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

FPGA module training

More on FPGAs

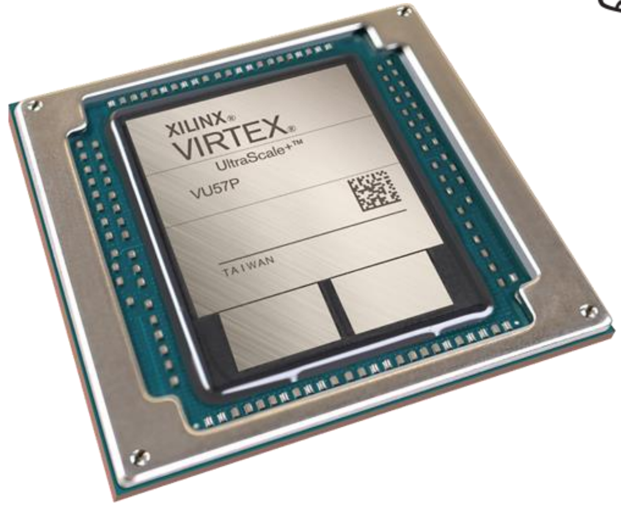
Lecture-4: February 24th 2026

Varun Sharma

University of Wisconsin – Madison, USA

So Far…

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



Today:

- More on FPGA's & HLS Setup

FPGA: **F**ield **P**rogrammable **G**ate **A**rray

TAC-HEP 2026

HLS Setup on cmstrigger02

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

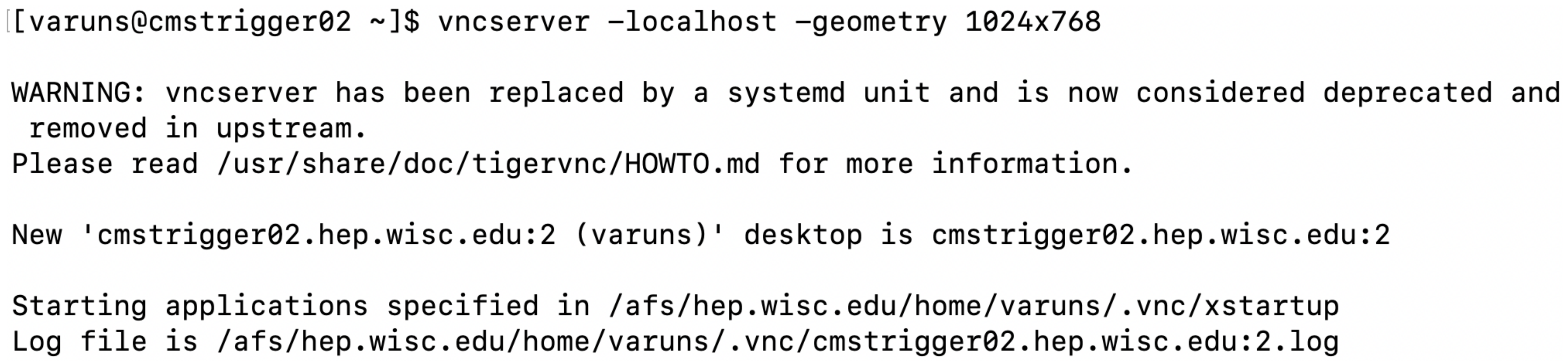
```
Host *
 ControlPath ~/.ssh/control/%C
 ControlMaster auto

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

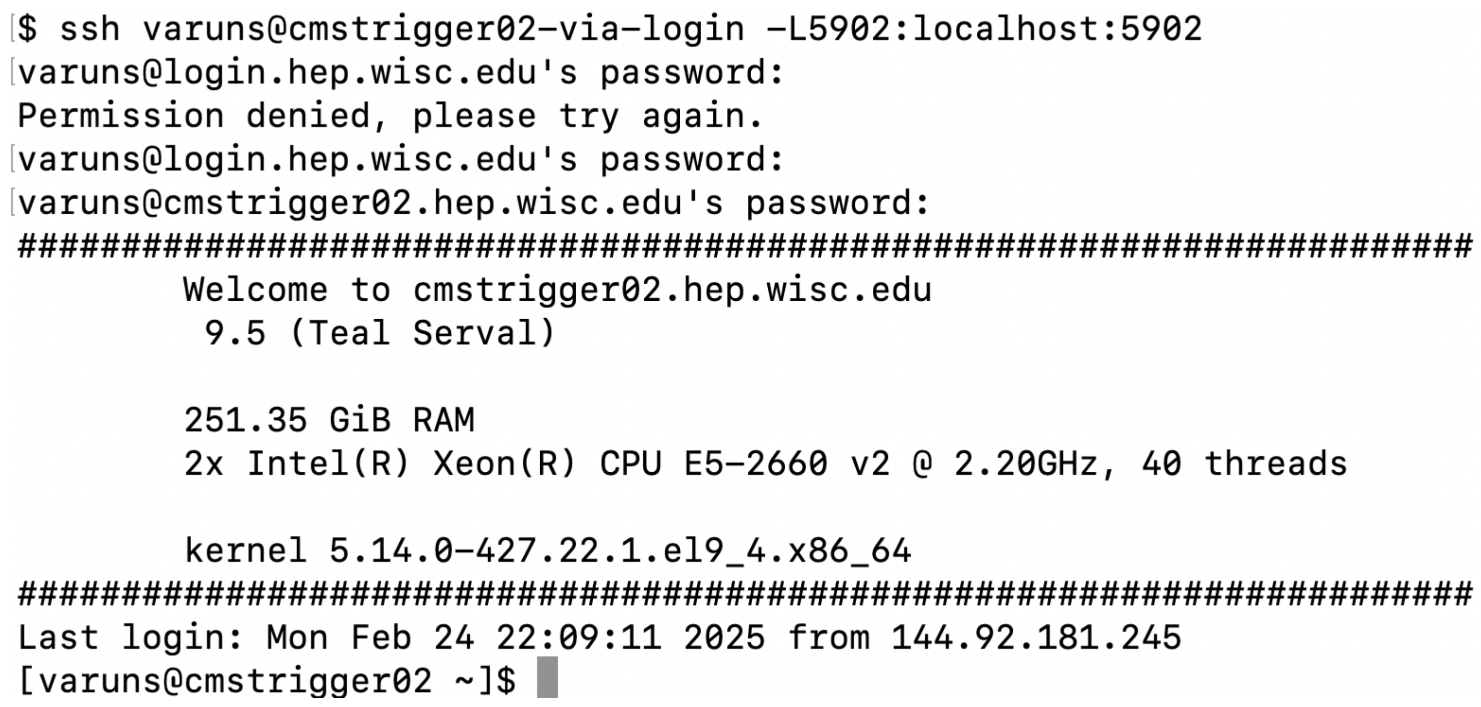
IP Port forwarding

- 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



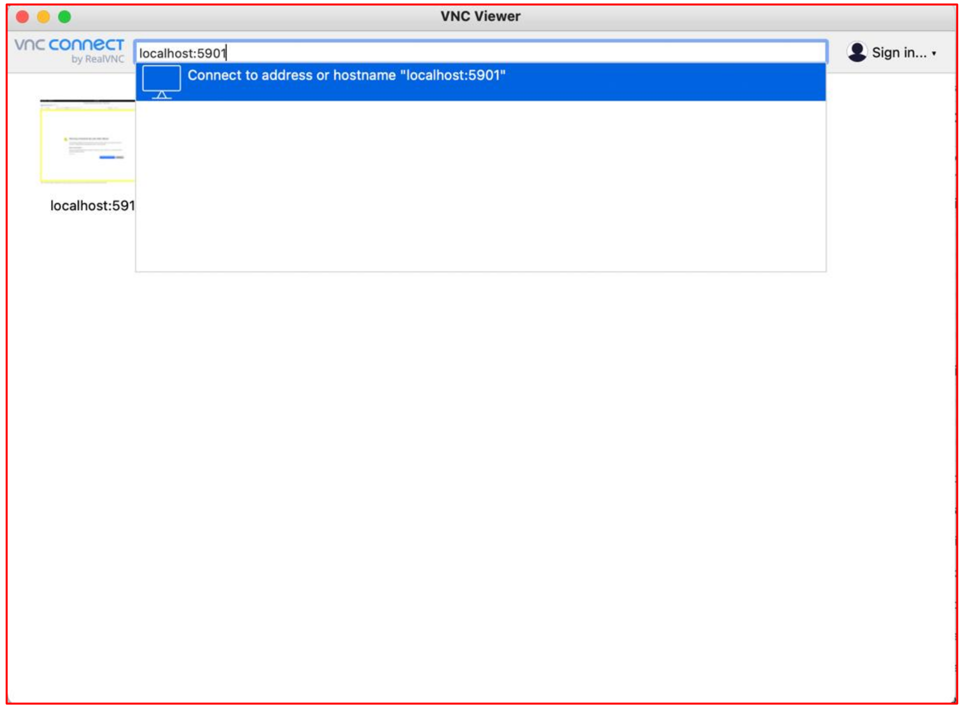
```
$ ssh varuns@cmstrigger02-via-login -L5902:localhost:5902
varuns@login.hep.wisc.edu's password:
Permission denied, please try again.
varuns@login.hep.wisc.edu's password:
varuns@cmstrigger02.hep.wisc.edu's password:
######################################################################
        Welcome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

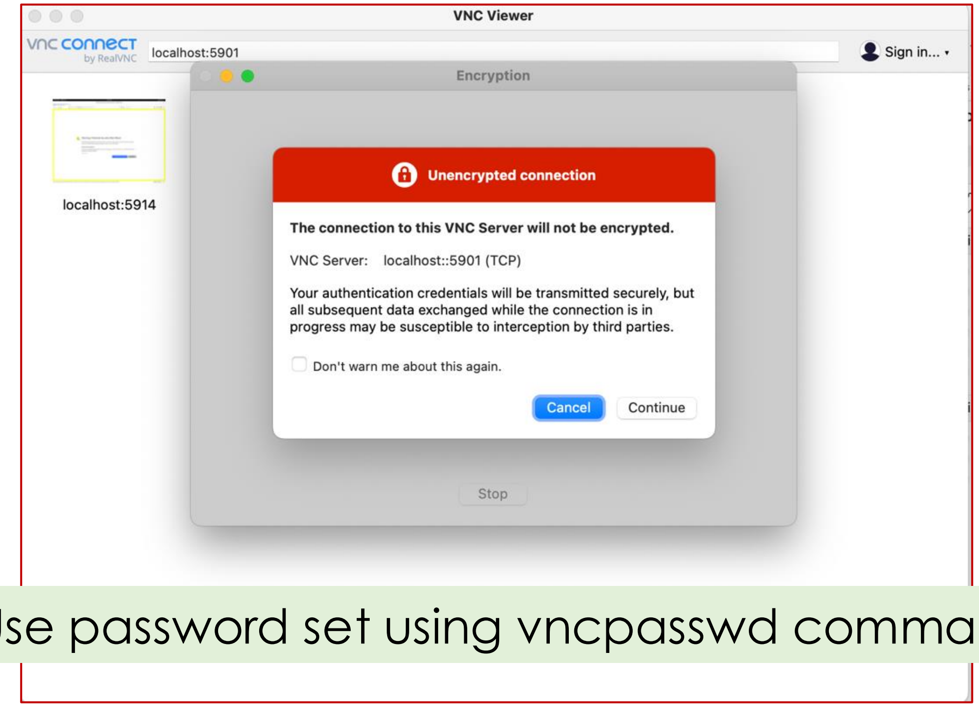


VNC Viewer

localhost:5901

Connect to address or hostname "localhost:5901"

localhost:591



VNC Viewer

localhost:5901

Encryption

Unencrypted connection

The connection to this VNC Server will not be encrypted.

VNC Server: localhost::5901 (TCP)

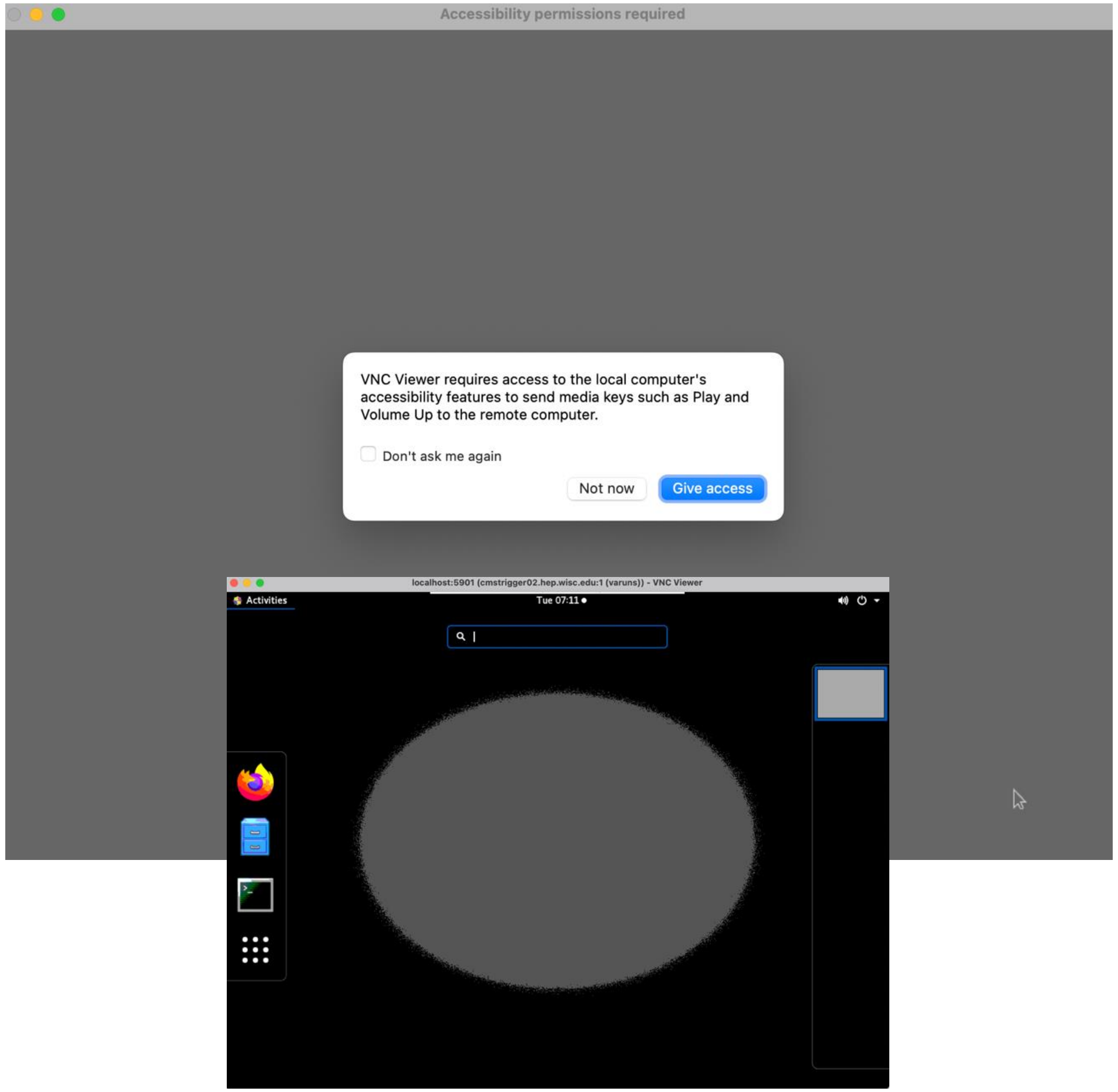
Your authentication credentials will be transmitted securely, but all subsequent data exchanged while the connection is in progress may be susceptible to interception by third parties.

Don't warn me about this again.

localhost:5914

Use password set using vncpasswd command

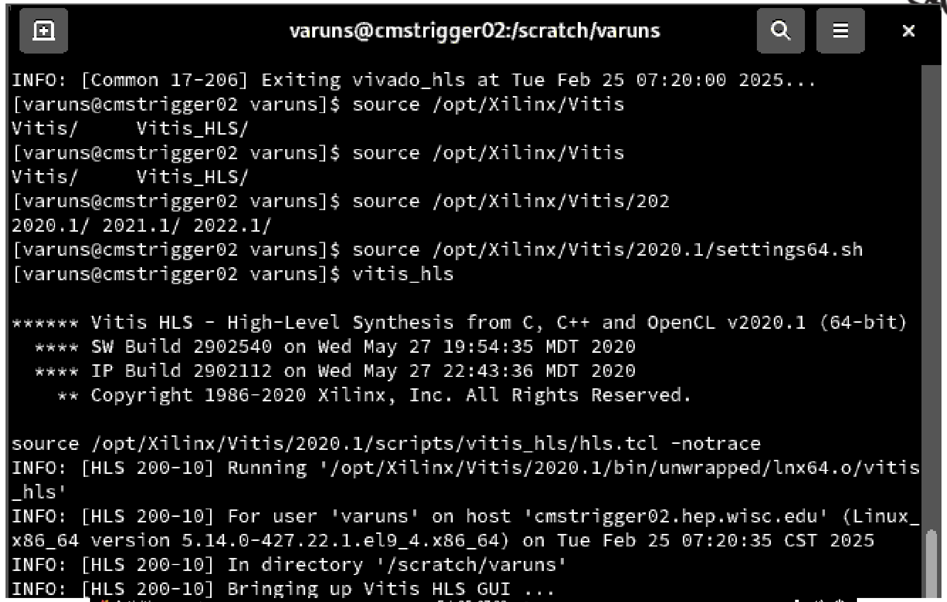
Connected…


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

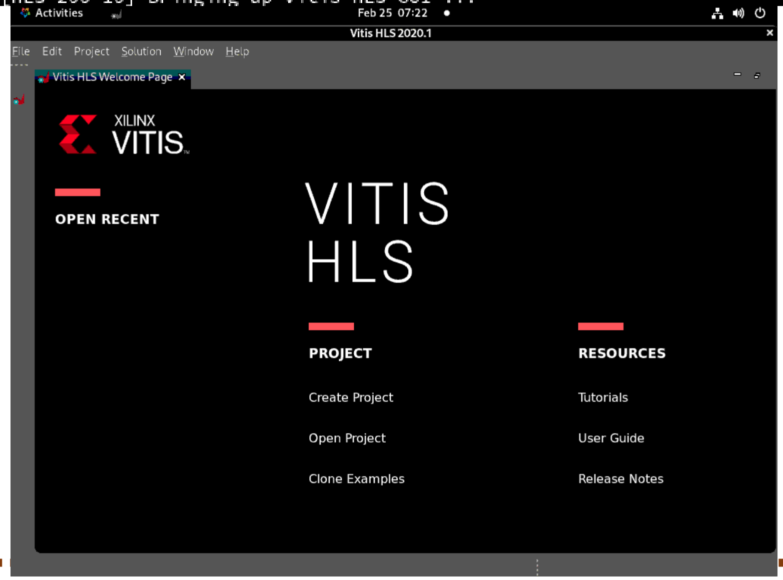
VNC Viewer requires access to the local computer's accessibility features to send media keys such as Play and Volume Up to the remote computer.

☐ Don't ask me again

Not now | Give access



```
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 ...
```



All set for hands-on

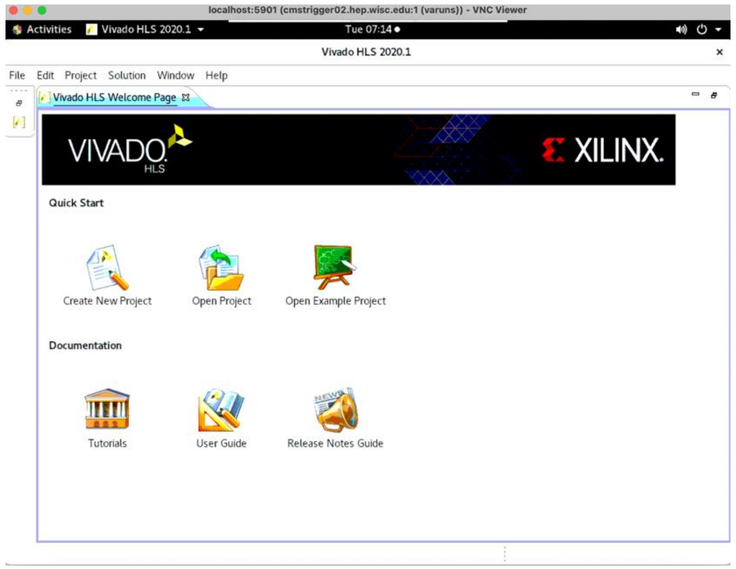
Everytime

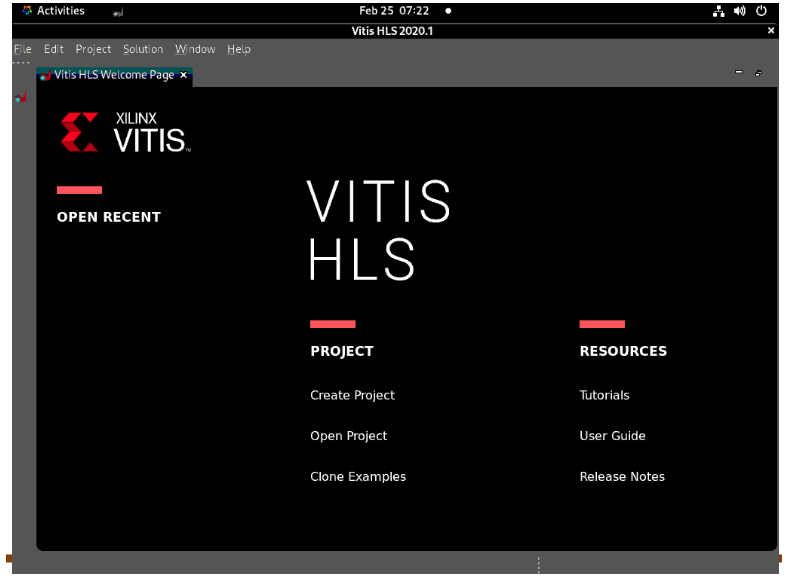
Summary

- 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





Example: Matrix Addition

```cpp
#include <iostream>
#define N 10

void matrix_add(int A[N][N], int B[N][N], int C[N][N]) {

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
}
```

1. Run on HLS and check the resource utilization

TAC-HEP 2026

Basic concepts of Hardware Design

FPGA Processing

FPGA: blank slate with a box of building blocks

- ***Vivado @ HLS compiler*** create processing architecture from the box of building blocks that best fits the software program

- The process of guiding the HLS compiler to create the best processing architecture requires fundamental knowledge about hardware design concepts

- In contrast, with a **processor**, computation architecture is fixed, and the job of compiler is to determine how to best fit the software application in the available processing structures

Clock Frequency

- **Important metric to determine the choice of processor**
- *In general*: High clock frequency means higher performance execution rate
    - Can be misleading

Maximum clock frequency:

FPGA	500 MHz
Processor	2 GHz

Which is better?

Stages		**Description**
IF	**Instructions Fetch**	Get the instruction from program memory
ID	**Instruction decode**	Decode the instruction to determine the operation and the operators
EXE	**Execute**	Execute the instruction on the available hardware
MEM	**Memory Operation**	Fetch data for the next instruction using memory operations
WB	**Write Back**	Write the results of the instruction to local registers/global memory

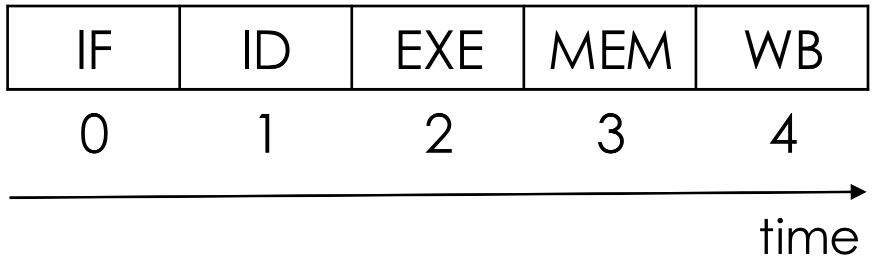
Regardless of processor type: Execution instructions

Clock Frequency

FPGA	500 MHz
Processor	2 GHz

4 times better?

- A **processor** is able to execute any program on a common hardware platform
- The compiler, which has a built-in understanding of the processor architecture, compiles the user software into a set of instructions

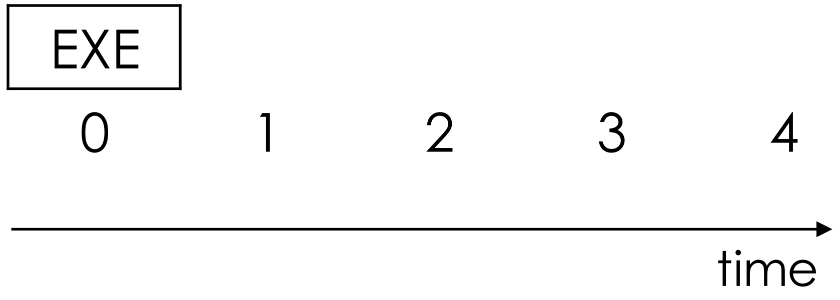


Processor instruction execution stages

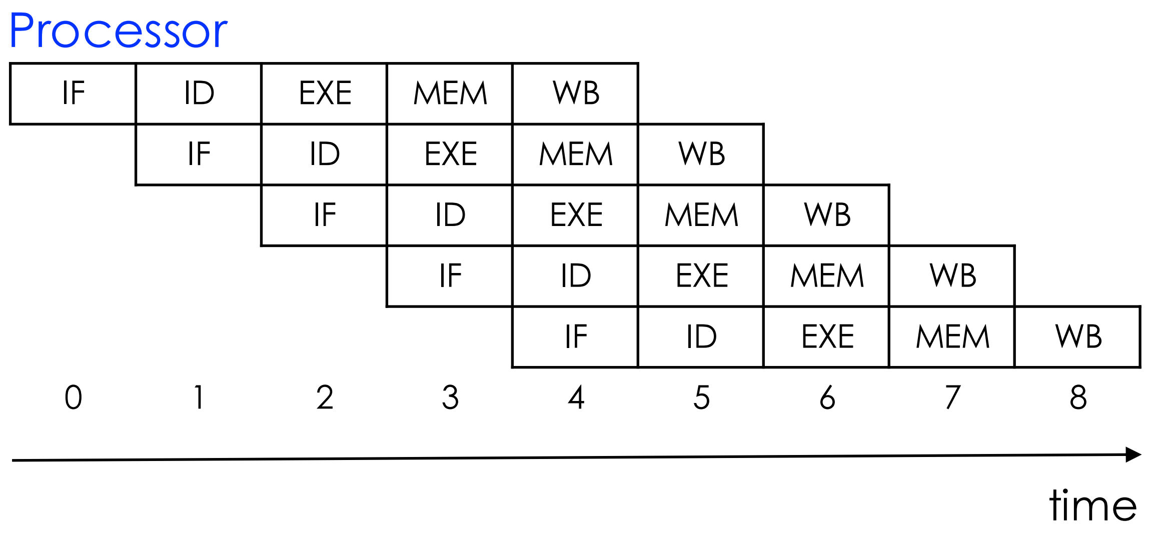
Clock Frequency: FPGA

- **FPGA** does not execute all software on a common computation platform.
- BUT executes on custom circuit for that program
  - Therefore, any modification to program changes the circuit in the FPGA.

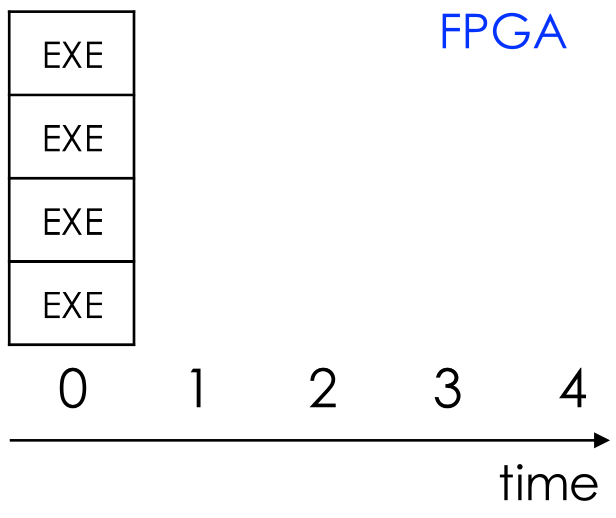
- Vivado HLS compiler does not need to account for overhead stages in the platform
  - Can find ways of maximizing instruction parallelism.



Clock Frequency



< 9x faster



Multiple instruction execution stages

FPGAs generally demonstrate at least 10x the performance

Approximate Power consumption = $\frac{1}{2}$ cF.V^2

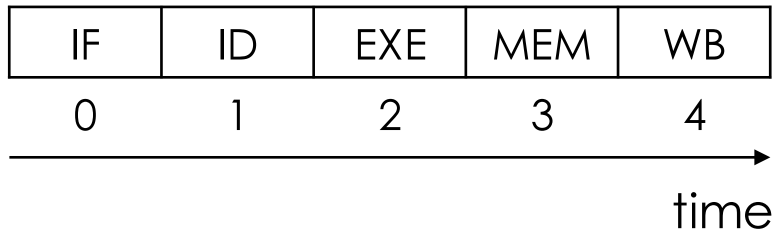
FPGA is able to run at a lower clock frequency with maximum parallelism

- Thus lower power for same computational workload

Latency and Pipelining

Latency and Pipelining

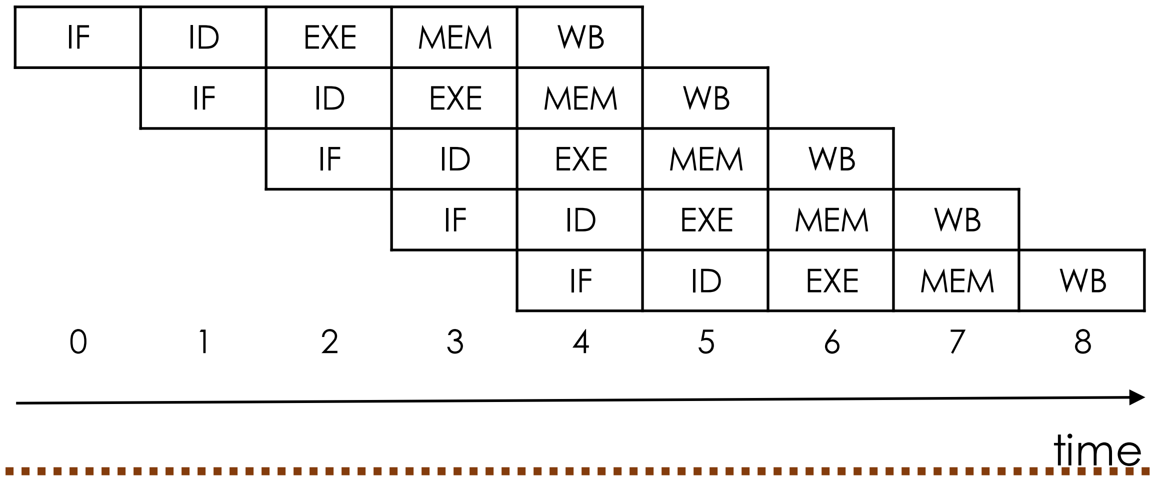
Latency: *number of clock cycles it takes to complete an instruction or set of instructions to generate an application result value*



Latency: **5 clock cycle**
For 5 set of instructions: **25 clock cycles**

Latency is another important key performance metric

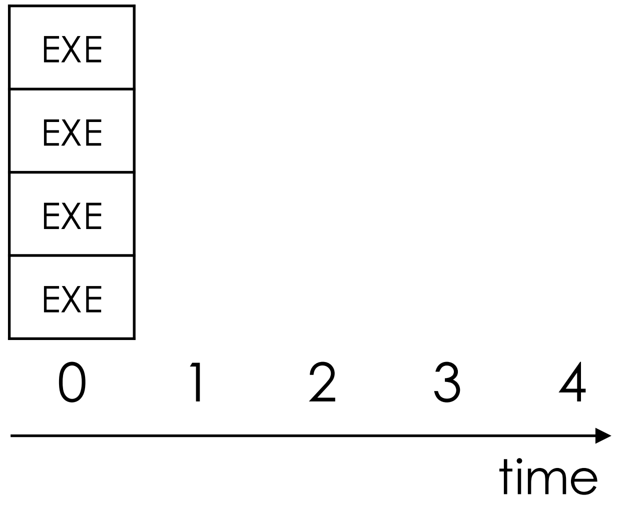
Latency can be improved via pipelining



Latency: **5 clock cycle**
For 5 set of instructions with pipelining
9 clock cycle

Latency and Pipelining

Parallelism also plays an important role in reducing latency

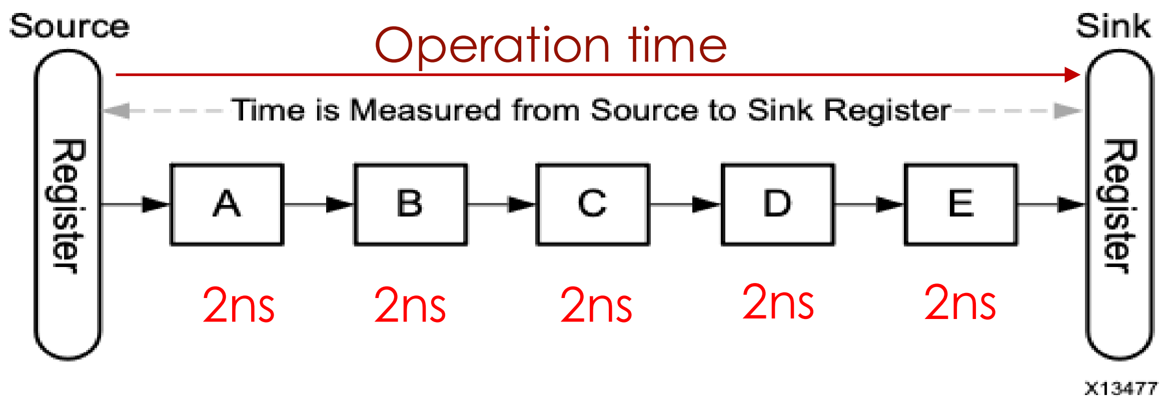


5 set of instructions
FPGA latency: 1 clock cycle

Do we need pipelining in One clock cycle latency of the FPGA?
- The reason for pipelining in an FPGA is to improve application performance

Latency and Pipelining

Reminder: FPGA is a blank slate with building blocks that must be connected to implement an application



FPGA implementation without pipelining

Example:

- Each block takes 2 ns to execute
- Current design (5 stages of implementation): 10 ns
- Latency: 1 clock cycle
- Clock frequency: $\frac{1}{5 \times 2\, ns} = \frac{1}{10\, ns}$ =100 MHz

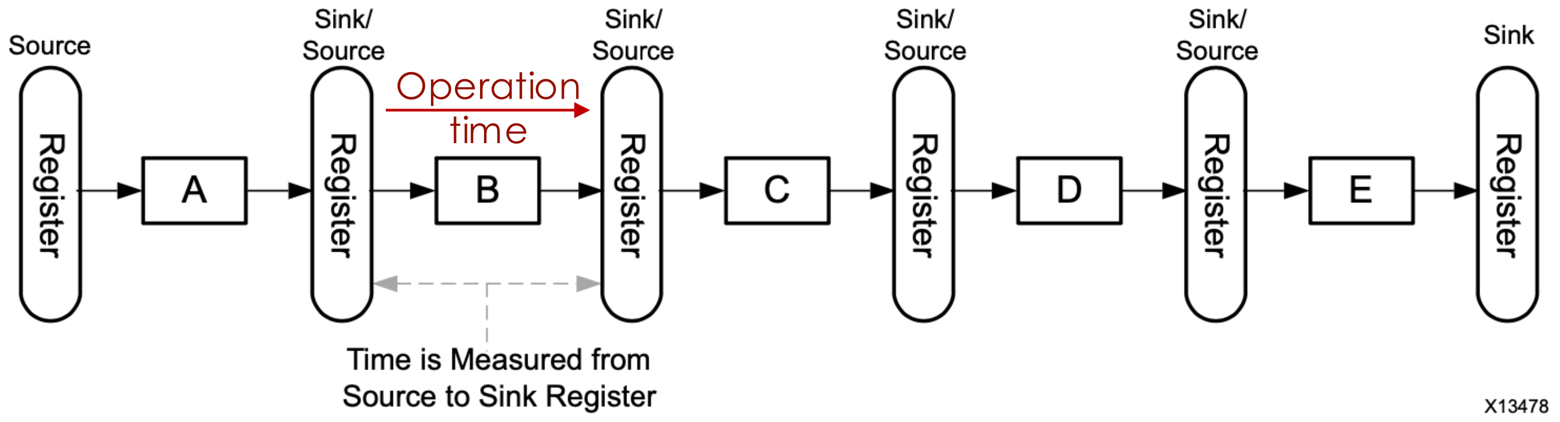
Clock Frequency: longest signal travel time between source and sink registers

Latency and Pipelining

Technique to avoid data dependencies and increase the level of parallelism

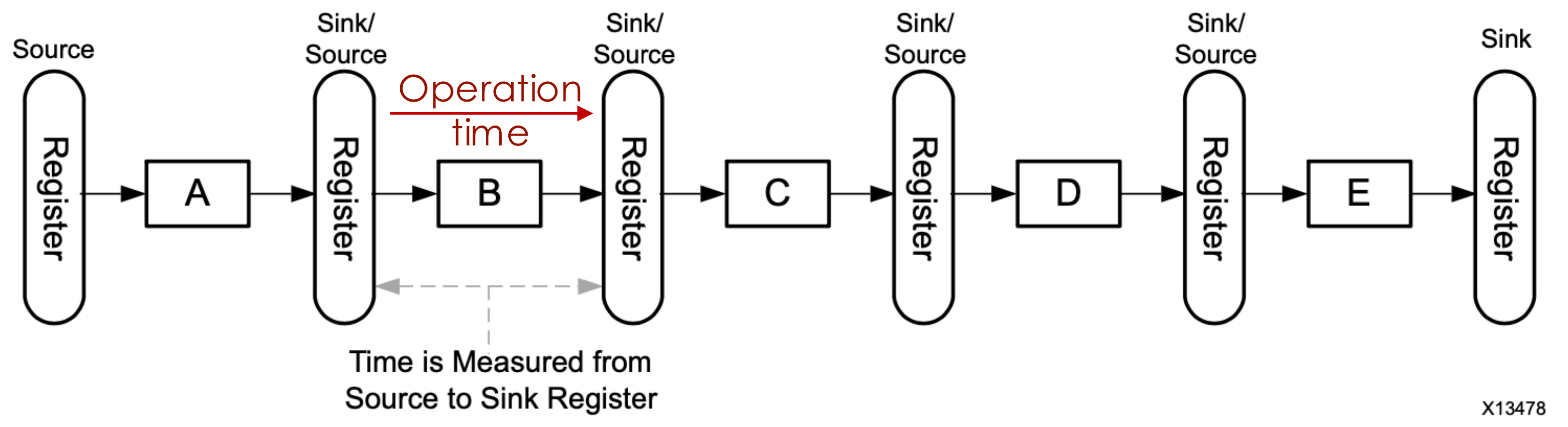
- Pipelining in an FPGA is the process of inserting more registers to break up large computation blocks into smaller segments.

- Partitioning of the computation increases the latency in absolute number of clock cycles but increases performance by allowing the custom circuit to run at a higher clock frequency



FPGA implementation with pipelining

Latency and Pipelining



FPGA implementation with pipelining

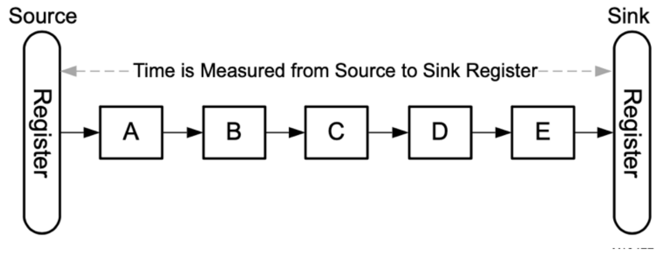
- Addition of registers reduces the timing requirement of the circuit from 10 ns to 2 ns,
- Results in a maximum clock frequency of 500 MHz.

- In addition, by separating the computation into separate register-bounded regions, each block is allowed to always be busy, which positively impacts the application **throughput**

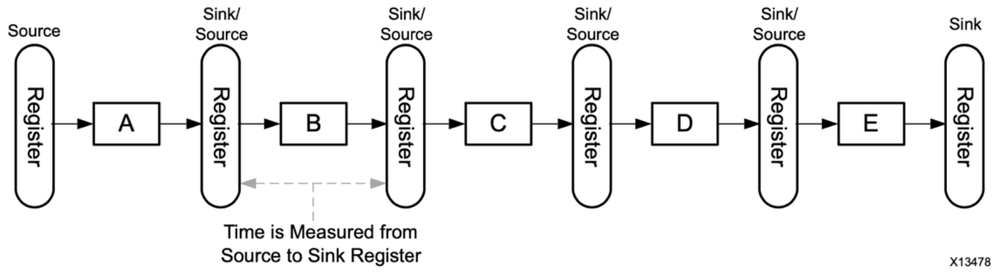
The latency caused by pipelining is one of the trade-offs to consider during FPGA design

Throughput

- Another another metric used to determine overall performance of an implementation
- Number of clock cycles it takes for the processing logic to accept the next input data sample
- Throughput changes with clock frequency



10 ns between input samples



2 ns between input samples

Second implementation has higher performance, because it can accept a higher input data rate

TAC-HEP 2026

Program execution on FPGA

Program execution on a Processor

A processor executes a program as **a sequence of instructions**

- Translated into useful computation for a software application
- Compiler transforms the C/C++ into assemble language

$z = a + b;$ $\rightarrow$ `ADD $R1,$R2,$R3`

- The assembly code defines the addition operation to compute the value of z in terms of the internal registers of a processor
- The complete assembly program to compute the value of z is as follows:

```
LD      a, $R1
LD      b, $R2
ADD     $R1,$R2,$R3
ST      $R3, c
```

- **Even a simple operation, such as the addition of two values, results in multiple assembly instructions**

Program execution on a Processor

- **Depending on the location of a and b, the LD operations take a different number of clock cycles to complete:**
  - Processor cache : few 10 clock cycles
  - DDR memory: ~100/~1000 clock cycles
  - Hard drives: even longer

- **Software engineers spend a lot of time restructuring their algorithms**
  - Increase the spatial locality of data in memory to increase the cache hit rate and decrease the processor time spent per instruction

Program execution on FPGA

FPGA is an inherently parallel processing fabric capable of implementing any logical and arithmetic function that can run on a processor

- Main difference: **Vivado HLS compiler**
  - Transforms software descriptions into RTL (Register-Transfer level),
  - Not hindered by the restrictions of a cache and a unified memory space

- Computation of z is compiled by Vivado HLS into several LUTs required to achieve the size of the output operand

- **E.g.**: In C code, variable a, b, and z are defined with the short data type (16-bit data container)
  - Variables gets implemented as 16 LUTs by Vivado HLS

General rule: 1 LUT is equivalent to 1 bit of computation

Program execution on FPGA

- LUTs used for the computation of **z** are exclusive to this operation ONLY.
  - Unlike a processor, where all computations share the same ALU
- FPGA implementation instantiates independent sets of LUTs for each computation in the software algorithm
- FPGA differs from processor: memory architecture & cost of memory access
- FPGA implementation, the Vivado HLS compiler arranges memories into multiple storage banks as close as possible to the point of use in the operation
  - Results in an instantaneous memory bandwidth, exceeding the capabilities of a processor

Execution steps on FPGA

- Vivado HLS compiler exercises the capabilities of the FPGA fabric using following processes:
  - Scheduling and binding
  - Pipelining
  - Dataflow

Transparent to the user, these processes are integral stages of the software compilation process that extract the best possible circuit-level implementation of the software application

Scheduling

Process of identifying the data and control dependencies between different operations

- To Determine which operation occur during each clock cycle based on:
  - Length of the clock cycle or clock frequency
  - Time it takes for the operation to complete, as defined by the target device
  - User-specified optimization directives

- Vivado HLS analyzes dependencies between adjacent operations as well as across time
- **Group operations to execute in the same clock cycle and set up the hardware to allow the overlap of function calls**

- Overlap of function call executions removes the processor restriction that requires the current function call to fully complete before the next function call to the same set of operations can begin -- ***Pipelining***

Scheduling

Process of identifying the data and control dependencies between different operations

- To Determine which operation occur during each clock cycle based on:
  - Length of the clock cycle or clock frequency
  - Time it takes for the operation to complete, as defined by the target device
  - User-specified optimization directives

- If the clock period is longer or a faster FPGA is targeted, more operations are completed within a single clock cycle, and all operations might complete in one clock cycle.

- Conversely, if the clock period is shorter or a slower FPGA is targeted, high-level synthesis automatically schedules the operations over more clock cycles, and some operations might need to be implemented as multicycle resources

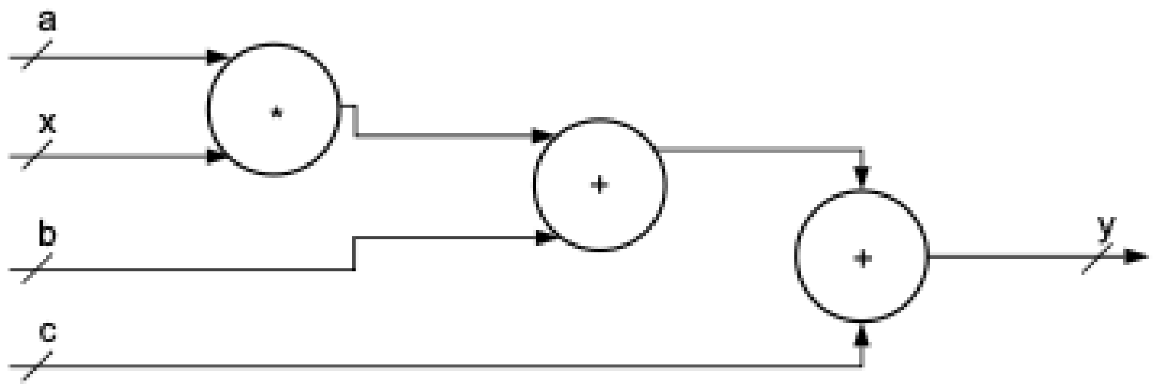
Binding

- Determines which hardware resource implements each scheduled operation

- To implement the optimal solution, high-level synthesis uses information about the target device

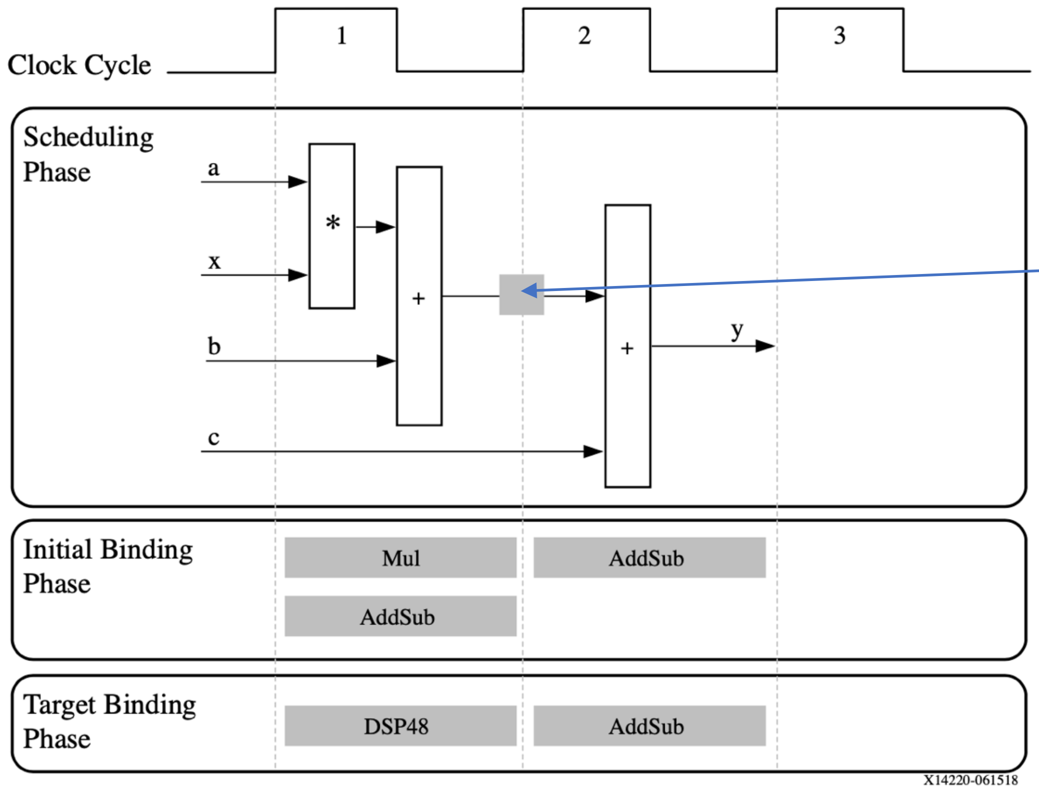
$$y = (a \times x) + b + c$$

Example



Scheduling & Binding

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



Scheduling

First cycle:

- Reads ***b***, ***a***, and ***b*** data ports

Second cycle:

- Reads data port **c**
- Generates output ***y***

Internal register storing a variable

Binding

First cycle:

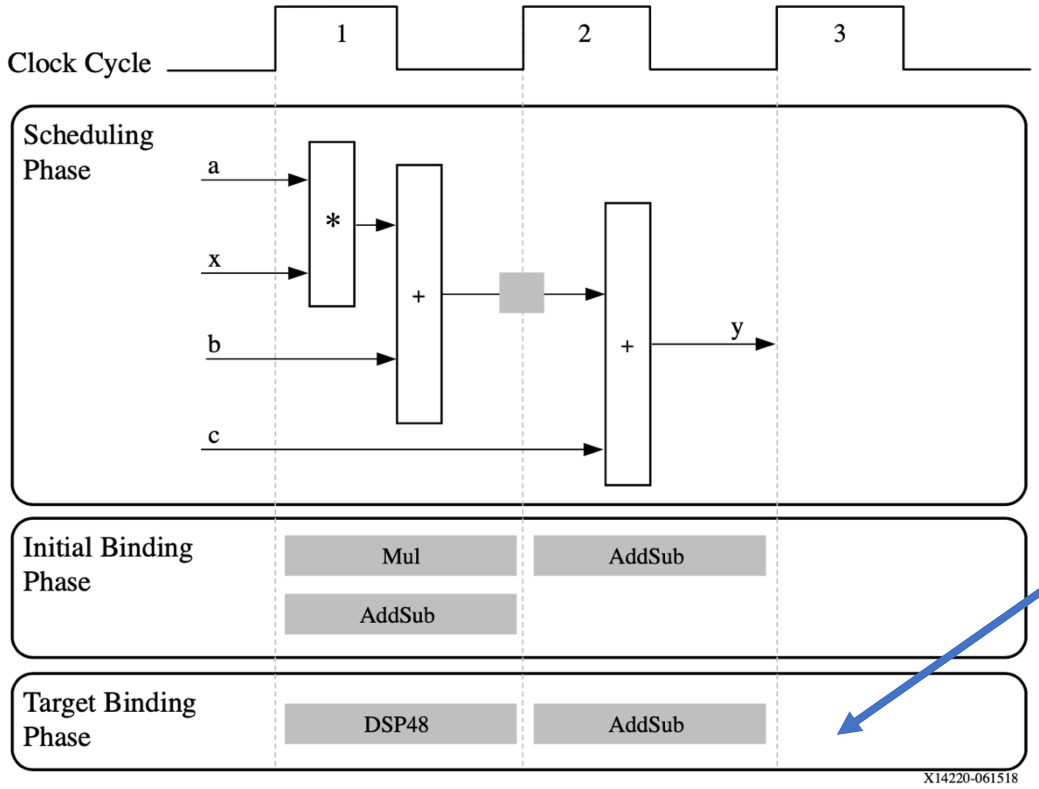
Multiplication & the first addition

Second cycle:

Second addition & output generation

Scheduling

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



In this example, the arguments are simple data ports but in hardware implementation they are I/O ports.

The input data ports are all **8-bits** wide ***(char)***.

Output data port is **32-bit** wide as function return is a 32-bit ***int*** data type

Optimised for the ideal balance of high-performance and efficient implementation

Pipelining

Technique to avoid data dependencies and increase the level of parallelism

- Preserving the original functionality, required circuit is divided into a chain of independent stages
- All stages in the chain run in parallel on the same clock cycle
- The only difference is the source of data for each stage
- Each stage in the computation receives its data values from the result computed by the preceding stage during the previous clock cycle

$$y = (a \times x) + b + c$$

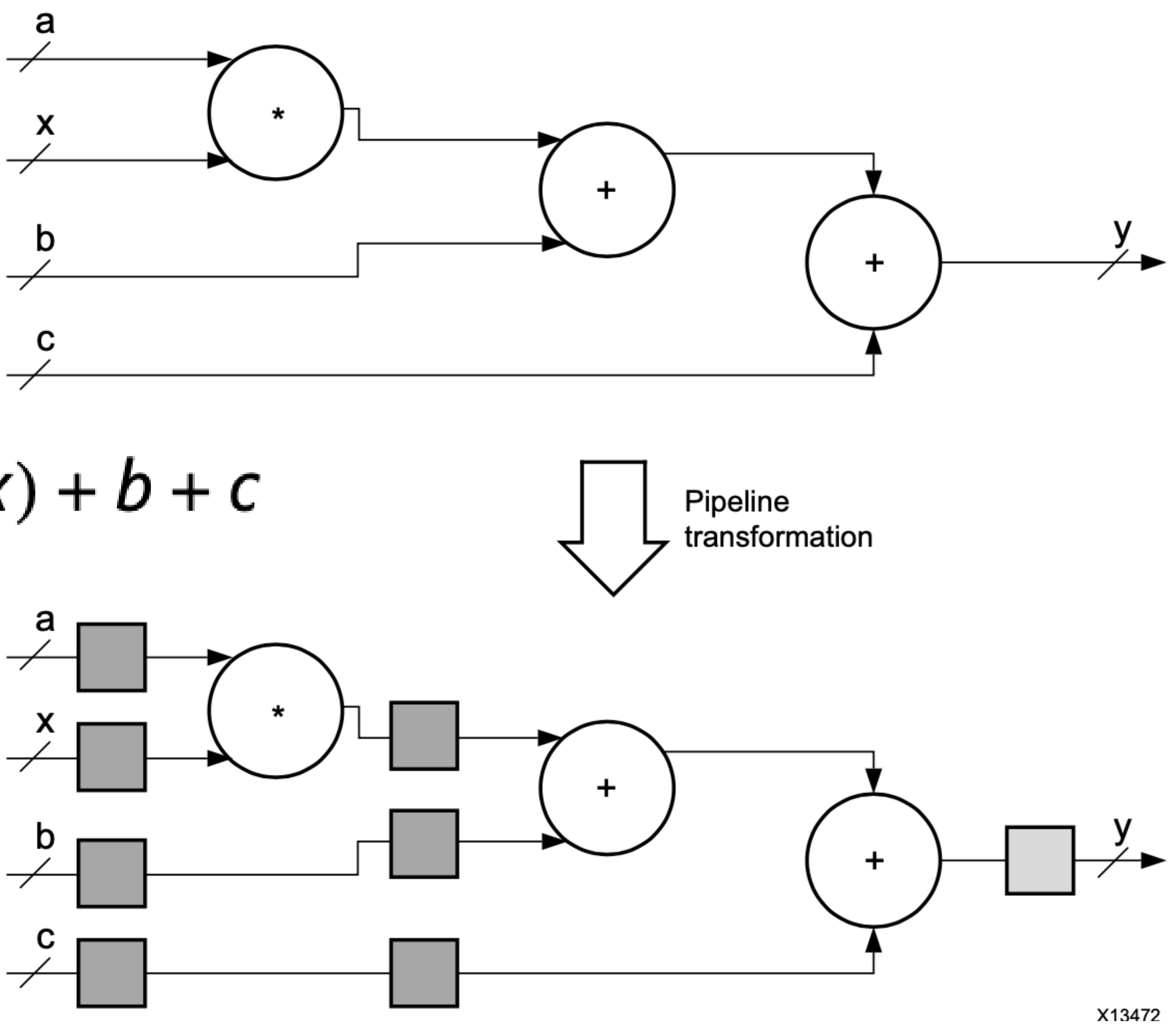
- Vivado HLS compiler instantiates one multiplier and two adder blocks for above example

Pipelining

C implementation

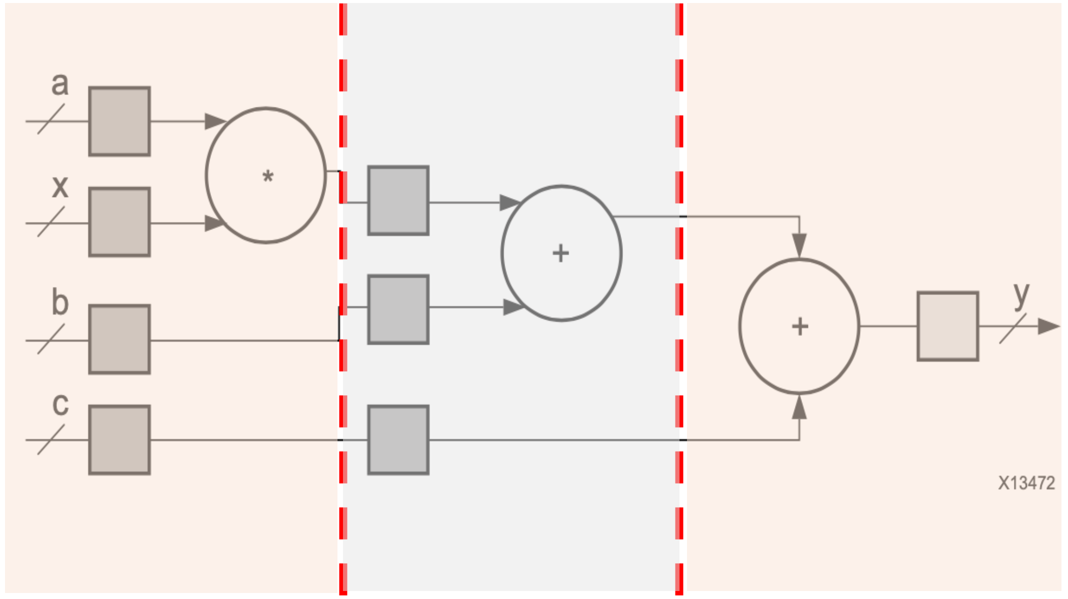
$$y = (a \times x) + b + c$$

Pipelined implementation



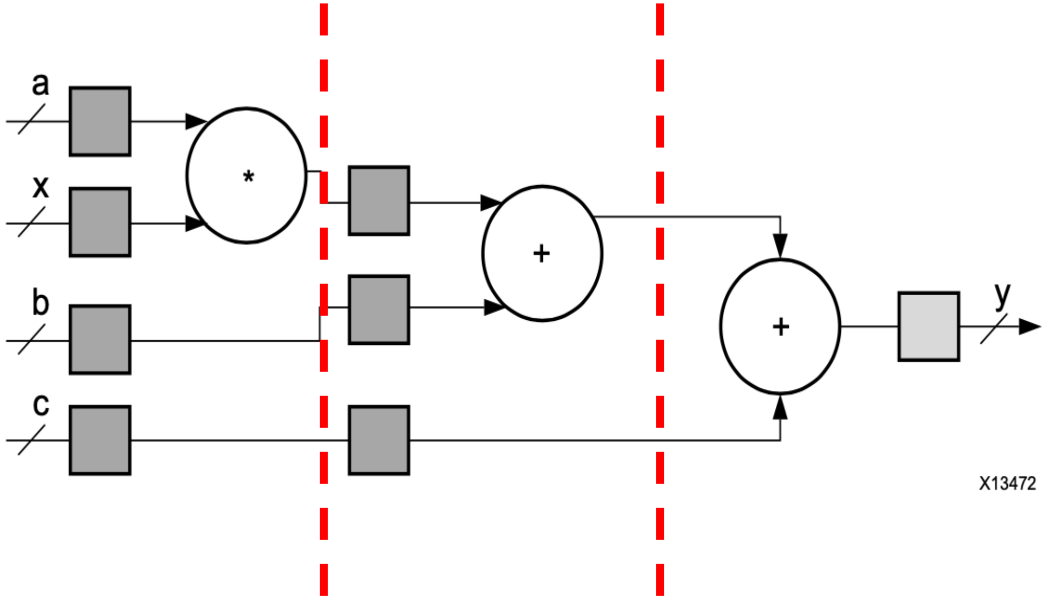
Pipelining

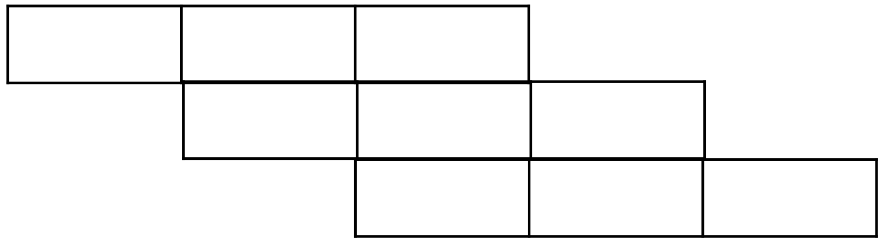
- Boxes: registers implemented by FF blocks
- Each box column counted as single clock cycle
- Result in 3 clock cycles.
- Addition of registers, leads to separated compute sections for each block
  - Multiplier & two adders can run in parallel and reduce latency



Pipelining

- Both sections of the datapath run in parallel
  - Essentially computing the ***y*** and ***y'*** in parallel
  - ***y'*** result of the next execution
  - First computation of y: **pipeline fill time** = 3 CLK
  - After this initial computation, a new value of y is available at the output on every clock cycle, because the computation pipeline contains overlapped data sets for the current and subsequent y computations



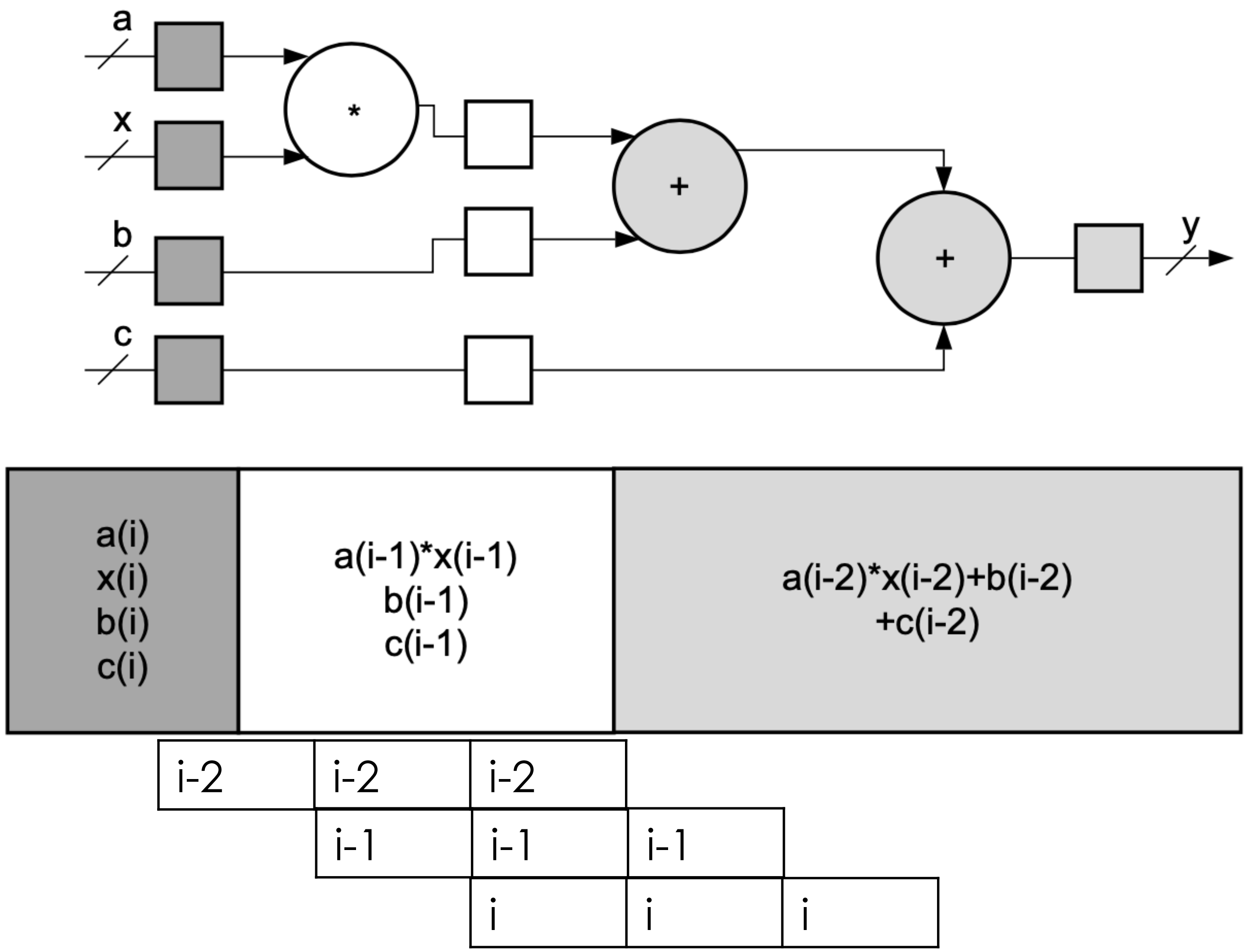


Pipelining

- Raw data: dark gray,
- Semi-computed data: white
- Final data: light gray

All exist simultaneously & each stage result is captured in its own set of registers

Although the latency for such computation is in multiple cycles, there is new result with every cycle



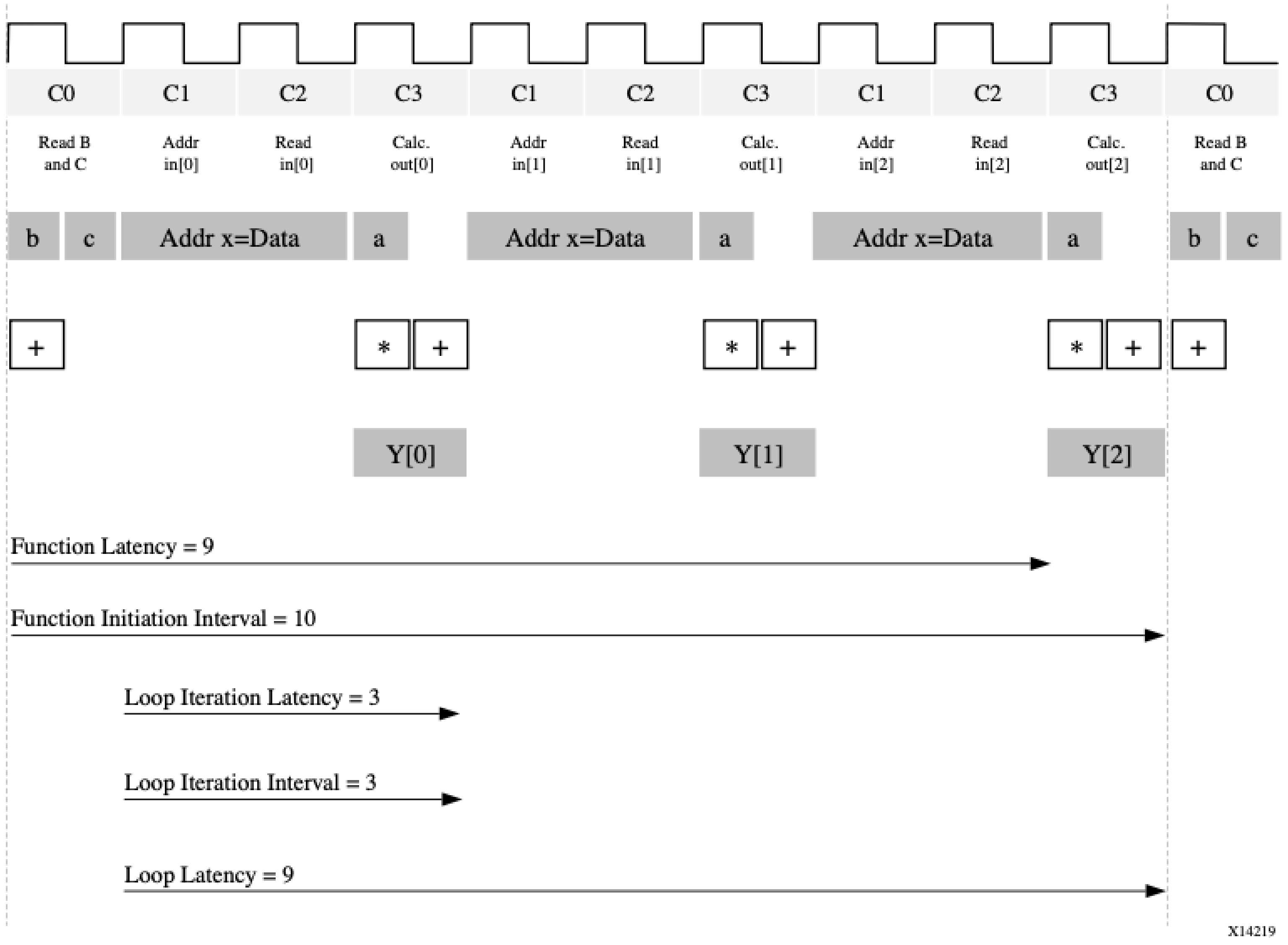
Dataflow

Consumer-producer scenario:

- Producer creates a complete data set before the consumer can start its operation
    - Parallelism by instantiating a pair of BRAM memories arranged as memory banks *ping* and *pong*
    - Each function can access only one memory bank, *ping* or *pong*, for the duration of a function call
    - Guarantees **functional correctness** but limits parallelism

- Consumer can start working with partial results from the producer
    - Both functions are connected through the use of a FIFO memory circuit
    - FIFO act as queue, provides data-level synchronization between the modules
    - both hardware modules are executing during any time of functional call
    - **Exception**: consumer module waits for some data to be available from the producer before beginning computation (***Initiation interval – II***)

Latency & Initiation Interval



Example: Matrix Addition

```
#include <iostream>
#define N 10

void matrix_add(int A[N][N], int B[N][N], int C[N][N]) {

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
}
```

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

Homework: 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

Connecting to “login” machines

- Connect to login machine:
  - ssh -X -Y <username>@login.hep.wisc.edu
  - mkdir /scratch/`whoami`   (If directory exist, go to next bullet)
  - cd /scratch/`whoami`

VNC Server setup

- 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
```

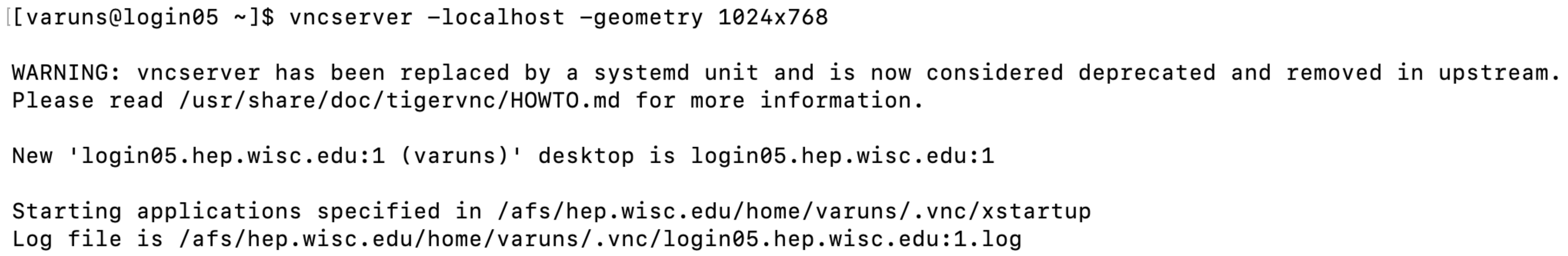
Can be copied from above link as well

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

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 ***login.hep.wisc.edu:1***, where **:1** is your display



```
[varuns@login05 ~]$ 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 'login05.hep.wisc.edu:1 (varuns)' desktop is login05.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/login05.hep.wisc.edu:1.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 ***login.hep.wisc.edu:1***, where **:1** is your display

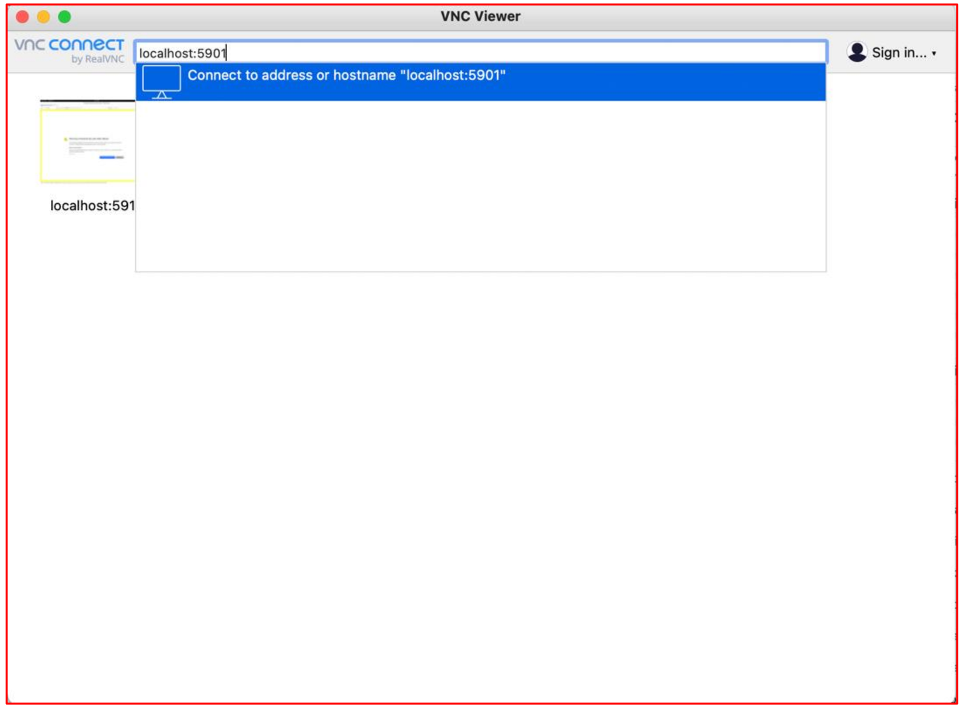
- ssh <username>@login.hep.wisc.edu -L5902:localhost:5902 [In separate terminal tab]
- Make sure you change "<username> to your user name, and "5902" to (5900 + your display number), say 5903, if vncserver told you 3!

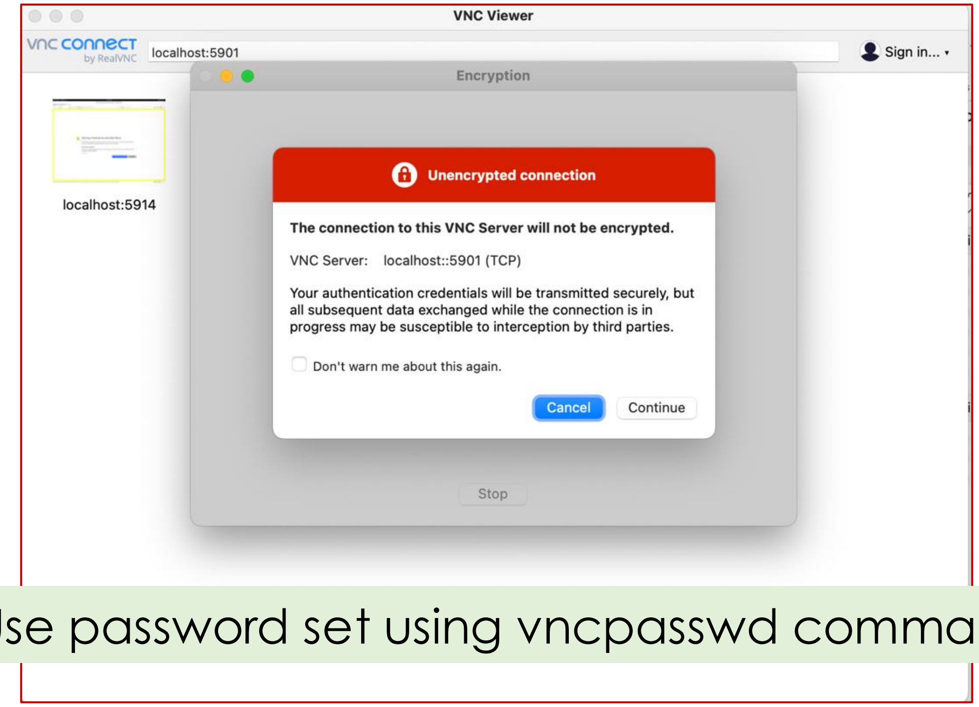
- You can kill your VNC server (:2) using the command:
  - vncserver –kill :2
  - vncserver –list   (check active servers and kill unnecessary ones)

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





The connection to this VNC Server will not be encrypted.

VNC Server: localhost::5901 (TCP)

Your authentication credentials will be transmitted securely, but all subsequent data exchanged while the connection is in progress may be susceptible to interception by third parties.

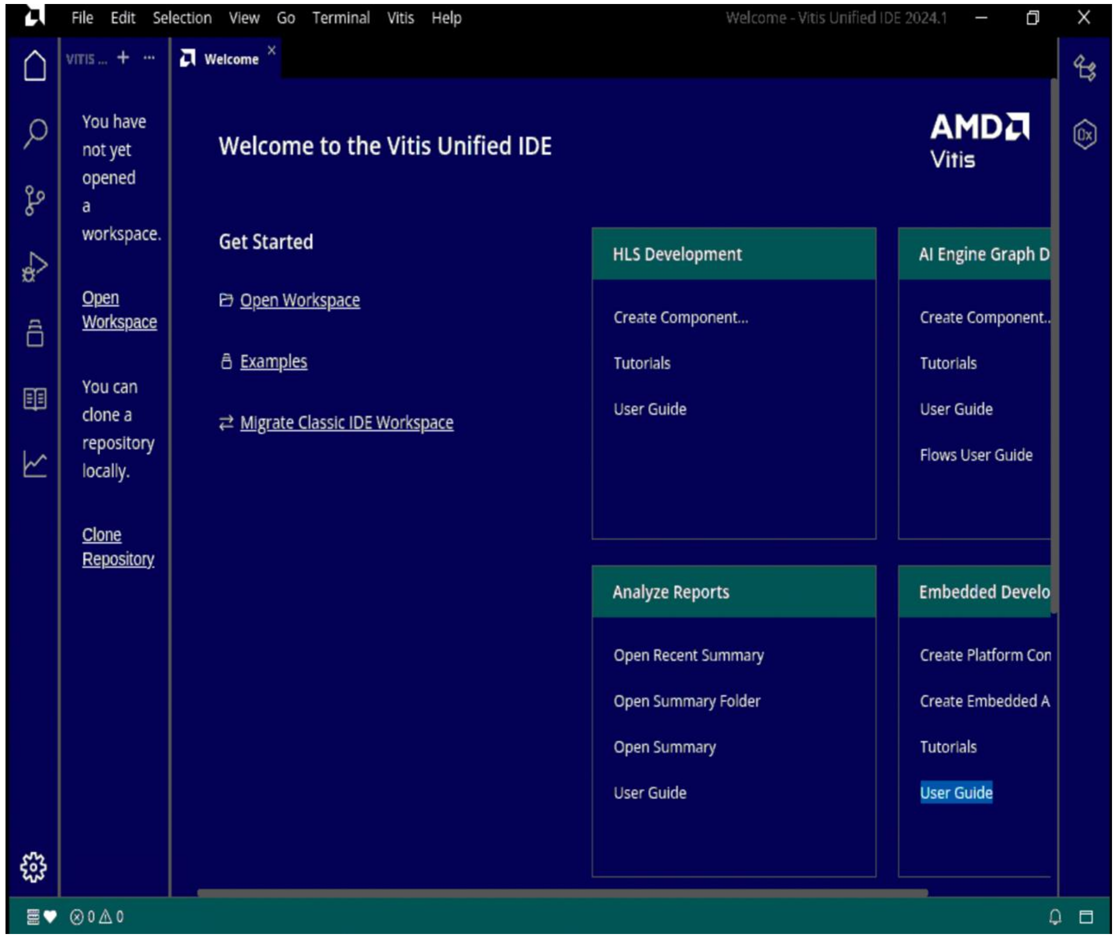
Use password set using vncpasswd command

Connected

Everytime

Summary

- ssh varuns@login.hep.wisc.edu -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/Vitis/2024.1/settings64.sh
  - vitis_hls



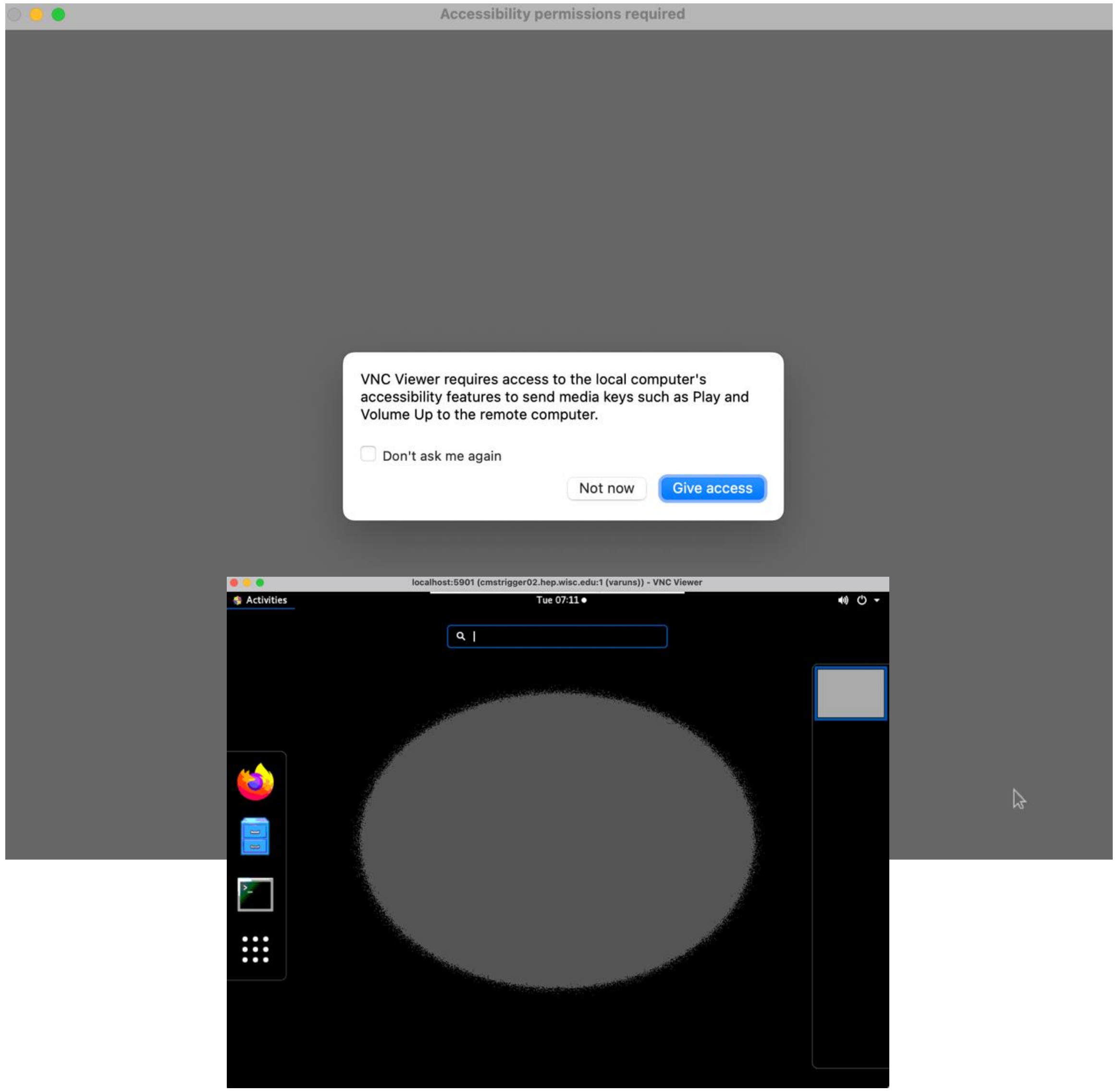
Welcome to the Vitis Unified IDE

You have not yet opened a workspace.

Open Workspace

You can clone a repository locally.

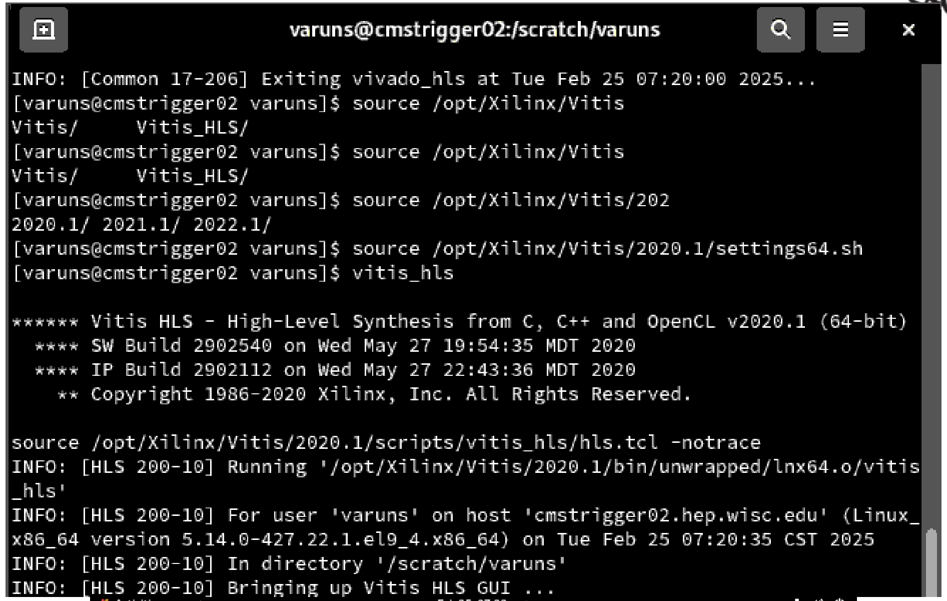
Connected


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

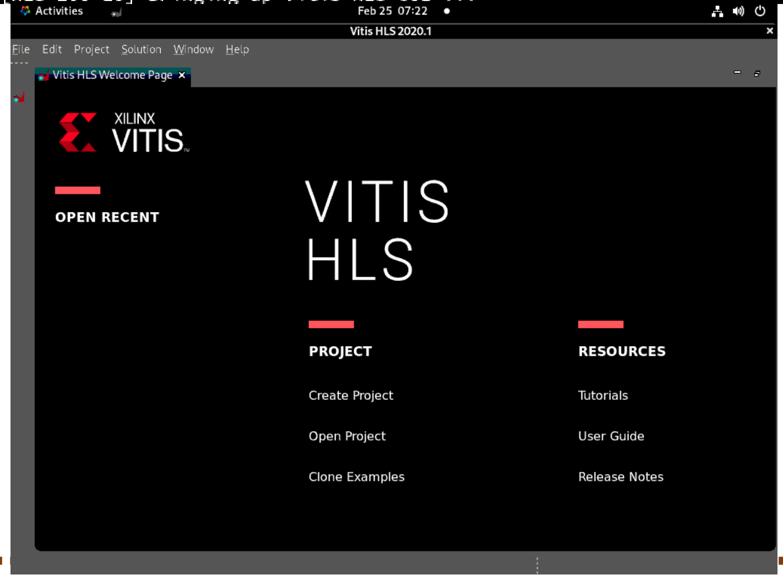
VNC Viewer requires access to the local computer's accessibility features to send media keys such as Play and Volume Up to the remote computer.

Don't ask me again

Not now | Give access



```
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