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

FPGA module training

Week-5

Lecture-9: 25/02/2025

Varun Sharma

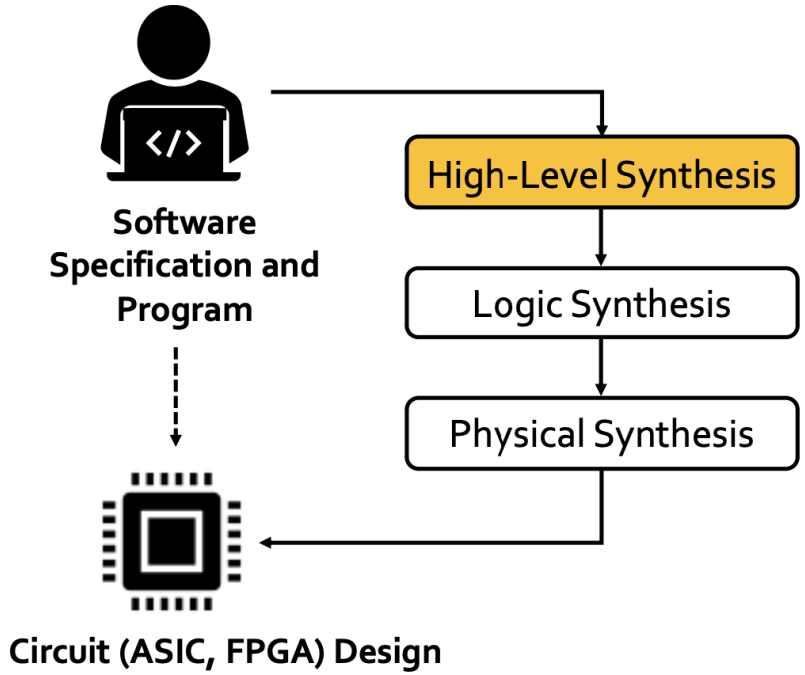
University of Wisconsin – Madison, USA

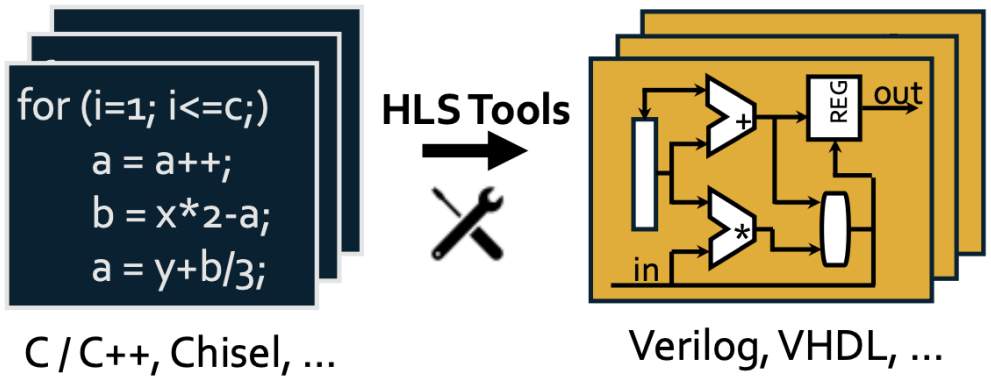
Content

- Vivado/Vitis HLS and its setup

What is HLS (High-Level Synthesis)

HLS is an automated design process that transforms a high-level functional specification to an optimized register-transfer level (RTL) descriptions for efficient hardware implementation

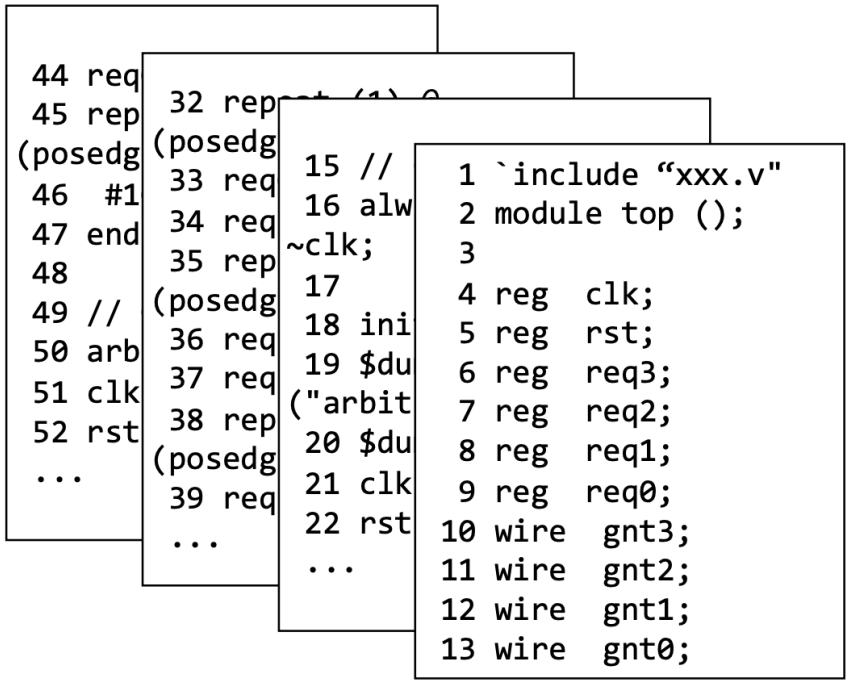




What is HLS

```
for(int h = 0; h < H; h++ )
  for(int w = 0; w < W; w++)
    for(int m = 0; m < K; m++)
      for(int n = 0; n < K; n++)
        ...
```

HLS Tools



Behavioral-level:
Expressive and Concise

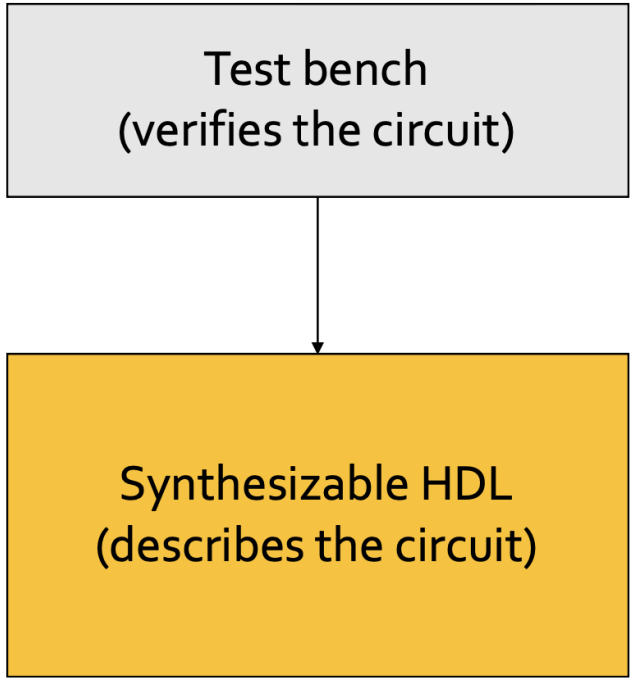
Why use HLS?

- Productivity
  - Lower design complexity and faster simulation speed
  - Ease of use
- Portability
  - Single source -> multiple implementations (different target devices)
- Permutability
  - Much more optimization opportunities at higher level
  - Rapid design space exploration

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)

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

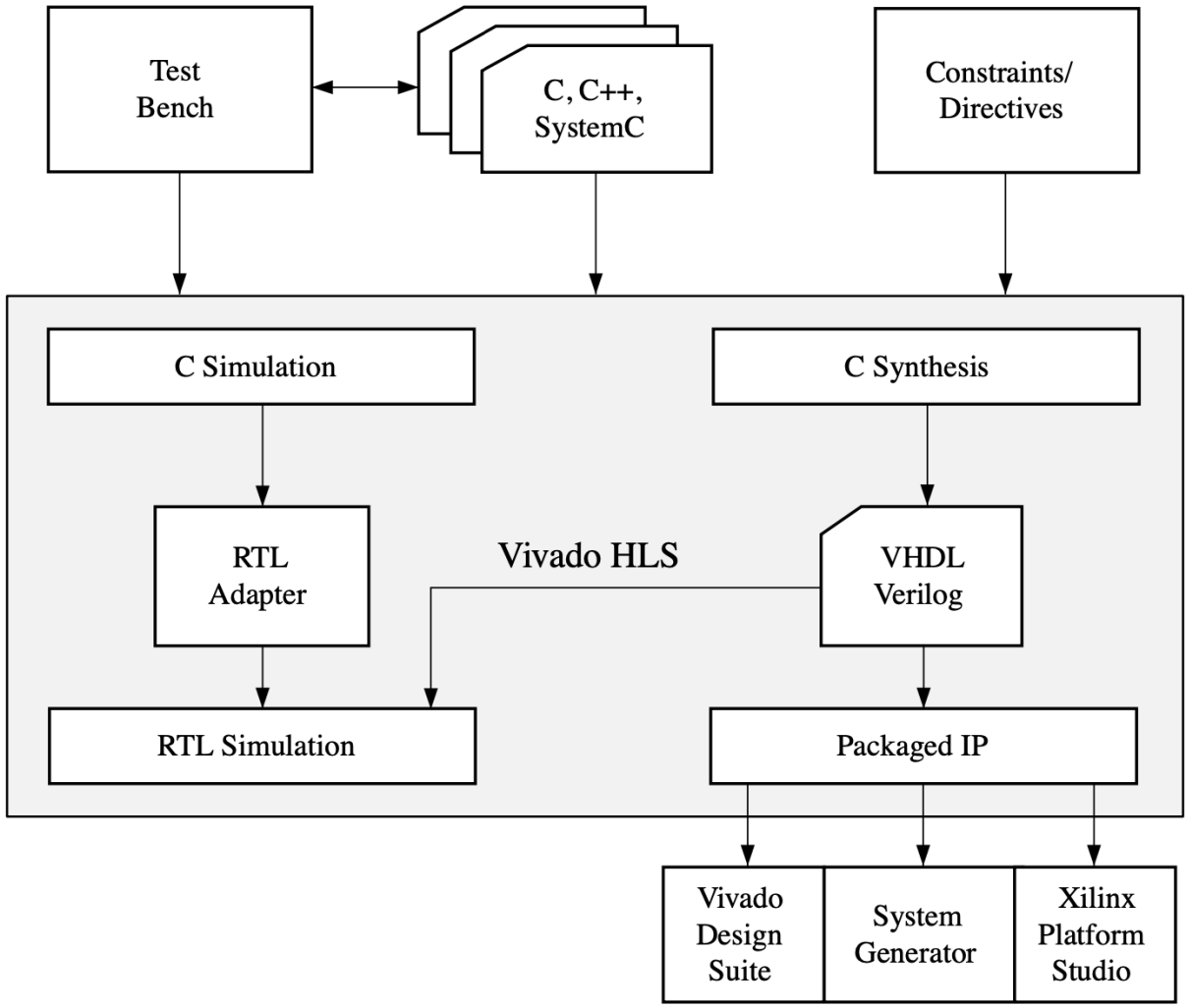


Fig.

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

Writing code in HLS

- Two strategies to develop in HLS
  - Write code in your favourite editor and use Vitis/Vivado HLS command line interface (CLI)
  - Use HLS's GUI to do both editing and synthesis

- Vitis/Vivado HLS command line does not provide all the tools
- HLS GUI is required when you need to investigate design performance in detail

Using VNC to use GUI is very handy

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 2025

HLS Setup

- Xilinx **Vitis/Vivado** HLS has a graphical user interface that we intend to use
- The goal is to run ***vivado_hls*** or ***vitis_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 /scratch/`whoami` (If directory exist, go to next bullet)
  - cd /scratch/`whoami`

https://github.com/varuns23/TAC-HEP-FPGA/tree/main/hls-setup

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:2***, where **:2** is your display

```
[varuns@cmstrigger02 ~]$ vncserver -localhost -geometry 1024x768

WARNING: vncserver has been replaced by a systemd unit and is now considered deprecated and
 removed in upstream.
Please read /usr/share/doc/tigervnc/HOWTO.md for more information.

New 'cmstrigger02.hep.wisc.edu:2 (varuns)' desktop is cmstrigger02.hep.wisc.edu:2

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:2.log
```

Ignore the warning

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:2***, where **:2** is your display

- 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 -L5902:localhost:5902. [In separate terminal tab]
  - Make sure you change "varuns" to your user name, and "5902" to (5900 + your display number), say 5903, if vncserver told you 3!

- You can kill your VNC server (:2) using the command:
  - vncserver –kill :2
  - vncserver –list (check active servers and kill unnecessary ones)

ssh varuns@cmstrigger02-via-login -L5902:localhost:5902

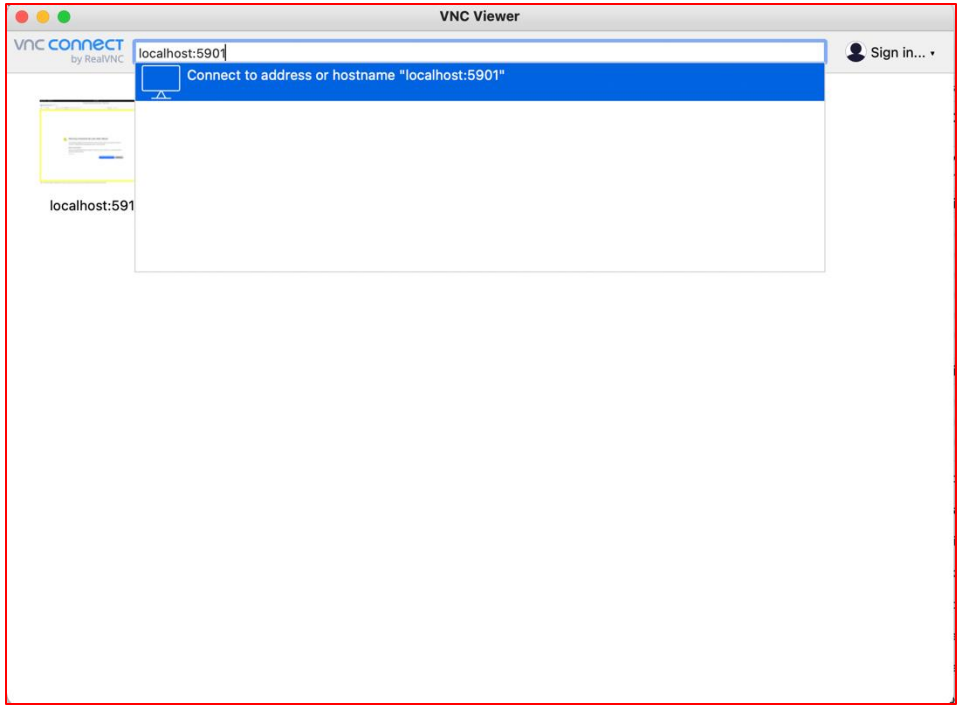
```
$ ssh varuns@cmstrigger02-via-login -L5902:localhost:5902
varuns@login.hep.wisc.edu's password:
Permission denied, please try again.
varuns@login.hep.wisc.edu's password:
varuns@cmstrigger02.hep.wisc.edu's password:
##############################################################################
        Welcome to cmstrigger02.hep.wisc.edu
         9.5 (Teal Serval)

        251.35 GiB RAM
        2x Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz, 40 threads

        kernel 5.14.0-427.22.1.el9_4.x86_64
##############################################################################
Last login: Mon Feb 24 22:09:11 2025 from 144.92.181.245
[varuns@cmstrigger02 ~]$
```

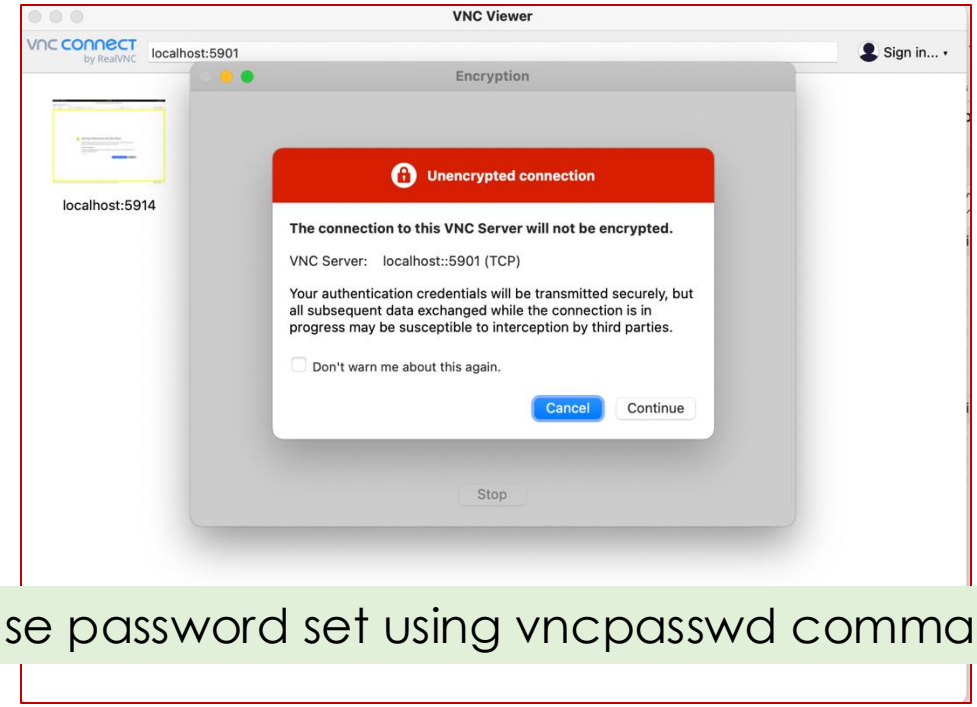
Enter password twice

Remote desktop client

One time only

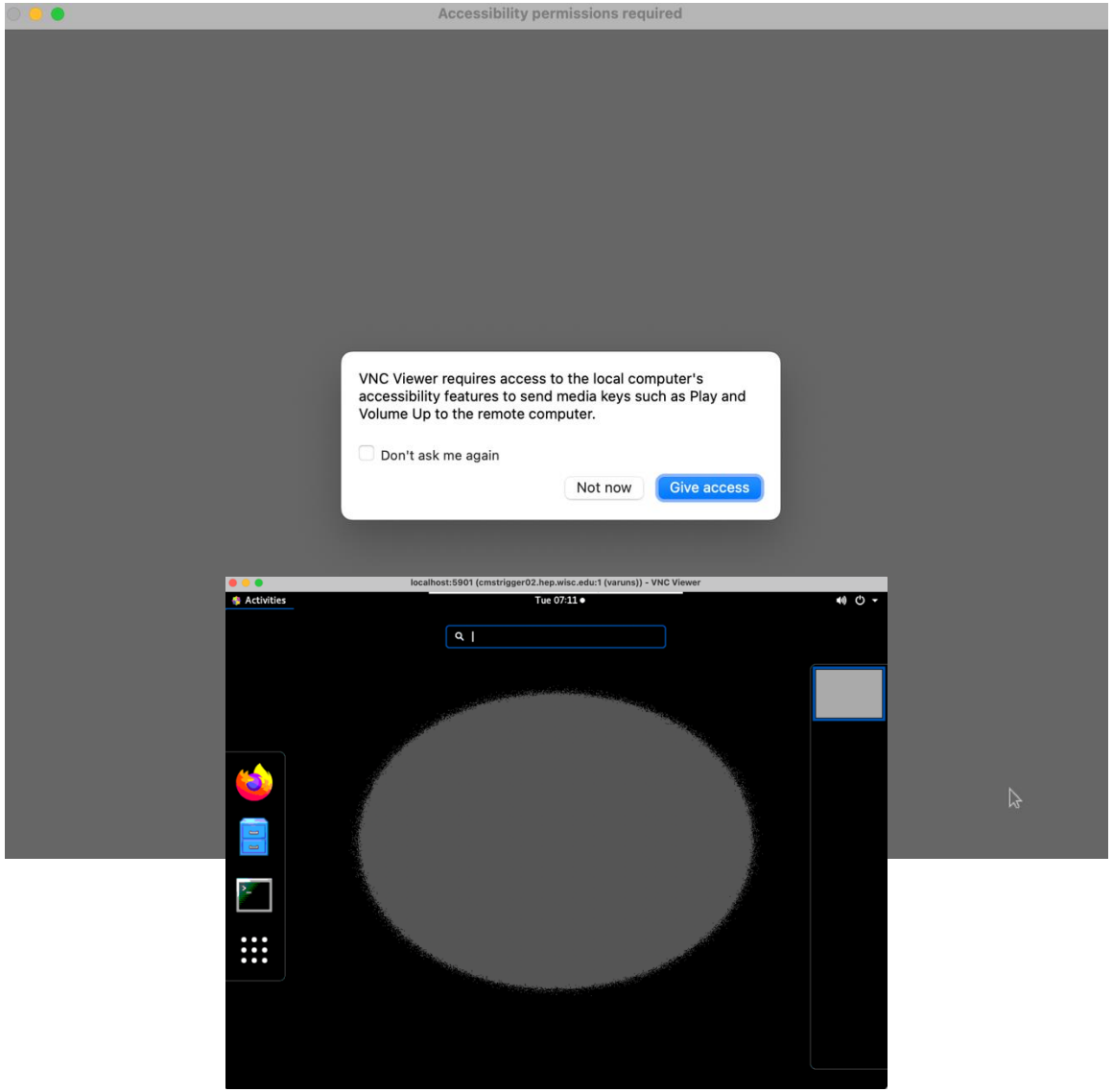
- Download VNC viewer: https://www.realvnc.com/en/connect/download/viewer/
  - You can choose any other remote desktop client but this is one of the stable one that I have used

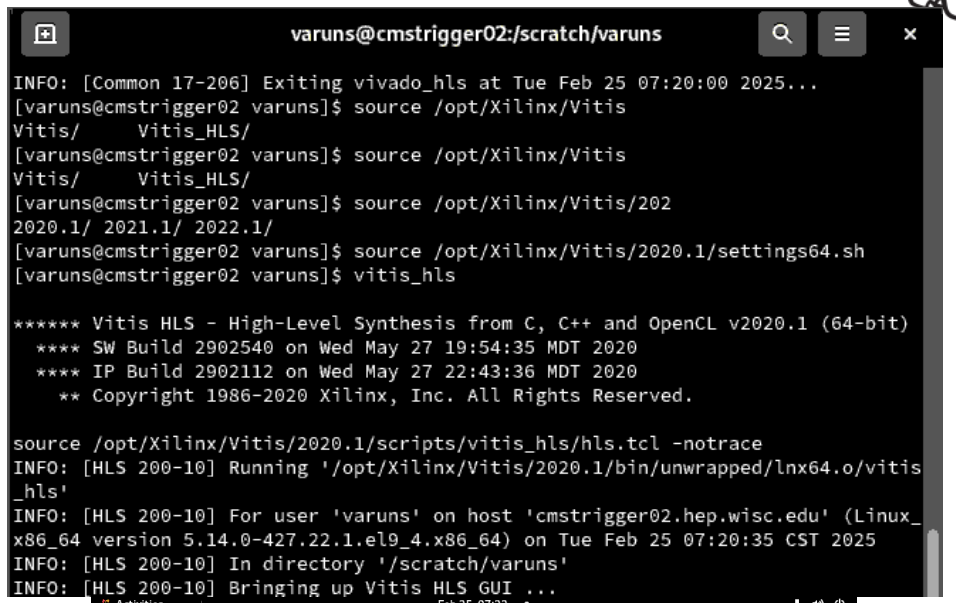


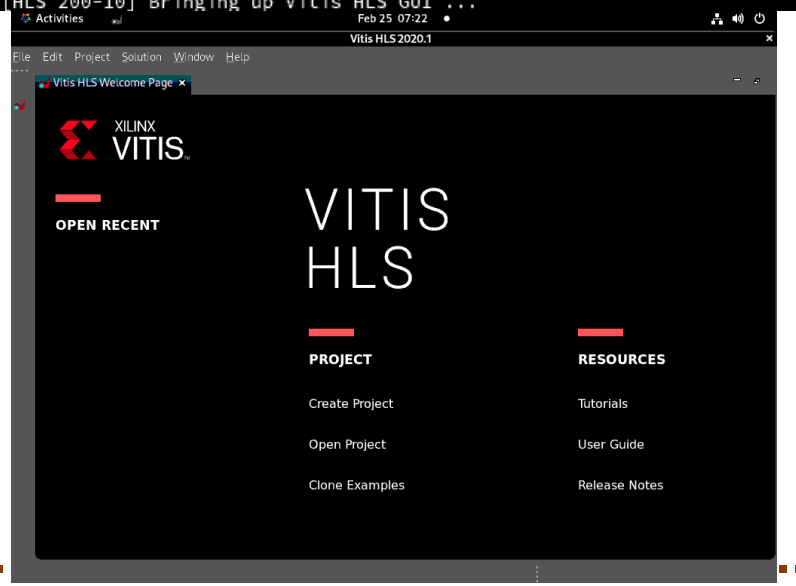


Use password set using vncpasswd command

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


varuns@cmstrigger02:/scratch/varuns
INFO: [Common 17-206] Exiting vivado_hls at Tue Feb 25 07:20:00 2025...
[varuns@cmstrigger02 varuns]$ source /opt/Xilinx/Vitis
Vitis/ Vitis_HLS/
[varuns@cmstrigger02 varuns]$ source /opt/Xilinx/Vitis
Vitis/ Vitis_HLS/
[varuns@cmstrigger02 varuns]$ source /opt/Xilinx/Vitis/202
2020.1/ 2021.1/ 2022.1/
[varuns@cmstrigger02 varuns]$ source /opt/Xilinx/Vitis/2020.1/settings64.sh
[varuns@cmstrigger02 varuns]$ vitis_hls
****** Vitis HLS - High-Level Synthesis from C, C++ and OpenCL v2020.1 (64-bit)
**** SW Build 2902540 on Wed May 27 19:54:35 MDT 2020
**** IP Build 2902112 on Wed May 27 22:43:36 MDT 2020
** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.
source /opt/Xilinx/Vitis/2020.1/scripts/vitis_hls/hls.tcl -notrace
INFO: [HLS 200-10] Running '/opt/Xilinx/Vitis/2020.1/bin/unwrapped/lnx64.o/vitis_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.x86_64) on Tue Feb 25 07:20:35 CST 2025
INFO: [HLS 200-10] In directory '/scratch/varuns'
INFO: [HLS 200-10] Bringing up Vitis HLS GUI ...


Activities
Feb 25 07:22
Vitis HLS 2020.1
File Edit Project Solution Window Help
Vitis HLS Welcome Page
XILINX
VITIS
OPEN RECENT
VITIS
HLS
PROJECT
Create Project
Open Project
Clone Examples
RESOURCES
Tutorials
User Guide
Release Notes

All set for hands-on

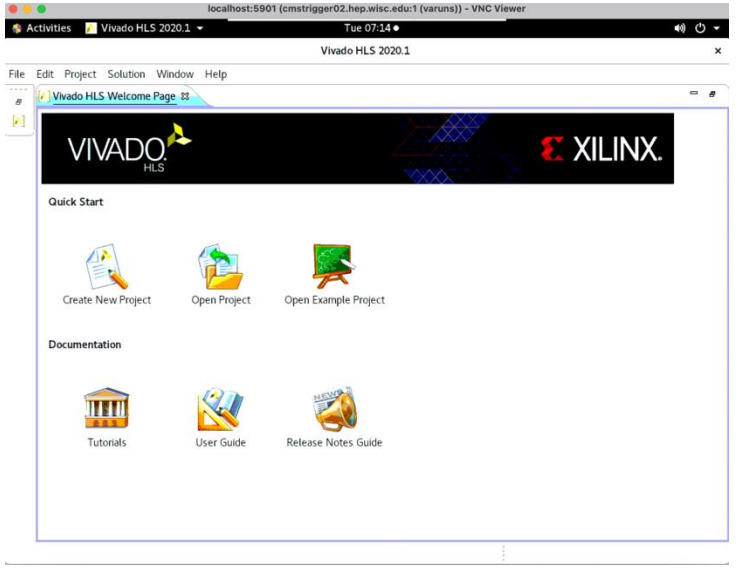
Everytime

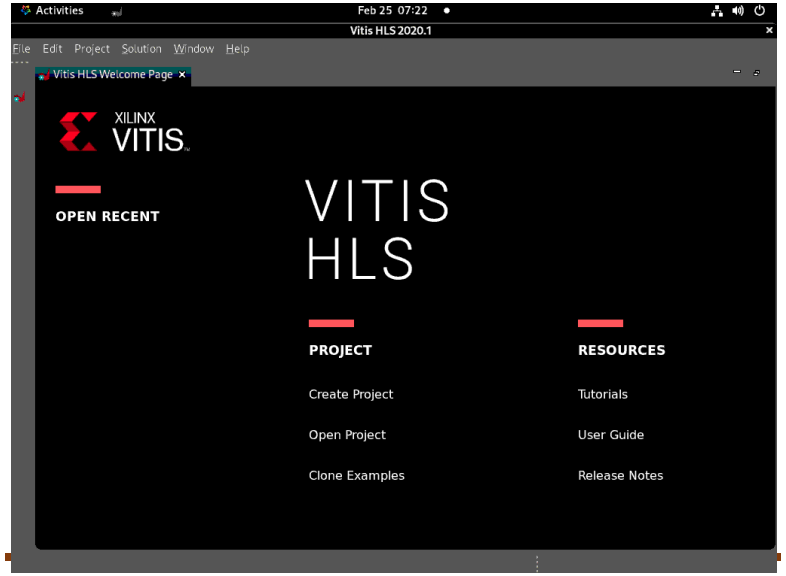
Summary

- 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

OR

  - Source /opt/Xilinx/Vitis/2020.1/settings64.sh
  - vitis_hls





Reminder: Assignments

- Assignment-1 (13-02-2025)
- Assignment-2 (18-02-2025)

Uploaded to cernbox: https://cernbox.cern.ch/s/gmUqRDHTxDLqx4M

Send via email: **varun.sharma@cern.ch**

Submit in 2 weeks from date of assignment

TAC-HEP 2025

Questions?

Acknowledgements:
- Some of these slides are from Isobel Ojalvo

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