Traineeships in Advanced Computing for High Energy Physics (TAC-HEP)

FPGA module training

More on FPGAs

Lecture-5: March 3rd 2026

Varun Sharma

University of Wisconsin – Madison, USA

So Far…

- Motivation
- Comparison: FPGAs/ASICs/GPU/CPU
- Domain specific Accelerators

Today:

- HLS Setup, create project, examples

FPGA: Field Programmable Gate Array

TAC-HEP 2026

HLS Setup on cmstrigger02

https://github.com/varuns23/TAC-HEP-FPGA/blob/main/hls-setup/readme.md

Setup

Setup Port Forwarding for RDP

- From your local machine, run:
  - `ssh -L 3389:localhost:3389 -J <username>@login.hep.wisc.edu <username>@cmstrigger02.hep.wisc.edu`
- Keep this terminal open - it maintains the RDP tunnel

Connect Using Remote Desktop (RDP Client)

- Use Microsoft's RDP client (called `Windows App`) available for macOS and Windows.
- **Setup:** Download & install the `Windows App`, Open the app and click the **+** icon, then select `Add PC`
- **Configure:** Enter the IP address: `localhost:3389` or `127.0.0.1:3389`
- **Connect:** Double-click the PC icon, enter your UW computing and , and click **Connect**
- You should now see the remote desktop of `cmstrigger02`

https://github.com/varuns23/TAC-HEP-FPGA/blob/main/hls-setup/readme.md

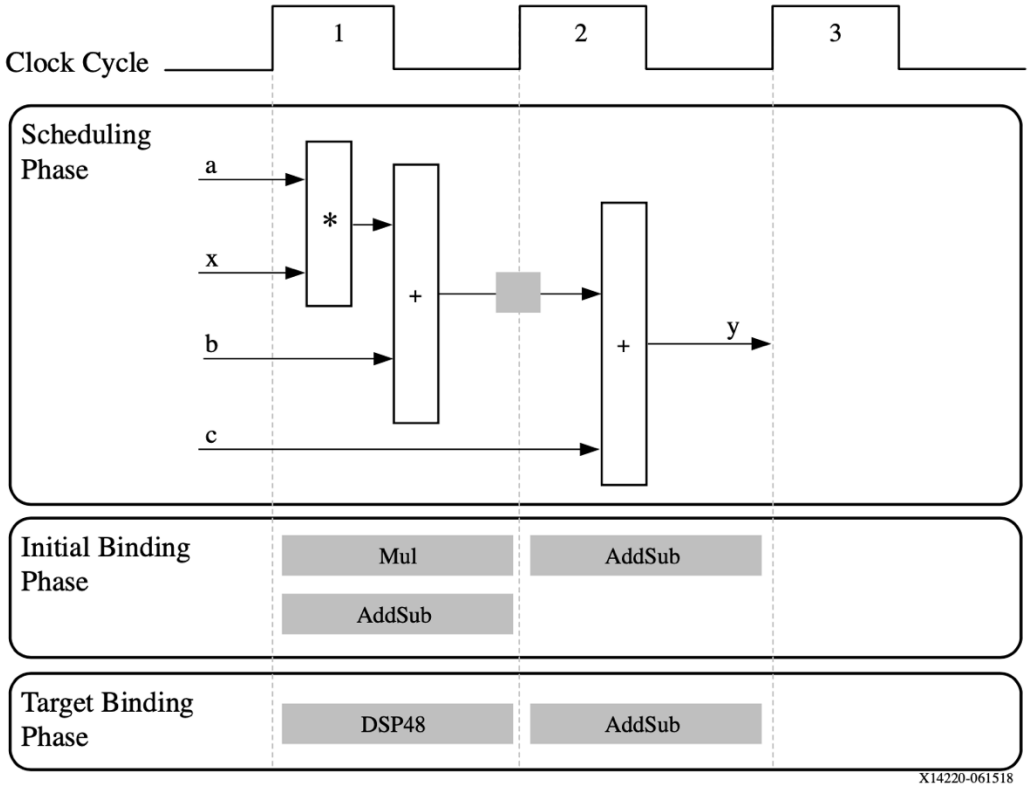
Setup workign Directory

After logging in via RDP, open a terminal inside the remote desktop

- `mkdir -p /scratch/`whoami`` (if not there already)
- `cd /scratch/`whoami``
  - **For Vitis HLS:**
    - `Source /opt/Xilinx/Vitis/2020.1/settings64.sh`
    - `cd /scratch/`whoami``
    - `vitis_hls`
  - **For Vivado HLS:**
    - `Source /opt/Xilinx/Vivado/2020.1/settings64.sh`
    - `cd /scratch/`whoami``
    - `vivado_hls`

Example

```
int foo(char x, char a, char b, char c) {
 char y;
 y = x*a+b+c;
 return y;
}
```



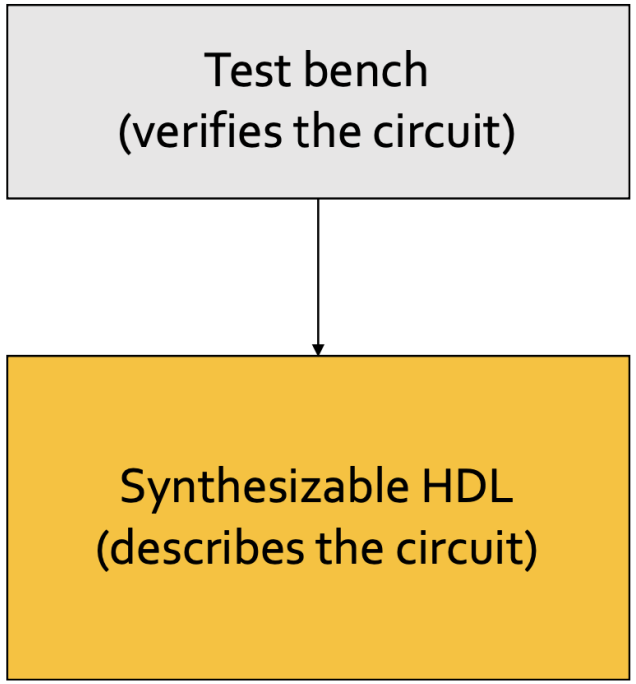
TAC-HEP 2026

Vitis/Vivado HLS

Simulation and Synthesis

The two major purposes of HDLs are logic simulation and synthesis:

- **During simulation**, inputs are applied to a module, and the outputs are checked to verify that the module operates correctly
- **During synthesis,** the textual description of a module is transformed into logic gates



HDL code is divided into synthesizable modules and a test bench:

- The synthesizable modules describe the hardware
- The test bench checks whether the output results are correct (only for simulation and cannot be synthesized)

Vitis/Vivado HLS

- HLS is a very big system
- Try to focus on ***WHAT*** can be done
  - ***HOW*** they are done will take more time (experience and learning)

- Vitis/Vivado is an Eclipse based integrated development environment (IDE)
  - Allows you to get going instantly

- The code setup has two major pieces:
  - A test harness
    - Runs only on the host
  - Top-level procedure
    - Code destined for FPGA

Vitis/Vivado HLS

- The AMD/Xilinx HLS tool synthesizes a C function into an IP block that you can integrate into a hardware system
- Tightly integrated with the rest of the AMD/Xilinx design tools and provides comprehensive language support and features for creating the optimal implementation for your C algorithm

- **Following is the Vivado HLS design flow:**
  1. Compile, execute (simulate), and debug the C algorithm
  2. Synthesize the C algorithm into an RTL implementation, optionally using user optimization directives
  3. Generate comprehensive reports and analyze the design
  4. Verify the RTL implementation using a pushbutton flow
  5. Package the RTL implementation into a selection of IP formats

C-Simulation

Inputs to HLS

- C function written in C, C++, or SystemC
  - Primary input and the function can contain a hierarchy of sub-functions
- Constraints
  - Constraints are required and include the clock period, clock uncertainty, and FPGA target
  - The clock uncertainty defaults to 12.5% of the clock period if not specified.
- Directives
  - Directives are optional and direct the synthesis process to implement a specific behavior or optimization
- C test bench and any associated files
  - Vivado HLS uses the C test bench to simulate the C function prior to synthesis and to verify the RTL output using C/RTL co-simulation

Inputs to Vivado HLS

- C function written in C, C++, or SystemC
  - Primary input and the function can contain a hierarchy of sub-functions
- Constraints

C input files, directives, and constraints can be added to project interactively using the Vivado HLS GUI

OR

- Using Tcl commands at the command prompt (Create a Tcl file and execute the commands in batch mode)

specific behavior or optimization

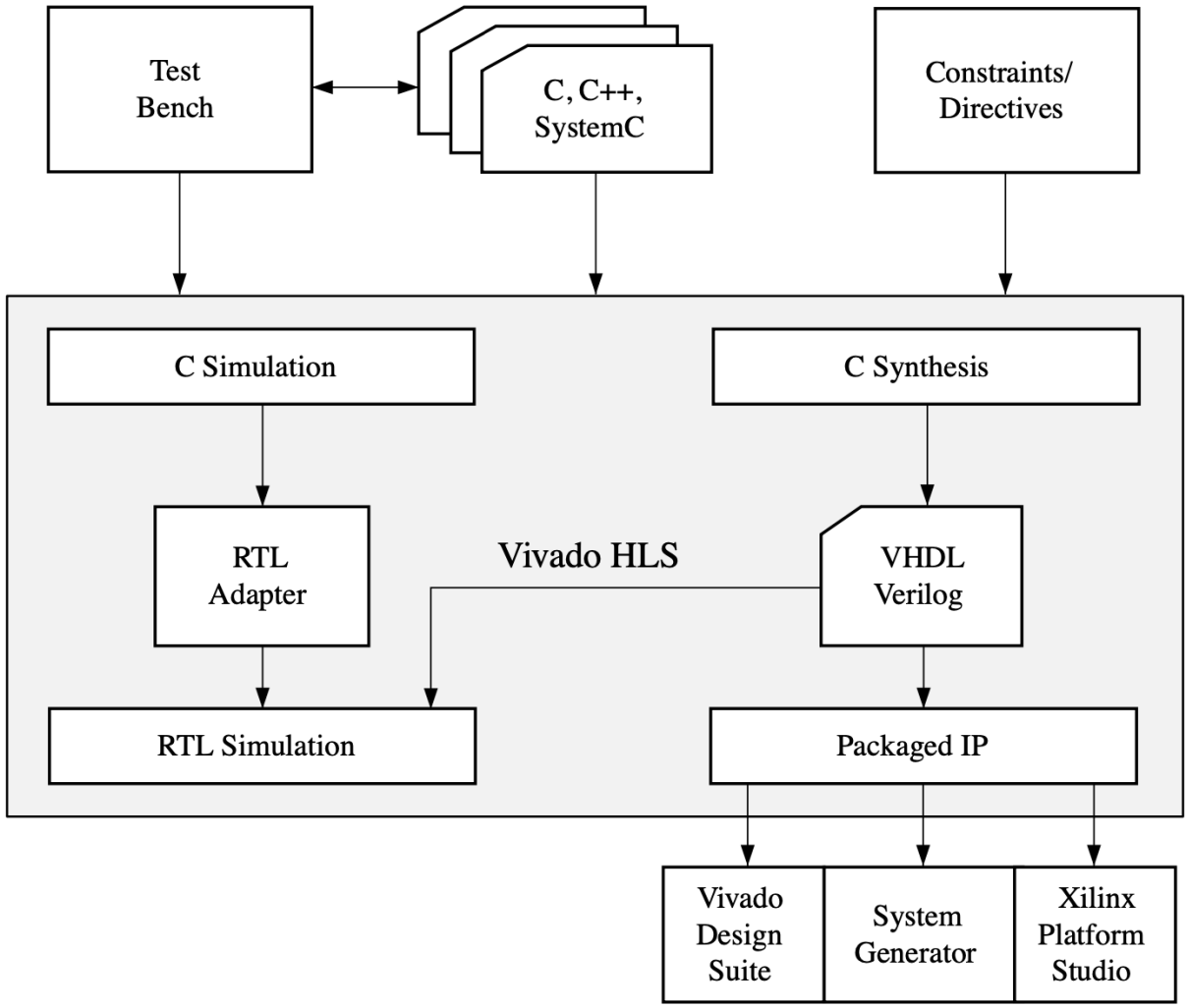
- C test bench and any associated files
  - Vivado HLS uses the C test bench to simulate the C function prior to synthesis and to verify the RTL output using C/RTL co-simulation

Outputs from Vivado HLS

- **Primary output:** RTL implementation files in hardware description language (HDL) formats
  - Using Vivado synthesis, you can synthesize the RTL into a gate-level implementation and an FPGA bitstream file
  - RTL is available in
    - Verilog
    - VHDL
  - Vivado HLS packages the implementation files as an IP block for use with other tools
  - Packed IP is synthesised into a bit stream

- **Report files**
  - Result of synthesis, C/RTL co-simulation, and IP packaging

Overview of HLS design flow



HLS Pragmas

"Pragmas": Instructions to tell your compiler how to build the hardware

- HLS tool provides different set of pragmas that can be used to optimize the design, reduce latency, improve performance etc. These pragmas can be directly added to the source code for the kernel.

Type	Attributes
Kernel Optimization	• pragma HLS aggregate • pragma HLS alias • pragma HLS disaggregate • pragma HLS expression_balance • pragma HLS latency • pragma HLS performance • pragma HLS protocol • pragma HLS reset • pragma HLS top • pragma HLS stable
Function Inlining	• pragma HLS inline
Interface Synthesis	• pragma HLS interface • pragma HLS stream
Task-level Pipeline	• pragma HLS dataflow • pragma HLS stream
Pipeline	• pragma HLS pipeline • pragma HLS occurrence
Loop Unrolling	• pragma HLS unroll • pragma HLS dependence
Loop Optimization	• pragma HLS loop_flatten • pragma HLS loop_merge • pragma HLS loop_tripcount
Array Optimization	• pragma HLS array_partition • pragma HLS array_reshape
Structure Packing	• pragma HLS aggregate • pragma HLS dataflow
Resource Utilization	• pragma HLS allocation • pragma HLS bind_op • pragma HLS bind_storage • pragma HLS function_instantiate

https://docs.xilinx.com/r/en-US/ug1399-vitis-hls/HLS-Pragmas

Terminology

- **HLS file:** C/C++ code that will be synthesised and run on FPGA
- **Test bench (TB) file:** C/C++ code that is run to test the HLS code. It calls the HLS functions and can run tests on their output, e.g. C asserts
- **Tcl scripts:** set of tcl instructions executed by the Vivado HLS shell

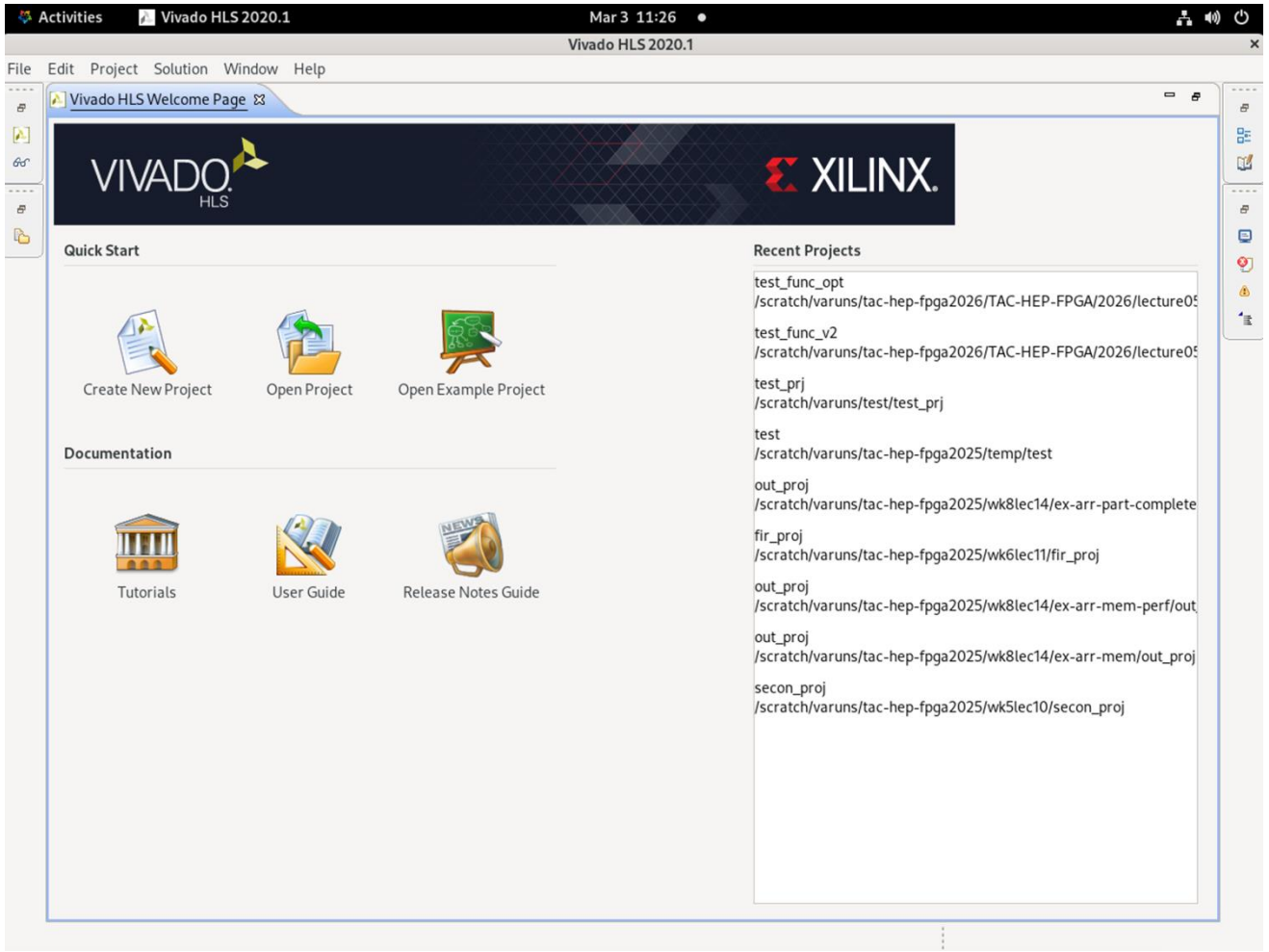
- **Synthesis:** C/C++ ➞ HDL lang (VHDL/Verilog)
- **Project:** Collection of HLS and test bench (TB) files
  - Has a top-level function name that is the starting point for synthesis
- **Solution:** specific implementation of project
  - Runs on a specific device at a specific clock frequency
- **C simulation:** HLS+TB files are compiled with gcc against HLS headers and lib and plainly run as any other executable
- **C/RTL cosimulation:** synthesized HLS code is run on simulator and results tested on the C/C++ test bench

TAC-HEP 2026

Create a new project

Using Vivado HLS

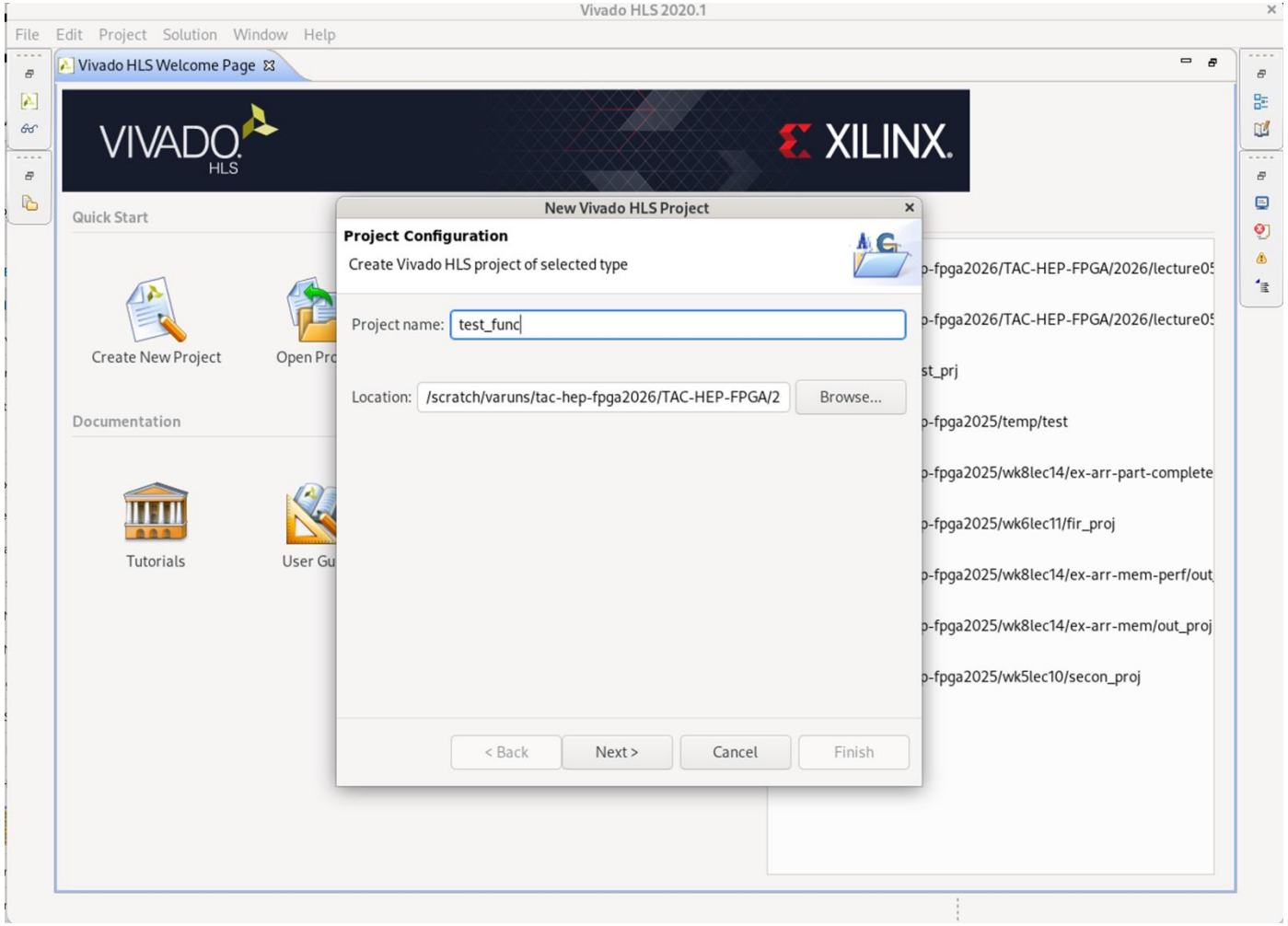
- Once connected to **cmstrigger02**
- Source the settings
- Go to **/scratch/`whoami`** directory
- Execute ***vivado_hls***



Similar instructions valid for Vitis (use vitis_hls and corresponding settings)

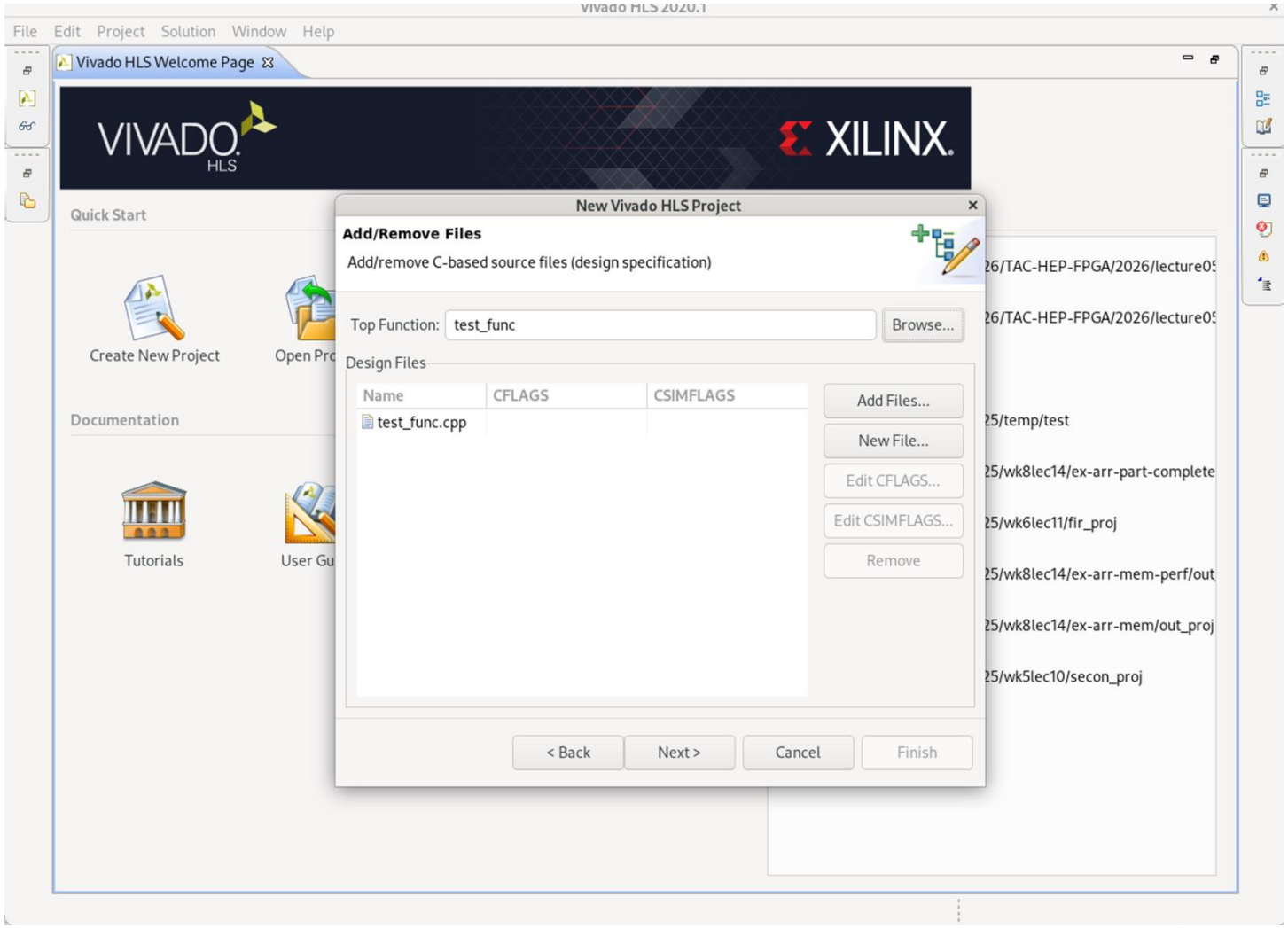
Creating Project

- Create New Project
- Enter the project name
- Click Browse to navigate to the location of the project directory
- Enter the location to be used for your project



Adding C-design

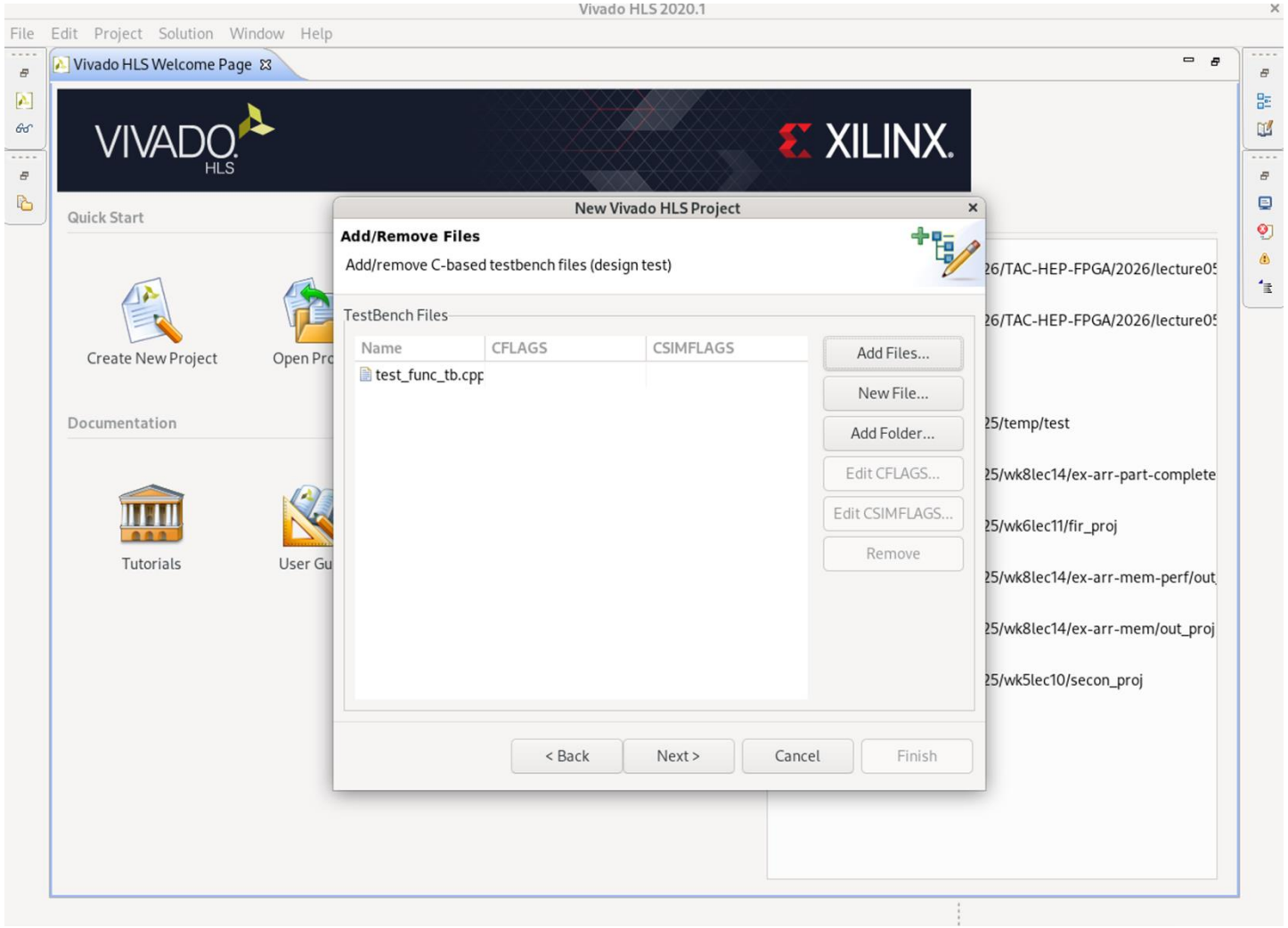
- Click Add Files
- Select all files that need to synthesized and click OK
- Specify the **top-level function** to be synthesised



Add test bench files

The test bench compares the output data from the **"top-level"** function with known good values

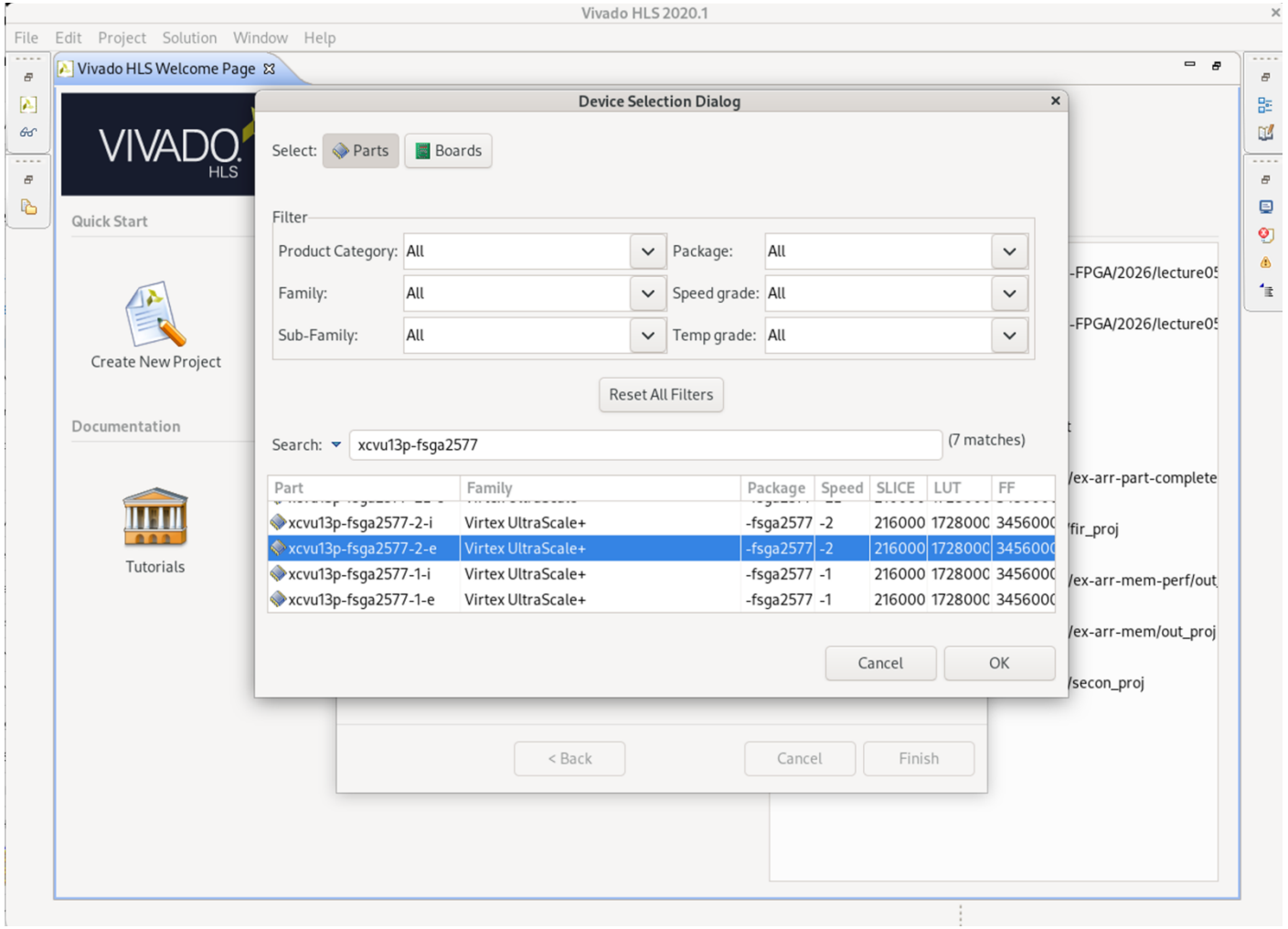
If you do not include all the files used by the test bench, **C** and **RTL simulation** might fail due to an inability to find the data files.



Select you target device

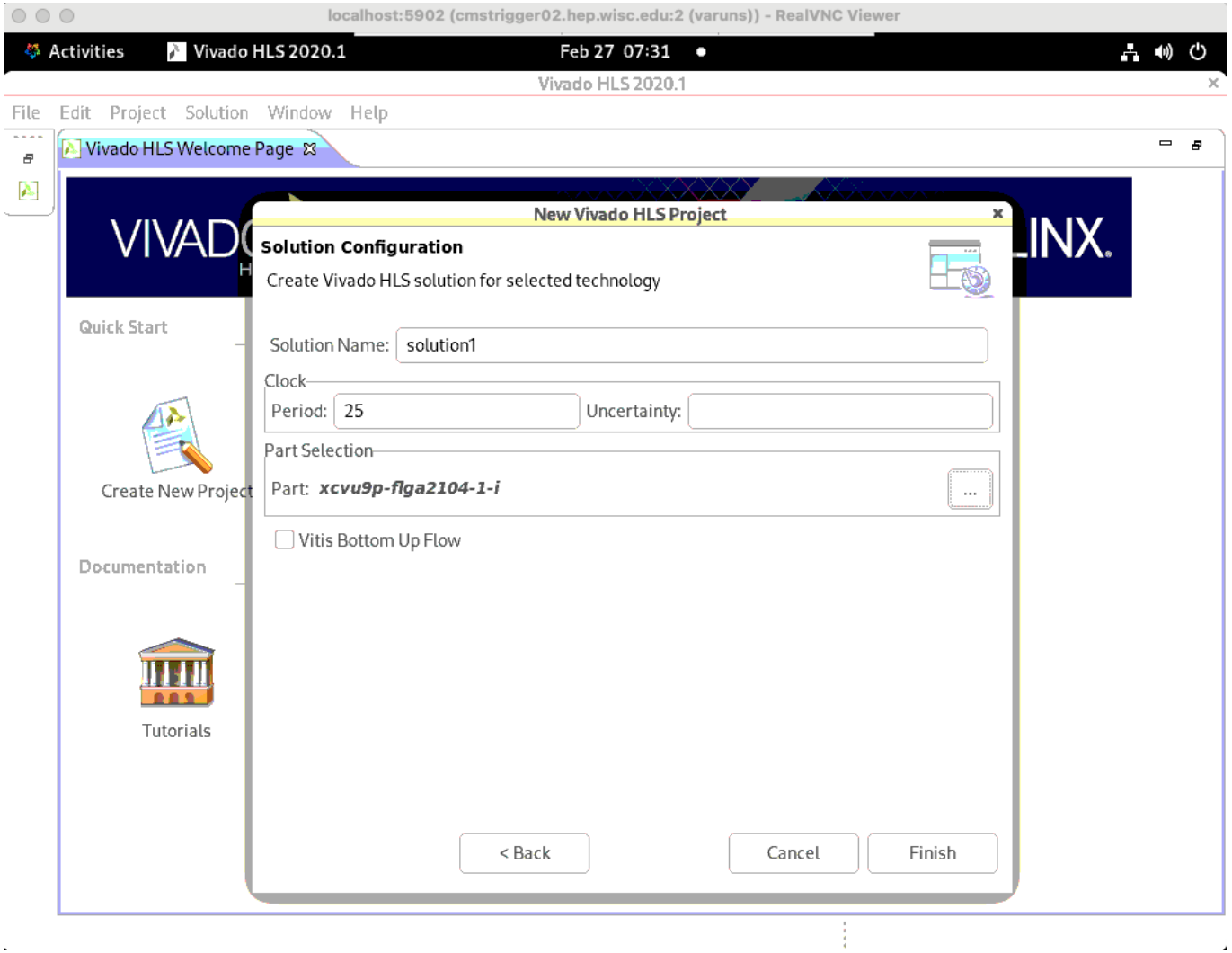
Check

- Family
- Package
- Speed grade,
- Resources available

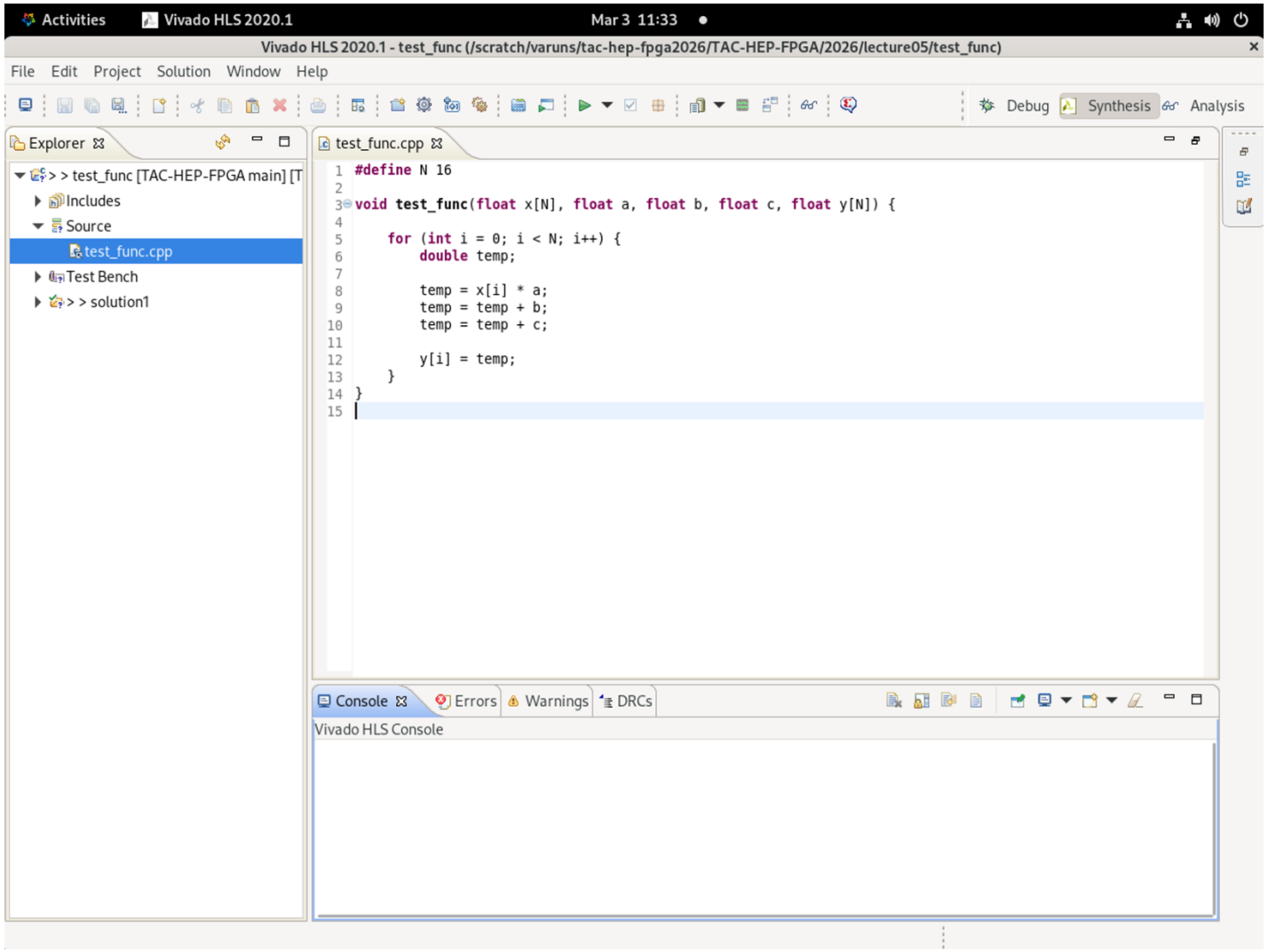


Target device

- **Solution Name**: of your choice
  - **solution1**
- **Clock Period**: in units of ns or a frequency value specified with the MHz suffix
  - **25ns**
- **Uncertainty**: If no value is given, default is 12.5%
- **Part**: Click to select the appropriate technology
  - **xcvu13p-fsga2577-2-e**



Vivado GUI


Activities
Vivado HLS 2020.1
Mar 3 11:33
Vivado HLS 2020.1 - test_func (/scratch/varuns/tac-hep-fpga2026/TAC-HEP-FPGA/2026/lecture05/test_func)
File Edit Project Solution Window Help
Debug
Synthesis
Analysis
Explorer
> > test_func [TAC-HEP-FPGA main] [T
Includes
Source
test_func.cpp
Test Bench
> > solution1
test_func.cpp
#define N 16
void test_func(float x[N], float a, float b, float c, float y[N]) {
for (int i = 0; i < N; i++) {
double temp;
temp = x[i] * a;
temp = temp + b;
temp = temp + c;
y[i] = temp;
}
}
Console
Errors
Warnings
DRCs
Vivado HLS Console

TAC-HEP 2026

Example

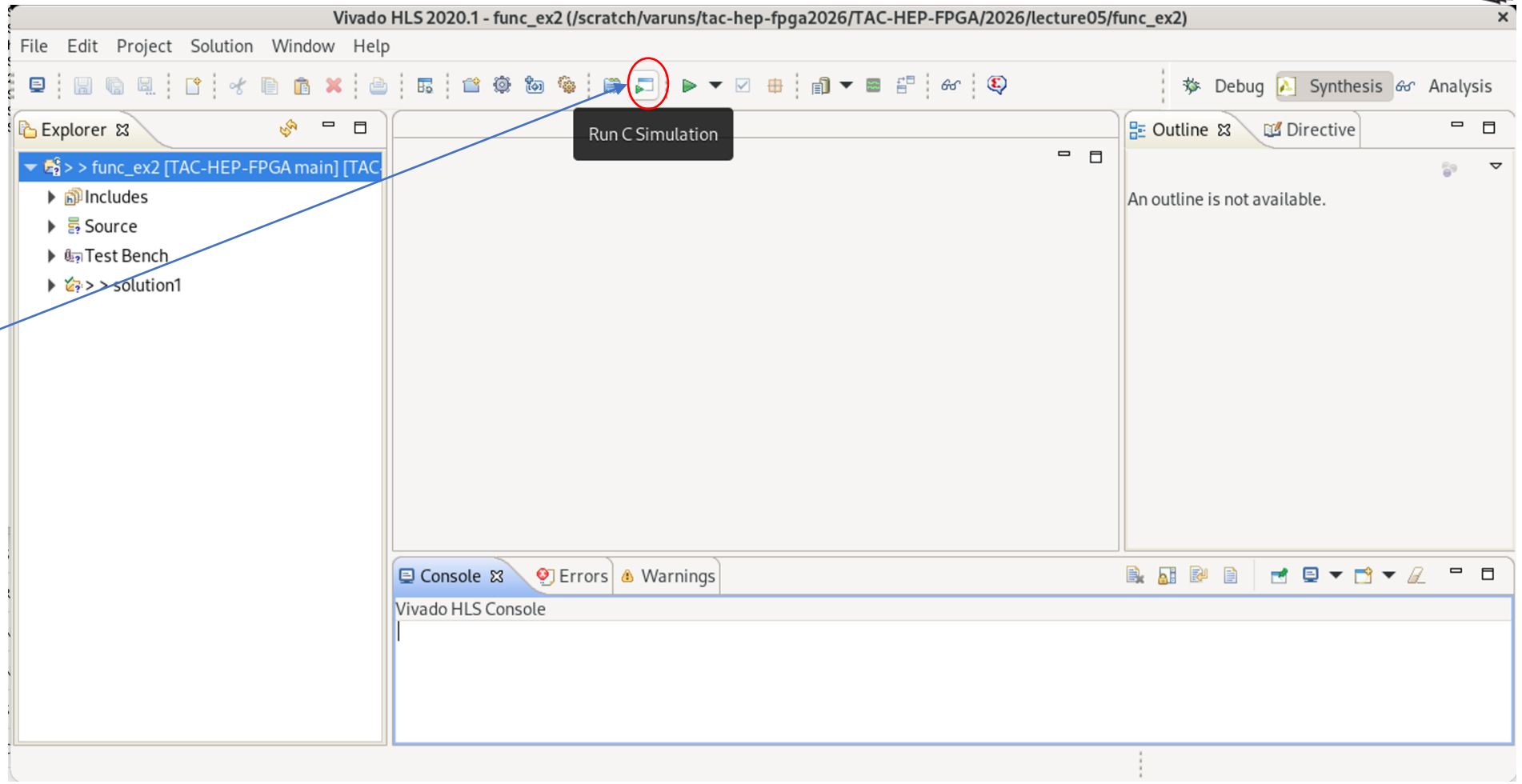
Example: func_ex2

```c
#include "func_ex2.h"

void func_ex2 (int *y, int c[N],  int x) {

   static int arr[N];
   int sum;
   int data;
   int i;

   sum=0;
Loop:
   for (i = N - 1; i >= 0; i--)
   {
      if (i == 0)
      {
         arr[0] = x;
         data = x;
      }
      else
      {
         arr[i] = arr[i - 1];
         data = arr[i];
      }
      sum += data * c[i];
      ;
   }
   *y = sum;
}
```

```c
#ifndef FUNC_EX2_H_
#define FUNC_EX2_H_
#define N       11

void func_ex2 (
  int *y,
  int c[N+1],
  int x
  );

#endif
```

```c
#include <stdio.h>
#include <math.h>
#include "func_ex2.h"

int main()
{
  const int samples = 600;
  FILE *oFile;

  int inp, output;
  int coef[N] = { 0, -10, -9, 23, 56, 63, 56, 23, -9, -10, 0};

  int i, rmp;
  inp = 0;
  rmp = 1;

  oFile = fopen("func_ex2_out.dat", "w");
  for (i = 0; i <= samples; i++)
  {
    if (rmp == 1)
      inp = inp + 1;
    else
      inp = inp - 1;

    // Execute the function with latest input
    func_ex2(&output, coef, inp);

    if ((rmp == 1) && (inp >= 75))
      rmp = 0;
    else if ((rmp == 0) && (inp <= -75))
      rmp = 1;

    // Save the results.
    fprintf(oFile, "%i %d %d\n", i, inp, output);
  }
  fclose(oFile);

  printf("Comparing against output data \n");
  if (system("diff -w func_ex2_out.dat func_ex2_ref_out.dat"))
  {

    fprintf(stdout, "*********************************************\n");
    fprintf(stdout, "FAIL: Output DOES NOT match the reference output\n");
    fprintf(stdout, "*********************************************\n");
    return 1;
  }
  else
  {
    fprintf(stdout, "*********************************************\n");
    fprintf(stdout, "PASS: The output matches the reference output!\n");
    fprintf(stdout, "*********************************************\n");
    return 0;
  }
}
```

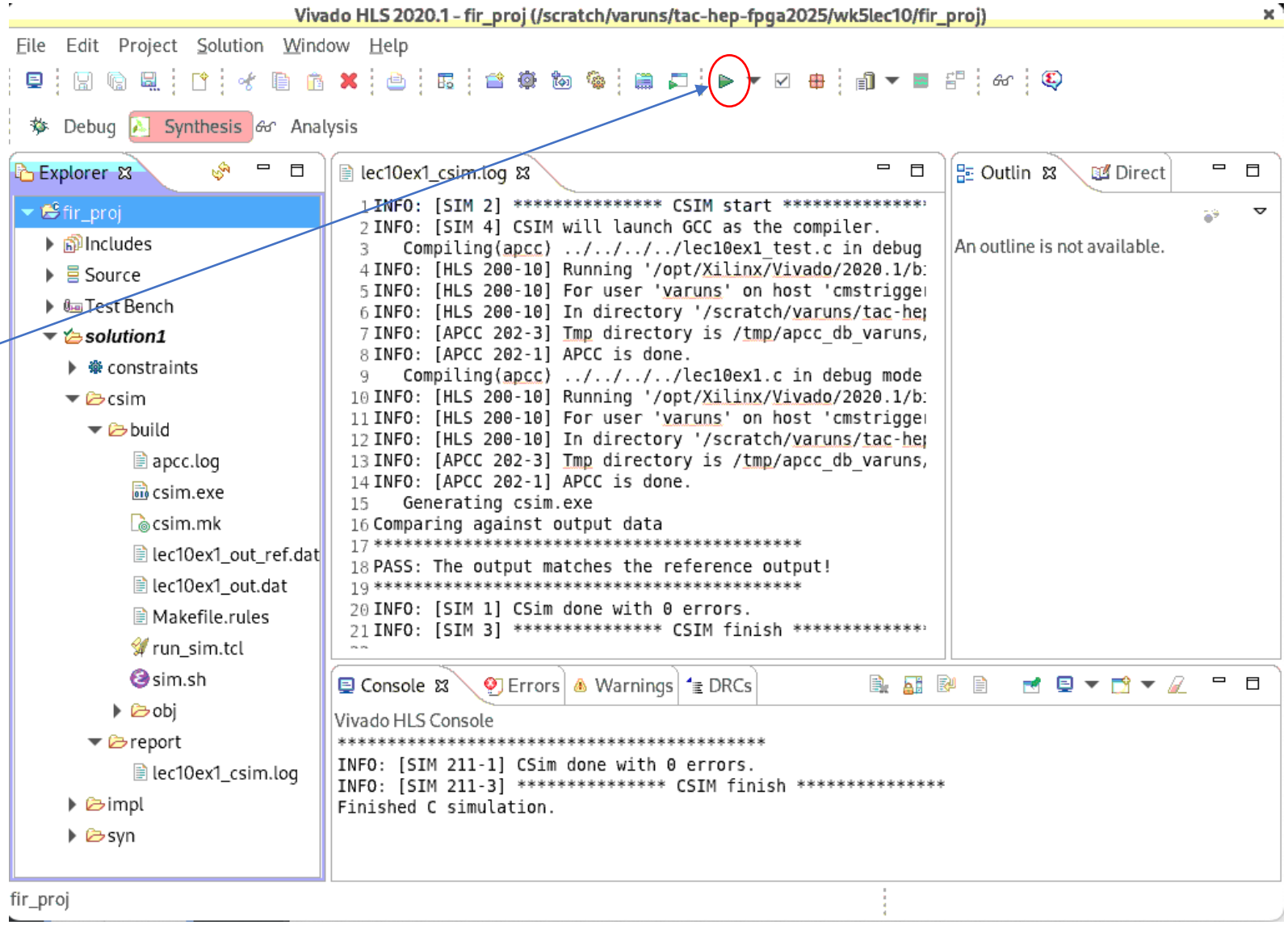
Step-2: C-Simulation



Run c-simulation

Step-3: C-Synthesis

Run C-Synthesis



Synthesis output printouts – 1/2

```
Starting C synthesis ...
/opt/Xilinx/Vivado/2020.1/bin/vivado_hls /scratch/varuns/tac-hep-fpga2025/wk5lec10/fir_proj/solution1/csynth.tcl
INFO: [HLS 200-10] Running '/opt/Xilinx/Vivado/2020.1/bin/unwrapped/lnx64.o/vivado_hls'
INFO: [HLS 200-10] For user 'varuns' on host 'cmstrigger02.hep.wisc.edu' (Linux_x86_64 version 5.14.0-427.22.1.el9_4
INFO: [HLS 200-10] In directory '/scratch/varuns/tac-hep-fpga2025/wk5lec10'
Sourcing Tcl script '/scratch/varuns/tac-hep-fpga2025/wk5lec10/fir_proj/solution1/csynth.tcl'
INFO: [HLS 200-10] Opening project '/scratch/varuns/tac-hep-fpga2025/wk5lec10/fir_proj'.
INFO: [HLS 200-10] Adding design file 'lec10ex1.c' to the project
INFO: [HLS 200-10] Adding design file 'lec10ex1.h' to the project
INFO: [HLS 200-10] Adding test bench file 'lec10ex1_out_ref.dat' to the project
INFO: [HLS 200-10] Adding test bench file 'lec10ex1_test.c' to the project
INFO: [HLS 200-10] Opening solution '/scratch/varuns/tac-hep-fpga2025/wk5lec10/fir_proj/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 25ns.
INFO: [HLS 200-10] Setting target device to 'xcvu9p-flga2104-1-i'
INFO: [SYN 201-201] Setting up clock 'default' with a period of 25ns.
INFO: [SCHED 204-61] Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequency constrair
INFO: [HLS 200-10] Analyzing design file 'lec10ex1.c' ...
INFO: [HLS 200-111] Finished Linking Time (s): cpu = 00:00:15 ; elapsed = 00:00:14 . Memory (MB): peak = 1629.840 ;
INFO: [HLS 200-111] Finished Checking Pragmas Time (s): cpu = 00:00:15 ; elapsed = 00:00:14 . Memory (MB): peak = 16
INFO: [HLS 200-10] Starting code transformations ...
INFO: [HLS 200-111] Finished Standard Transforms Time (s): cpu = 00:00:16 ; elapsed = 00:00:16 . Memory (MB): peak =
INFO: [HLS 200-10] Checking synthesizability ...
INFO: [HLS 200-111] Finished Checking Synthesizability Time (s): cpu = 00:00:16 ; elapsed = 00:00:16 . Memory (MB):
INFO: [HLS 200-111] Finished Pre-synthesis Time (s): cpu = 00:00:16 ; elapsed = 00:00:16 . Memory (MB): peak = 1629.
INFO: [HLS 200-472] Inferring partial write operation for 'arr' (lec10ex1.c:16:10)
INFO: [HLS 200-472] Inferring partial write operation for 'arr' (lec10ex1.c:21:10)
INFO: [HLS 200-111] Finished Architecture Synthesis Time (s): cpu = 00:00:16 ; elapsed = 00:00:16 . Memory (MB): pea
INFO: [HLS 200-10] Starting hardware synthesis ...
INFO: [HLS 200-10] Synthesizing 'lec10ex1' ...
```

Synthesis output printouts – 2/2

Vivado HLS Console

```
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'lec10ex1'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-11] Finished scheduling.
INFO: [HLS 200-111]  Elapsed time: 15.98 seconds; current allocated memory: 138.226 MB.
INFO: [BIND 205-100] Starting micro-architecture generation ...
INFO: [BIND 205-101] Performing variable lifetime analysis.
INFO: [BIND 205-101] Exploring resource sharing.
INFO: [BIND 205-101] Binding ...
INFO: [BIND 205-100] Finished micro-architecture generation.
INFO: [HLS 200-111]  Elapsed time: 0.04 seconds; current allocated memory: 138.392 MB.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-10] -- Generating RTL for module 'lec10ex1'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [RTGEN 206-500] Setting interface mode on port 'lec10ex1/y' to 'ap_vld'.
INFO: [RTGEN 206-500] Setting interface mode on port 'lec10ex1/c' to 'ap_memory'.
INFO: [RTGEN 206-500] Setting interface mode on port 'lec10ex1/x' to 'ap_none'.
INFO: [RTGEN 206-500] Setting interface mode on function 'lec10ex1' to 'ap_ctrl_hs'.
INFO: [RTGEN 206-100] Finished creating RTL model for 'lec10ex1'.
INFO: [HLS 200-111]  Elapsed time: 0.06 seconds; current allocated memory: 138.740 MB.
INFO: [HLS 200-790] **** Loop Constraint Status: All loop constraints were satisfied.
INFO: [HLS 200-789] **** Estimated Fmax: 173.25 MHz
INFO: [RTMG 210-278] Implementing memory 'lec10ex1_arr_ram (RAM)' using distributed RAMs with power-on initializati
INFO: [HLS 200-111] Finished generating all RTL models Time (s): cpu = 00:00:17 ; elapsed = 00:00:17 . Memory (MB):
INFO: [VHDL 208-304] Generating VHDL RTL for lec10ex1.
INFO: [VLOG 209-307] Generating Verilog RTL for lec10ex1.
INFO: [HLS 200-112] Total elapsed time: 17.1 seconds; peak allocated memory: 138.740 MB.
Finished C synthesis.
```

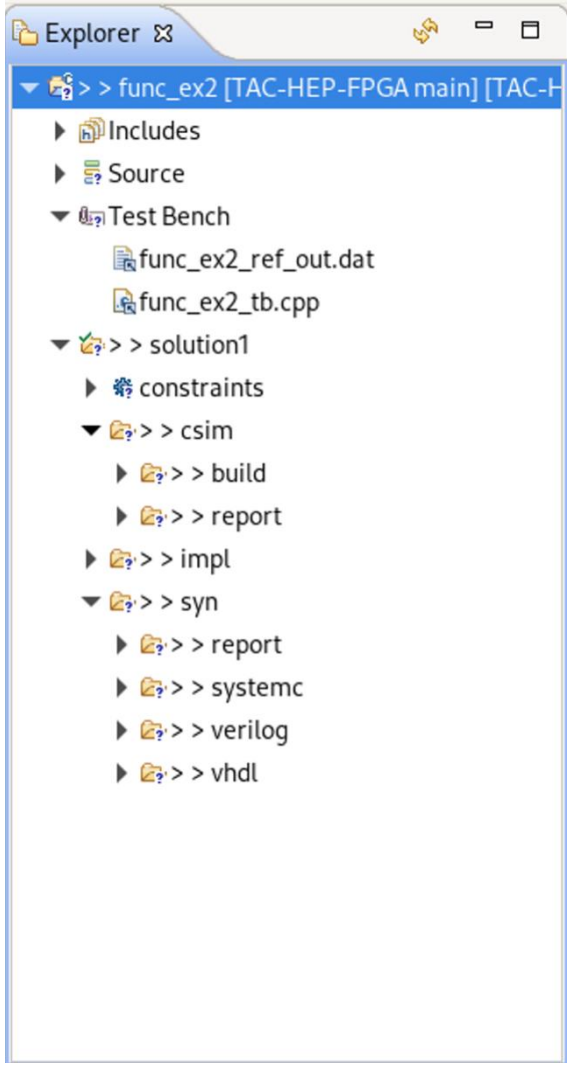
Step-3: Synthesis Report Review

The **syn** folder contains four sub-folder

A report folder and one folder for each of the RTL output formats.

The report folder contains a report file for the **top-level function** and one for every sub-function in the design

The **verilog**, **vhdl**, and **systemc** folders contain the output RTL files



Synthesis Report

Performance of the target device for desired algorithm

func_ex2_csim.log | Synthesis(solution1)(func_ex2_csynth.rpt)

Synthesis Report for 'func_ex2'

General Information

Date:	Tue Mar 3 11:57:41 2026
Version:	2020.1 (Build 2897737 on Wed May 27 20:21:37 MDT 2020)
Project:	func_ex2
Solution:	solution1
Product family:	virtexuplus
Target device:	xcvu13p-fsga2577-2-e

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	25.00 ns	4.644 ns	3.12 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
34	34	0.850 us	0.850 us	34	34	none

Detail

- Instance
- Loop

Synthesis Report: Resource Utilization

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	3	0	85	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	0	-	64	6	0
Multiplexer	-	-	-	105	-
Register	-	-	111	-	-
Total	0	3	175	196	0
Available	5376	12288	3456000	1728000	1280
Available SLR	1344	3072	864000	432000	320
Utilization (%)	0	~0	~0	~0	0
Utilization SLR (%)	0	~0	~0	~0	0

Homework

```
1   #include <iostream>
2   #define N 10
3
4   void matrix_add(int A[N][N], int B[N][N], int C[N][N]) {
5
6       for (int i = 0; i < N; i++) {
7           for (int j = 0; j < N; j++) {
8               C[i][j] = A[i][j] + B[i][j];
9           }
10      }
11  }
```

1. Run on HLS for the above example of matrix addition and check the resource utilization
2. Compare with a vector addition and matrix multilplication.

Share your resource utilization for all three cases and your observation about the same.

TAC-HEP 2026

Questions?

TAC-HEP 2026

Extra Slides

Jargons

- **ICs - Integrated chip:** assembly of hundreds of millions of transistors on a minor chip
- **PCB:** Printed Circuit Board
- **LUT - Look Up Table aka 'logic'** - generic functions on small bitwidth inputs. Combine many to build the algorithm
- **FF - Flip Flops** - control the flow of data with the clock pulse. Used to build the pipeline and achieve high throughput
- **DSP - Digital Signal Processor** - performs multiplication and other arithmetic in the FPGA
- **BRAM - Block RAM** - hardened RAM resource. More efficient memories than using LUTs for more than a few elements
- **PCIe or PCI-E - Peripheral Component Interconnect Express**: is a serial expansion bus standard for connecting a computer to one or more peripheral devices
- **InfiniBand** is a computer networking communications standard used in high-performance computing that features very high throughput and very low latency
- **HLS** - High Level Synthesis - compiler for C, C++, SystemC into FPGA IP cores
- **HDL** - Hardware Description Language - low level language for describing circuits
- **RTL** - Register Transfer Level - the very low level description of the function and connection of logic gates
- **FIFO** – First In First Out memory
- **Latency** - time between starting processing and receiving the result
  - Measured in clock cycles or seconds
- **II - Initiation Interval** - time from accepting first input to accepting next input