Lab 5: Matrix Multiplication

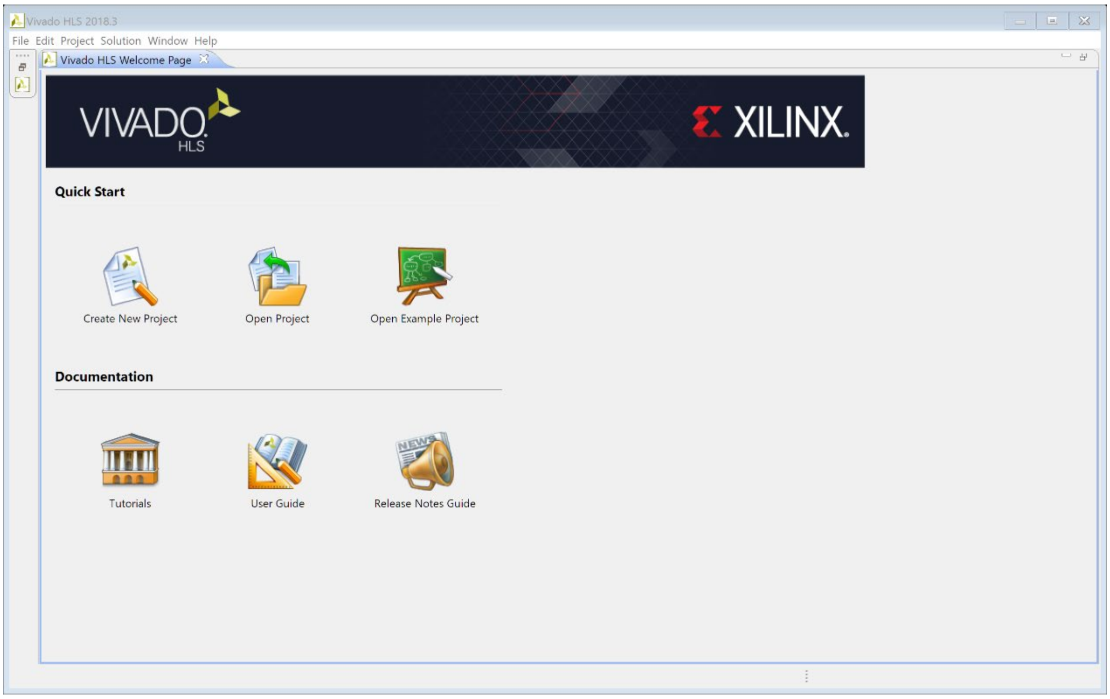
In this Lab, we are going to using Vivado High Level Synthesis (HLS) tool to create a hardware accelerator for Matrix Multiplication.

1. Vivado HLS

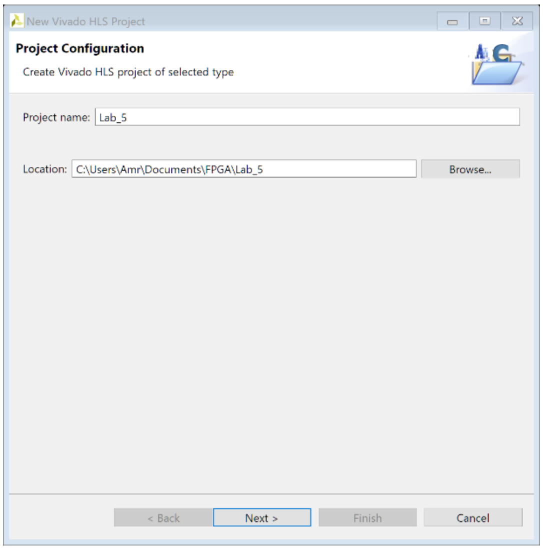
This section will walk you through creating a new Vivado HLS project

a. Download the lab5_template.zip and extract it.

b. Open Vivado HLS and click on New Project.



c. Choose a project name and make sure that the path to it doesn’t have any spaces at all! Click Next.

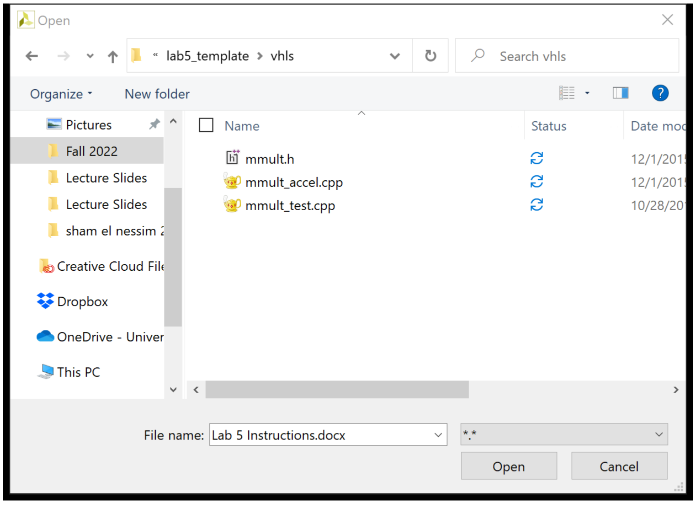


Project name: Lab_5

Location: C:\Users\Amr\Documents\FPGA\Lab_5

d. Copy the folder name “vhls” under lab5_template folder and paste in the folder of the project.

e. Choose the following files as your sources:

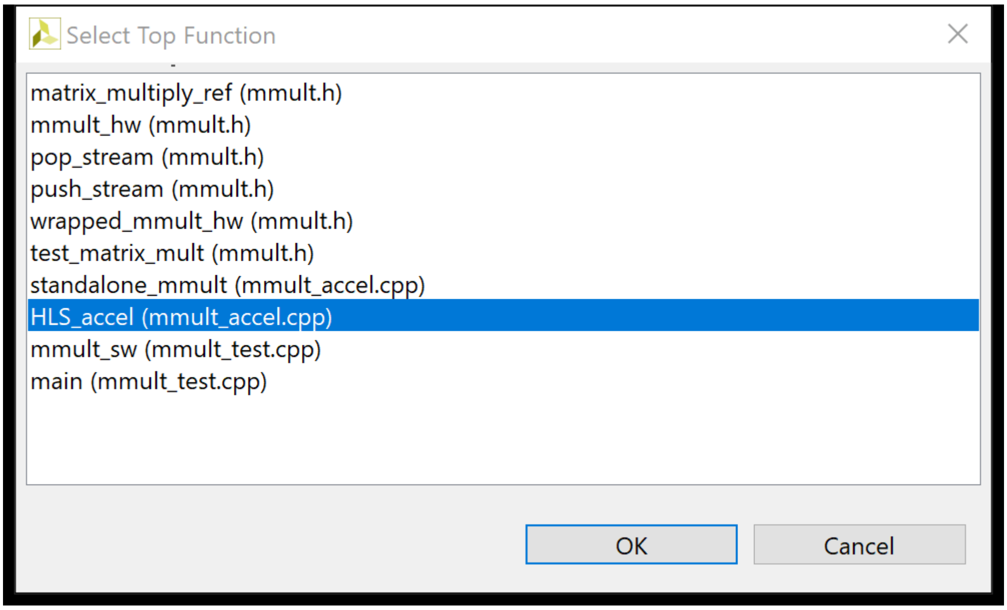


Name	Status	Date mod
mmult.h		12/1/201
mmult_accel.cpp		12/1/201
mmult_test.cpp		10/28/20

File name: Lab 5 Instructions.docx

.

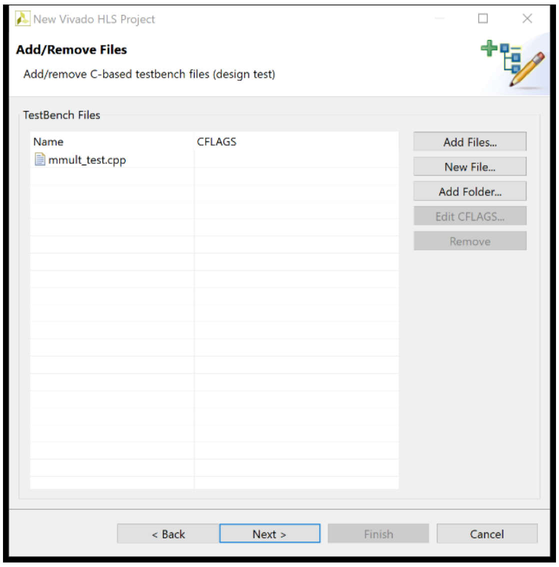
f. Next to Top Function, click on browse and choose HLS_accel (mmult_accel.cpp) as the top function. Click Next.



Select Top Function

- matrix_multiply_ref (mmult.h)
- mmult_hw (mmult.h)
- pop_stream (mmult.h)
- push_stream (mmult.h)
- wrapped_mmult_hw (mmult.h)
- test_matrix_mult (mmult.h)
- standalone_mmult (mmult_accel.cpp)
- HLS_accel (mmult_accel.cpp)
- mmult_sw (mmult_test.cpp)
- main (mmult_test.cpp)

g. Choose the mmult_test.cpp as your simulation file. Click Next.



New Vivado HLS Project

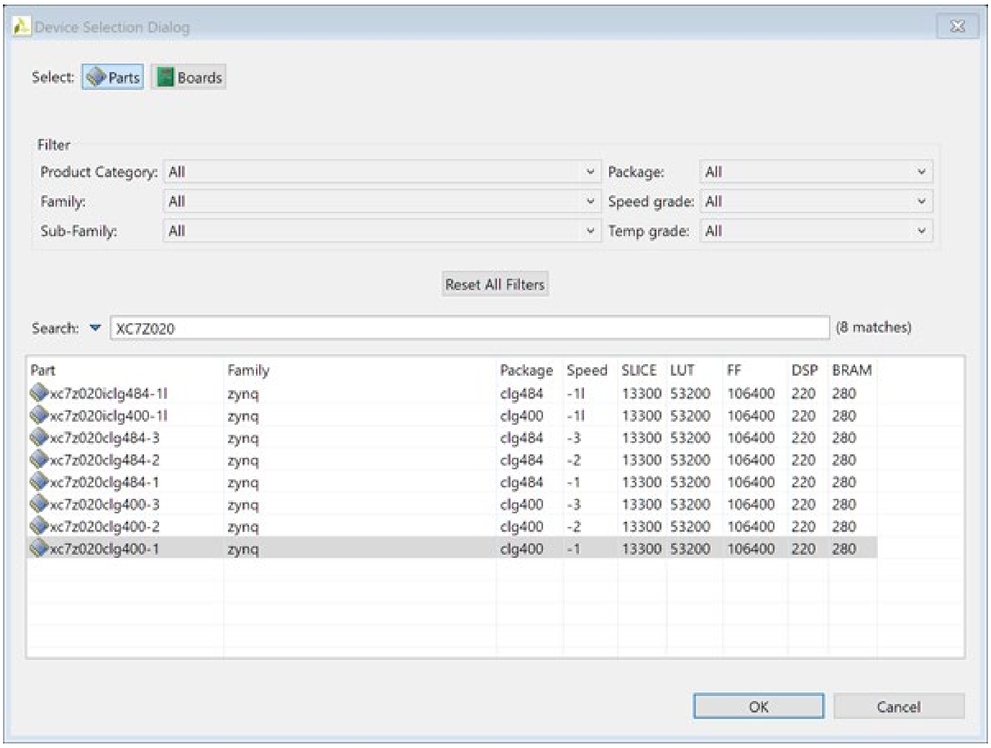
Add/Remove Files

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

TestBench Files

Name	CFLAGS
mmult_test.cpp	

h. In this window, we are going to choose ZYNQ Part number: search for: xc7z020clg and choose the following part number: Click Ok.

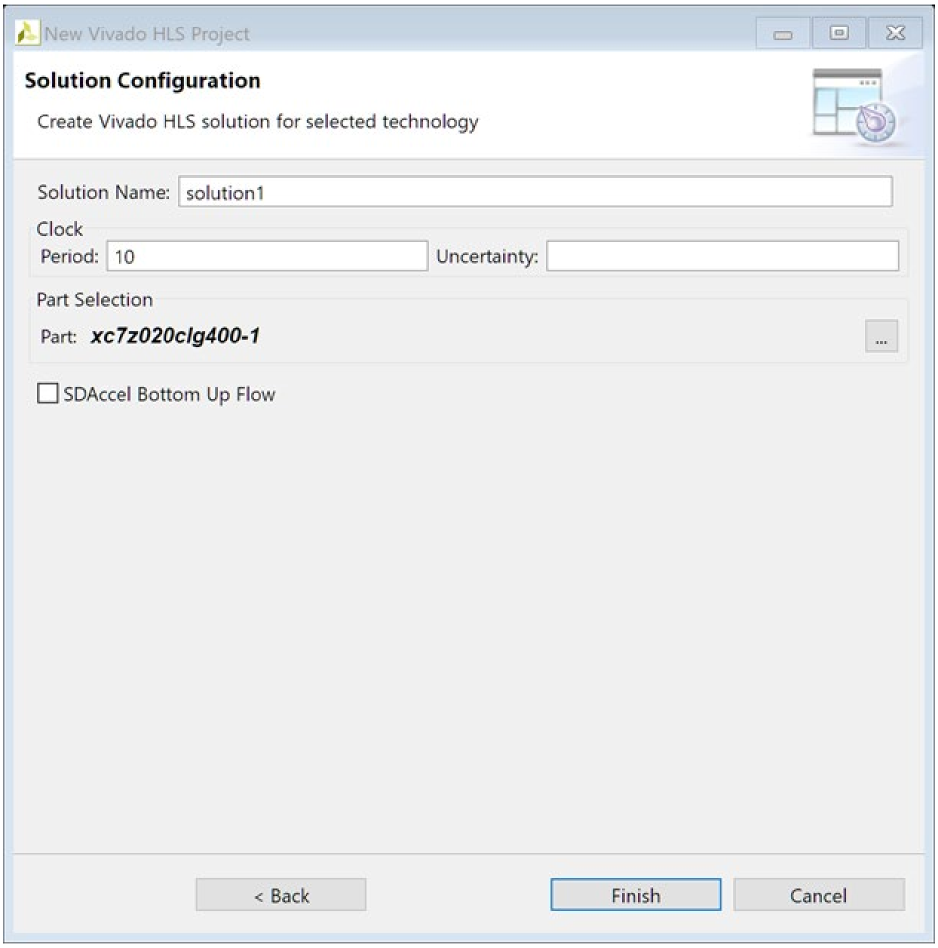


Filter
Product Category: All
Package: All
Family: All
Speed grade: All
Sub-Family: All
Temp grade: All

Search: XC7Z020 (8 matches)

Part	Family	Package	Speed	SLICE	LUT	FF	DSP	BRAM
xc7z020iclg484-1I	zynq	clg484	-1I	13300	53200	106400	220	280
xc7z020iclg400-1I	zynq	clg400	-1I	13300	53200	106400	220	280
xc7z020clg484-3	zynq	clg484	-3	13300	53200	106400	220	280
xc7z020clg484-2	zynq	clg484	-2	13300	53200	106400	220	280
xc7z020clg484-1	zynq	clg484	-1	13300	53200	106400	220	280
xc7z020clg400-3	zynq	clg400	-3	13300	53200	106400	220	280
xc7z020clg400-2	zynq	clg400	-2	13300	53200	106400	220	280
xc7z020clg400-1	zynq	clg400	-1	13300	53200	106400	220	280

i. Click Finish here



Solution Configuration

Create Vivado HLS solution for selected technology

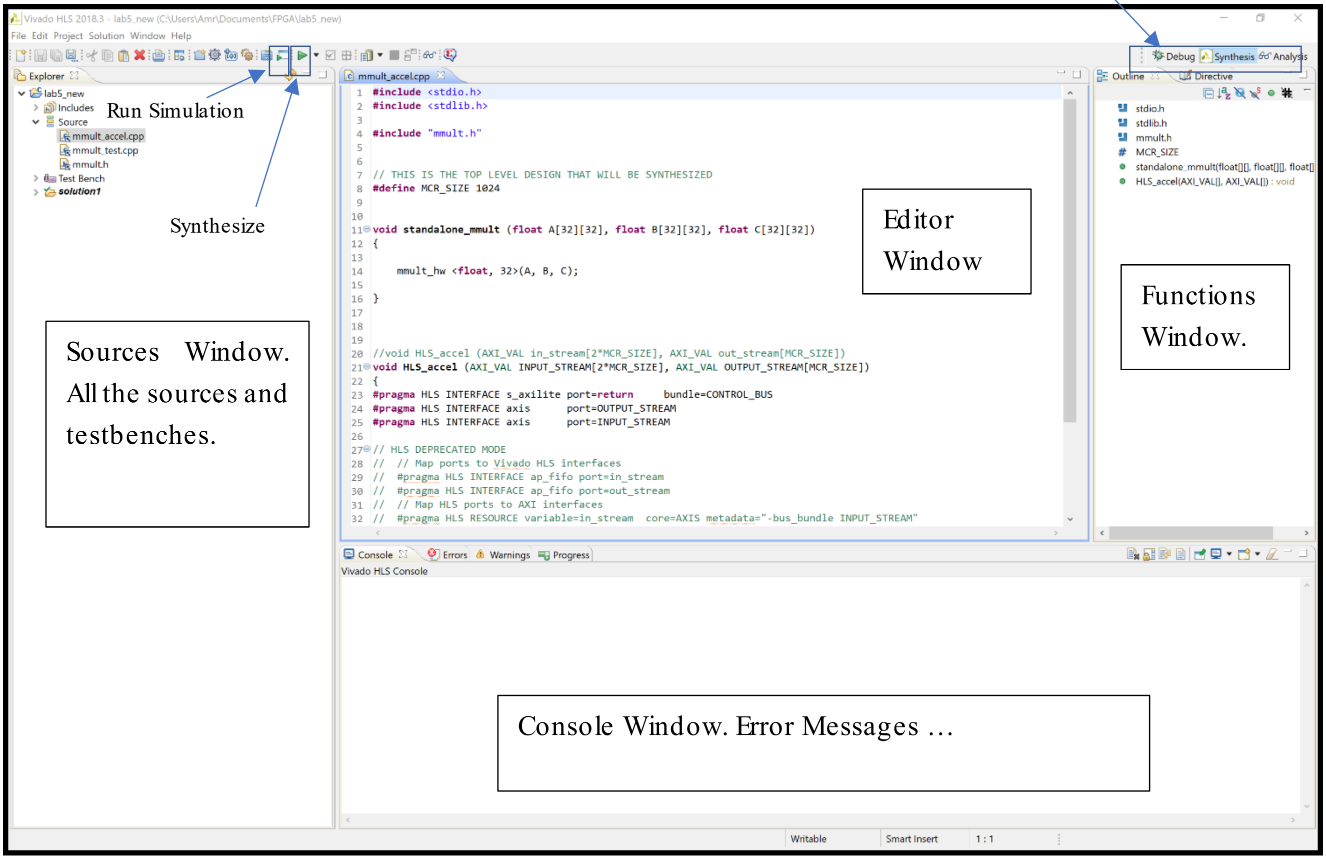
Solution Name: solution1

Clock
Period: 10 Uncertainty:

Part Selection
Part: ***xc7z020clg400-1***

☐ SDAccel Bottom Up Flow

j. This is what the project should look like now.


Switch View between Synthesis and Simulation (Debug)
Run Simulation
Synthesize
Sources Window. All the sources and testbenches.
Editor Window
Functions Window.
Console Window. Error Messages …
Vivado HLS 2018.3 - lab5_new (C:\Users\Amr\Documents\FPGA\lab5_new)
File Edit Project Solution Window Help
Explorer
lab5_new
Includes
Source
mmult_accel.cpp
mmult_test.cpp
mmult.h
Test Bench
solution1
mmult_accel.cpp
Debug
Synthesis
Analysis
Outline
Directive
stdio.h
stdlib.h
mmult.h
MCR_SIZE
standalone_mmult(float[][], float[][], float[]
HLS_accel(AXI_VAL[], AXI_VAL[]) : void
Console
Errors
Warnings
Progress
Vivado HLS Console
Writable
Smart Insert
1 : 1

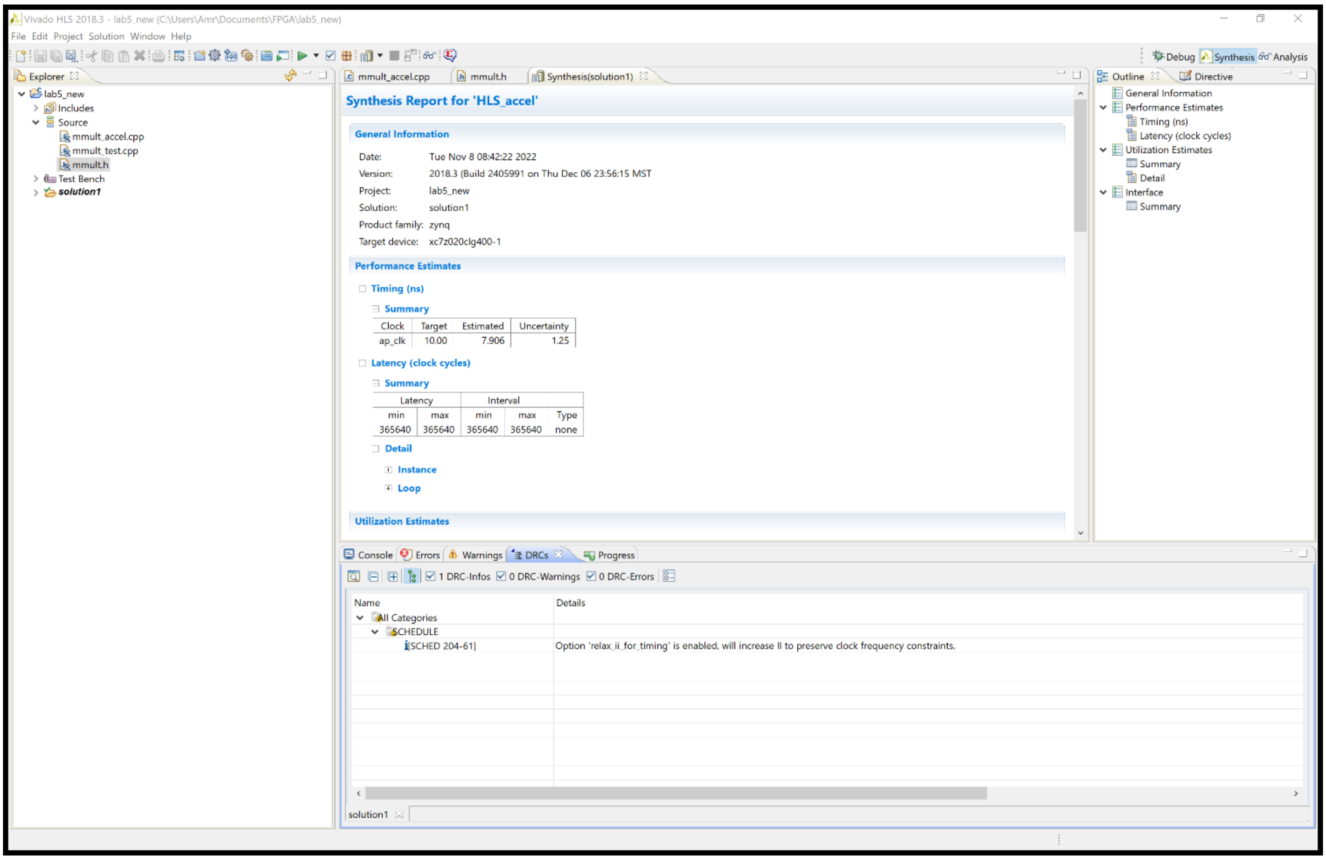
```
#include <stdio.h>
#include <stdlib.h>

#include "mmult.h"


// THIS IS THE TOP LEVEL DESIGN THAT WILL BE SYNTHESIZED
#define MCR_SIZE 1024


void standalone_mmult (float A[32][32], float B[32][32], float C[32][32])
{

    mmult_hw <float, 32>(A, B, C);

}



//void HLS_accel (AXI_VAL in_stream[2*MCR_SIZE], AXI_VAL out_stream[MCR_SIZE])
void HLS_accel (AXI_VAL INPUT_STREAM[2*MCR_SIZE], AXI_VAL OUTPUT_STREAM[MCR_SIZE])
{
#pragma HLS INTERFACE s_axilite port=return     bundle=CONTROL_BUS
#pragma HLS INTERFACE axis      port=OUTPUT_STREAM
#pragma HLS INTERFACE axis      port=INPUT_STREAM

// HLS DEPRECATED MODE
//  // Map ports to Vivado HLS interfaces
//  #pragma HLS INTERFACE ap_fifo port=in_stream
//  #pragma HLS INTERFACE ap_fifo port=out_stream
//  // Map HLS ports to AXI interfaces
//  #pragma HLS RESOURCE variable=in_stream  core=AXIS metadata="-bus_bundle INPUT_STREAM"
```

2. Synthesizing Top Function

Now you should open the *mmult.h* and start implementing the function named *mmult_hw*. Note that you would need the mmult_hw to be completed before running the synthesize step. In this step, Vivado HLS will try and generate hardware (.vhd) file representing this Top Function. It will take into consideration all the directive written and will try to optimize your design as it can.

a. When you are done implementing your C function, click on the Synthesize button.

b. If there were no errors in the synthesis and implementation, you should see the following window. Any errors will be displayed in the console window, you can go back and fix them.



Synthesis Report for 'HLS_accel'

General Information

Date: Tue Nov 8 08:42:22 2022
Version: 2018.3 (Build 2405991 on Thu Dec 06 23:56:15 MST
Project: lab5_new
Solution: solution1
Product family: zynq
Target device: xc7z020clg400-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.906	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
365640	365640	365640	365640	none

Detail

Instance

Loop

Utilization Estimates

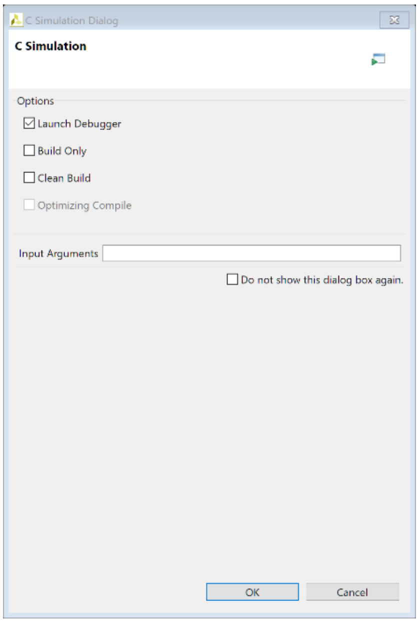
☑ 1 DRC-Infos ☑ 0 DRC-Warnings ☑ 0 DRC-Errors

Name	Details
All Categories	
SCHEDULE	
[SCHED 204-61]	Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequency constraints.

c. This is the generated Synthesis Report, which has an estimate of the Performance (Timing, maximum Clock frequency, Utilization, no. of each of the resources used, and the Interface ports), as discussed in the lecture.

Please take a screenshot of this report and name it: Trial 1, no optimization.

d. Now you are ready to Simulate. Click on the Run Simulation Icon, and Check Launch Debug in the following window.
Note: you might also want to check Clean Build in between runs, just to make sure that nothing gets written over.



C Simulation

Options

☑ Launch Debugger

☐ Build Only

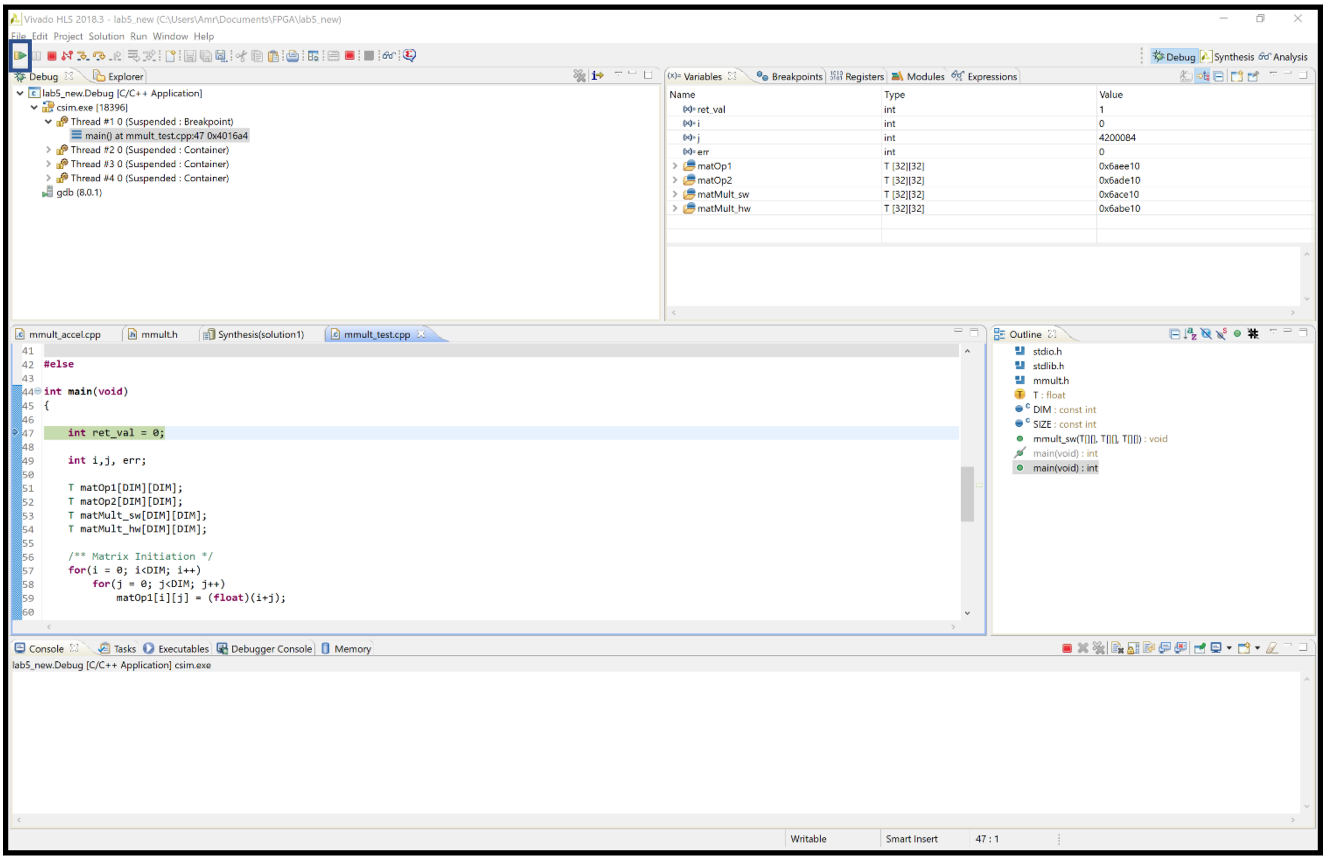
☐ Clean Build

☐ Optimizing Compile

Input Arguments

☐ Do not show this dialog box again.

e. If the simulation was able to start successfully, then you the view will be switched to the Debug view as in the following window. This is the Debug mode, so you are able to go through your code step by step. If you wish to run your code entirely, Click on Resume (the first icon to the left) or hit F8. In the following Simulation runs, if you don't wish to go in the Debug mode and run the code directly, uncheck the "Launch Debugger" in the previous step.



lab5_new.Debug [C/C++ Application]
csim.exe [18396]
Thread #1 0 (Suspended : Breakpoint)
main() at mmult_test.cpp:47 0x4016a4
Thread #2 0 (Suspended : Container)
Thread #3 0 (Suspended : Container)
Thread #4 0 (Suspended : Container)
gdb (8.0.1)

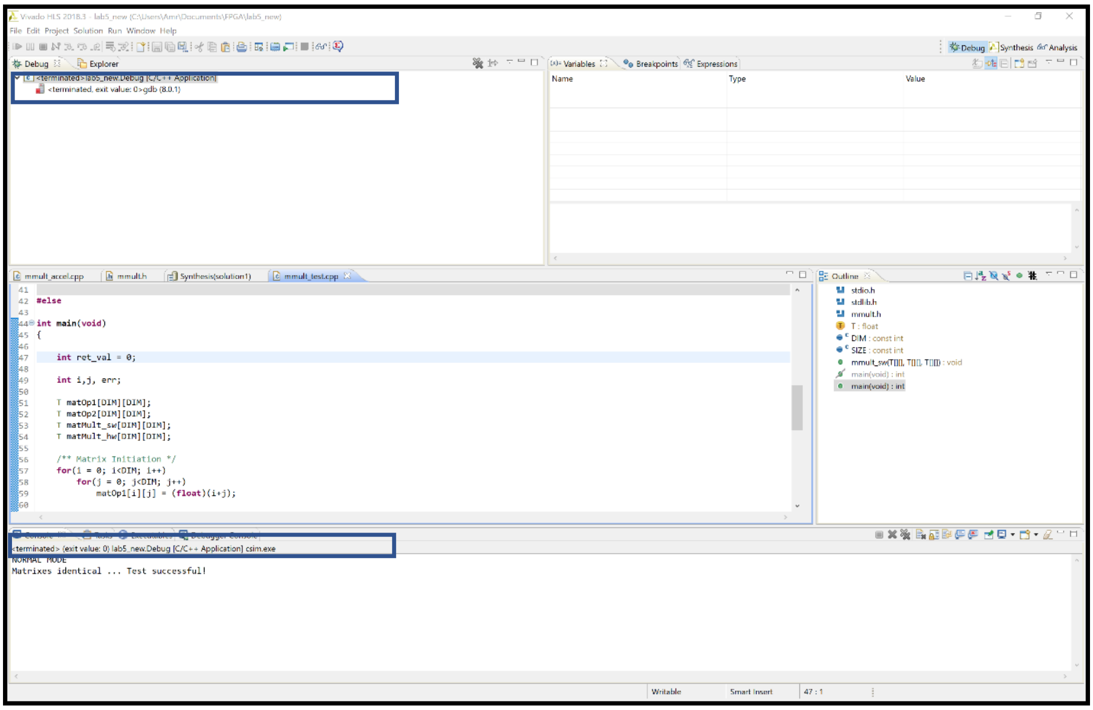
Name	Type	Value
ret_val	int	1
i	int	0
j	int	4200084
err	int	0
matOp1	T [32][32]	0x6aee10
matOp2	T [32][32]	0x6ade10
matMult_sw	T [32][32]	0x6ace10
matMult_hw	T [32][32]	0x6abe10

```
#else

int main(void)
{

    int ret_val = 0;

    int i,j, err;

    T matOp1[DIM][DIM];
    T matOp2[DIM][DIM];
    T matMult_sw[DIM][DIM];
    T matMult_hw[DIM][DIM];

    /** Matrix Initiation */
    for(i = 0; i<DIM; i++)
        for(j = 0; j<DIM; j++)
            matOp1[i][j] = (float)(i+j);

```

stdio.h
stdlib.h
mmult.h
T : float
DIM : const int
SIZE : const int
mmult_sw(T[][], T[][], T[][]) : void
main(void) : int
main(void) : int

lab5_new.Debug [C/C++ Application] csim.exe

f. Any errors, will appear in the console window, you can go back and fix them. If the code was run in its entirety with no errors, you will see the following window. Note that the exit code is zero, meaning no issues.



<terminated>lab5_new.Debug [C/C++ Application]
<terminated, exit value: 0>gdb (8.0.1)

```
#else

int main(void)
{

    int ret_val = 0;

    int i,j, err;

    T matOp1[DIM][DIM];
    T matOp2[DIM][DIM];
    T matMult_sw[DIM][DIM];
    T matMult_hw[DIM][DIM];

    /** Matrix Initiation */
    for(i = 0; i<DIM; i++)
        for(j = 0; j<DIM; j++)
            matOp1[i][j] = (float)(i+j);

```

stdio.h
stdlib.h
mmult.h
T : float
DIM : const int
SIZE : const int
mmult_sw(T[][], T[][], T[][]) : void
main(void) : int
main(void) : int

```
<terminated> (exit value: 0) lab5_new.Debug [C/C++ Application] csim.exe
NORMAL MODE
Matrixes identical ... Test successful!
```

3. Optimizing the Design, Trial 2

Now that you were able to synthesize the function, it’s time to add some optimization to it.

In your implemented code for matrix multiplication, add the following pragma directive as the first line inside your second for loop:

#pragma HLS PIPELINE II=1

You can copy and paste it from here. This line is going to do full pipelining for your accelerator to speed it.

Synthesize your design as you did before and take a screenshot of the generated report and name it: Trial 2, fully pipelined.

Notice how the latency drops drastically.

Simulate your new function and make sure that you pass the test cases.

4. Optimizing the Design, Trial 3

Uncomment the pragma directives on line 52 and 53 in mmult.h file:

```
//#pragma HLS array_partition variable=a block factor=FACTOR dim=2
//#pragma HLS array_partition variable=b block factor=FACTOR dim=1
```

Keep the pipeline directive as is from the second trial.

Synthesize your design as you did before and take a screenshot of the generated report and name it: Trial 3, fully pipelined with partitioning.

Notice how the latency drops drastically, even from trial 2.

Simulate your new function and make sure that you pass the test cases.

What to do:

1. You should design the mmult_hw function in the provided mmult.h template, as described above.
2. Synthesize the Top Function and generate the Synthesis Report. You will be writing a report for this Lab so make sure to get screenshots that would cover all parts of the report.
3. Simulate the c testbench and make sure the test cases are passed. Take screenshots showing that the test case was passed as they will be included in the report.
4. Modify your implementation to achieve better performance, as described in Trial 2 and Trial 3.
5. Repeat step 2 and 3 for each trial and make sure to take screenshots.

Deliverables:

1. You will be delivering all the C codes that you have written, The submission will be on Canvas Assignment for Lab 5. You can just Zip the top Vivado HLS project folder. You will also include make a report and add all the necessary screenshots mentioned in the What to do section. Please make sure to submit this report along with the zipped project folder and **NOT** inside the zipped folder. So submit two files, the zipped project folder and the report.
2. Check-offs will be done on the due date for this assignment. The rubric for the check-offs will be provided later.