Physical Design of My Hardware Accelerator

Tools I Used

1. OpenLane 2 (Open-Source EDA Framework)
   (followed instructions from:
https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html)

2. My own hardware accelerator RTL (Viterbi PE and Top modules)
   These were developed and tested fully in Verilog/SystemVerilog during previous challenges.

3. Vivado (FPGA Flow)
   Also explored Vivado's synthesis flow briefly to cross-check frequency estimation.

Steps Performed:

1. RTL Preparation (from Challenge #15)
   a. My fully working Verilog design for the Viterbi PE and top-level modules were cleaned up, made synthesizable, and made ready for physical synthesis.
   b. Verified that all modules had clean synthesis constraints:
      i. No latches
      ii. No unconnected nets
      iii. Fully static design
2. Installed & Used OpenLane
   a. Read OpenLane2 Newcomers guide carefully.
   b. Installed OpenLane locally on my Linux environment.
   c. Verified prerequisite packages: Docker, make, python3-venv, etc.
   d. Successfully ran default spm (simple processor module) example as demo.

   Prompt used with LLM:
   "How do I install OpenLane 2 on Ubuntu and test my own Verilog design?"

3. Integrated My Viterbi Design
   a. Created OpenLane config directory:
      ```
      designs/viterbi/
      ├── config.tcl
      ├── src/viterbi_top.v
      ├── src/viterbi_pe.v
      ```
   b. Defined clock constraints (config.tcl)
   c. Successfully ran full OpenLane flow: synthesis → floorplanning → placement → CTS → routing → GDS export.

4. FPGA Flow
   a. Installed Vivado (WebPACK version) locally.
   b. Imported same RTL into Vivado design flow.
   c. Synthesized for Artix-7 FPGA device.
   d. Achieved timing: Max frequency: ~110 MHz

Physical Design of My Hardware Accelerator

Prompt used with LLM:
"How do I run a simple Vivado synthesis for my Verilog design and get maximum frequency?"

This challenge helped me connect my functional hardware accelerator design to actual implementation-level metrics.
I now understand how to estimate real-world chip performance, and feel more comfortable transitioning between software models, RTL design, and physical hardware prototypes.