FL24 CSE565M - Lab 2

Instructor: Dr. Anthony Cabrera

Due Date: November 1, 2024

TOC

Initial Setup for Environment

1. SSH into NERC build server
2. Source the scripts for environment:

   ```
   source /tools/Xilinx/Vitis/2023.1/settings64.sh
   source /opt/xilinx/xrt/setup.sh
   ```

3. Clone a fresh copy of this repository

Overview of Starter Code

Here's an overview of the directory

```
fir11
├── input.dat
├── Makefile
├── makefile_us_alveo.mk
├── out.gold.fir11.dat
├── src
│   ├── fir11_host.cpp
│   ├── fir11_kernel.cpp
│   ├── fir11_kernel.h
│   ├── fir11_kernel_solution.cpp
│   └── fir11_test.cpp
├── utils.mk
└── xrt.ini
```

- The `*.dat` files contain the input and expected data.
- The `*.mk` files and the `Makefile` contain the build infra
- The `xrt.ini` file contains a runtime configuration

Baseline Code

1. Look at `fir11_host.cpp`. Describe what this code does in a few sentences.

   > **Tip**
   >
   > In part, the print statements in the code can help you here.

2. Populate `fir11_kernel.cpp` with a baseline version of an 11 point `fir` filter.

   > **Tip**
   >
   > You may use the implementation from Studio 1

Software Emulation

1. Build and run the software emulation version of your baseline.

   ```
   make all TARGET=sw_emu
   PLATFORM=/opt/xilinx/platforms/xilinx_u280_gen3x16_xdma_1_202211_1/xilinx_u280_gen3x16_xdma_1_202211_1.xpfm
   ./fir11_xrt -x
   build_dir.sw_emu.xilinx_u280_gen3x16_xdma_1_202211_1/fir11.xclbin
   ```

2. Verify that the test passes and take a screenshot of the output

Hardware Emulation

Repeate the steps from Software Emulation but replace instances of `sw_emu` with `hw_emu`.

Specifically, build the `hw_emu` version and verify the output.

At this point, you'll have a baseline implementation of an 11-tap filter running through everything but the actual hardware implementation.

"Optimize" the Baseline

Perform any optimization on the baseline

Recall from previous lectures that we took a look at a variety of different optimizations for the FIR filter in class.

Pick any one of these optimizations, implement them, and repeat the software and hardware steps for your optimized kernel.

Setting Up a VNC Server

Before we proceed, we're going to need access a GUI interface to the NERC build server.

Recall the instructions from here and/or here to use a VNC server for GUI access to the build server.

For those who used `vncserver` for a GUI interface to the build server in Lab 0, check to see if your `vncserver` instance is still running using:

```
vncserver -list
```

If the server is still up , you should see something like this

```
cabrera@fpga-tools:~$ vncserver -list

TigerVNC server sessions:

X DISPLAY #     RFB PORT #      PROCESS ID
:3              5903            1541908
```

If the output says anything about a stale handle, remove the vncserver instance using

```
vncserver -kill :<X Display #>
```

e.g.,

```
vncserver -kill :3
```

Then access the server using the previously reference instructions.

Fine-Grained Kernel Information with the Vitis GUI

The following exercise is to familiarize yourself with some of the Vitis tooling you can use to get a better understanding of your kernel(s).

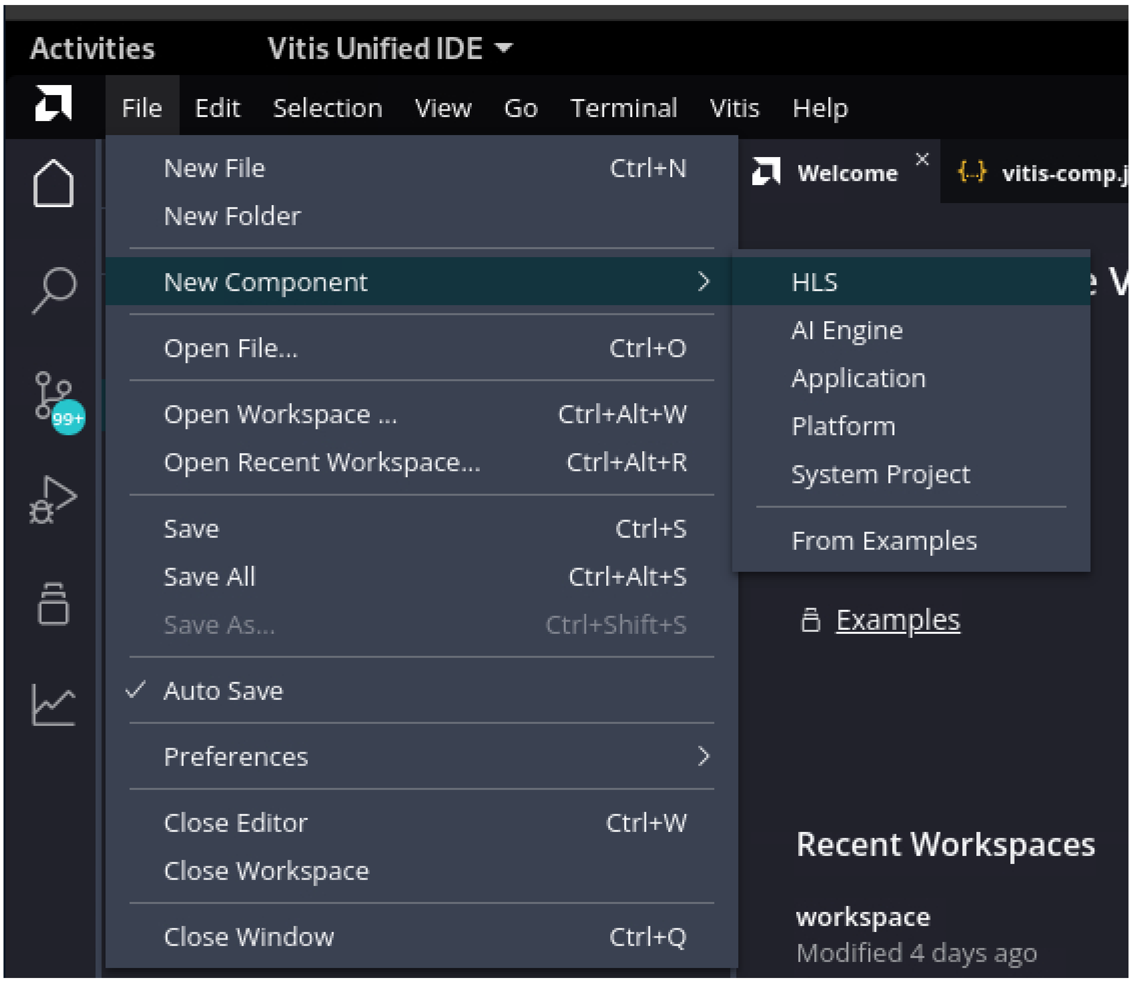
While in the GUI desktop for the build server, open up a terminal and source the commands from earlier.

Issue the command

```
vitis -new -w cse565m_workspace
```

After a minute or so, you should be greeted with the Vitis GUI environment.

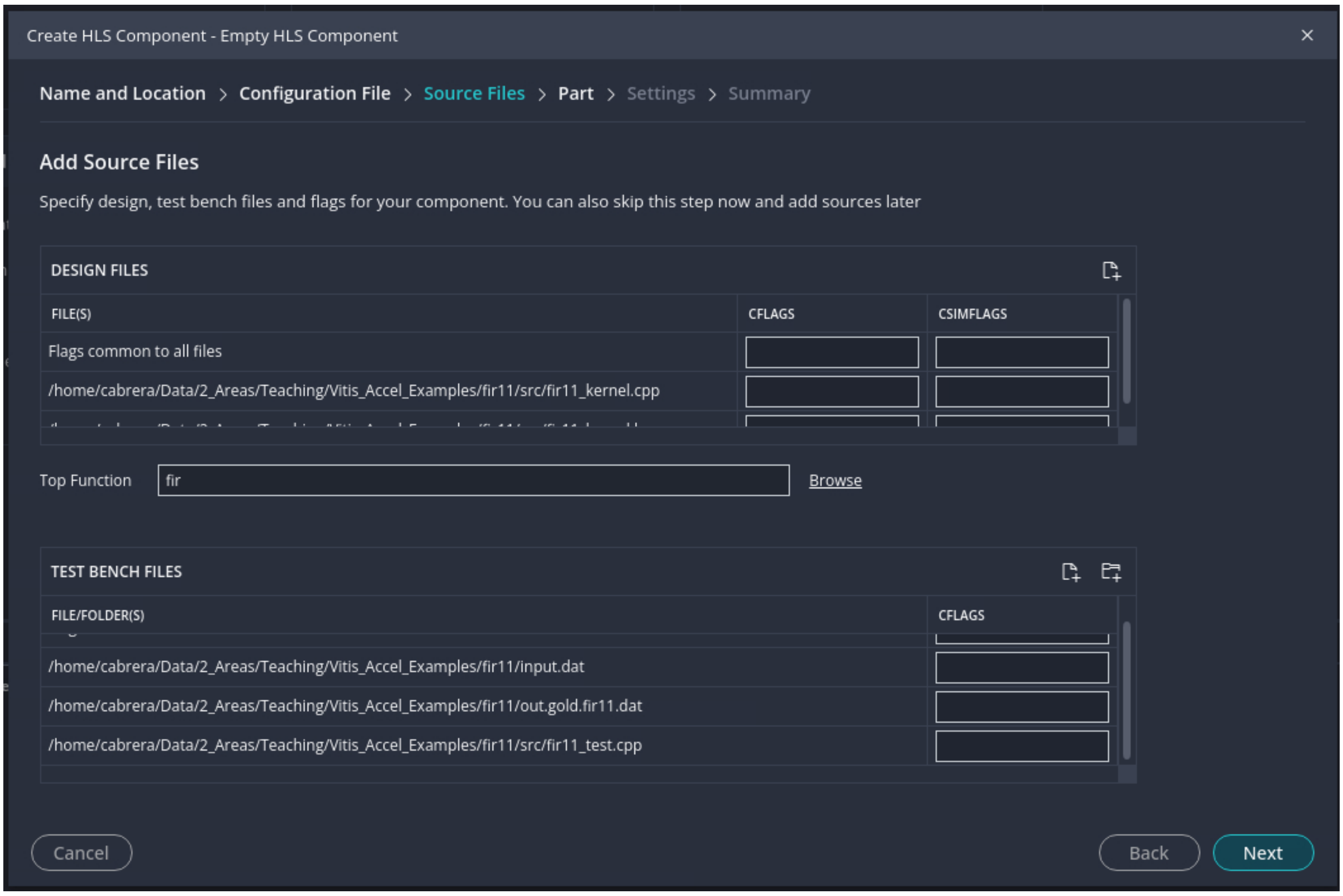
Next, create an HLS Component for your Optimized FIR Kernel



For component name, use `fir11_hls` and click `Next`.

For configuration file, make sure the `Empty File` radio button is selected and click `Next`.

For source files, make sure you add `fir11_kernel.{h,cpp}` and set the `top` function to `fir`. For test bench files, include the input, expected output, and `fir11_test.cpp` testbench file. Click `Next`.


Create HLS Component - Empty HLS Component

Name and Location > Configuration File > Source Files > Part > Settings > Summary

Add Source Files

Specify design, test bench files and flags for your component. You can also skip this step now and add sources later

DESIGN FILES

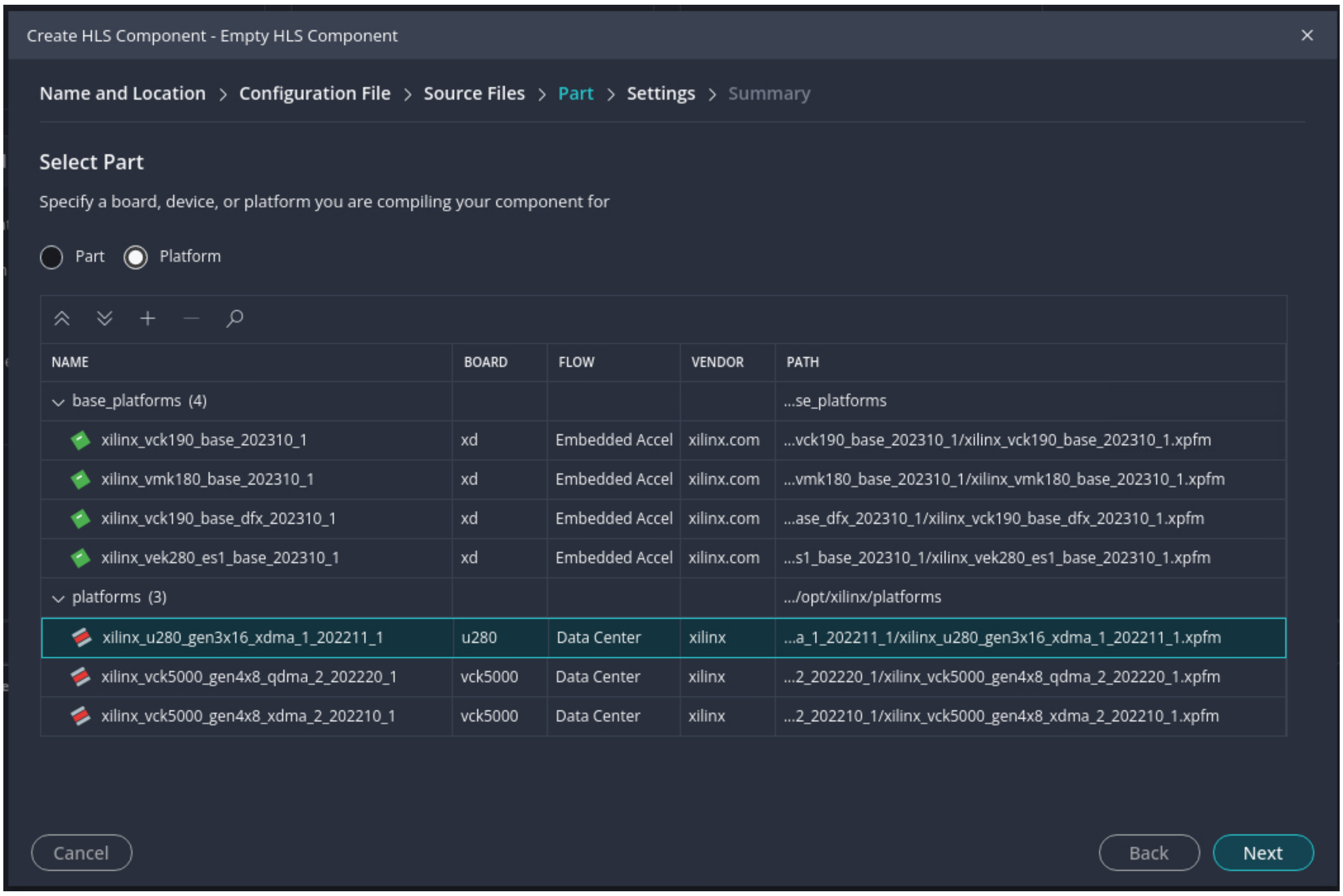
FILE(S)	CFLAGS	CSIMFLAGS
Flags common to all files		
/home/cabrera/Data/2_Areas/Teaching/Vitis_Accel_Examples/fir11/src/fir11_kernel.cpp		

Top Function: fir

TEST BENCH FILES

FILE/FOLDER(S)	CFLAGS
/home/cabrera/Data/2_Areas/Teaching/Vitis_Accel_Examples/fir11/input.dat	
/home/cabrera/Data/2_Areas/Teaching/Vitis_Accel_Examples/fir11/out.gold.fir11.dat	
/home/cabrera/Data/2_Areas/Teaching/Vitis_Accel_Examples/fir11/src/fir11_test.cpp	

For the Part settings, click the `Platform` radio button and select the `u280` platform option.


Create HLS Component - Empty HLS Component
Name and Location > Configuration File > Source Files > Part > Settings > Summary
Cancel
Back
Next

Select Part

Specify a board, device, or platform you are compiling your component for

Part [✓ Platform]

NAME	BOARD	FLOW	VENDOR	PATH
base_platforms (4)				...se_platforms
xilinx_vck190_base_202310_1	xd	Embedded Accel	xilinx.com	...vck190_base_202310_1/xilinx_vck190_base_202310_1.xpfm
xilinx_vmk180_base_202310_1	xd	Embedded Accel	xilinx.com	...vmk180_base_202310_1/xilinx_vmk180_base_202310_1.xpfm
xilinx_vck190_base_dfx_202310_1	xd	Embedded Accel	xilinx.com	...ase_dfx_202310_1/xilinx_vck190_base_dfx_202310_1.xpfm
xilinx_vek280_es1_base_202310_1	xd	Embedded Accel	xilinx.com	...s1_base_202310_1/xilinx_vek280_es1_base_202310_1.xpfm
platforms (3)				.../opt/xilinx/platforms
xilinx_u280_gen3x16_xdma_1_202211_1	u280	Data Center	xilinx	...a_1_202211_1/xilinx_u280_gen3x16_xdma_1_202211_1.xpfm
xilinx_vck5000_gen4x8_qdma_2_202220_1	vck5000	Data Center	xilinx	...2_202220_1/xilinx_vck5000_gen4x8_qdma_2_202220_1.xpfm
xilinx_vck5000_gen4x8_xdma_2_202210_1	vck5000	Data Center	xilinx	...2_202210_1/xilinx_vck5000_gen4x8_xdma_2_202210_1.xpfm

In the Settings window, make sure the `Vitis Kernel Flow` radio button is selected and click `Next`.

Review the summary and click `Finish`.

C Simulation

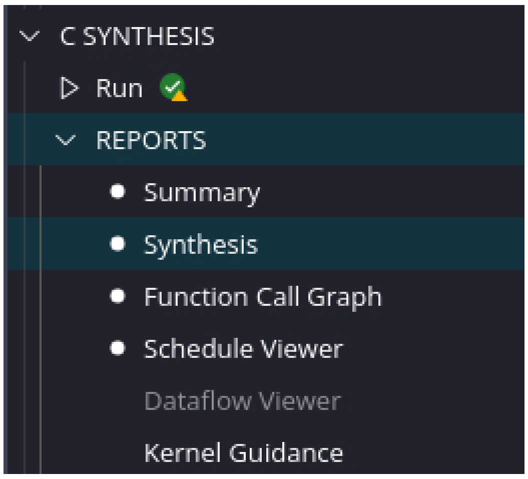
To simulate the kernel using the testbench code, click the `Run` option under `C Simulation` in the pane in the bottom left-hand corner.

Verify that the simulation was successful and take a screenshot of the resulting terminal.

C Synthesis

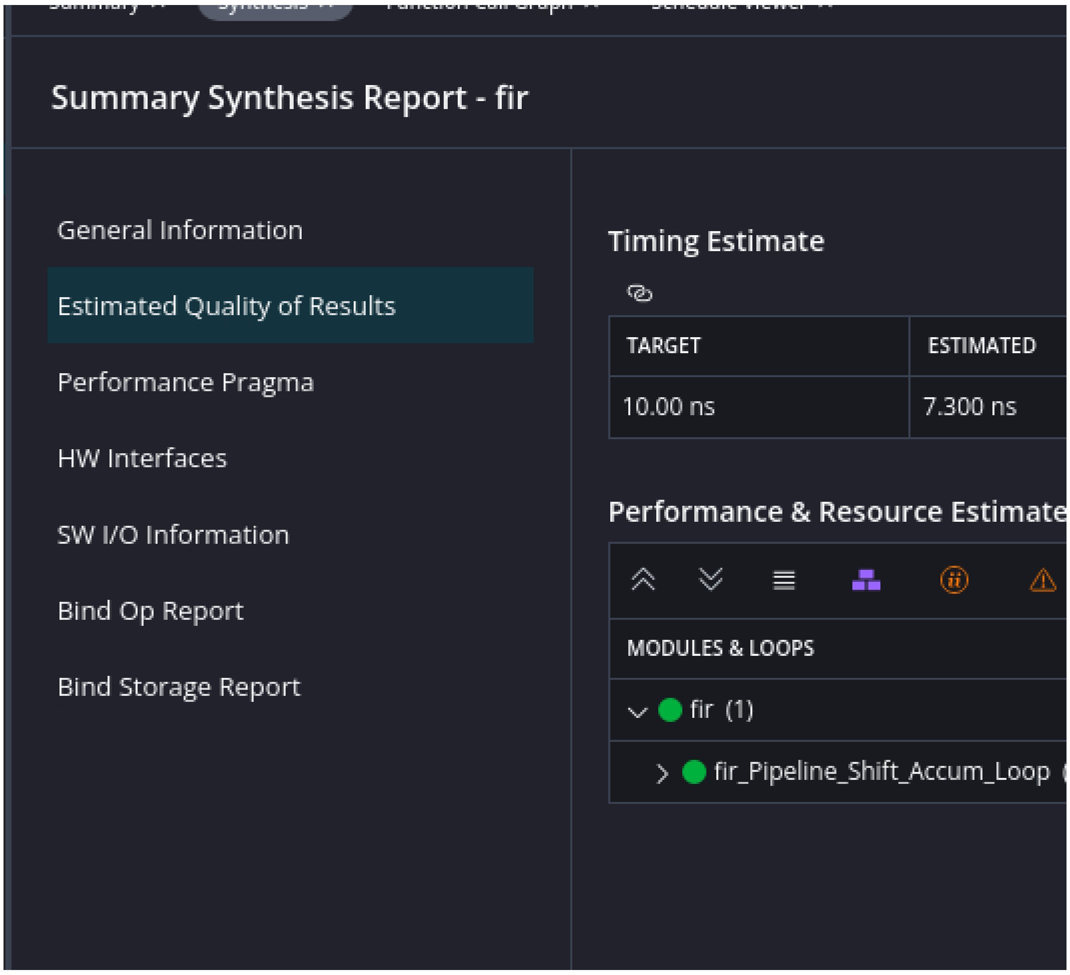
Run the C synthesis of this kernel by clicking the `Run` option under C synthesis.

Verify that the kernel synthesizes with no errors. Then, take a look at the reports generated.



Take a screenshot of the report you find most interesting and describe it.

Note that each report has many things you can interact with, e.g., the synthesis option has the following reports:



Summary Synthesis Report - fir

Timing Estimate

TARGET	ESTIMATED
10.00 ns	7.300 ns

Performance & Resource Estimate

Implementaiton

Finally, click the `Run` option under implementation.

Take a look at the reports and summarize what you observe.

Running on the Alveo U280

The next part can be done outside of the GUI and instead in a terminal window.

Repeate the steps from Software Emulation but replace instances of `sw_emu` with `hw`.

> **Tip**
>
> Recall the instructions from Lab 1 for building the hardware. It will probably take about ~2 hours, so `tmux` is your friend here.

Allocate an FPGA instance.

You'll need to copy over the host binary -- `fir11_xrt` -- the `fir11.xclbin` from the `hw` build directory, and the `xrt.ini` file over to the FPGA node.

Run the binary with

```
$ ./fir11_xrt -x fir11.xclbin
```

Verify the kernel acts as expected and take a screenshot of the results.

Vitis Analyzer

The last thing to do is to take a look at the `xrt.run_summary` file generated after running the kernel. We will want to copy that back over to the build server. I will initiate `scp` from the build server.

For me, that looks something like

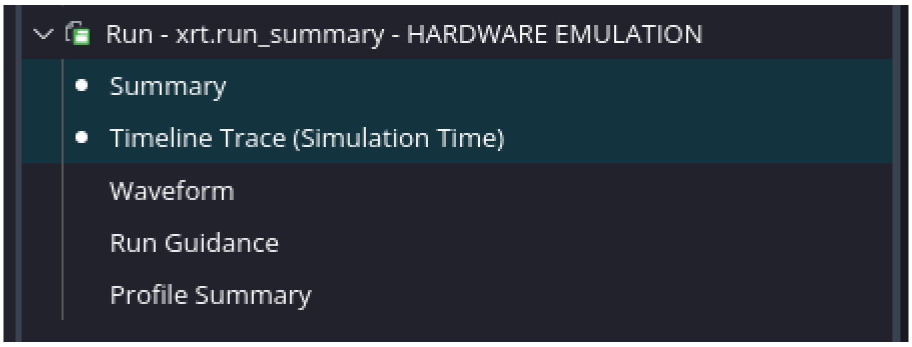
```
scp -i ~/.ssh/carondelet_rsa
cabrera@pc171.cloudlab.umass.edu:/users/cabrera/xrt.run_summary
```

Back in the GUI for the build server, issue

```
vitis_analyzer xrt.run_summary
```

Poke around at the different reports in the right-hand pane.

Click on and take a screenshot of the output of the Timeline Trace summary.


Run - xrt.run_summary - HARDWARE EMULATION
Summary
Timeline Trace (Simulation Time)
Waveform
Run Guidance
Profile Summary

Describe what you see in the Timeline Trace.