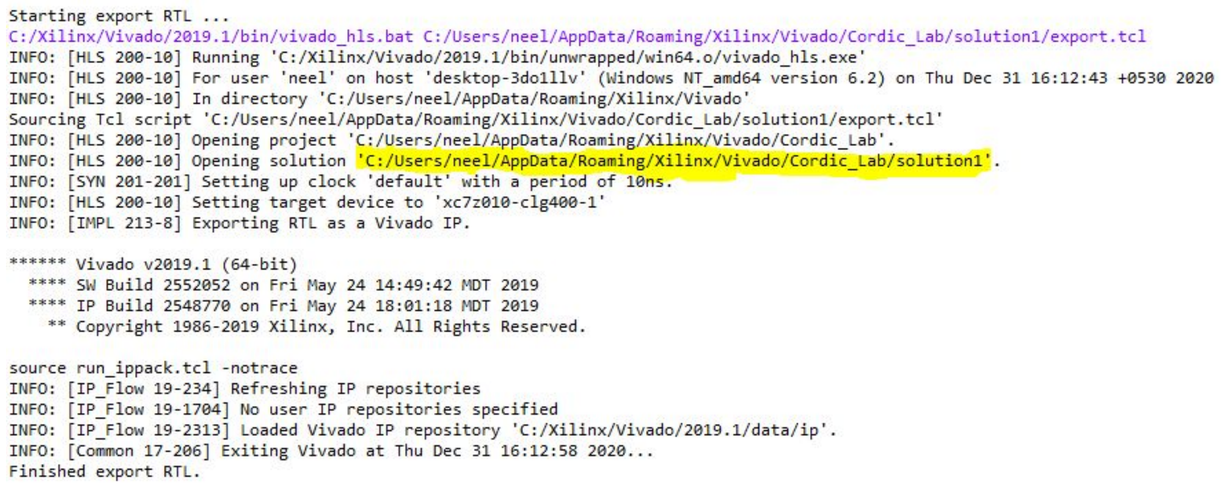
C:/Users/neel/AppData/Roaming/Xilinx/Vivado/cordic3/solution1/impl/ip

Note: Default IP repository search paths will be applied only at new project creation.

```
update_ip_catalog
INFO: [IP_Flow 19-234] Refreshing IP r
INFO: [IP_Flow 19-1700] Loaded user IE
create_bd_design "design_1"
Wrote  : <C:\Users\neel\ELD_labs\cordic_lab\cordic_lab.srcs\sources_1\bd\design_1\design_1.bd>
create_bd_design: Time (s): cpu = 00:00:09 ; elapsed = 00:00:10 . Memory (MB): peak = 854.785 ; gain = 0.000
update_compile_order -fileset sources_1
```

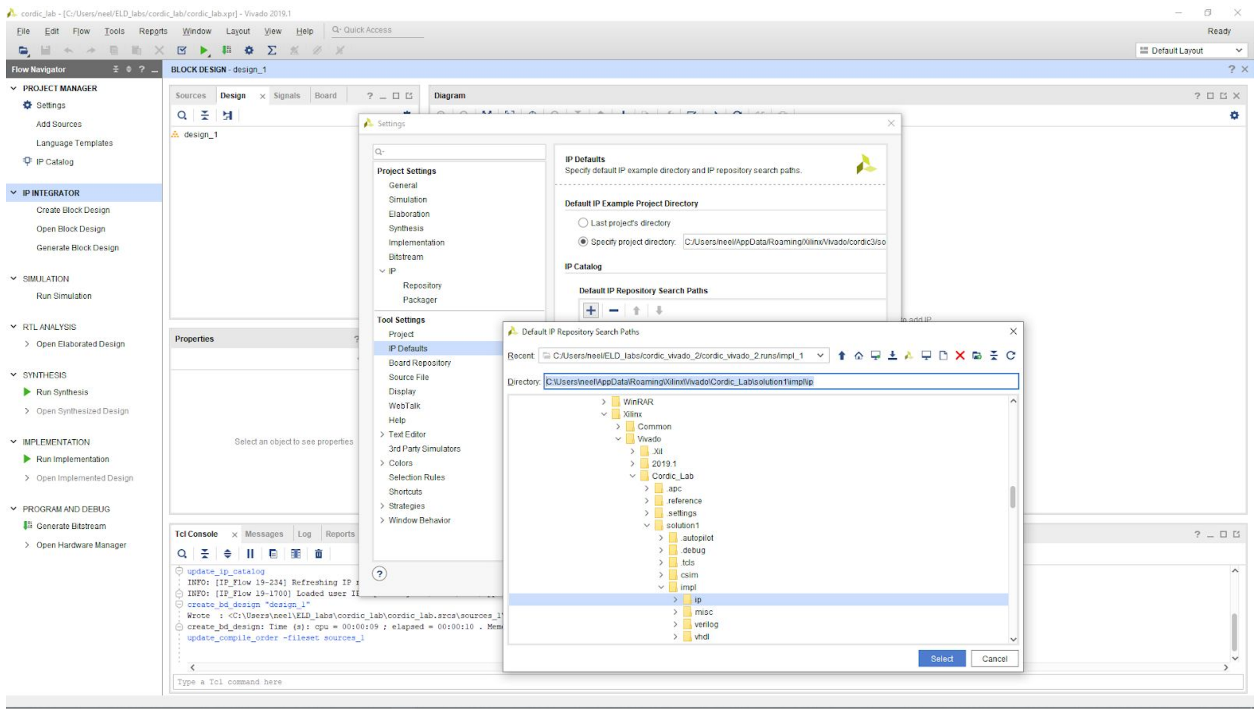
14) add a default IP search path.

You can find this path in vivado HLS after you exported the project as an IP.



```
Starting export RTL ...
C:/Xilinx/Vivado/2019.1/bin/vivado_hls.bat C:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab/solution1/export.tcl
INFO: [HLS 200-10] Running 'C:/Xilinx/Vivado/2019.1/bin/unwrapped/win64.o/vivado_hls.exe'
INFO: [HLS 200-10] For user 'neel' on host 'desktop-3do1llv' (Windows NT_amd64 version 6.2) on Thu Dec 31 16:12:43 +0530 2020
INFO: [HLS 200-10] In directory 'C:/Users/neel/AppData/Roaming/Xilinx/Vivado'
Sourcing Tcl script 'C:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab/solution1/export.tcl'
INFO: [HLS 200-10] Opening project 'C:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab'.
INFO: [HLS 200-10] Opening solution 'C:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-10] Setting target device to 'xc7z010-clg400-1'
INFO: [IMPL 213-8] Exporting RTL as a Vivado IP.

****** Vivado v2019.1 (64-bit)
  **** SW Build 2552052 on Fri May 24 14:49:42 MDT 2019
  **** IP Build 2548770 on Fri May 24 18:01:18 MDT 2019
    ** Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

source run_ippack.tcl -notrace
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2019.1/data/ip'.
INFO: [Common 17-206] Exiting Vivado at Thu Dec 31 16:12:58 2020...
Finished export RTL.
```

While defining the path you should make sure that you go into impl1>>ip to explicitly tell where the IP is located.



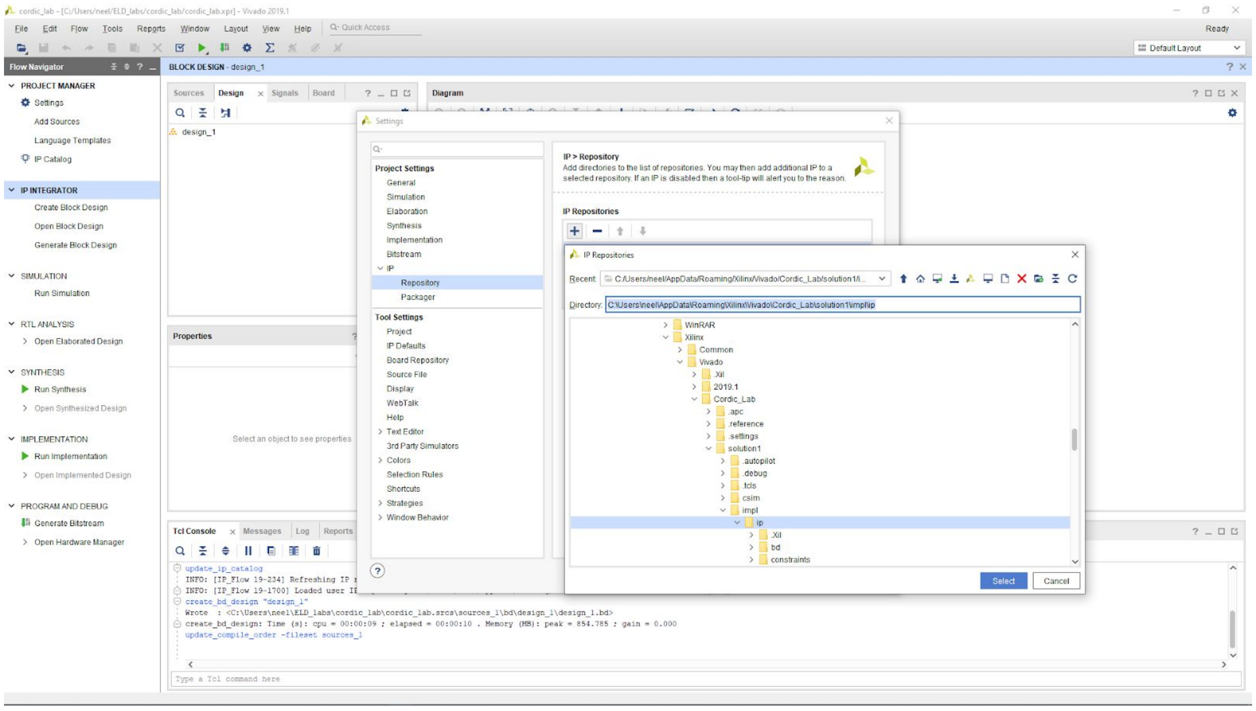
IP Defaults
Specify default IP example directory and IP repository search paths.

Default IP Example Project Directory
( ) Last project's directory
(●) Specify project directory: C:/Users/neel/AppData/Roaming/Xilinx/Vivado/cordic3/so

IP Catalog
Default IP Repository Search Paths

Default IP Repository Search Paths
Recent: C:/Users/neel/ELD_labs/cordic_vivado_2/cordic_vivado_2.runs/impl_1
Directory: C:\Users\neel\AppData\Roaming\Xilinx\Vivado\Cordic_Lab\solution1\impl\ip

15) now click on IP >> Repository and do the same thing.



IP > Repository

Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason.

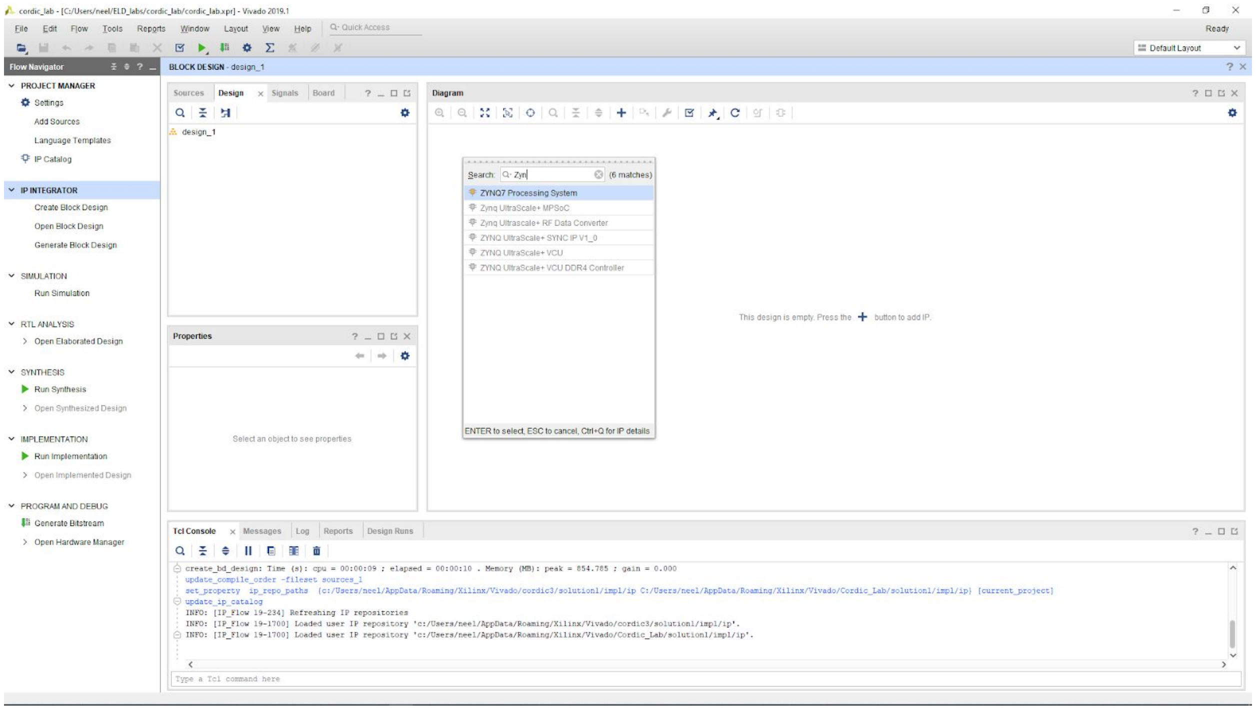
Recent: C:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab/solution1/i...

Directory: C:\Users\neel\AppData\Roaming\Xilinx\Vivado\Cordic_Lab\solution1\impl\ip

```
update_ip_catalog
INFO: [IP_Flow 19-234] Refreshing IP r
INFO: [IP_Flow 19-1700] Loaded user IP
create_bd_design "design_1"
Wrote  : <C:\Users\neel\ELD_labs\cordic_lab\cordic_lab.srcs\sources_1\bd\design_1\design_1.bd>
create_bd_design: Time (s): cpu = 00:00:09 ; elapsed = 00:00:10 . Memory (MB): peak = 854.785 ; gain = 0.000
update_compile_order -fileset sources_1
```

16) click on apply

17) Now add the Zync7 processing system to the block design.



Search: Zyn (6 matches)

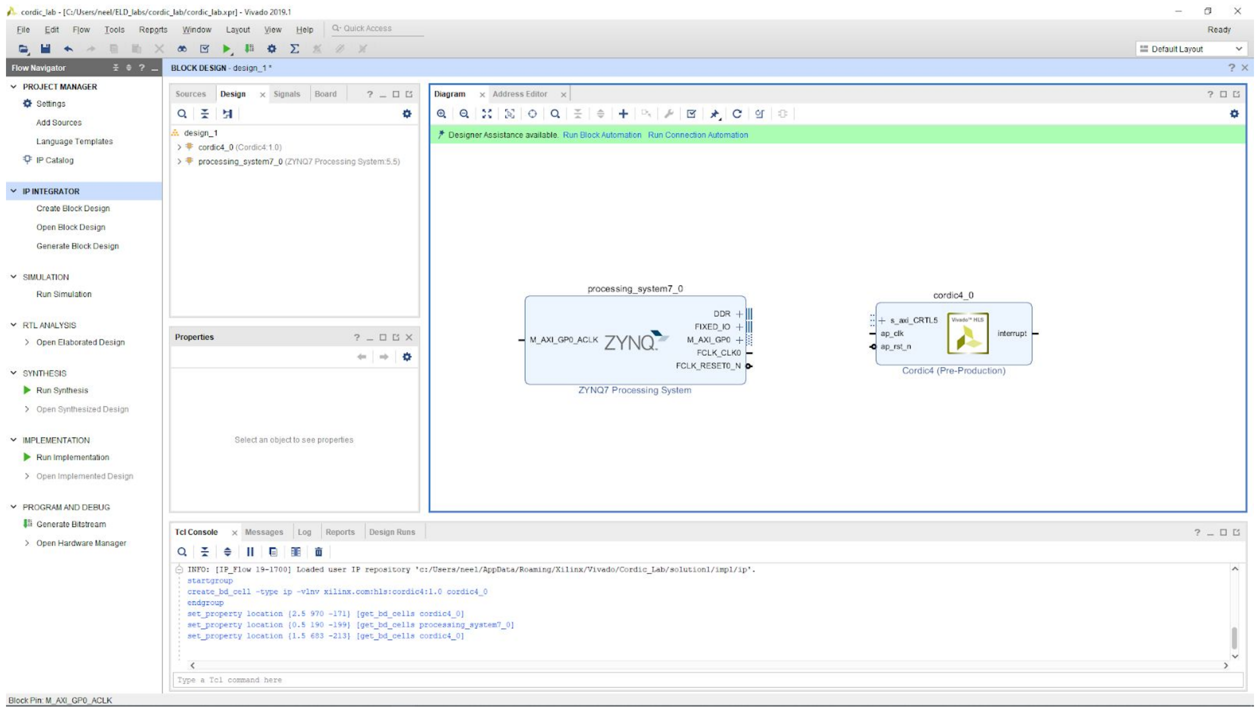
- ZYNQ7 Processing System
- Zynq UltraScale+ MPSoC
- Zynq Ultrascale+ RF Data Converter
- ZYNQ UltraScale+ SYNC IP V1_0
- ZYNQ UltraScale+ VCU
- ZYNQ UltraScale+ VCU DDR4 Controller

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

```
create_bd_design: Time (s): cpu = 00:00:09 ; elapsed = 00:00:10 . Memory (MB): peak = 854.785 ; gain = 0.000
update_compile_order -fileset sources_1
set_property  ip_repo_paths  {c:/Users/neel/AppData/Roaming/Xilinx/Vivado/cordic3/solution1/impl/ip C:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab/solution1/impl/ip} [current_project]
update_ip_catalog
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/Users/neel/AppData/Roaming/Xilinx/Vivado/cordic3/solution1/impl/ip'.
INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab/solution1/impl/ip'.
```

18) similarly search for HLS IP that we generated in the same place and add it to the design block. The name of the IP should be the name of the top_function that we synthesized. Mine is **cordic4** yours might differ.

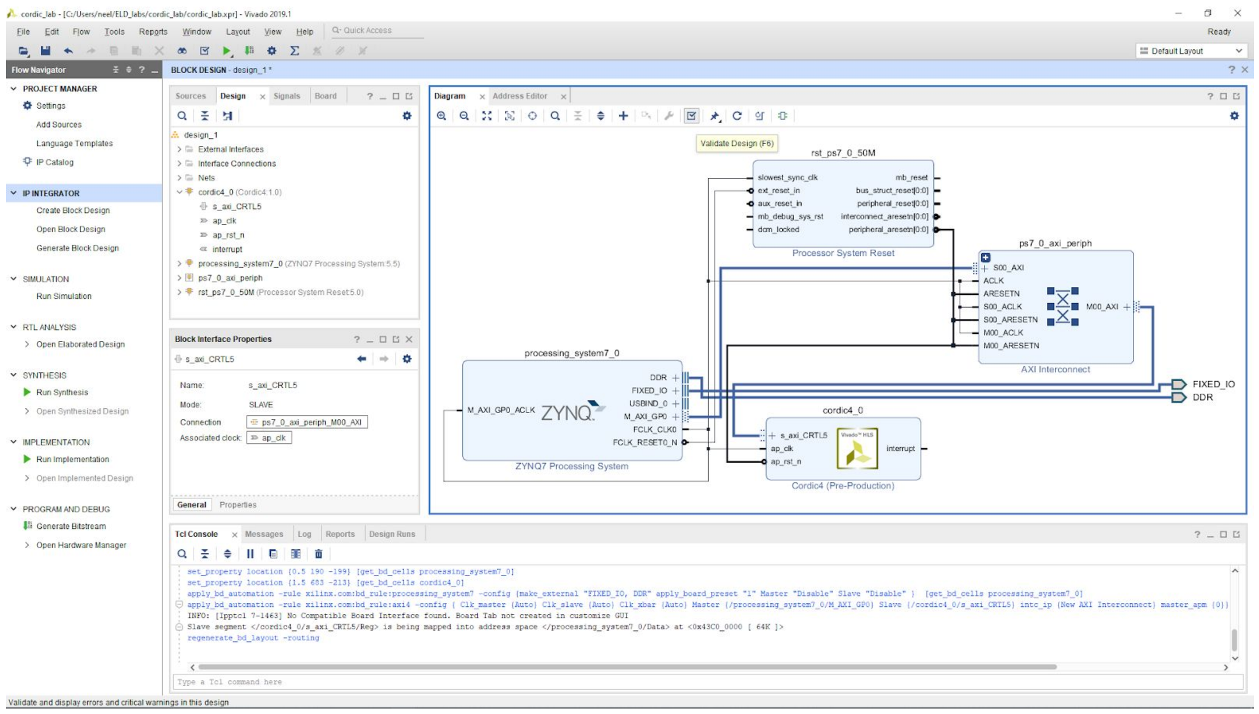
Finally, your block design should look like this.


cordic_lab - [C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.xpr] - Vivado 2019.1
BLOCK DESIGN - design_1 *
design_1
cordic4_0 (Cordic4:1.0)
processing_system7_0 (ZYNQ7 Processing System:5.5)
Designer Assistance available. Run Block Automation Run Connection Automation
processing_system7_0
M_AXI_GP0_ACLK
DDR
FIXED_IO
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
cordic4_0
s_axi_CRTLS
ap_clk
ap_rst_n
interrupt
Cordic4 (Pre-Production)
Block Pin: M_AXI_GP0_ACLK

```
INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/Users/neel/AppData/Roaming/Xilinx/Vivado/Cordic_Lab/solution1/impl/ip'.
startgroup
create_bd_cell -type ip -vlnv xilinx.com:hls:cordic4:1.0 cordic4_0
endgroup
set_property location {2.5 970 -171} [get_bd_cells cordic4_0]
set_property location {0.5 190 -199} [get_bd_cells processing_system7_0]
set_property location {1.5 683 -213} [get_bd_cells cordic4_0]
```

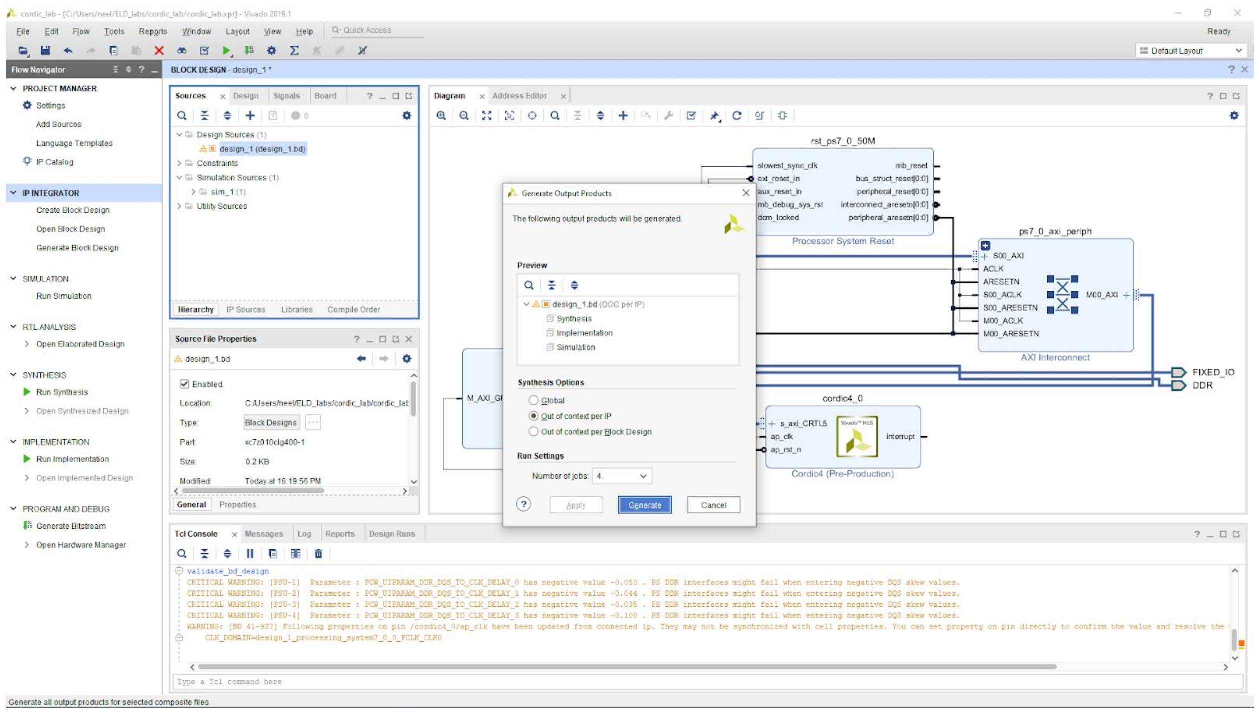
19) now click on **Run block automation** and **Run connection automation**. then validate the design.

Your block design should finally look as following:

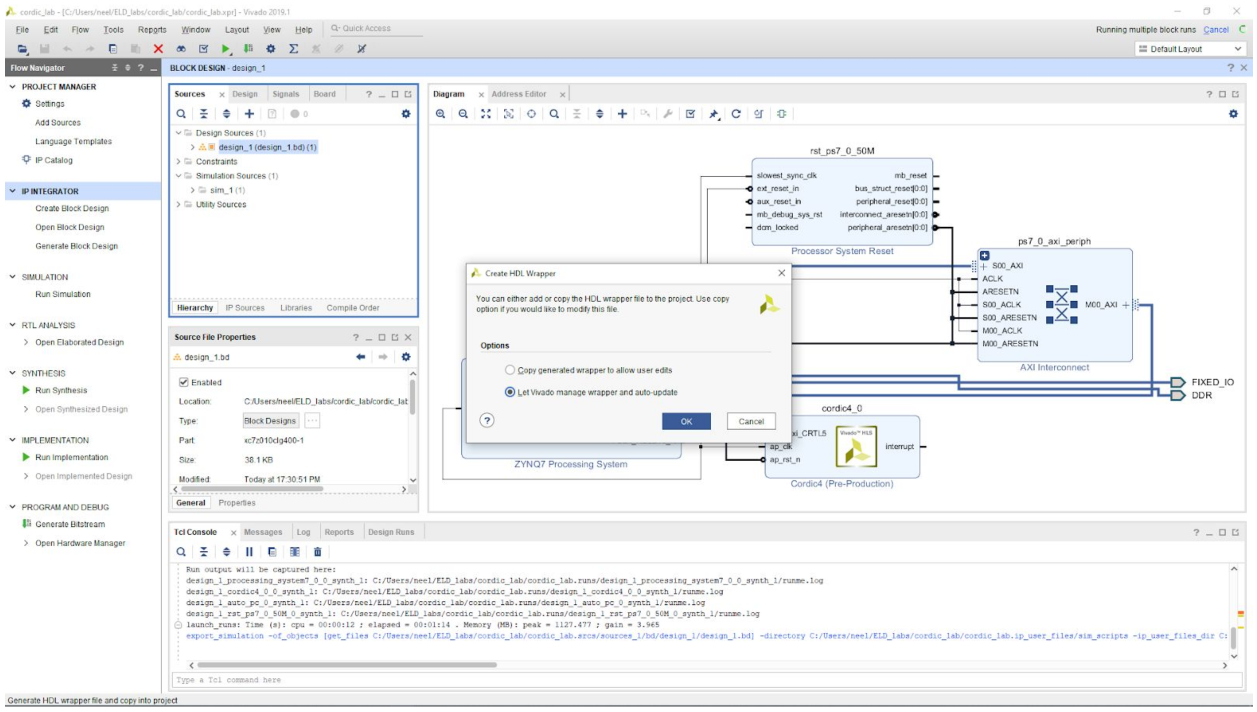


```
set_property location {0.5 190 -199} [get_bd_cells processing_system7_0]
set_property location {1.5 683 -213} [get_bd_cells cordic4_0]
apply_bd_automation -rule xilinx.com:bd_rule:processing_system7 -config {make_external "FIXED_IO, DDR" apply_board_preset "1" Master "Disable" Slave "Disable" }  [get_bd_cells processing_system7_0]
apply_bd_automation -rule xilinx.com:bd_rule:axi4 -config { Clk_master {Auto} Clk_slave {Auto} Clk_xbar {Auto} Master {/processing_system7_0/M_AXI_GP0} Slave {/cordic4_0/s_axi_CRTL5} intc_ip {New AXI Interconnect} master_apm {0}}
INFO: [Ipptcl 7-1463] No Compatible Board Interface found. Board Tab not created in customize GUI
Slave segment </cordic4_0/s_axi_CRTL5/Reg> is being mapped into address space </processing_system7_0/Data> at <0x43C0_0000 [ 64K ]>
regenerate_bd_layout -routing
```

20) after validation go-to sources and right-click on design-1 to **Generate output product.** After which click on **create HDL wrapper**



```
validate_bd_design
CRITICAL WARNING: [PSU-1]  Parameter : PCW_UIPARAM_DDR_DQS_TO_CLK_DELAY_0 has negative value -0.050 . PS DDR interfaces might fail when entering negative DQS skew values.
CRITICAL WARNING: [PSU-2]  Parameter : PCW_UIPARAM_DDR_DQS_TO_CLK_DELAY_1 has negative value -0.044 . PS DDR interfaces might fail when entering negative DQS skew values.
CRITICAL WARNING: [PSU-3]  Parameter : PCW_UIPARAM_DDR_DQS_TO_CLK_DELAY_2 has negative value -0.035 . PS DDR interfaces might fail when entering negative DQS skew values.
CRITICAL WARNING: [PSU-4]  Parameter : PCW_UIPARAM_DDR_DQS_TO_CLK_DELAY_3 has negative value -0.100 . PS DDR interfaces might fail when entering negative DQS skew values.
WARNING: [BD 41-927] Following properties on pin /cordic4_0/ap_clk have been updated from connected ip. They may not be synchronized with cell properties. You can set property on pin directly to confirm the value and resolve the
    CLK_DOMAIN=design_1_processing_system7_0_0_FCLK_CLK0
```



Create HDL Wrapper

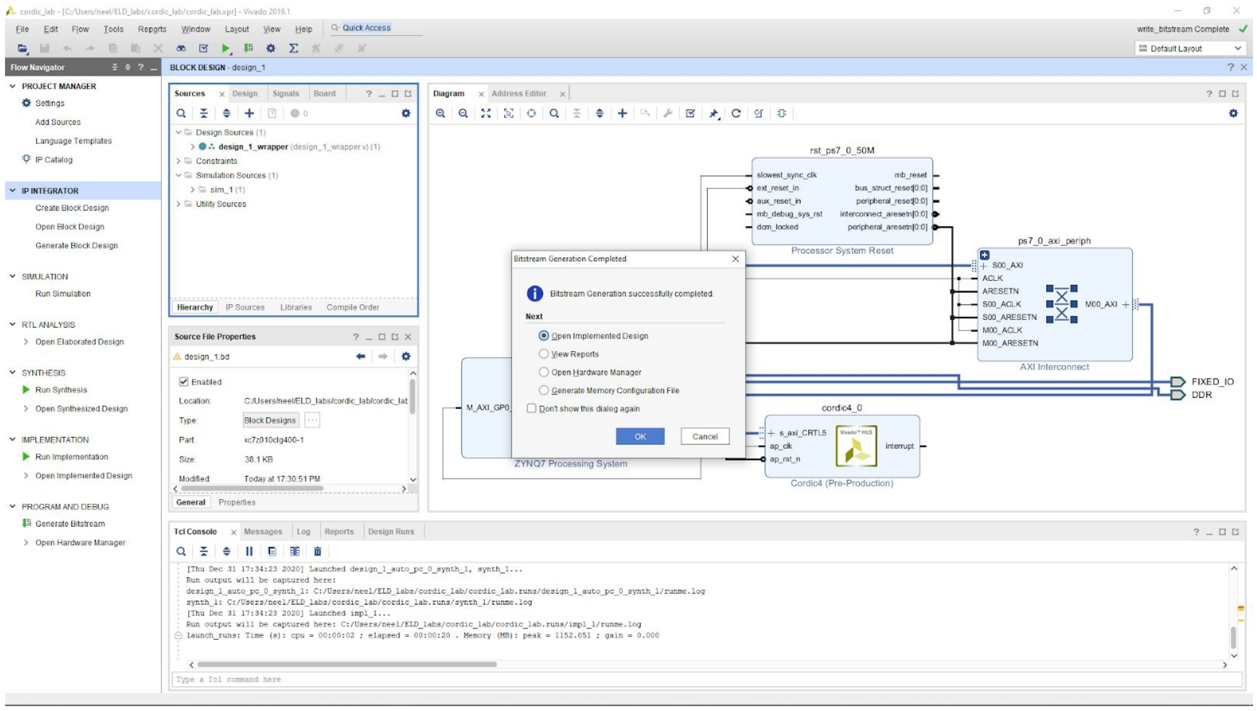
You can either add or copy the HDL wrapper file to the project. Use copy option if you would like to modify this file.

Options

( ) Copy generated wrapper to allow user edits
(●) Let Vivado manage wrapper and auto-update

```
Run output will be captured here:
design_1_processing_system7_0_0_synth_1: C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.runs/design_1_processing_system7_0_0_synth_1/runme.log
design_1_cordic4_0_0_synth_1: C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.runs/design_1_cordic4_0_0_synth_1/runme.log
design_1_auto_pc_0_synth_1: C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.runs/design_1_auto_pc_0_synth_1/runme.log
design_1_rst_ps7_0_50M_0_synth_1: C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.runs/design_1_rst_ps7_0_50M_0_synth_1/runme.log
launch_runs: Time (s): cpu = 00:00:12 ; elapsed = 00:01:14 . Memory (MB): peak = 1127.477 ; gain = 3.965
export_simulation -of_objects [get_files C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.srcs/sources_1/bd/design_1/design_1.bd] -directory C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.ip_user_files/sim_scripts -ip_user_files_dir C:
```

21) Now click on **generate bitstream.** After generation click **ok**



Bitstream Generation Completed

Bitstream Generation successfully completed.

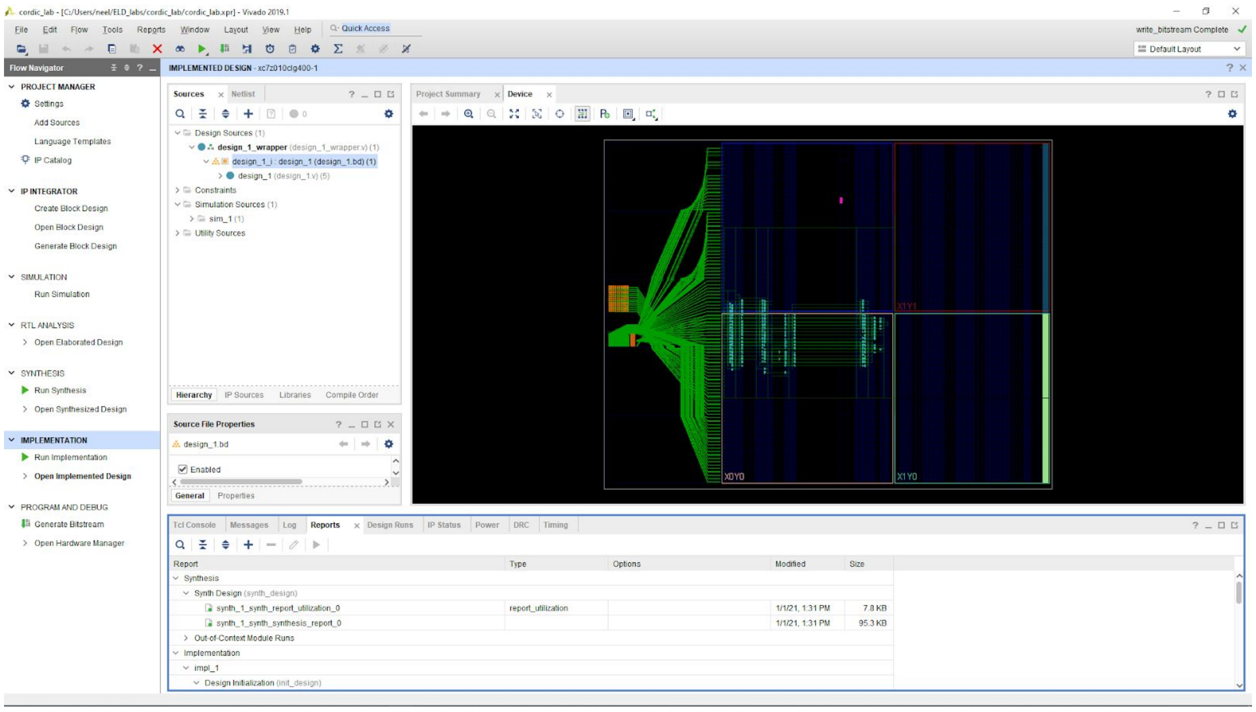
Next

(●) Open Implemented Design
( ) View Reports
( ) Open Hardware Manager
( ) Generate Memory Configuration File

[ ] Don't show this dialog again

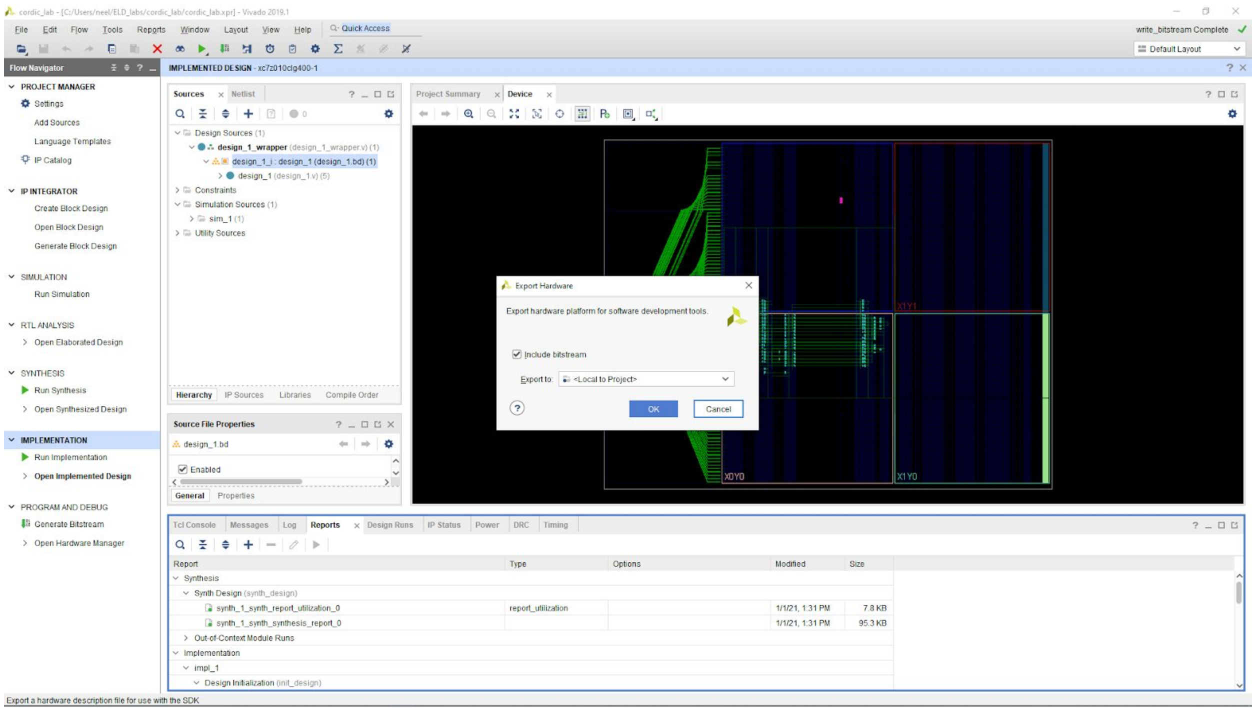
```
[Thu Dec 31 17:34:23 2020] Launched design_1_auto_pc_0_synth_1, synth_1...
Run output will be captured here:
design_1_auto_pc_0_synth_1: C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.runs/design_1_auto_pc_0_synth_1/runme.log
synth_1: C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.runs/synth_1/runme.log
[Thu Dec 31 17:34:23 2020] Launched impl_1...
Run output will be captured here: C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.runs/impl_1/runme.log
launch_runs: Time (s): cpu = 00:00:02 ; elapsed = 00:00:20 . Memory (MB): peak = 1152.051 ; gain = 0.000
```

Your device should look like this:



Report	Type	Options	Modified	Size
Synthesis				
Synth Design (synth_design)				
synth_1_synth_report_utilization_0	report_utilization		1/1/21, 1:31 PM	7.8 KB
synth_1_synth_synthesis_report_0			1/1/21, 1:31 PM	95.3 KB
Out-of-Context Module Runs				
Implementation				
impl_1				
Design Initialization (init_design)				

22) next go to **File>>export>>export hardware** and export the hardware, make sure you click on **include bitstream** to include the bitstream file.


cordic_lab - [C:/Users/neel/ELD_labs/cordic_lab/cordic_lab.xpr] - Vivado 2019.1
write_bitstream Complete
IMPLEMENTED DESIGN - xc7z010clg400-1
Export Hardware
OK
Cancel
Export a hardware description file for use with the SDK

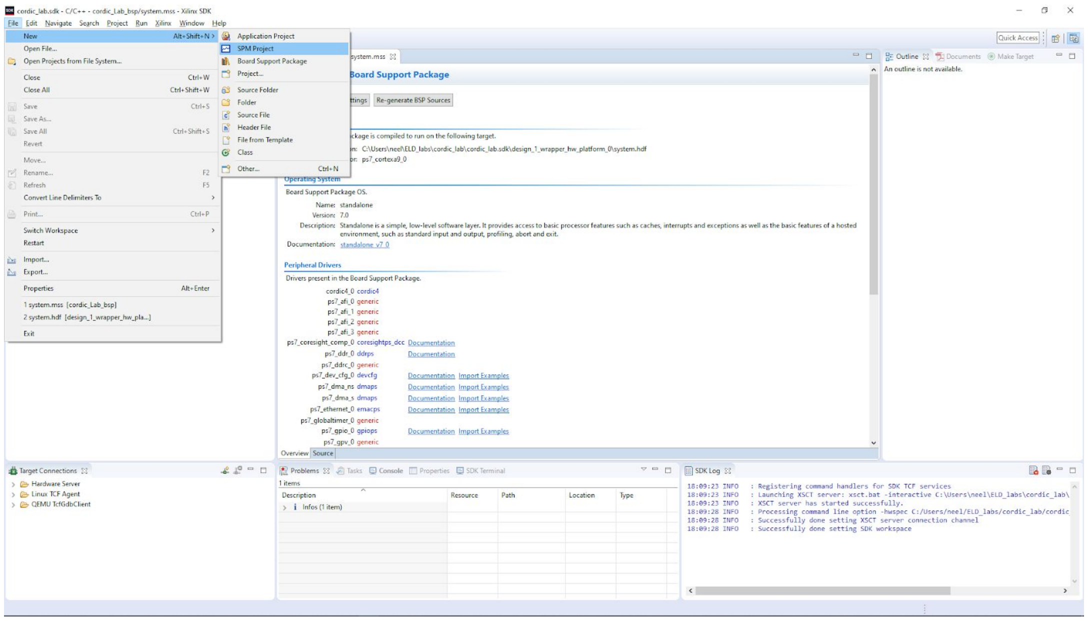
Export hardware platform for software development tools.

☑ Include bitstream

Export to: <Local to Project>

23) next click on **file>>Launch SDK** to open the SDK application and program the PS part of the board and debug it if necessary.

24) create a new application project in both c and c++ languages. We will write the code in the c++ project but we need some header files to copy from our c project.



Board Support Package

ckage is compiled to run on the following target.

n: C:\Users\neel\ELD_labs\cordic_lab\cordic_lab.sdk\design_1_wrapper_hw_platform_0\system.hdf

or: ps7_cortexa9_0

Operating System

Board Support Package OS.

Name:	standalone
Version:	7.0
Description:	Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.
Documentation:	standalone_v7_0

Peripheral Drivers

Drivers present in the Board Support Package.

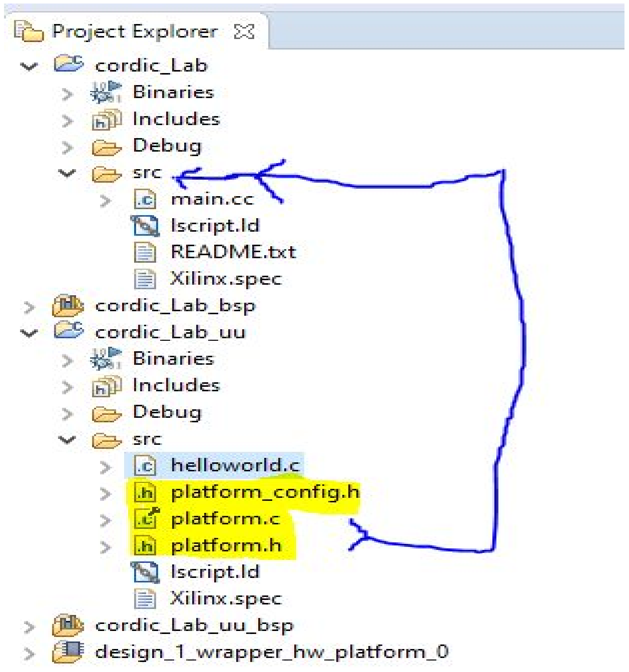
cordic4_0	cordic4		
ps7_afi_0	generic		
ps7_afi_1	generic		
ps7_afi_2	generic		
ps7_afi_3	generic		
ps7_coresight_comp_0	coresightps_dcc	Documentation	
ps7_ddr_0	ddrps	Documentation	
ps7_ddrc_0	generic		
ps7_dev_cfg_0	devcfg	Documentation	Import Examples
ps7_dma_ns	dmaps	Documentation	Import Examples
ps7_dma_s	dmaps	Documentation	Import Examples
ps7_ethernet_0	emacps	Documentation	Import Examples
ps7_globaltimer_0	generic		
ps7_gpio_0	gpiops	Documentation	Import Examples
ps7_gpv_0	generic		

1 items

Description	Resource	Path	Location	Type
Infos (1 item)				

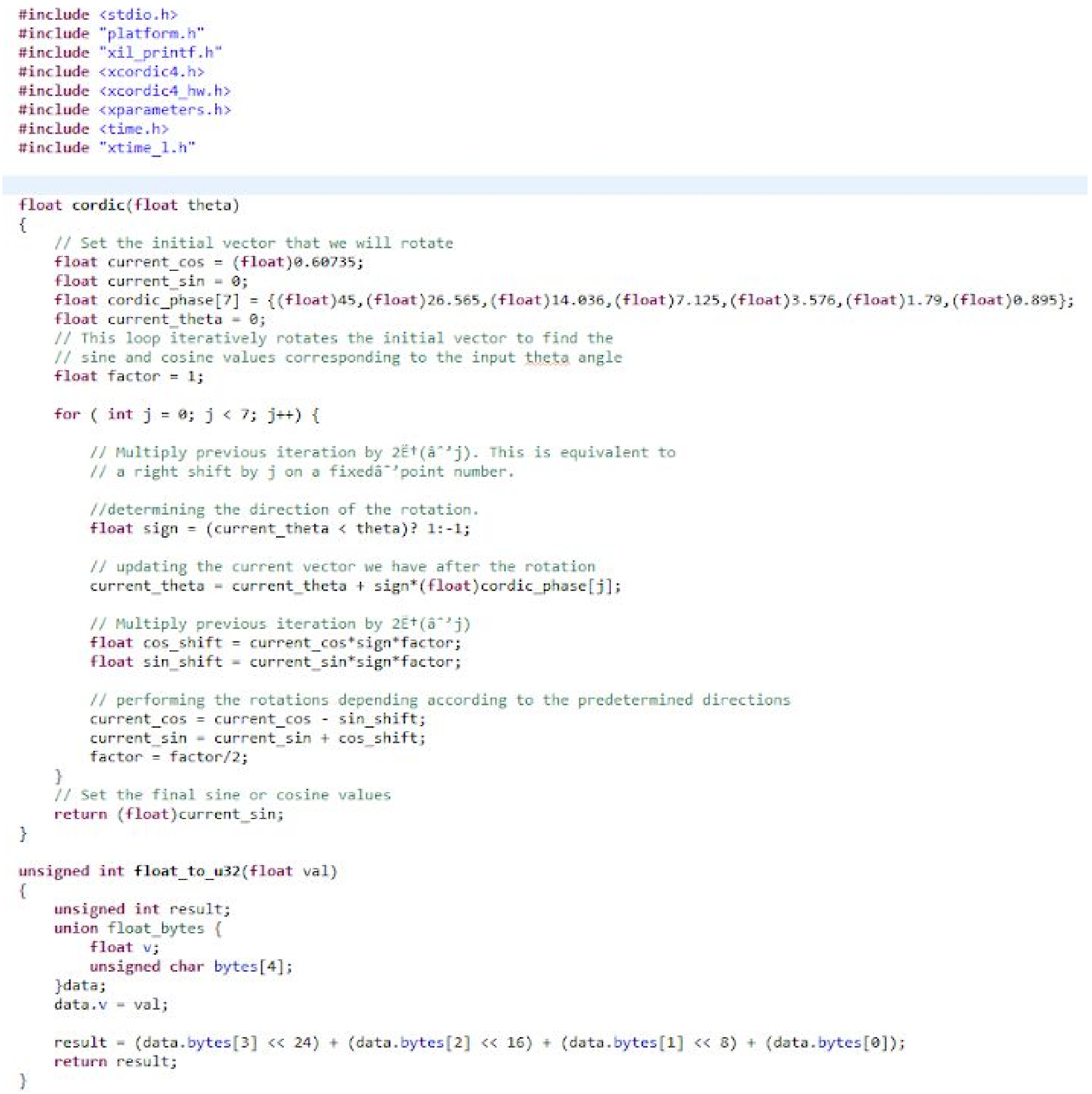
```
18:09:23 INFO   : Registering command handlers for SDK TCF services
18:09:23 INFO   : Launching XSCT server: xsct.bat -interactive C:\Users\neel\ELD_labs\cordic_lab\
18:09:23 INFO   : XSCT server has started successfully.
18:09:28 INFO   : Processing command line option -hwspec C:/Users/neel/ELD_labs/cordic_lab/cordic
18:09:28 INFO   : Successfully done setting XSCT server connection channel
18:09:28 INFO   : Successfully done setting SDK workspace
```

Following are the header file that you need to copy and paste into the CPP project. Paste these files in the src folder that is located in your non-BSP folder.

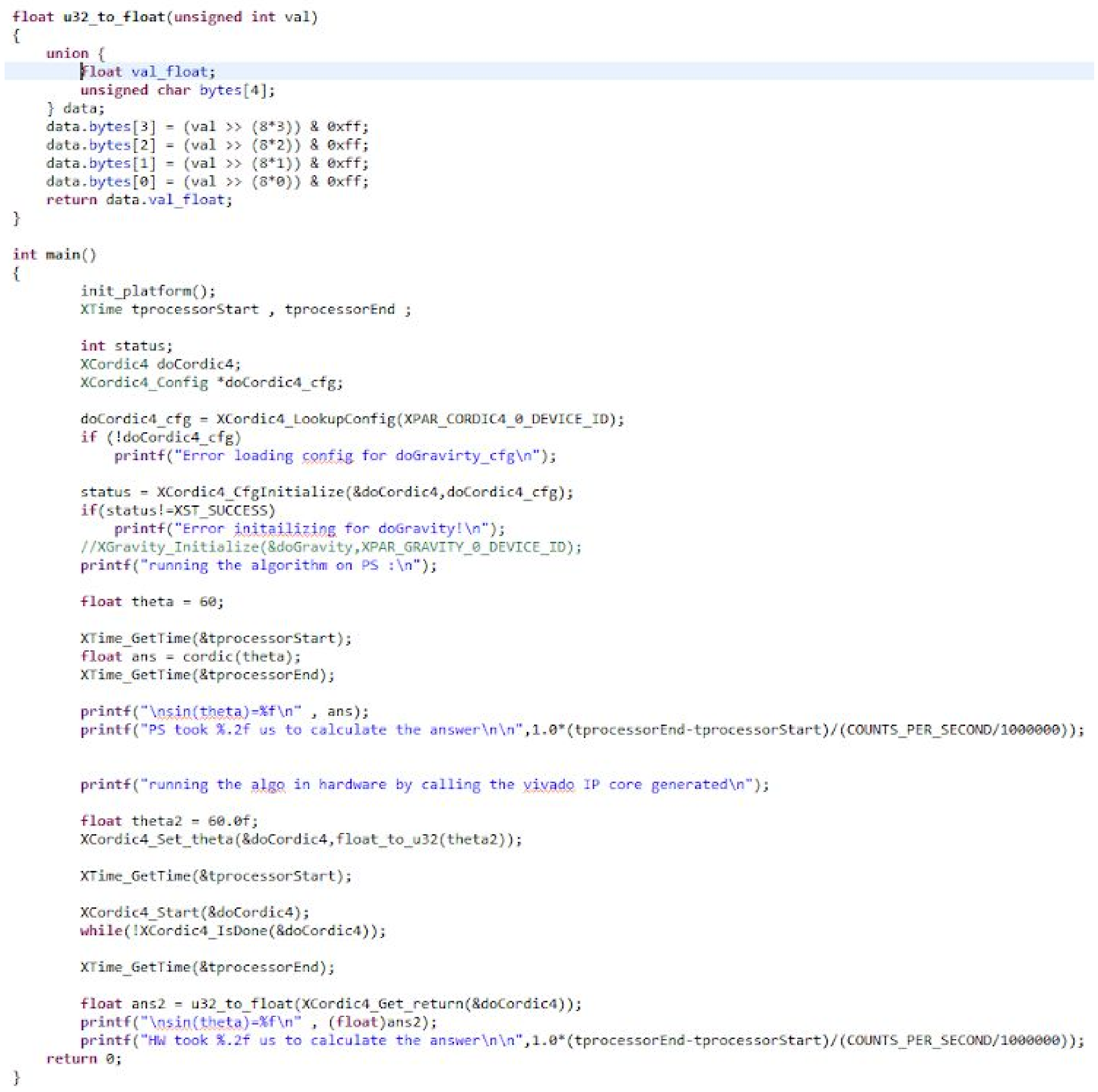


Now if you want to, you can delete the c project[mine is named “cordic_Lab_uu”].

25) using the c++ code given below you can use your Vivado HLS IP generated to calculate the sin(theta) and can observe the difference when we use hardware and software for calculating sin(theta) using the same algorithm.



```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include <xcordic4.h>
#include <xcordic4_hw.h>
#include <xparameters.h>
#include <time.h>
#include "xtime_l.h"


float cordic(float theta)
{
    // Set the initial vector that we will rotate
    float current_cos = (float)0.60735;
    float current_sin = 0;
    float cordic_phase[7] = {(float)45,(float)26.565,(float)14.036,(float)7.125,(float)3.576,(float)1.79,(float)0.895};
    float current_theta = 0;
    // This loop iteratively rotates the initial vector to find the
    // sine and cosine values corresponding to the input theta angle
    float factor = 1;

    for ( int j = 0; j < 7; j++) {

        // Multiply previous iteration by 2Ë†(âˆ’j). This is equivalent to
        // a right shift by j on a fixedâˆ’point number.

        //determining the direction of the rotation.
        float sign = (current_theta < theta)? 1:-1;

        // updating the current vector we have after the rotation
        current_theta = current_theta + sign*(float)cordic_phase[j];

        // Multiply previous iteration by 2Ë†(âˆ’j)
        float cos_shift = current_cos*sign*factor;
        float sin_shift = current_sin*sign*factor;

        // performing the rotations depending according to the predetermined directions
        current_cos = current_cos - sin_shift;
        current_sin = current_sin + cos_shift;
        factor = factor/2;
    }
    // Set the final sine or cosine values
    return (float)current_sin;
}

unsigned int float_to_u32(float val)
{
    unsigned int result;
    union float_bytes {
        float v;
        unsigned char bytes[4];
    }data;
    data.v = val;

    result = (data.bytes[3] << 24) + (data.bytes[2] << 16) + (data.bytes[1] << 8) + (data.bytes[0]);
    return result;
}
```



```c
float u32_to_float(unsigned int val)
{
    union {
        float val_float;
        unsigned char bytes[4];
    } data;
    data.bytes[3] = (val >> (8*3)) & 0xff;
    data.bytes[2] = (val >> (8*2)) & 0xff;
    data.bytes[1] = (val >> (8*1)) & 0xff;
    data.bytes[0] = (val >> (8*0)) & 0xff;
    return data.val_float;
}

int main()
{
        init_platform();
        XTime tprocessorStart , tprocessorEnd ;

        int status;
        XCordic4 doCordic4;
        XCordic4_Config *doCordic4_cfg;

        doCordic4_cfg = XCordic4_LookupConfig(XPAR_CORDIC4_0_DEVICE_ID);
        if (!doCordic4_cfg)
            printf("Error loading config for doGravirty_cfg\n");

        status = XCordic4_CfgInitialize(&doCordic4,doCordic4_cfg);
        if(status!=XST_SUCCESS)
            printf("Error initailizing for doGravity!\n");
        //XGravity_Initialize(&doGravity,XPAR_GRAVITY_0_DEVICE_ID);
        printf("running the algorithm on PS :\n");

        float theta = 60;

        XTime_GetTime(&tprocessorStart);
        float ans = cordic(theta);
        XTime_GetTime(&tprocessorEnd);

        printf("\nsin(theta)=%f\n" , ans);
        printf("PS took %.2f us to calculate the answer\n\n",1.0*(tprocessorEnd-tprocessorStart)/(COUNTS_PER_SECOND/1000000));


        printf("running the algo in hardware by calling the vivado IP core generated\n");

        float theta2 = 60.0f;
        XCordic4_Set_theta(&doCordic4,float_to_u32(theta2));

        XTime_GetTime(&tprocessorStart);

        XCordic4_Start(&doCordic4);
        while(!XCordic4_IsDone(&doCordic4));

        XTime_GetTime(&tprocessorEnd);

        float ans2 = u32_to_float(XCordic4_Get_return(&doCordic4));
        printf("\nsin(theta)=%f\n" , (float)ans2);
        printf("HW took %.2f us to calculate the answer\n\n",1.0*(tprocessorEnd-tprocessorStart)/(COUNTS_PER_SECOND/1000000));
    return 0;
}
```

26) Since we are remotely accessing the board, we will use JTAG Terminal for Output. Add the target, configure the STDIN and STDOUT in BSP, and launch the debug configurations. There is **no standard input** in this code but it can be easily implemented by using **scanf** and passing the data, in this case, it is theta.