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

GPU & FPGA module training: Part-2

Week-2: Introduction to Vivado HLS, Setup

Lecture-4: March 29th 2023

Varun Sharma

University of Wisconsin – Madison, USA

WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

So Far...

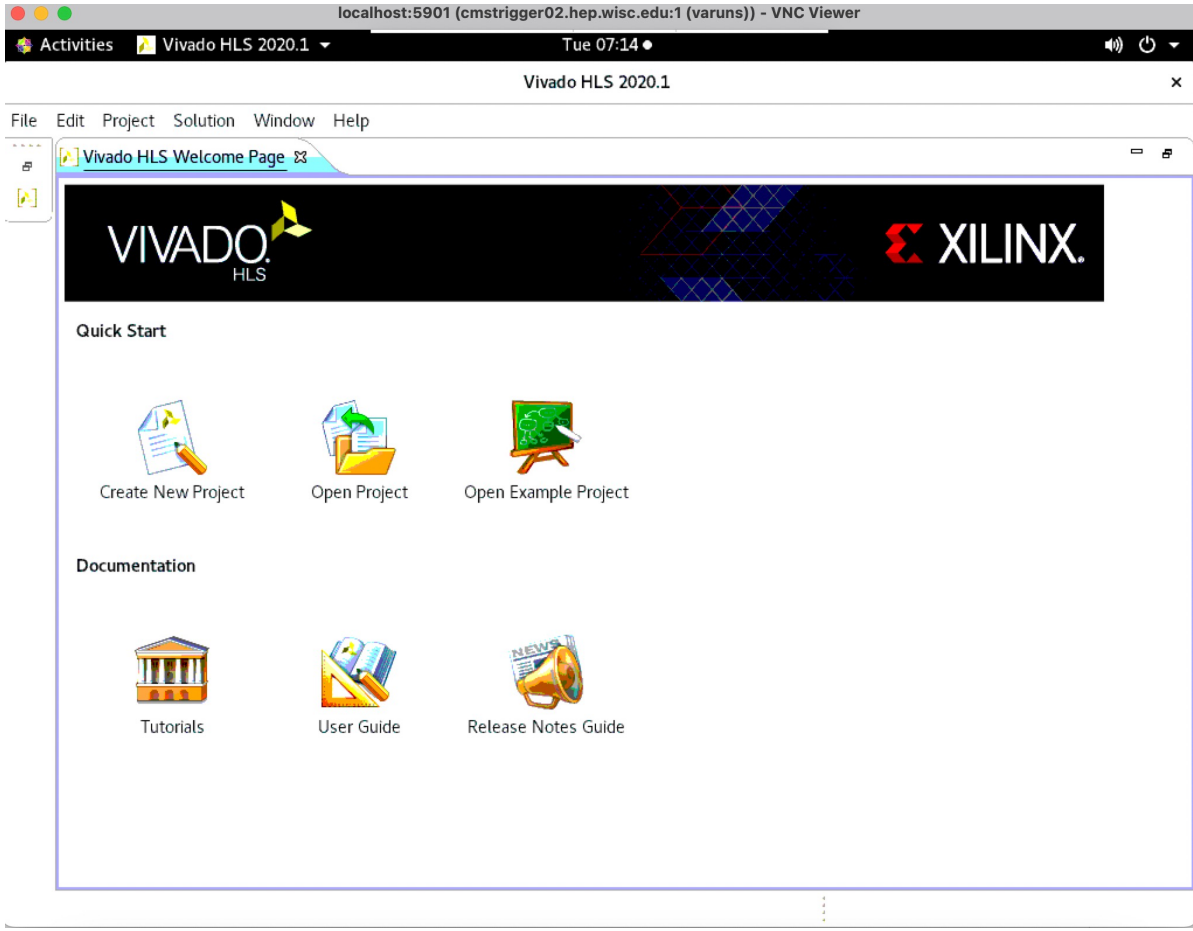
- **FPGA and its architecture**
  - Registor/Flip-Flops, LUTs/Logic Cells, DSP, BRAMs
  - Clock Frequency, Latency
  - Extracting control logic & Implementing I/O ports
- **Parallelism in FPGA**
  - Scheduling, Pipelining, DataFlow

Today:

- Vivado HLS setup
- Introduction to Vivado HLS
- Some Vivado HLS hands-on

Are you ready?

- ssh varuns@cmstrigger02-via-login -L5901:localhost:5901
  - Or whatever ***:1*** display number
  - Sometimes you may need to run vncserver -localhost -geometry 1024x768 again to start new vnc server
- Connect to VNC server (remote desktop) client
- Open terminal
- Source /opt/Xilinx/Vivado/2020.1/settings64.sh
- vivado_hls



Vivado HLS

- Vivado 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)

- 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

Vivado HLS

- The Xilinx Vivado 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 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 Vivado 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

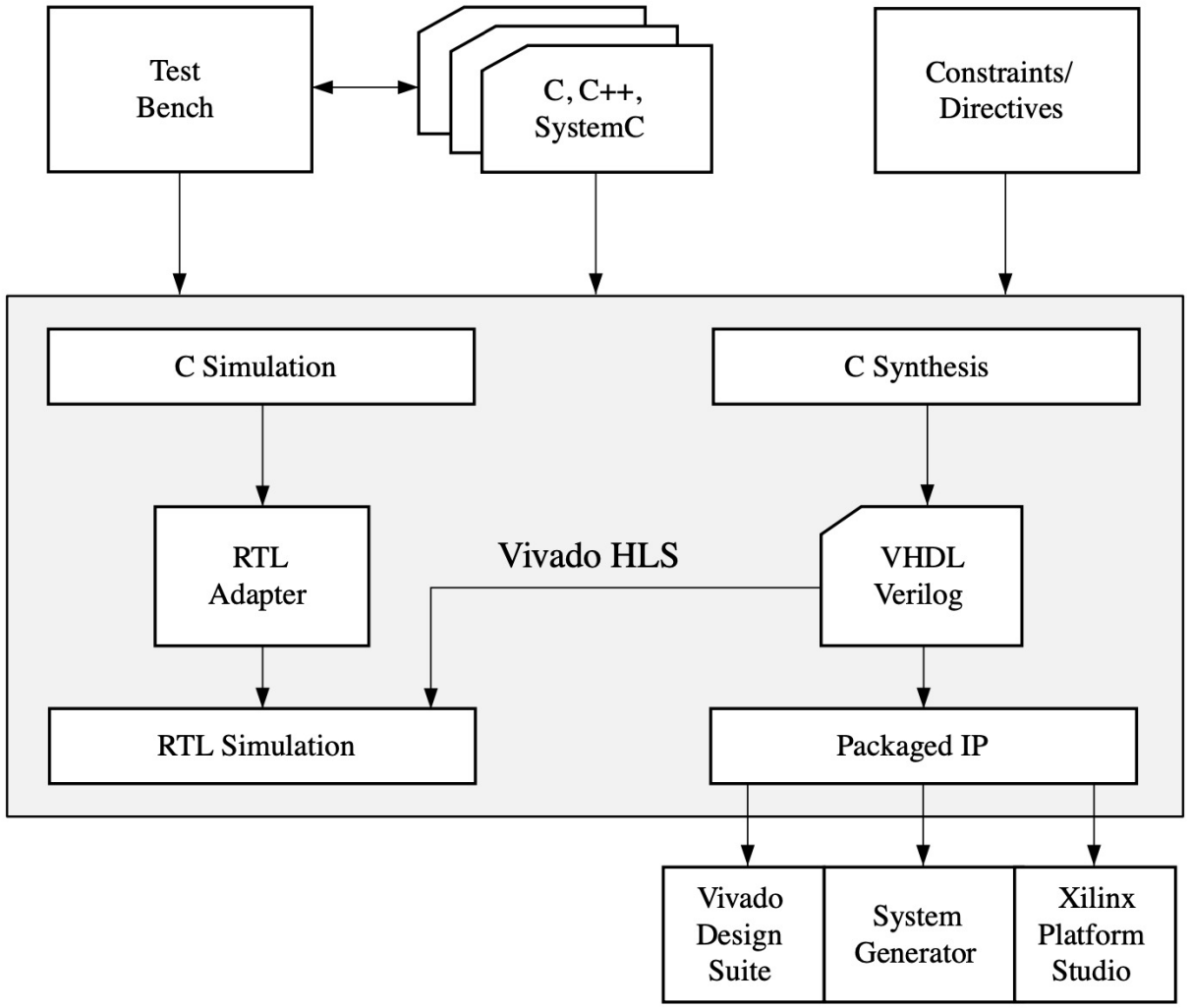
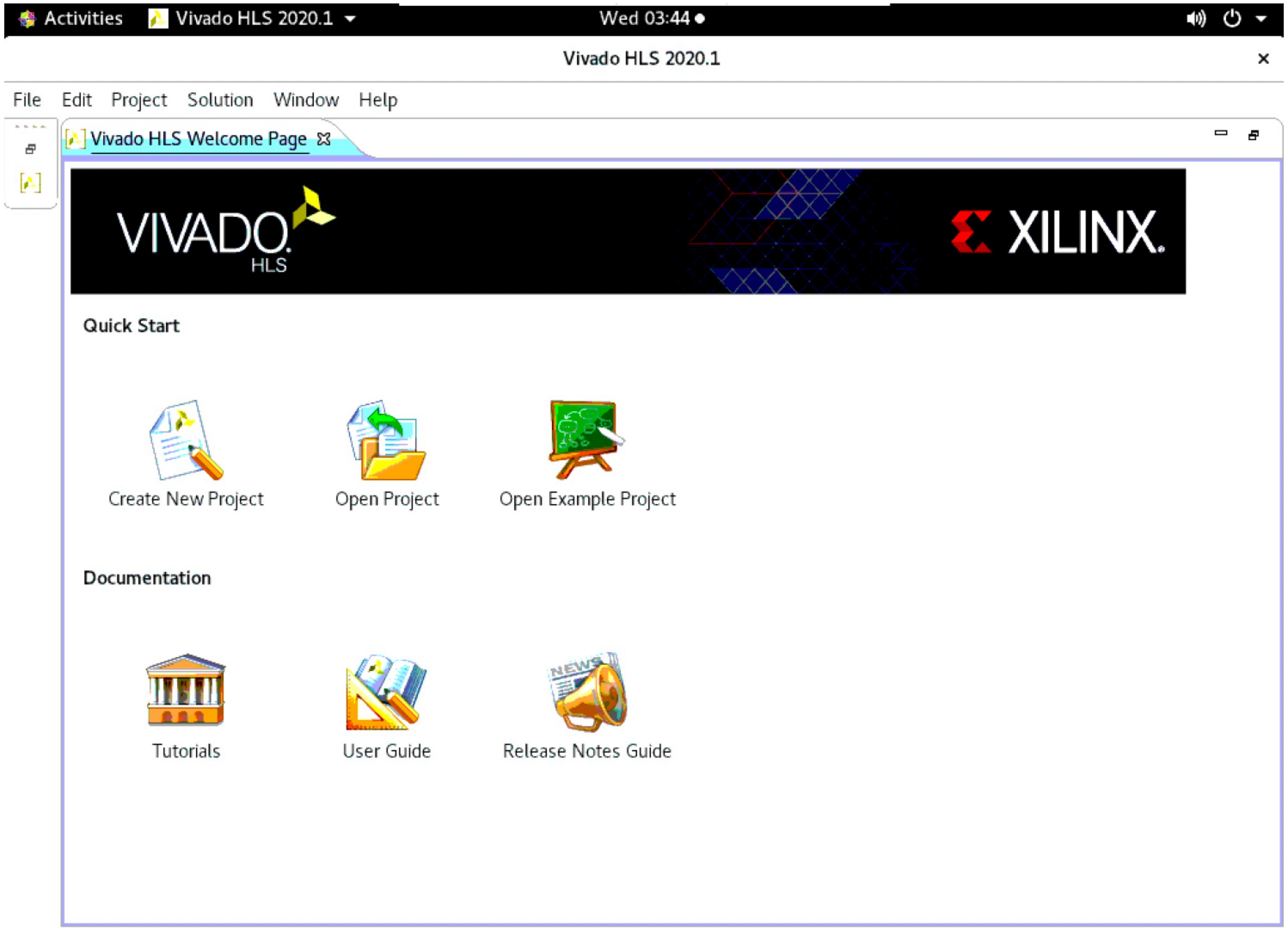


Fig.

TAC-HEP 2023

Using Vivado HLS

Using Vivado HLS

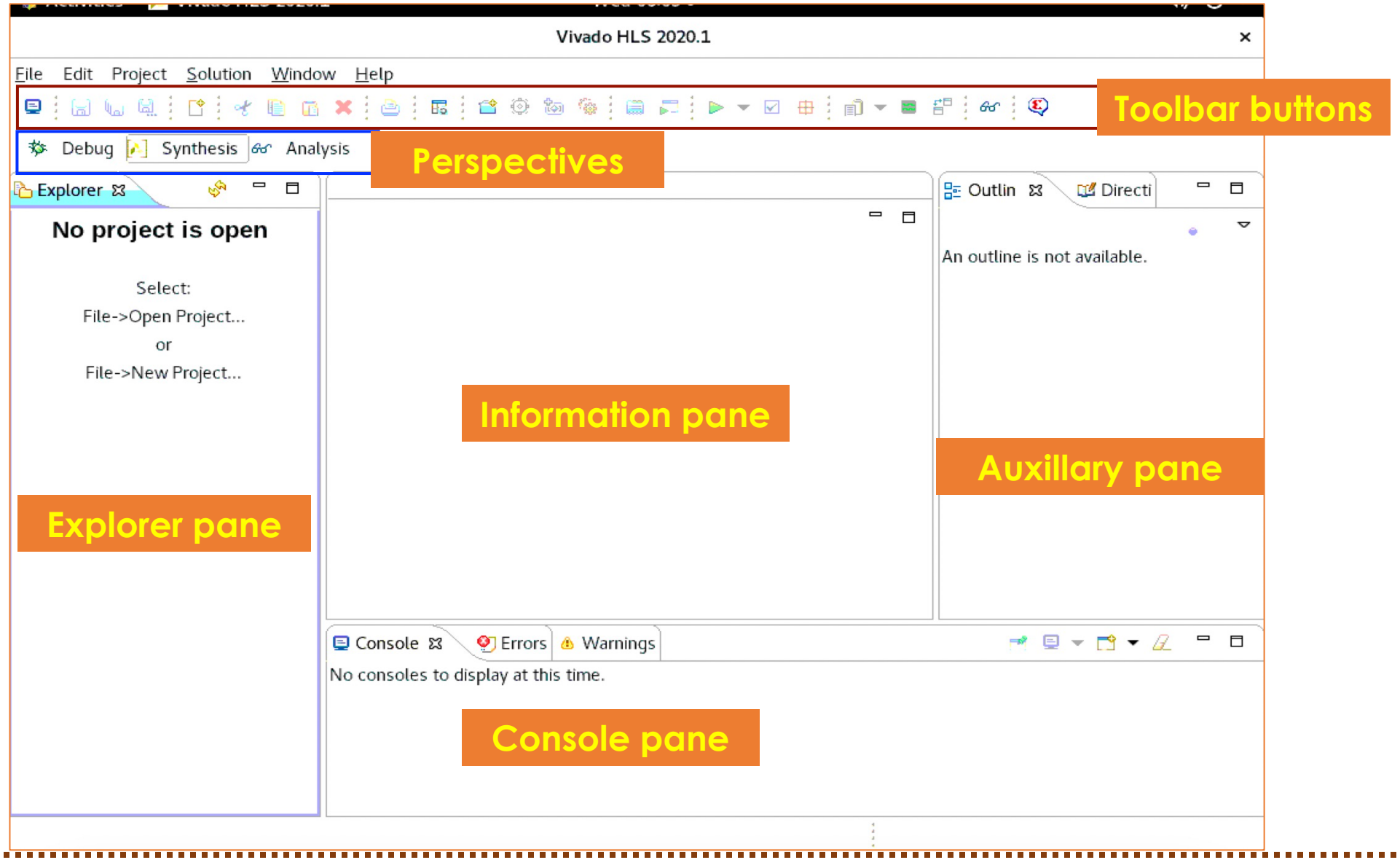


Once connected to cmstrigger02

Source the settings

Execute *vivado_hls*

Understanding GUI



Understanding GUI

Explorer Pane

- Shows the project hierarchy.
- As you proceed through the validation, synthesis, verification, and IP packaging steps, sub-folders with the results of each step are created automatically inside the solution directory (named csim, syn, sim, and impl respectively)

Information Pane

- Shows the contents of any files opened from the Explorer pane.
- When operations complete, the report file opens automatically in this pane

Auxiliary Pane

- Cross-links with the Information pane.
- The information shown in this pane dynamically adjusts, depending on the file open in the Information pane

Console Pane

- Shows the messages produced when Vivado HLS runs.
- Errors and warnings appear in Console pane tabs

Understanding GUI

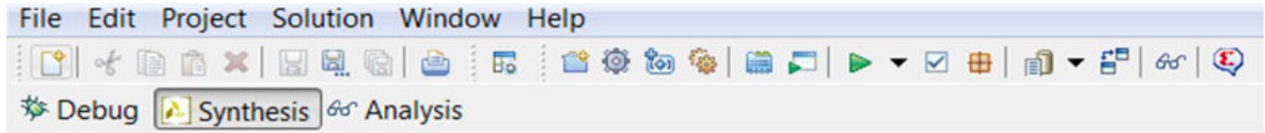
Toolbar Buttons

- Can perform the most common operations using the Toolbar buttons.
- When you hold the cursor over the button, a popup dialog box opens, explaining the function.
- Each button also has an associated menu item available from the pulldown menu

Perspectives

- *Synthesis Perspective:* Allows to synthesize designs, run simulations, and package the IP.
- *Debug Perspective:* Includes panes associated with debugging the C code.
  - Can be used only after the C code compiles
- *Analysis Perspective:* Windows in this perspective are configured to support analysis of synthesis results.
  - Can be used only after synthesis completes.

Toolbar buttons



- **Create New Project** opens the new project wizard
- **Project Settings** allows the current project settings to be modified
- **New Solution** opens the new solution dialog box
- **Solution Settings** allows the current solution settings to be modified

The next group of toolbar buttons control the tool operation:

- **Index C Source** refreshes the annotations in the C source
- **Run C Simulation** opens the C Simulation dialog box
- **C Synthesis** starts C source code in Vivado HLS
- **Run C/RTL Cosimulation** verifies the RTL output
- **Export RTL** packages the RTL into the desired IP output format

The final group of toolbar buttons are for design analysis:

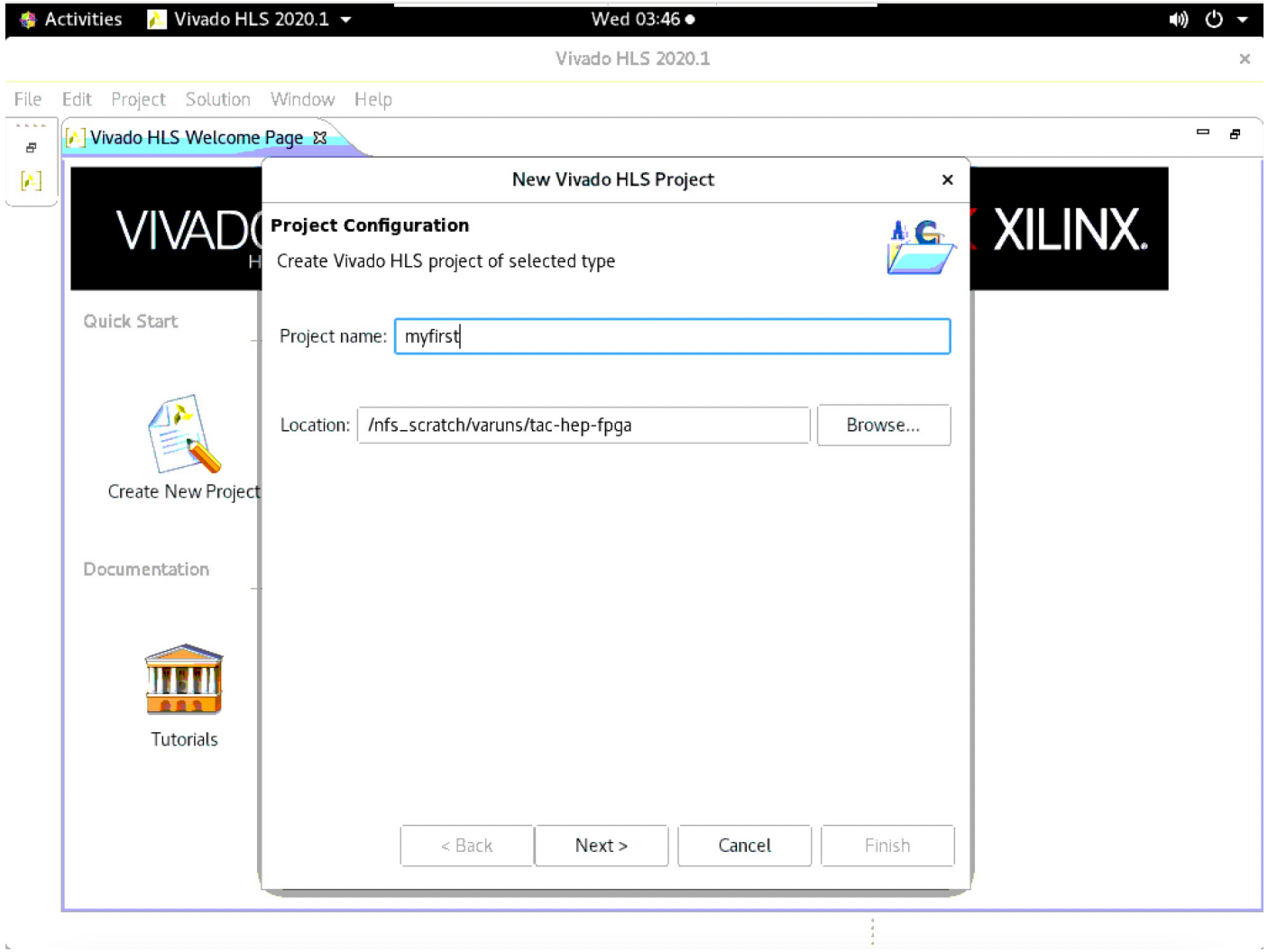
- **Open Report** opens the C synthesis report or drops down to open other reports
- **Compare Reports** allows the reports from different solutions to be compared

Steps to IP Creation

- **Step-1: Creating a New Project/Opening an existing project**
- **Step-2: Validating the C-source code**
- **Step-3: High Level Synthesis**
- **Step-4: RTL Verification**
- **Step-5: IP Creation**

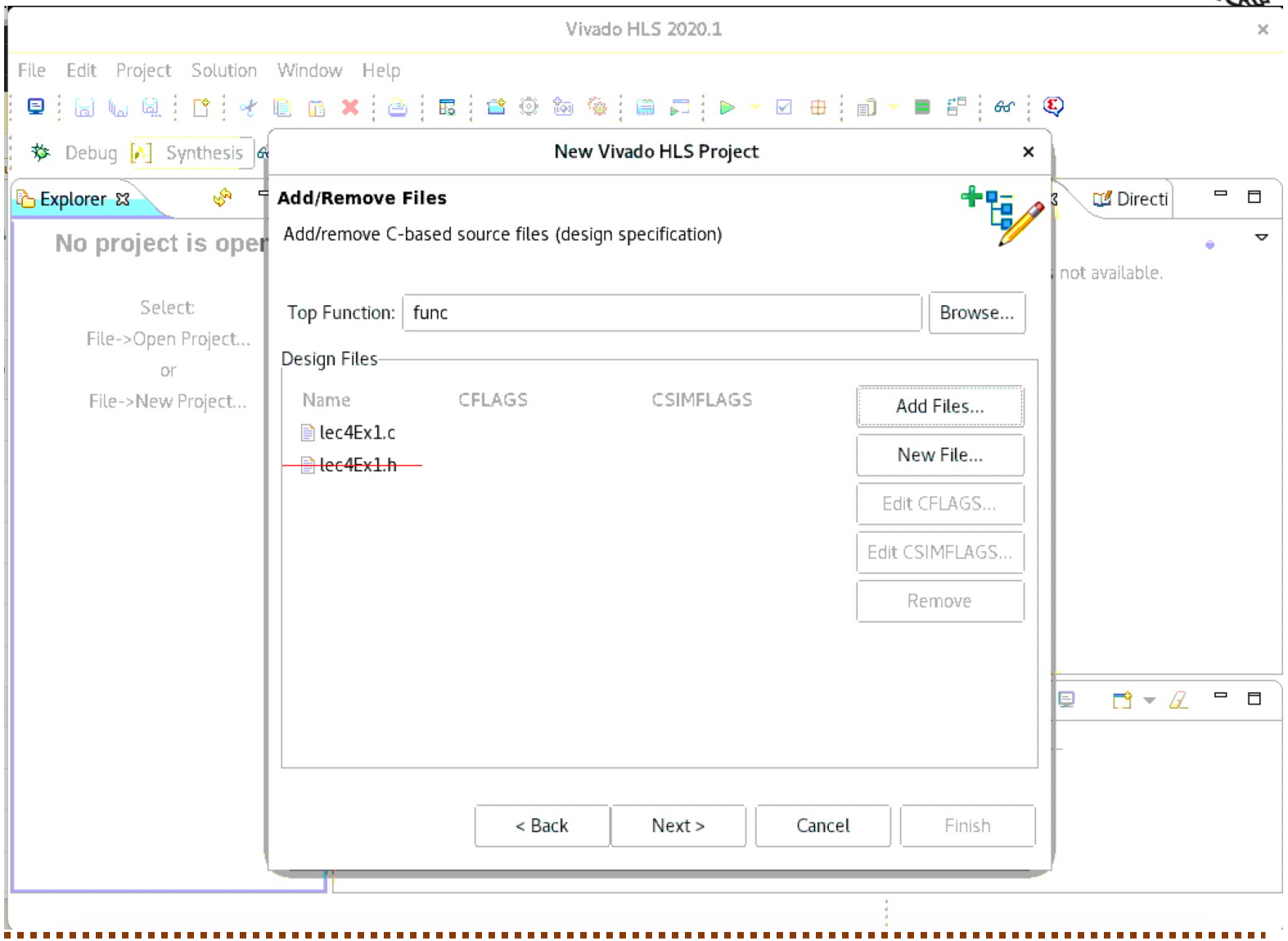
Step-1: Creating Project

- Create New Project
- Enter the project name
- Enter the location to be used for your project

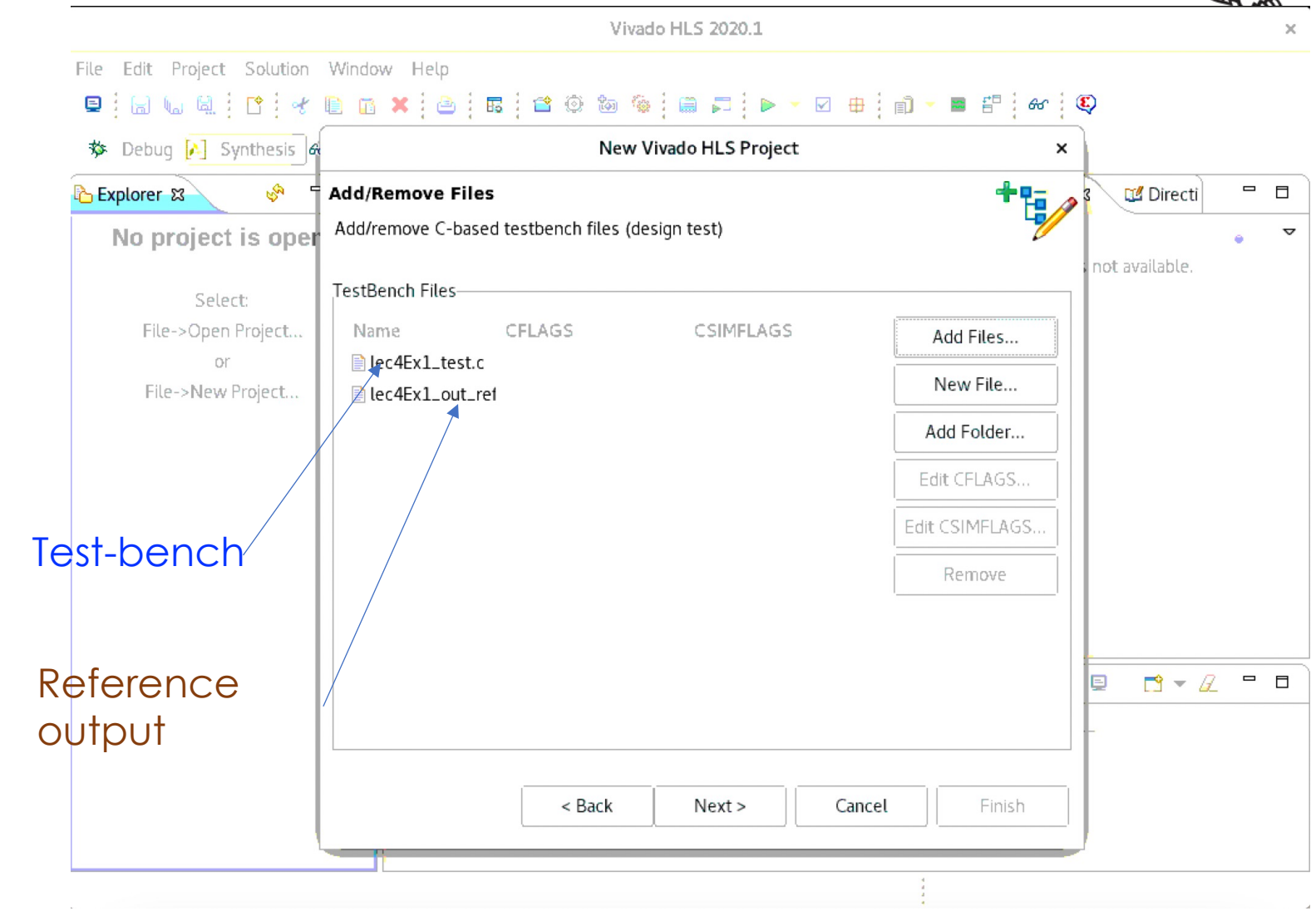


Step-2: Adding C-design

- Enter information to specify the **C design** files (no header files)
- Specify the **top-level function** to be synthesised



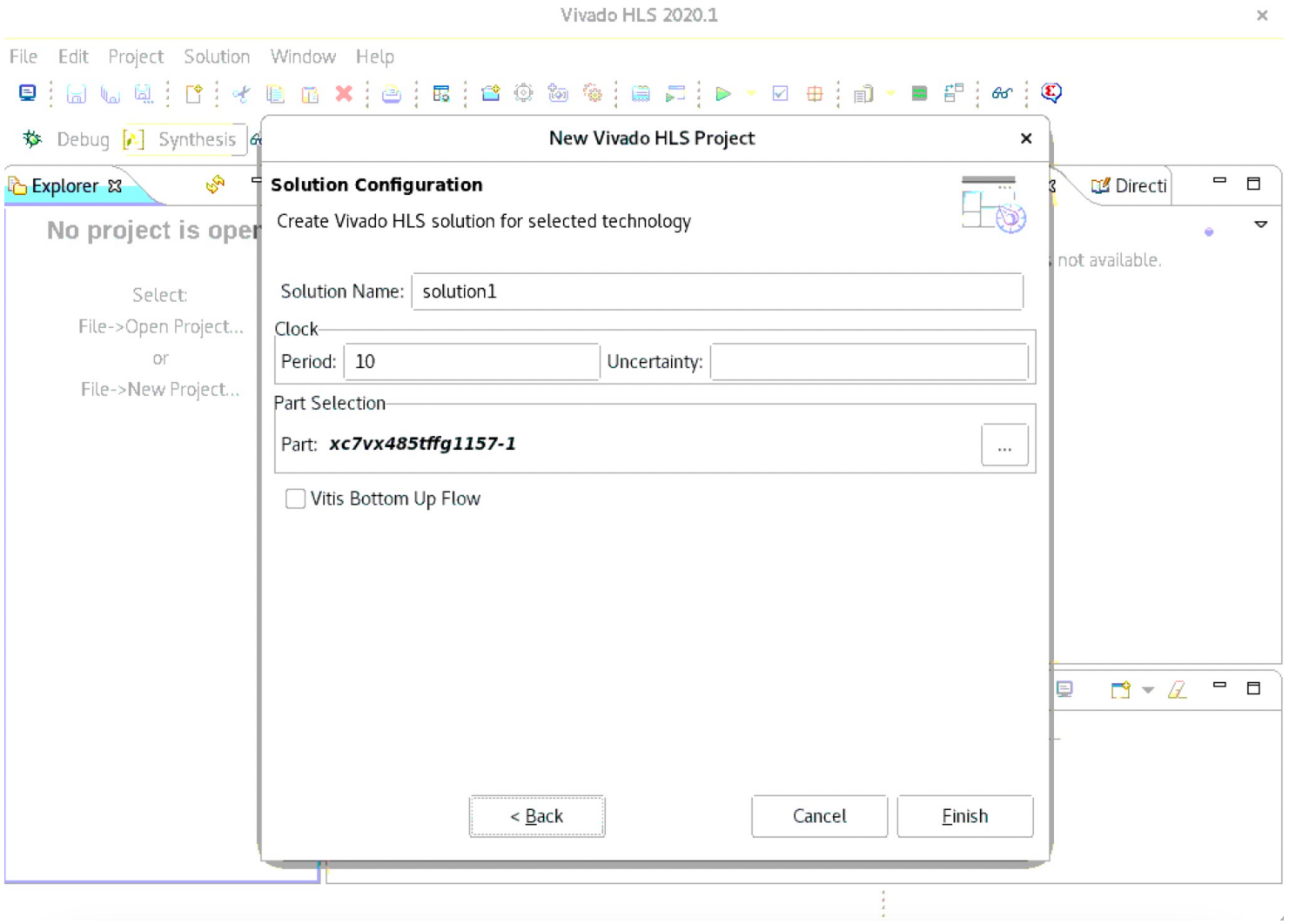
Step-2: Validation



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

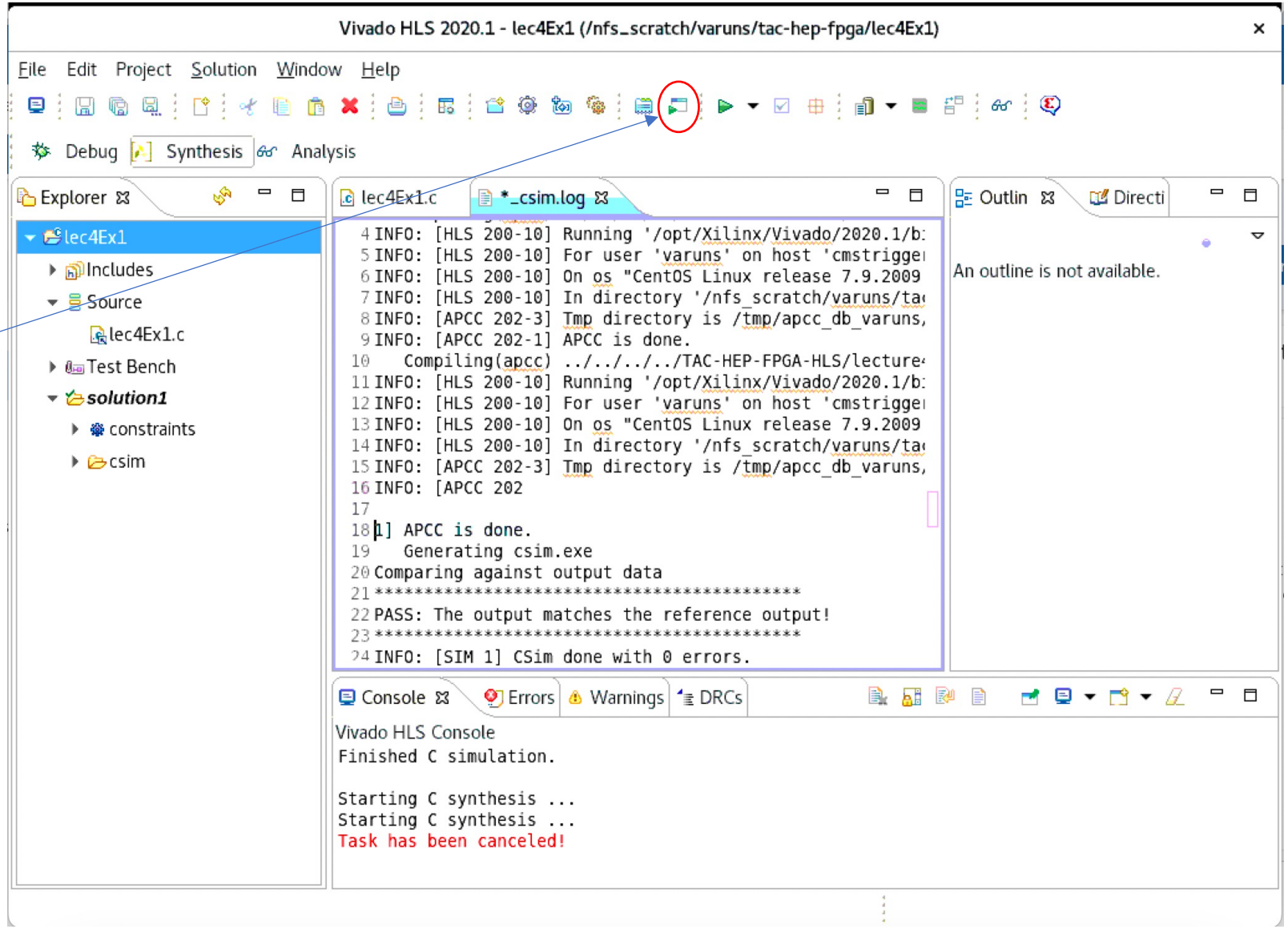
Step-2: Constraints

- **Solution Name**: of your choice
- **Clock Period**: in units of ns or a frequency value specified with the MHz suffix
- **Uncertainty**: If no value is given, default is 12.5%
- **Part**: Click to select the appropriate technology
  - Eg. xc7vx485tffg1157



Step-2: C-Simulation

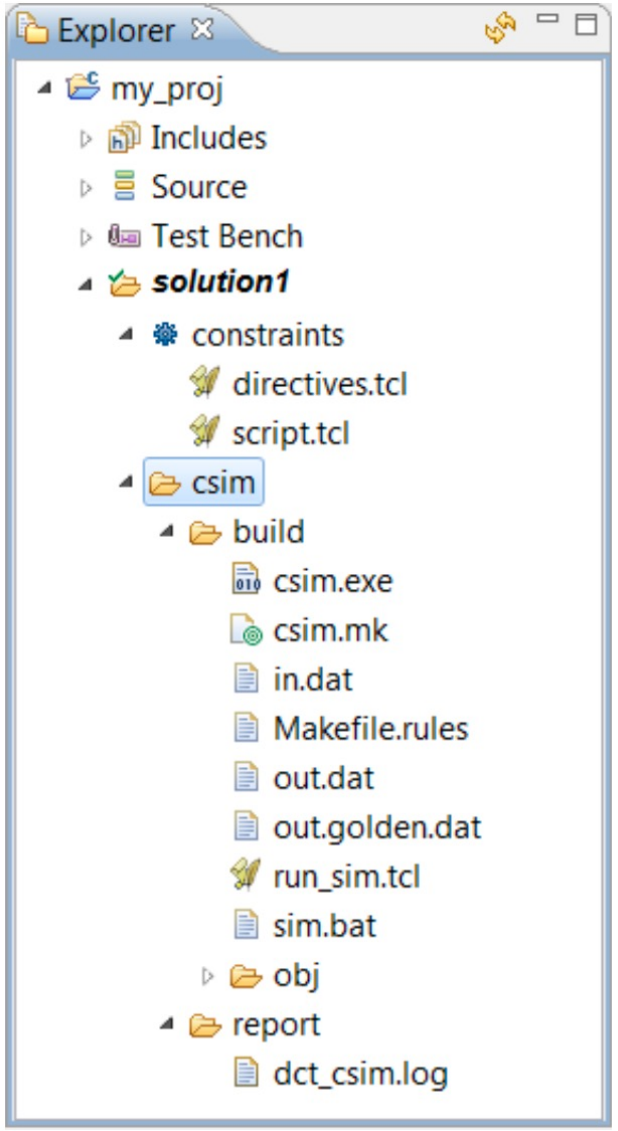
Run c-simulation



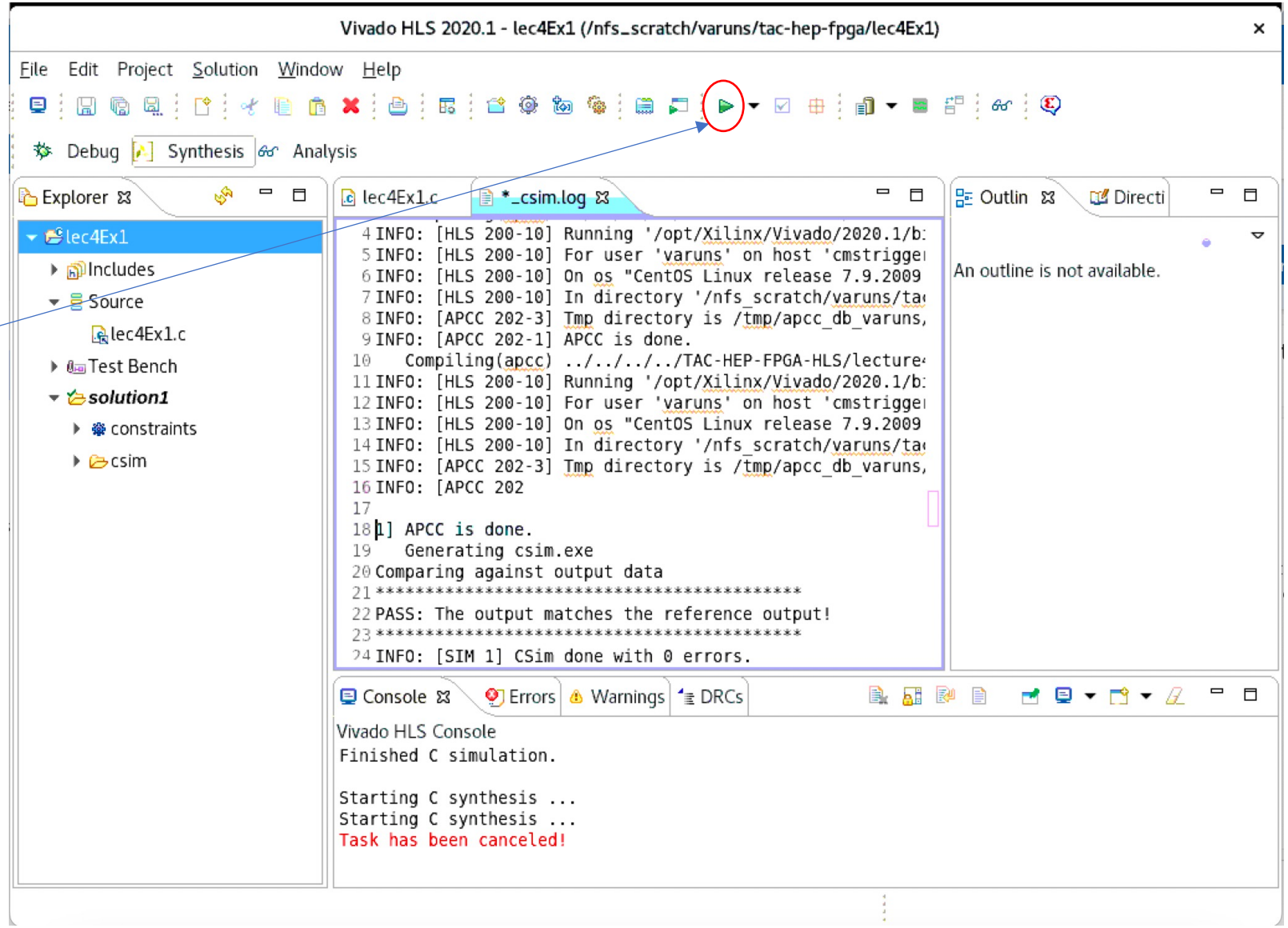
Step-2: C-Simulation Complete

C-simulation complete and ready to be synthesized

```
INFO: [HLS 200-10] Running '/opt/Xilinx/Vivado/2020.1/b
INFO: [HLS 200-10] For user 'varuns' on host 'cmstrigge
INFO: [HLS 200-10] On os "CentOS Linux release 7.9.2009
INFO: [HLS 200-10] In directory '/nfs_scratch/varuns/ta
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_varuns,
INFO: [APCC 202-1] APCC is done.
   Compiling(apcc) ../../../../TAC-HEP-FPGA-HLS/lecture
INFO: [HLS 200-10] Running '/opt/Xilinx/Vivado/2020.1/b
INFO: [HLS 200-10] For user 'varuns' on host 'cmstrigge
INFO: [HLS 200-10] On os "CentOS Linux release 7.9.2009
INFO: [HLS 200-10] In directory '/nfs_scratch/varuns/ta
INFO: [APCC 202-3] Tmp directory is /tmp/apcc_db_varuns,
INFO: [APCC 202
1] APCC is done.
   Generating csim.exe
Comparing against output data
*********************************************
PASS: The output matches the reference output!
*********************************************
INFO: [SIM 1] CSim done with 0 errors.
```

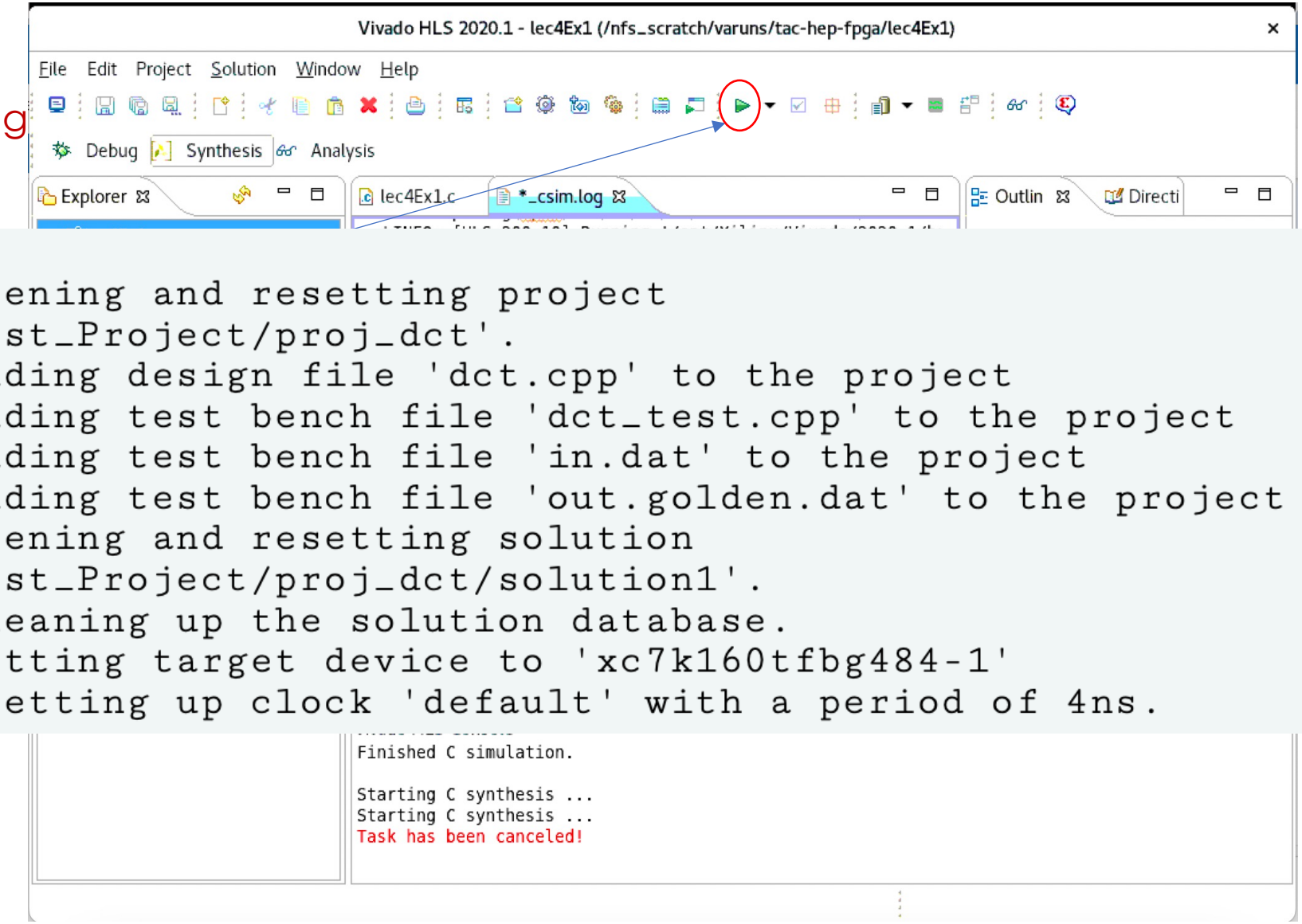


Step-3: Synthesis



Run Synthesis

Step-3: Synthesis Progressing

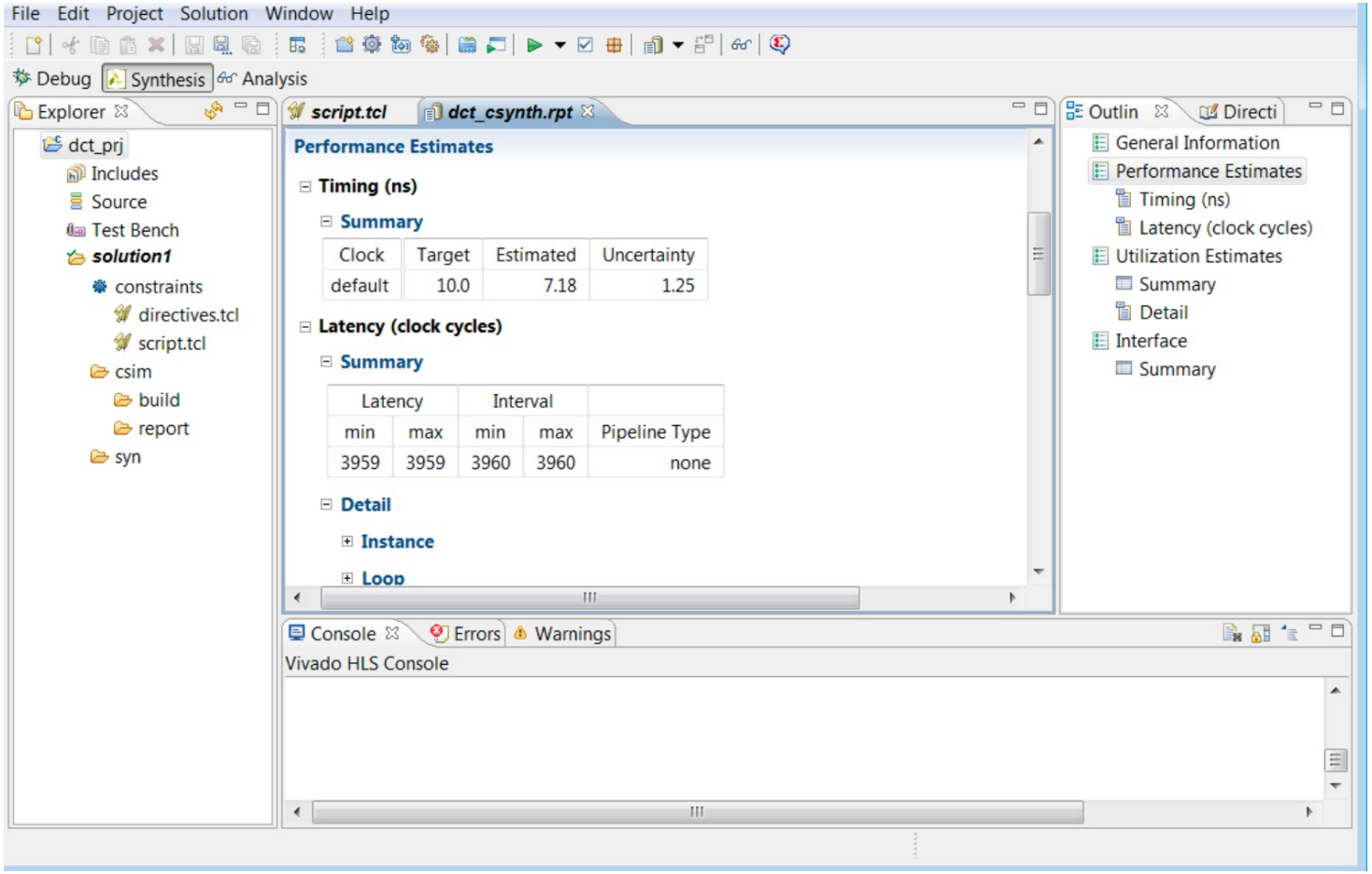


Synthesis process is proceeding

```
INFO: [HLS 200-10] Opening and resetting project
'C:/Vivado_HLS/My_First_Project/proj_dct'.
INFO: [HLS 200-10] Adding design file 'dct.cpp' to the project
INFO: [HLS 200-10] Adding test bench file 'dct_test.cpp' to the project
INFO: [HLS 200-10] Adding test bench file 'in.dat' to the project
INFO: [HLS 200-10] Adding test bench file 'out.golden.dat' to the project
INFO: [HLS 200-10] Opening and resetting solution
'C:/Vivado_HLS/My_First_Project/proj_dct/solution1'.
INFO: [HLS 200-10] Cleaning up the solution database.
INFO: [HLS 200-10] Setting target device to 'xc7k160tfbg484-1'
INFO: [SYN 201-201] Setting up clock 'default' with a period of 4ns.
```

Step-3: Synthesis Report

Performance of the target device for desired algorithm



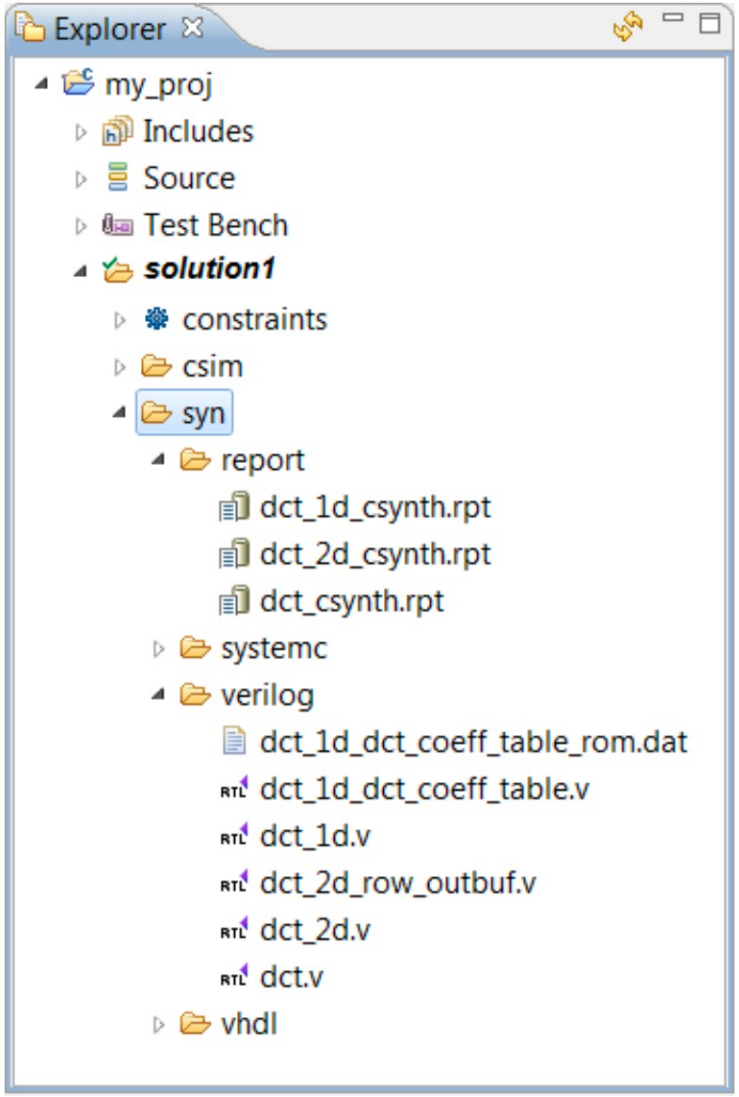
Step-3: Synthesis Report Review

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



TAC-HEP 2023

Example

Example file

- Git clone: https://github.com/varuns23/TAC-HEP-FPGA-HLS.git

lec4Ex1.c	example for lecture-4	1 minute ago
lec4Ex1.h	example for lecture-4	1 minute ago
lec4Ex1_test.c	example for lecture-4	1 minute ago
out_ref.dat	example for lecture-4	1 minute ago

Example program

Src (.C) file

26 lines (23 sloc) | 348 Bytes

```
1   #include "lec4Ex1.h"
2
3   void lec4Ex1 (
4     int *y,
5     int c[N],
6     int x
7     ) {
8
9      static int arr[N];
10     int sum;
11     int data;
12     int i;
13
14     sum=0;
15  Loop: for (i=N-1;i>=0;i--) {
16           if (i==0) {
17              arr[0]=x;
18              data = x;
19           } else {
20              arr[i]=arr[i-1];
21              data = arr[i];
22           }
23           sum+=data*c[i];;
24        }
25        *y=sum;
26  }
```

Header (.h) file

11 lines (9 sloc) | 123 Bytes

```
1   #ifndef LEC4EX1_H_
2   #define LEC4EX1_H_
3   #define N       11
4
5   void lec4Ex1 (
6     int *y,
7     int c[N+1],
8     int x
9     );
10
11  #endif
```

Test bench file

49 lines (40 sloc) | 1.21 KB

```
1   #include <stdio.h>
2   #include <math.h>
3   #include "lec4Ex1.h"
4
5   int main () {
6     const int    samples=600;
7     FILE         *oFile;
8
9     int inp, output;
10    int coef[N] = {0,-10,-9,23,56,63,56,23,-9,-10,0,
11
12    int i, rmp;
13    inp = 0;
14    rmp = 1;
15
16    oFile=fopen("out.dat","w");
17    for (i=0;i<=samples;i++) {
18          if (rmp == 1)
19                inp = inp + 1;
20          else
21                inp = inp - 1;
22
23          // Execute the function with latest input
24      lec4Ex1(&output,coef,inp);
25
26      if ((rmp == 1) && (inp >= 75))
27          rmp = 0;
28      else if ((rmp == 0) && (inp <= -75))
29          rmp = 1;
30
31          // Save the results.
32      fprintf(oFile,"%i %d %d\n",i,inp,output);
33    }
34    fclose(oFile);
```

Assignment

- Use the code on slide-28 and follow steps:1-3 (slide-17-26) to get the synthesis report

- Use target device: **xc7k160tfbg484-2**
- Clock period of 10ns

TAC-HEP 2023

Questions?

TAC-HEP 2023

Additional material

Correct Time

From 03.28.2023 onwards

- Tuesdays: 9:00-10:00 CT / 10:00-11:00 ET / 16:00-17:00 CET
- Wednesday: 11:00-12:00 CT / 12:00-13:00 ET / 18:00-19:00 CET

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

TAC-HEP 2023

HLS Setup

- Xilinx Vivado HLS has a graphical user interface that we intend to use
- The goal is to run vivado_hls on *cmstrigger02* machine but be able to do so remotely
- So, we want to display the *cmstrigger02* screen on your desktop (Mac or Windows or Linux)
- In principle one can use X-Windows directly. However, that will be very slow over WAN
- Therefore, we suggest using a VNC server on cmstrigger02 and a remote machine

Connecting to cmstrigger02

- Connect to login machine:
  - ssh -X -Y <username>@login.hep.wisc.edu

- From 'login' machine connect to 'cmstrigger02' machine - All of you should have access
  - ssh cmstrigger02
  - mkdir /nfs_scratch/`whoami` (If directory exist, go to next bullet)
  - cd /nfs_scratch/`whoami`

VNC Server setup

One time only

- Log into cmstrigger02
- Set your VNC password using the linux command: vncpasswd
  - **Do NOT use an important password** here, as it is NOT secure
- Follow this instruction at http://red.ht/1fSVIUc to set up your X-Windows session
- Namely, you need to create a file ~/.vnc/xstartup with content:

```
#!/bin/sh
# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
#xsetroot -solid grey
#vncconfig -iconic &
#xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#twm &
if test -z "$DBUS_SESSION_BUS_ADDRESS" ; then
    eval `dbus-launch --sh-syntax ?exit-with-session`
    echo "D-BUS per-session daemon address is: \
    $DBUS_SESSION_BUS_ADDRESS"
fi
exec gnome-session
```

Can be copied from above link as well

- You need to set execute permission for the startup file
  - chmod +x ~/.vnc/xstartup

Setting direct tunnelling

One time only

- Add to your (laptop or computer) ~/.ssh/config

```
Host *
  ControlPath ~/.ssh/control/%C
  ControlMaster auto

Host cmstrigger02-via-login
    User varuns
    HostName cmstrigger02.hep.wisc.edu
    ProxyCommand ssh login.hep.wisc.edu nc %h %p

Host *.wisc.edu
    User varuns
```

Replace "varuns" with your <username>

- If all is done correctly, following command should directly take you to *cmstrigger02* machine (enter passwd twice)
  - ssh cmstrigger02-via-login

IP Port forwarding

- Start the VNC server - you do this command after you stopped vncserver by hand or otherwise, using:
  - vncserver -localhost -geometry 1024x768
- This command, vncserver, tells you the number of your X-Windows Display, example *cmstrigger02.hep.wisc.edu:1*, where **:1** is your display

```
[varuns@cmstrigger02 tac-hep-fpga]$ vncserver -localhost -geometry 1024x768


New 'cmstrigger02.hep.wisc.edu:1 (varuns)' desktop is cmstrigger02.hep.wisc.edu:1

Starting applications specified in /afs/hep.wisc.edu/home/varuns/.vnc/xstartup
Log file is /afs/hep.wisc.edu/home/varuns/.vnc/cmstrigger02.hep.wisc.edu:1.log
```

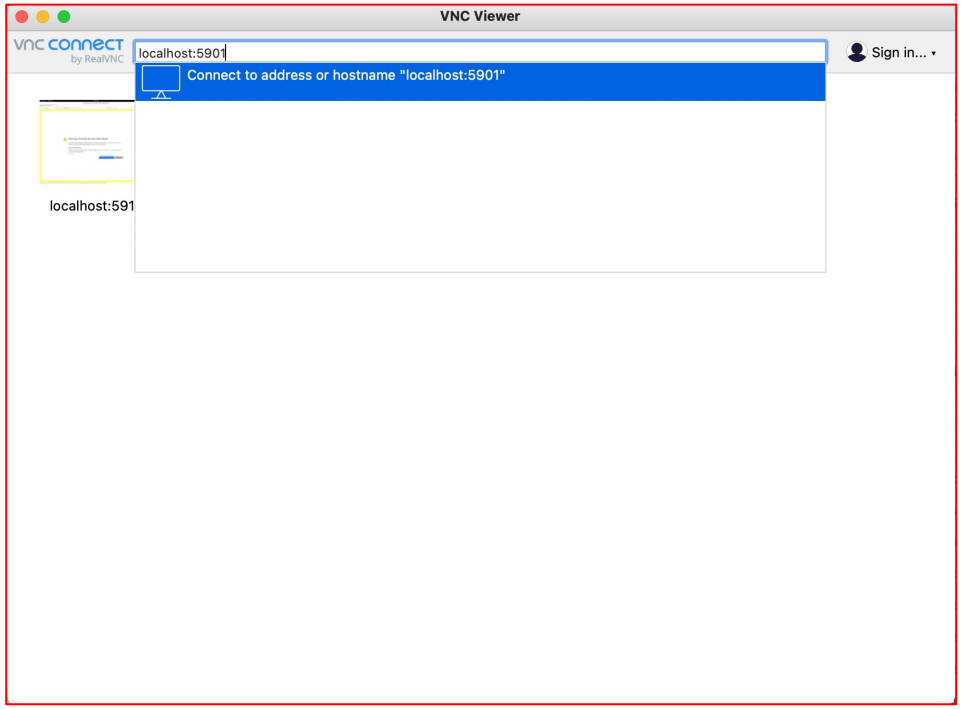
- We use an IP forwarding tunnel to *cmstrigger02.hep.wisc.edu* to see your *cmstrigger02* display on your laptop/desktop. The command to make that magic is:
  - ssh varuns@cmstrigger02-via-login -L5901:localhost:5901
  - Make sure you change "varuns" to your user name, and "5901" to (5900 + your display number), say 5903, if vncserver told you 3!

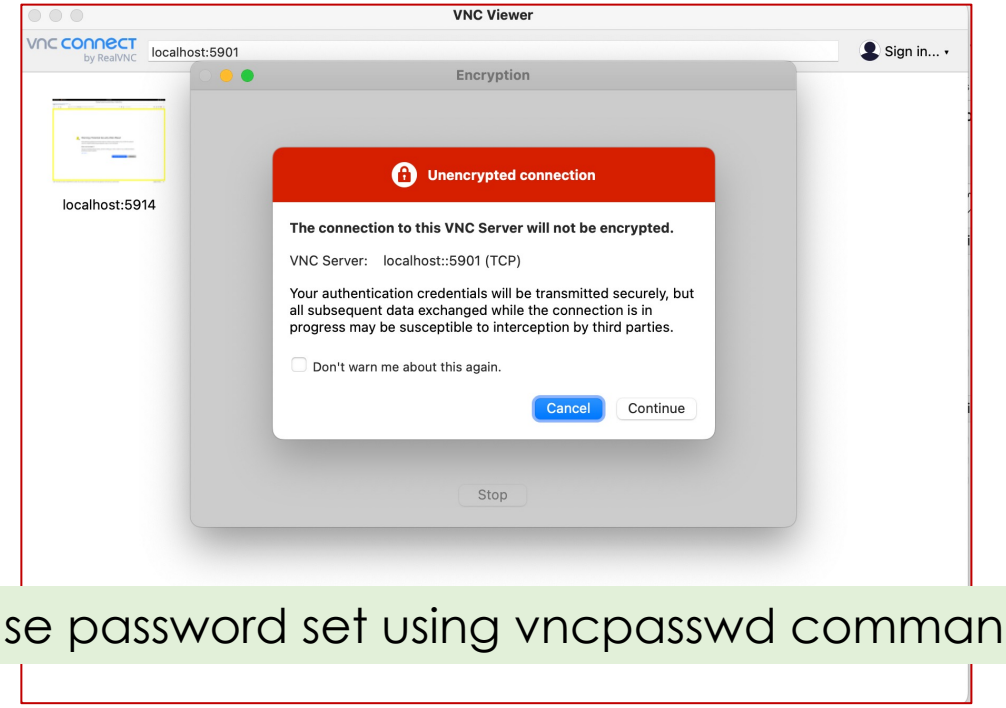
- You can kill your VNC server (:3) using the command:
  - vncserver -kill :1

Remote desktop client

One time only

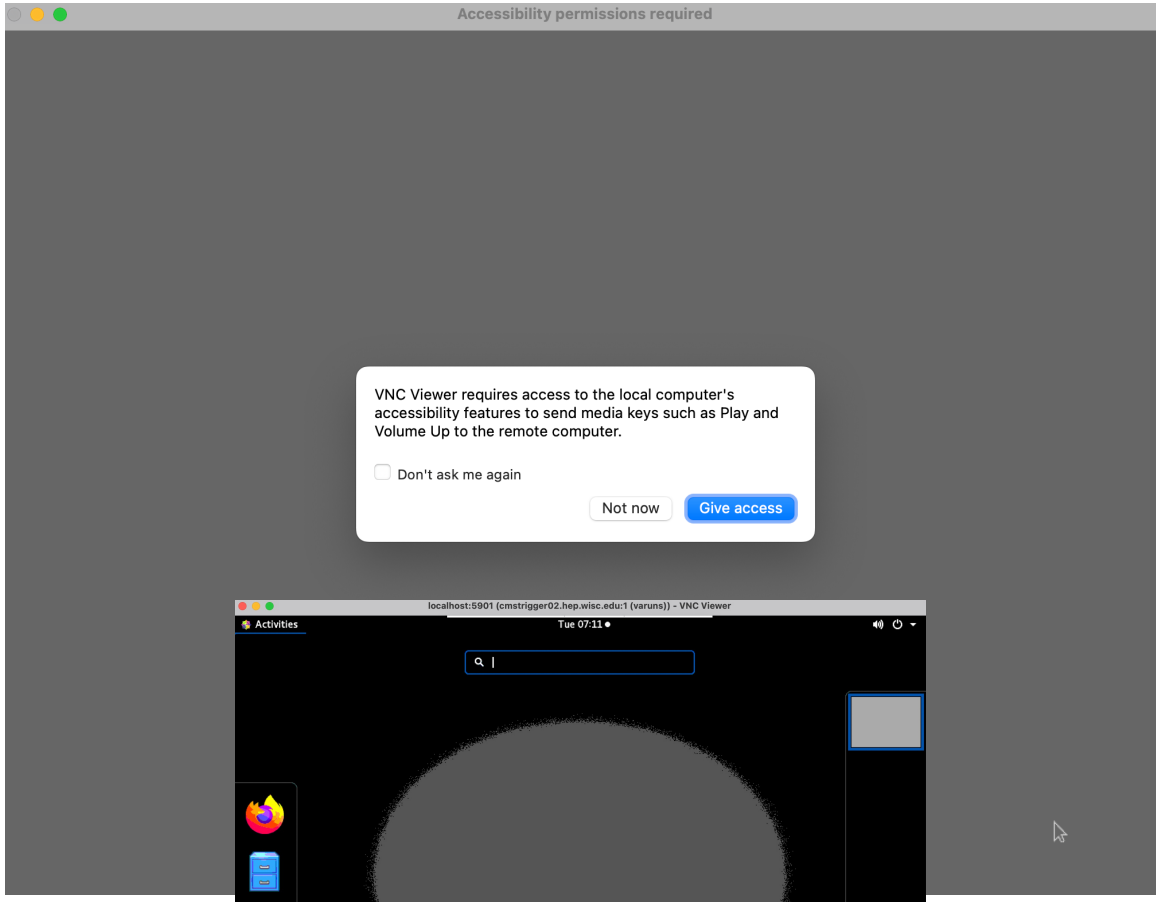
- Download VNC viewer: https://www.realvnc.com/en/connect/download/viewer/
  - You can choose any other remote desktop client but this is one of the stable one that I have used





Use password set using vncpasswd command

Connected...


Accessibility permissions required
VNC Viewer requires access to the local computer's accessibility features to send media keys such as Play and Volume Up to the remote computer.
Don't ask me again
Not now
Give access

localhost:5901 (cmstrigger02.hep.wisc.edu:1 (varuns)) - VNC Viewer
Activities
Tue 07:11

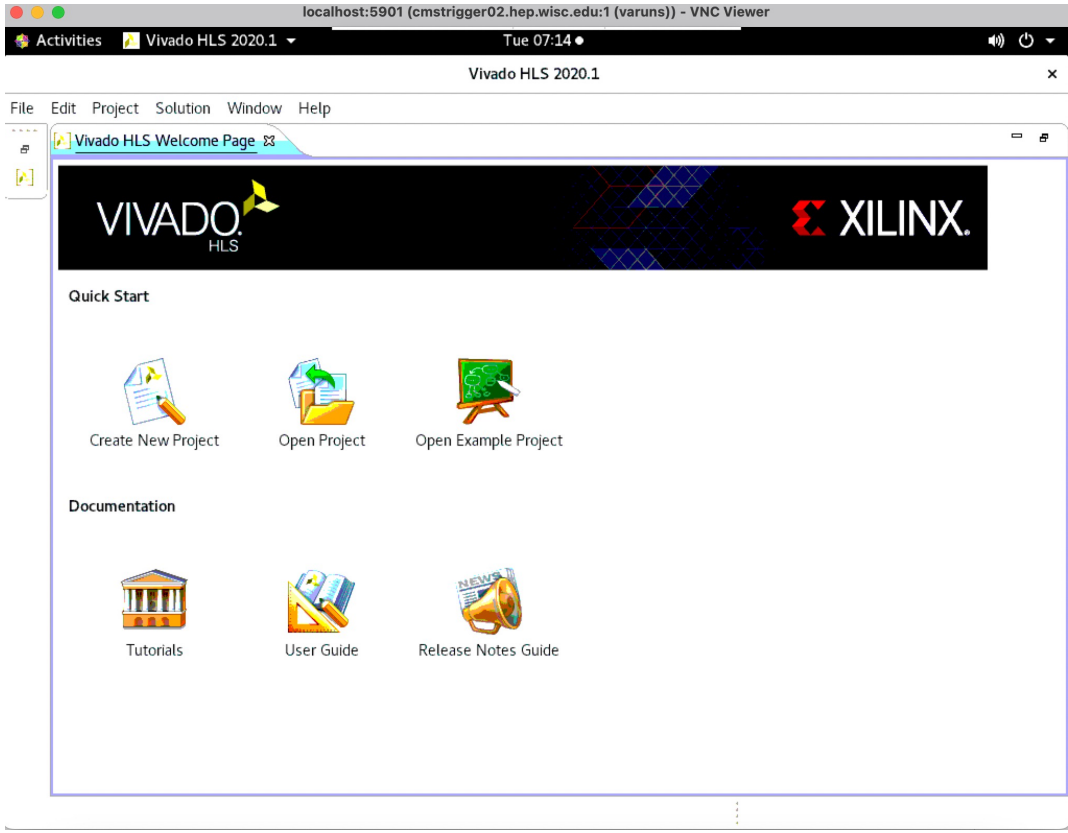

localhost:5901 (cmstrigger02.hep.wisc.edu:1 (varuns)) - VNC Viewer
Activities
Terminal
Tue 07:13
varuns@cmstrigger02:/afs/hep.wisc.edu/user/varuns
File Edit View Search Terminal Help
[varuns@cmstrigger02 varuns]$ source /opt/Xilinx/Vivado/2020.1/settings64.sh
[varuns@cmstrigger02 varuns]$ vivado_hls
VIVADO HLS
2020.1
XILINX
Copyright 1986-2020 Xilinx, Inc.
All Rights Reserved.

All set for hands-on

Everytime

Summary

- ssh varuns@cmstrigger02-via-login -L5901:localhost:5901
  - Or whatever ***:1*** display number
  - Sometimes you may need to run vncserver -localhost -geometry 1024x768 again to start new vnc server
- Connect to VNC server (remote desktop) client
- Open terminal
- Source /opt/Xilinx/Vivado/2020.1/settings64.sh
- vivado_hls



Homework: You are able to connect and bring this screen
Let me know in case of any issue