ICLAD 2025 GenAI Hackathon

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

CEDA

SIGDA

Qualcomm

Google

Hackathon Overview

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

CEDA

SIGDA

Qualcomm

Google

Problem Categories and Track

🏞️ Hackathon Tracks

The hackathon features two exciting tracks:

💻 SLM Challenge

- Solve design tasks using **small language models (SLMs)** on **low-cost laptops** provided at the event.

☁️ Open Challenge

- Use **any GenAI models**, including **large language models (LLMs)** running in the **cloud**. For this hackathon, we have support for Gemini APIs via Google Cloud Platform (GCP).

Participants choose any one track

Problem Categories and Track

- 🛠️ **GenAI for Spec2Tapeout (ASU)**
  Build an agent that generates RTL and GDS from design specs—automated start-to-finish flow.
- 🔍 **GenAI for Design Verification (Google)**
  Given buggy RTL and specs, create testbenches to catch sneaky logic bugs 🐞⚡
- 🌀 **Miscellaneous CVDP Problems (NVIDIA)**
  Different tasks that include RTL debugging, testbench integration, design edits, and EDA workflow setup.
- 🔧 **GenAI for Logic Synthesis (Qualcomm, NXP)**
  Generate buggy RTL from the buggy netlist and correct RTL files.

Each category has **multiple problems** with varying levels of difficulty. Solve as many problems as you can across multiple categories. Winners will be named **in each category and track**.

Participants choose any number of problem categories that can be solved with either track

Acknowledging the Group Effort

Animesh Basak Chowdhury
Qualcomm/NXP

Guillaume Shippee
Qualcomm

Mark Ho
NVIDIA

Vidya Chhabria
Arizona State University

Ivan Lobov
Google

Problem contributors:
People you can find and ask questions!

NVIDIA Team:
- Mark Ren
- Nathaniel Pickney
- Mark Ho

Qualcomm Team:
- Rajeev Jain
- Ankita Nayak
- Guillaume Shippee
- Lindsay Kostas
- Animesh Basak Chowdhury (NXP)
- Adrian Nunez-Rocha

ICLAD General Chairs:
- Mark Ren, NVIDIA
- Siddharth Garg, NYU

Google Team:
- Sergio Guadarrama
- Ivan Lobov
- Nathalie Beauguerlange

ASU Team:
- Vidya A. Chhabria
- Vikram Gopalakrishnan
- Sai Harika Julakanti

Problem Categories

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® sigda Qualcomm Google

ASU Spec2Tapeout

Vidya A. Chhabria

62 THE CHIPS TO SYSTEMS CONFERENCE

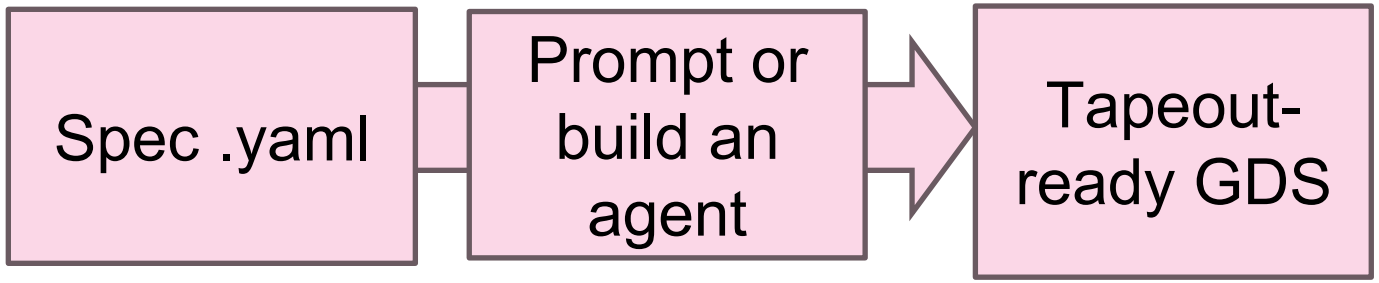
SPONSORED BY CEDA® IEEE Council on Electronic Design Automation SIGDA acm

Problem Description

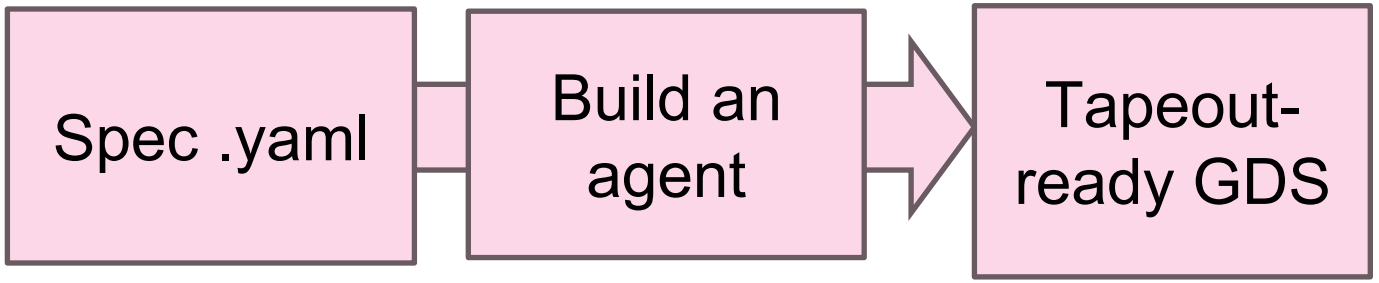
Your goal:

- Use an LLM to write RTL for a given specification
- Evaluate the RTL using the provided testbench or use an agent to generate a testbench
- Use an LLM to write a script that runs OpenROAD flow-scripts by generating the required intermediate files

Visible problems (5 problems)



Hidden problems (5 hidden problems)

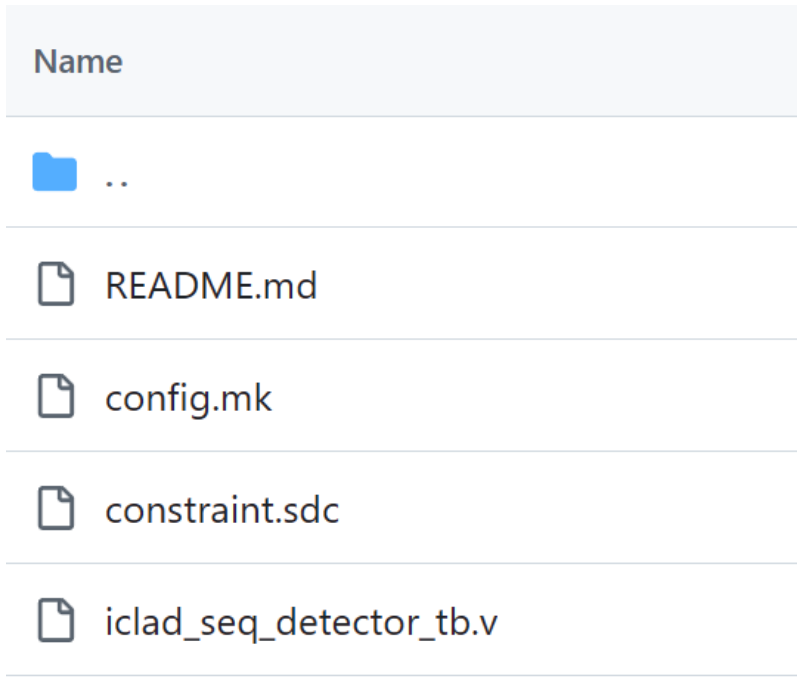


Inputs and Outputs

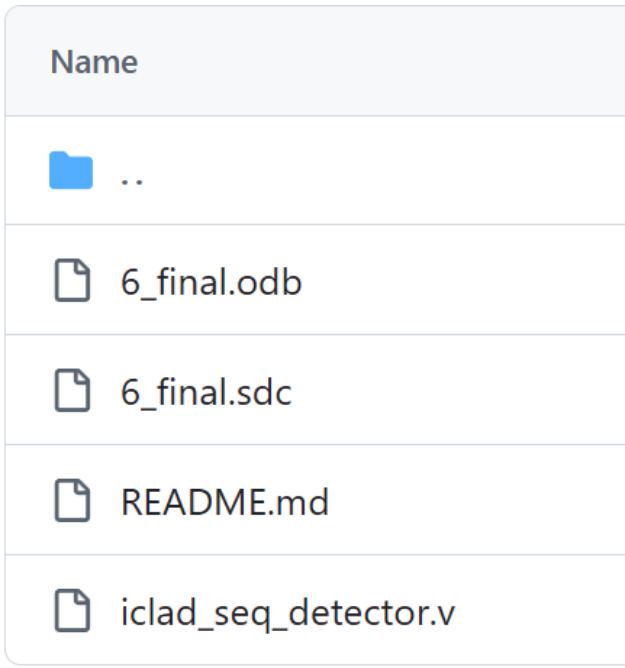
Input p1.yaml

```yaml
seq_detector_0011:
  description: Detects a binary sequence "0011" in the input stream.
  tech_node: SkyWater 130HD
  clock_period: 1.1ns
  ports:
    - name: clk
      direction: input
      type: logic
      description: Clock input
    - name: reset
      direction: input
      type: logic
      description: Synchronous reset (active high)
    - name: data_in
      direction: input
      type: logic
      description: Serial data input
    - name: detected
      direction: output
      type: logic
      description: Asserted high for one cycle when '0011' is detected.
  module_signature: |
    module seq_detector_0011(
        input clk,
        input reset,
        input data_in,
        output reg detected
    );
  sequence_to_detect: '0011'
  sample_input: '0001100110110010'
  sample_output: '0000010001000000'
```

Intermediate files



Outputs



One possible path to arrive at the solution

Submission and Evaluations

Submission Instructions

- Place **visible solutions** inside the solutions/ folder in the GH repository for each problem.
- For **hidden problems**, include clear **documentation** on how to run your agent.
- Once your solutions are ready:
  - **Email** the path of the solutions/ folder in your VM, and your **VM name**, to: iclad2025-hackathon@googlegroups.com
  - **Fork** the ASU repository to your own GitHub account and e**mail the link** to your forked repository to the same address.

Evaluation

- Testbenches and scripts are provided to verify your solutions
  - Functional correctness via testbench
  - Metrics correctness for specification via OpenROAD
- Each problem has a different score based on difficulty
- Hidden problems carry more weight

Google Design Verification

Ivan Lobov

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation SIGDA ACM

Problem Description: Inputs and Outputs

Goal: Build an AI agent that automatically writes a *Verilog testbench* based on a **natural language specification**.

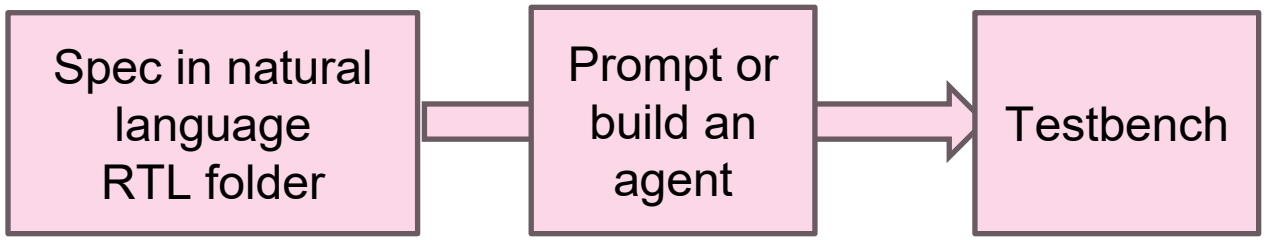
- **Inputs:**
  - A **natural language spec** describing RTL behavior
  - **31 RTL module implementations**, only one is correct
- **Outputs:**
- Testbench that passes for the correct RTL but fails for the rest

Instructions to build agent

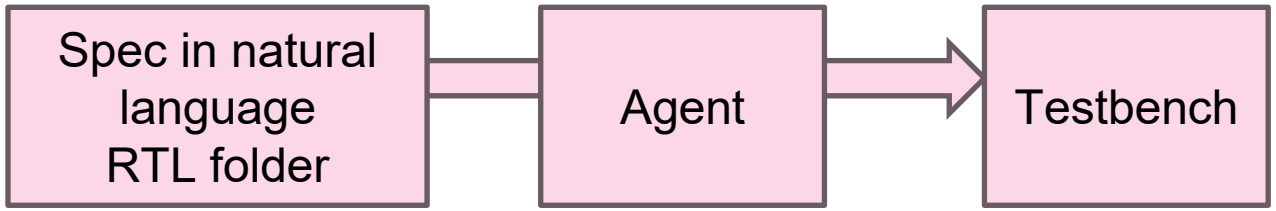
Agent Implementation:

- Implement the generate_testbench() function in test_harness/agent.py
- Must output a complete **Verilog testbench** string
- Execution must finish within 5 minutes
- Use iverilog (via subprocess.run) to simulate

Visible problems (10 problems)



Hidden problems (10 problems)



Submissions and Evaluation

Submission Instructions

- Place testbenches for visible problems in the GH repository for each problem (replace the current dummy testbench)
- Replace the test_harness/agent.py file for hidden problems
- Once your solutions are ready:
  - **Email** the path of the repo in your VM, and your **VM name**, to: iclad2025-hackathon@googlegroups.com
  - **Fork** the repository to your own GitHub account and e**mail the link** to your forked repository to the same address.E

Evaluation Details

https://github.com/ICLAD-Hackathon/Google-Verification-ICLAD25-Hackathon

Qualcomm Logic Synthesis

Animesh Basak Chowdhury, Guillaume Shippee

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation sigda acm

Problem: Regenerate Buggy RTL and Smart ECO patch generation

- cd Qualcomm-Problems directory
- Use golden_rtl + buggy_netlist to recreate buggy_rtl
- Testbenches provided under tb/

📦 Directory Structure

```
.
├── buggy_netlist/
│   └── aes_bug_v1/
├── buggy_rtl/
│   ├── aes_bug_v1/
│   └── team_01/
│       ├── aes_bug_v2/
├── golden_rtl/
│   └── aes/
├── patched_netlist/
│   └── team_01/
├── tb/
│   ├── buggy_v1/
│   └── golden/
```

🧠 Objective

1) Reverse-engineer the **buggy RTL** from the **synthesized netlist**, using the **golden RTL** as a reference.
2) Generate minimal patch to buggy netlist to correct functionality.

You'll be evaluated on whether your reconstructed RTL passes the testbench designed for the buggy version.

Details: RTL re-construction and ECO generation

Problem: Regenerate Buggy RTL and Smart ECO patch generation

🔬 Golden RTL Test

```
cd tb/golden
iverilog -o golden_test
tb_aes_cipher_top.v
../../golden_rtl/*.v
vvp golden_test
```

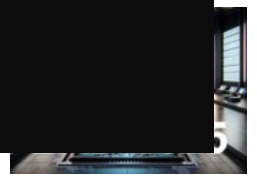
```
(base) gmrd01@gmrd:~/iclad_hackathon/ICLAD-Hackathon-2025/problem-categories/Qualcomm-Problems/golden_rtl/aes$ iverilog -o golden_test ../../tb/gold
en/tb_aes_cipher_top.v *.v
(base) gmrd01@gmrd:~/iclad_hackathon/ICLAD-Hackathon-2025/problem-categories/Qualcomm-Problems/golden_rtl/aes$ vvp golden_test
VCD info: dumpfile aes.vcd opened for output.
------------------------------------------------
Test 1:
Key       : 2b7e151628aed2a6abf7158809cf4f3c
Plaintext : 6bc1bee22e409f96e93d7e117393172a
Expected  : 3ad77bb40d7a3660a89ecaf32466ef97
Output    : 3ad77bb40d7a3660a89ecaf32466ef97
Test 1 PASSED
------------------------------------------------
Test 2:
Key       : 000102030405060708090a0b0c0d0e0f
Plaintext : 00112233445566778899aabbccddeeff
Expected  : 69c4e0d86a7b0430d8cdb78070b4c55a
Output    : 69c4e0d86a7b0430d8cdb78070b4c55a
Test 2 PASSED
------------------------------------------------
Test 3:
Key       : 10a58869d74be5a374cf867cfb473859
Plaintext : 00000000000000000000000000000000
Expected  : 6d251e6944b051e04eaa6fb4dbf78465
Output    : 6d251e6944b051e04eaa6fb4dbf78465
Test 3 PASSED
------------------------------------------------
Test 4:
Key       : 00000000000000000000000000000000
Plaintext : ffffffffffffffffffffffffffffffff
Expected  : 3f5b8cc9ea855a0afa7347d23e8d664e
Output    : 3f5b8cc9ea855a0afa7347d23e8d664e
Test 4 PASSED
------------------------------------------------
PERCENTAGE TEST PASSED: 100% (4/4)
------------------------------------------------
(base) gmrd01@gmrd:~/iclad_hackathon/ICLAD-Hackathon-2025/problem-categories/Qualcomm-Problems/golden_rtl/aes$
```

🐞 Buggy RTL Test

Place your generated RTL inside buggy_rtl/team_name/.

```
cd tb/buggy_v1
iverilog -o buggy_test
b_aes_cipher_top.v
../../buggy_rtl/team_name/*.v
vvp buggy_test
```

```
(base) gmrd01@gmrd:~/iclad_hackathon/ICLAD-Hackathon-2025/problem-categories/Qualcomm-Problems/buggy_rtl/aes_bug_v1$ iverilog -o buggy_test ../../tb
/buggy_v1/tb_aes_cipher_top.v *.v
(base) gmrd01@gmrd:~/iclad_hackathon/ICLAD-Hackathon-2025/problem-categories/Qualcomm-Problems/buggy_rtl/aes_bug_v1$ vvp buggy_test
VCD info: dumpfile aes.vcd opened for output.
------------------------------------------------
Test 1:
Key       : 2b7e151628aed2a6abf7158809cf4f3c
Plaintext : 6bc1bee22e409f96e93d7e117393172a
Expected  : c34feed2a9e8a2b46032d2cce21f0c52
Output    : c34feed2a9e8a2b46032d2cce21f0c52
Test 1 PASSED
------------------------------------------------
Test 2:
Key       : 000102030405060708090a0b0c0d0e0f
Plaintext : 00112233445566778899aabbccddeeff
Expected  : d1b7eb0ef5440d98b2399bcd60ec4866
Output    : d1b7eb0ef5440d98b2399bcd60ec4866
Test 2 PASSED
------------------------------------------------
Test 3:
Key       : 10a58869d74be5a374cf867cfb473852
Plaintext : 00000000000000000000000000000000
Expected  : bb7bcc52c0c0fbb1fc149625a8c7854b
Output    : bb7bcc52c0c0fbb1fc149625a8c7854b
Test 3 PASSED
------------------------------------------------
Test 4:
Key       : 00000000000000000000000000000022
Plaintext : ffffffffffffffffffffffffffffffff
Expected  : e20f147b056f5bfa5946e2a40d0d6598
Output    : e20f147b056f5bfa5946e2a40d0d6598
Test 4 PASSED
------------------------------------------------
PERCENTAGE TEST PASSED: 100% (4/4)
------------------------------------------------
(base) gmrd01@gmrd:~/iclad_hackathon/ICLAD-Hackathon-2025/problem-categories/Qualcomm-Problems/buggy_rtl/aes_bug_v1$
```



How to submit the solutions?

Submit your solution as a ZIP file: team_name.zip

On unzipping, it should contain:

```
ICLAD-Hackathon-2025/
└── problem-categories/
    └── Qualcomm-Problems/
        ├── buggy_rtl/
        |   └── team_name/
        ├── patched_netlist/
        |   └── team_name/
        └── logs/
            └── team_name/   <- Chat logs, code, notes
```

💡 **Include your LLM prompt logs and intermediate explorations for extra credit!**

🤝 **Code of conduct:** We expect participants to use only the SLMs in solving problems.

FAQ on SLM submission for other tracks:

For **ASU-ICLAD** and **Google** tracks, create under their subproblem:

- submission/<team_name>/problemK/
- logs/<team_name>/

For OPEN challenge, refer to the readme for more details on the submission category for each submission category.

🖼️ Snapshot Diagrams (For PPT/Docs)

Add the following diagrams to logs:

- Workflow for EDA tool invocation
- Agentic/Non-agentic approaches used to interact with SLMs and corresponding logs

NVIDIA ICLAD25 MISC

Mark Ho

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

NVIDIA-ICLAD25 ENV

- Agent under test docker env for every problem -> Participant provide their agent code and build the docker image
- Evaluation test harness docker env

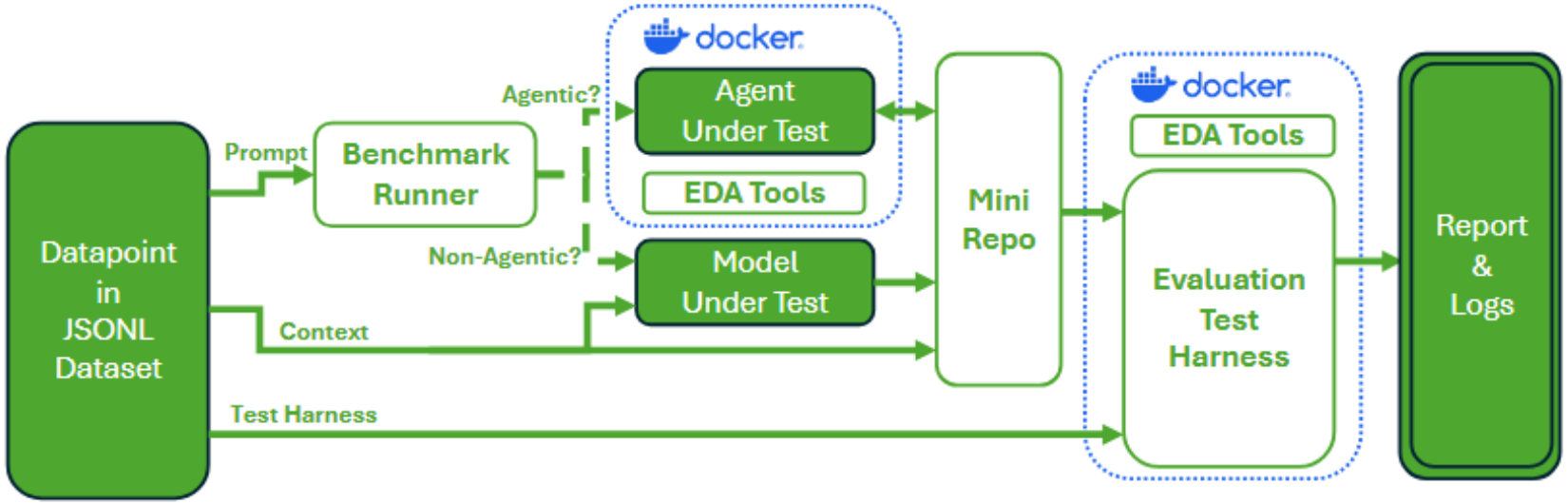


Figure 1: Benchmark Evaluation Flow.

NVIDIA-ICLAD25 Installation Steps

Step 1: Install the framework required python libraries

Setup Instructions

1. **Create a virtual environment** (recommended):

```
# Create virtual environment
python -m venv agent_env

# Activate virtual environment
# On Linux/macOS:
source agent_env/bin/activate
# On Windows:
agent_env\Scripts\activate
```

2. **Install Python dependencies:**

```
pip install -r requirements.txt
```

NVIDIA-ICLAD25 Installation Steps

Step 2: Build your agent

```
# Copy the complete agent example
cp -r examples/agent/ ./my-agent/
cd my-agent/

# Build using the provided script
./build_agent.sh
```

```
#!/bin/sh

# SPDX-FileCopyrightText: Copyright (c) 2025 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
# SPDX-License-Identifier: Apache-2.0

docker build -f Dockerfile-base -t example-agent-base .
docker build -f Dockerfile-agent -t example-agent --no-cache .
~
```

If you only changes some code, skip the example-agent-base docker build.

NVIDIA-ICLAD25 Run benchmark

Recommended Steps:

Step 1: Use the dummy agent to run through the bench mode to create all the problems in work directory

```
python run_benchmark.py -f ./dataset/hackathon-agentic-obfuscated_final_corrected.jsonl -l -g example-agent
# check the run and each problem
cd work
```

Step 2: Example work directory

```
(llm_env) [chiatungh@nvrdarmarco work]$ ls
cvdp_agentic_amber_eagle_lambda      cvdp_agentic_compass_breeze_obsidian  cvdp_agentic_echo_obsidian_lunar     cvdp_agentic_horizon_whisper_panther  cvdp_agentic_nebula_nova_castle       prompt_response.jsonl
cvdp_agentic_amber_prism_tiger       cvdp_agentic_crimson_river_butterfly  cvdp_agentic_ember_meadow_sunrise    cvdp_agentic_ivory_cloud_ocean        cvdp_agentic_starlight_phoenix_comet  raw_result.json
cvdp_agentic_azure_sapphire_tiger    cvdp_agentic_diamond_dolphin_whisper  cvdp_agentic_falcon_willow_dragon    cvdp_agentic_lagoon_dragon_diamond    cvdp_agentic_summit_horizon_diamond   report.json
cvdp_agentic_breeze_crystal_zenith   cvdp_agentic_diamond_orbit_valley     cvdp_agentic_forest_fountain_river   cvdp_agentic_meadow_canyon_sunrise    cvdp_agentic_sunrise_ivory_glacier    report.txt
cvdp_agentic_breeze_velvet_violet    cvdp_agentic_eagle_crystal_lunar      cvdp_agentic_garden_sapphire_amber   cvdp_agentic_meadow_marble_castle     cvdp_agentic_thunder_compass_river    run.log
cvdp_agentic_comet_cloud_tiger       cvdp_agentic_eagle_jade_forest        cvdp_agentic_glacier_amber_cosmic    cvdp_agentic_meadow_river_prism       cvdp_agentic_thunder_diamond_horizon
```

Step 3: Check one case and develop agentic approach

```
cd work/cvdp_agentic_programmable_fsm_dynamic_state_encoding/harness/1/
# invoke the agent run
./run_docker_agent.sh
# debug the agent
./run_docker_agent.sh -d
# run evaluation
./run_docker_harness_direct.sh
```

NVIDIA-ICLAD25 Additional info

Check individual run reports
cd NVIDIA-ICLAD25-Hackathon/work/cvdp_agentic_diamond_orbit_valley/reports

1_agent.txt: agent run log file
1.txt: direct evaluation log file

Debug and run the test
cd NVIDIA-ICLAD25-Hackathon/work/cvdp_agentic_diamond_orbit_valley/harness/1/

The evaluation identify the .patch file. The agent needs to write out answers to the specified /rtl directory.

./run_docker_agent.sh -> run the agent on the problem
./run_docker_harness_<problem>.sh -> run the individual test

Network license issue:
Solution: docker network create licnetwork

NVIDIA-ICLAD25 Evaluation Weights

Problem Name	Evaluate Weight (based on difficulty)
cvdp_agentic_eagle_crystal_lunar_2638	1.15
cvdp_agentic_crimson_river_butterfly_1181	1.15
cvdp_agentic_garden_sapphire_amber_0307	1.15
cvdp_agentic_forest_fountain_river_0702	1.3
cvdp_agentic_echo_obsidian_lunar_0001	1.3
cvdp_agentic_meadow_canyon_sunrise_0439	1.45
cvdp_agentic_nebula_nova_castle_8184	1.6
cvdp_agentic_starlight_phoenix_comet_6246	1.75
cvdp_agentic_glacier_amber_cosmic_3144	1.75
cvdp_agentic_diamond_dolphin_whisper_4792	1.75
cvdp_agentic_meadow_river_prism_8091	1.75
cvdp_agentic_meadow_marble_castle_0301	1.75
cvdp_agentic_azure_sapphire_tiger_4476	1.75
cvdp_agentic_sunrise_ivory_glacier_9089	1.75
cvdp_agentic_amber_eagle_lambda_0001	1.75
cvdp_agentic_ember_meadow_sunrise_1765	1.75
cvdp_agentic_horizon_whisper_panther_1434	1.75
cvdp_agentic_thunder_diamond_horizon_0119	1.75
cvdp_agentic_compass_breeze_obsidian_0016	1.75
cvdp_agentic_summit_horizon_diamond_3314	1.8
cvdp_agentic_amber_prism_tiger_1687	2
cvdp_agentic_eagle_jade_forest_0681	2
cvdp_agentic_ivory_cloud_ocean_3516	2
cvdp_agentic_breeze_velvet_violet_7060	2
cvdp_agentic_lagoon_dragon_diamond_4339	2
cvdp_agentic_comet_cloud_tiger_9313	2
cvdp_agentic_falcon_willow_dragon_8753	2
cvdp_agentic_thunder_compass_river_8383	2
cvdp_agentic_breeze_crystal_zenith_1259	2
cvdp_agentic_diamond_orbit_valley_0001	2

Final Submission

- Agent source code and a Dockerfile
  - Path to both of these files inside your VM or fork and upload the repo in your own GH account.
  - Email the path and the link to : iclad2025-hackathon@googlegroups.com
- Evaluate the benchmark with participants' agent docker image
- Good luck and Enjoy!

Infrastructure

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® sigda Qualcomm Google

Open Challenge Infrastructure

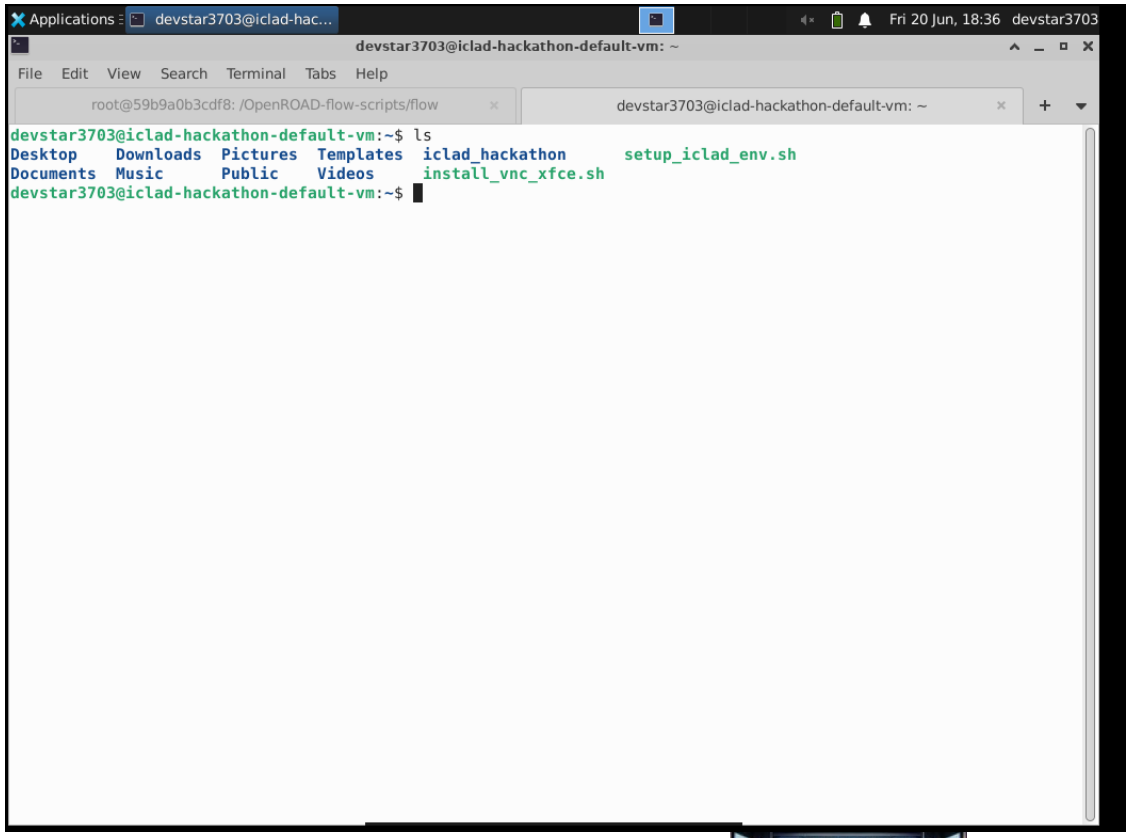
Google Cloud Platform (GCP)

Setup and login to GCP VM

- Username/Password: Provided/will be provided via email.
- VNC Viewer: External IP address and VNC password provided for existing VM (recommended)
- Optional: Documentation to create more VMs is also provided

Virtual Machine Environment

- The ICLAD Hackathon Repository is already pre-cloned and available
  - ~/iclad_hackathon/ICLAD-Hackathon-2025
- The problem categories repository has submodule to all the problems.
- Each problem category:
  - Detailed description
  - Submission process
  - Evaluation



Docker Image Setup For Prerequisites

For the following problem categories:

- GenAI for Spec2Tapeout (ASU)
- GenAI for Design Verification (Google)
- GenAI for Logic Synthesis (Qualcomm

Prerequisites:
OpenROAD-flow-scripts,
iVerilog gtkwave

In the VNC viewer

Start the docker image with the installed prerequisites and mount your repository

- docker run -it --rm -v ~/iclad_hackathon:/workspace/iclad_hackathon iclad_hackathon:latest bash

The docker has the workspace mounted and it will have preinstalled prerequites

Docker Image Setup For Prerequisites

For the following problem categories:

- Miscellaneous NVIDIA CVDP problems
- ~/iclad_hackathon/ICLAD-Hackathon-2025/problem-categories

From the virtual machine:

- Follow instructions available https://github.com/ICLAD-Hackathon/NVIDIA-ICLAD25-Hackathon/README.md
- Docker image will be pre-built for you on the VM to save some time

Interacting with Gemini on GCP

Hello World with Gemini on GCP

```
from vertexai.preview.generative_models import GenerativeModel
import vertexai
vertexai.init(location="us-central1")
model = GenerativeModel("gemini-2.0-flash-001")
response = model.generate_content("Say hi from inside the ICLAD Hackathon container!")
print(response.candidates[0].text)
```

SLM Track Laptop Setup

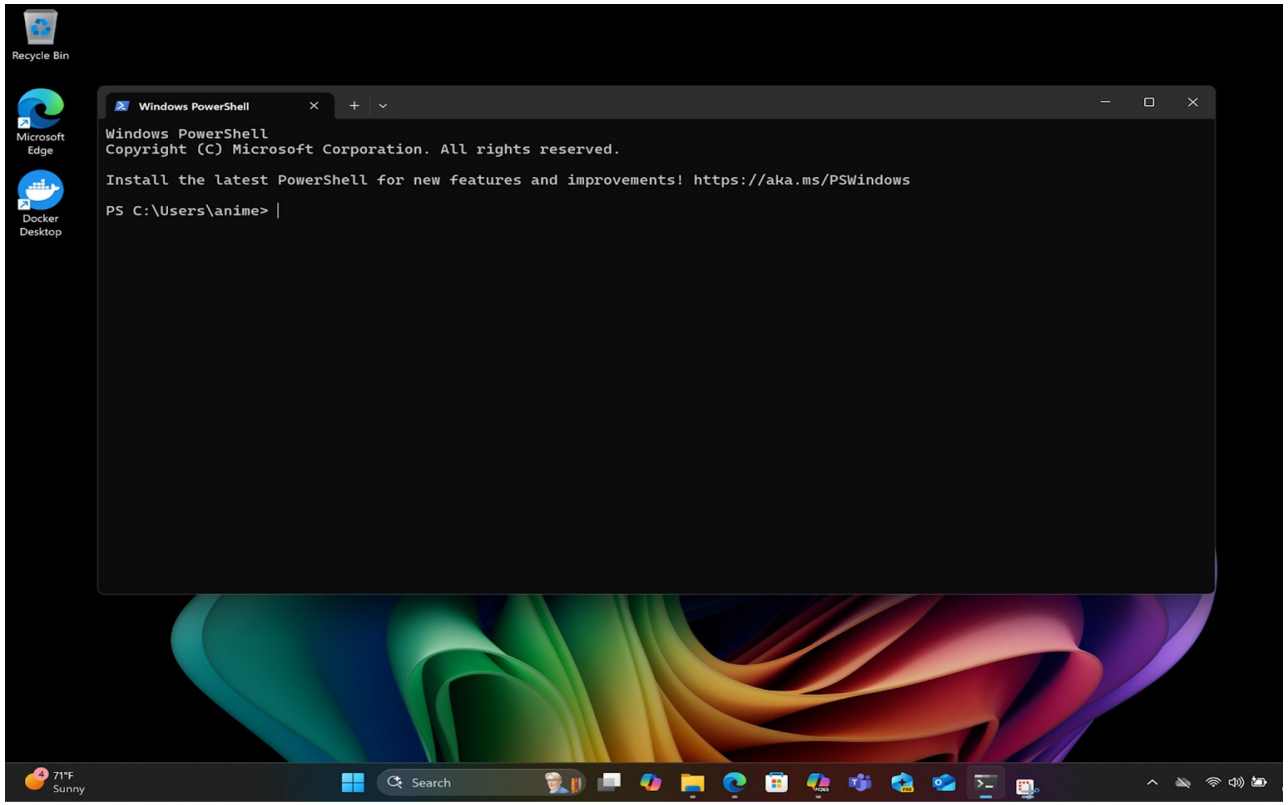
Qualcomm

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation sigda acm

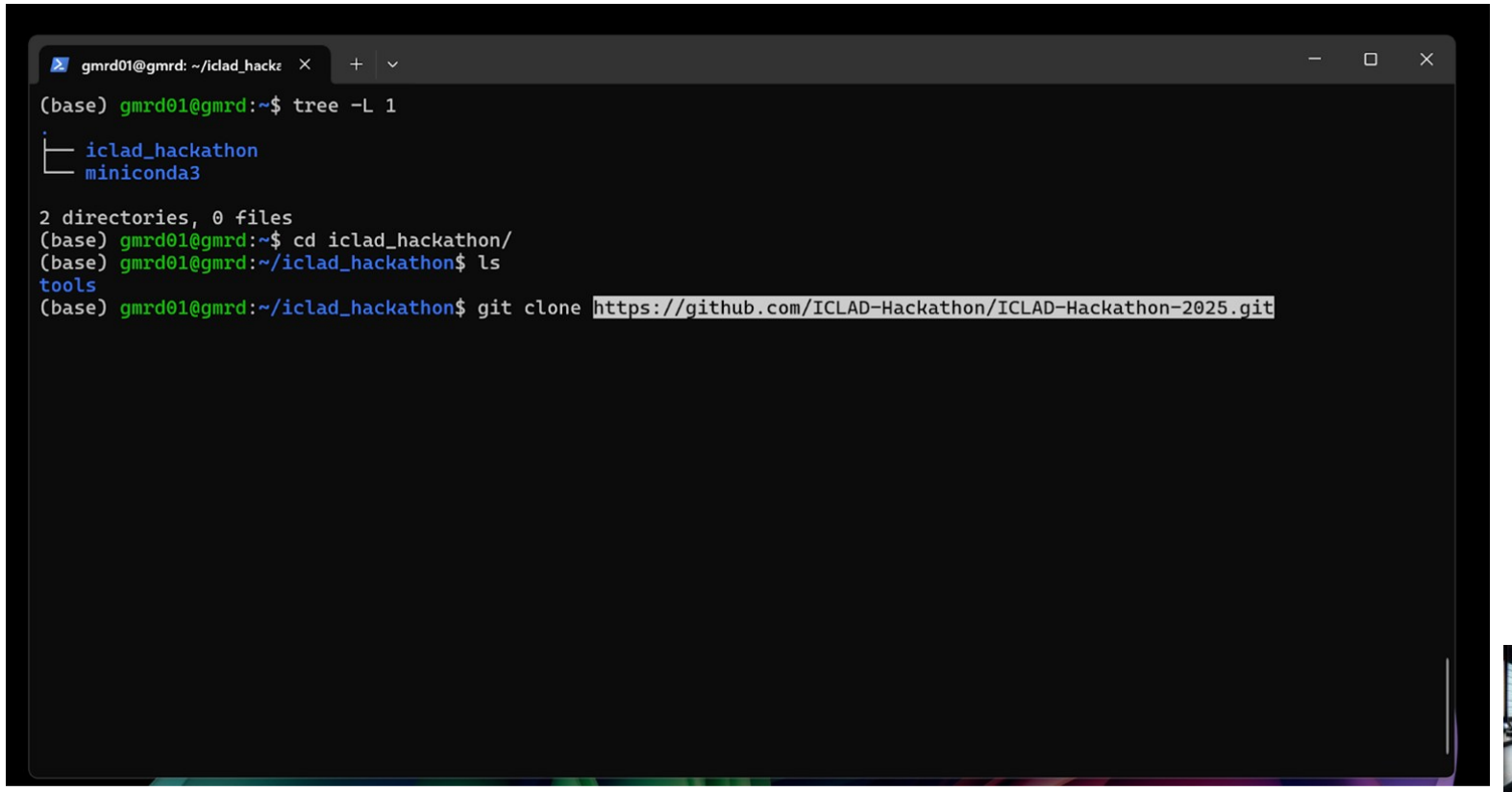
Setup and login

- Username/Password: To be provided
- Launch WSL: wsl --list; wsl -d iclad_2025
- Go to the directory: cd /home/gmrd01
- Run: su gmrd01



Clone ICLAD Repo

- Navigate to iclad_hackathon
- git clone https://github.com/ICLAD-Hackathon/ICLAD-Hackathon-2025.git --recurse-submodules

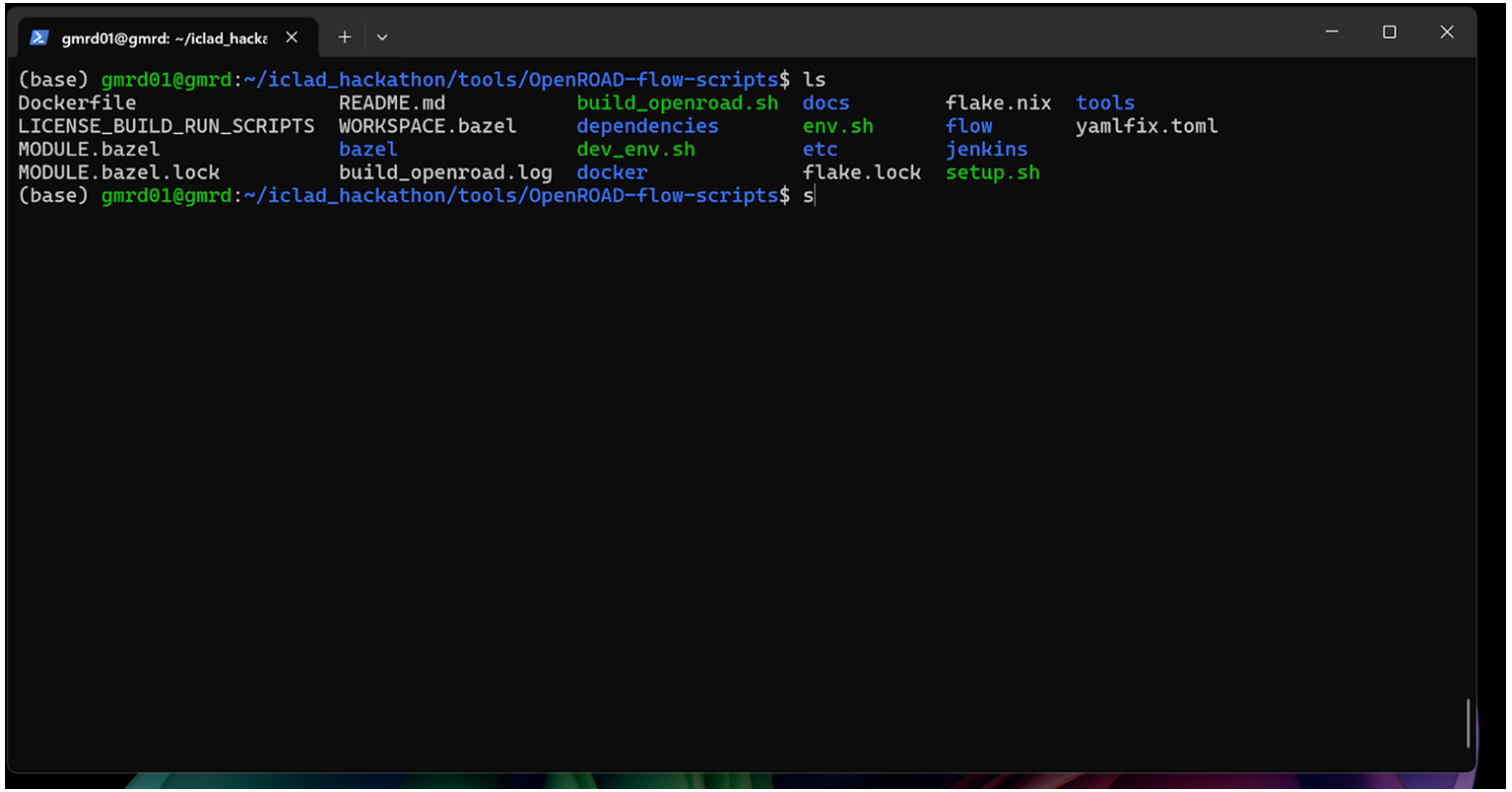


Test EDA tools

- OpenROAD: source env.sh; make (RTL2GDSII flow for benchmark GCD)
- Icarus Verilog: iverilog + vvp hello (Verilog simulation)
- GTKWave: gtkwave dump.vcd (Testbench visualizer)
- Verilator: Verilog simulator with C++ backend

Test EDA tools

```
cd ~/iclad_hackathon/tools/OpenROAD-flow-scripts
source env.sh
cd flow/
make
```



Test EDA tools

```
cd ~/iclad_hackathon/tools/OpenROAD-flow-scripts
source env.sh
cd flow/
make
```

```
(base) gmrd01@gmrd:~/iclad_hackathon/tools/OpenROAD-flow-scripts/flow$ make
/home/gmrd01/iclad_hackathon/tools/OpenROAD-flow-scripts/flow/util/preprocessLib.py -i /home/gmrd01/iclad_hackathon/tools/OpenROAD-flow-scripts/flo
/platforms/nangate45/lib/NangateOpenCellLibrary_typical.lib -o objects/nangate45/gcd/base/lib/NangateOpenCellLibrary_typical.lib
Opening file for replace: /home/gmrd01/iclad_hackathon/tools/OpenROAD-flow-scripts/flow/platforms/nangate45/lib/NangateOpenCellLibrary_typical.lib
Commented 0 lines containing "original_pin"
Replaced malformed functions 0
Replaced capacitive_load_unit 0
Writing replaced file: objects/nangate45/gcd/base/lib/NangateOpenCellLibrary_typical.lib
mkdir -p results/nangate45/gcd/base/
echo 0.46 > results/nangate45/gcd/base/clock_period.txt
/home/gmrd01/iclad_hackathon/tools/OpenROAD-flow-scripts/flow/scripts/synth.sh /home/gmrd01/iclad_hackathon/tools/OpenROAD-flow-scripts/flow/script
/synth_canonicalize.tcl ./logs/nangate45/gcd/base/1_1_yosys_canonicalize.log
Using ABC area script.
Extracting clock period from SDC file: ./results/nangate45/gcd/base/clock_period.txt
Setting clock period to 0.46
1. Executing Liberty frontend: ./objects/nangate45/gcd/base/lib/NangateOpenCellLibrary_typical.lib
2. Executing Liberty frontend: ./objects/nangate45/gcd/base/lib/NangateOpenCellLibrary_typical.lib
3. Executing Verilog-2005 frontend: /home/gmrd01/iclad_hackathon/tools/OpenROAD-flow-scripts/flow/designs/src/gcd/gcd.v
4. Executing Verilog-2005 frontend: /home/gmrd01/iclad_hackathon/tools/OpenROAD-flow-scripts/flow/platforms/nangate45/cells_clkgate.v
5. Executing HIERARCHY pass (managing design hierarchy).
6. Executing AST frontend in derive mode using pre-parsed AST for module `\gcd'.
6.1. Analyzing design hierarchy..
6.2. Executing AST frontend in derive mode using pre-parsed AST for module `\GcdUnitDpathRTL_0x4d0fc71ead8d3d9e'.
6.3. Executing AST frontend in derive mode using pre-parsed AST for module `\GcdUnitCtrlRTL_0x4d0fc71ead8d3d9e'.
6.4. Analyzing design hierarchy..
6.5. Executing AST frontend in derive mode using pre-parsed AST for module `\RegRst_0x9f365fdf6c8998a'.
6.6. Executing AST frontend in derive mode using pre-parsed AST for module `\RegEn_0x68db79c4ec1d6e5b'.
6.7. Executing AST frontend in derive mode using pre-parsed AST for module `\Subtractor_0x422b1f52edd46a85'.
6.8. Executing AST frontend in derive mode using pre-parsed AST for module `\Mux_0xdd6473406d1a99a'.
6.9. Executing AST frontend in derive mode using pre-parsed AST for module `\Mux_0x683fa1a418b072c9'.
6.10. Executing AST frontend in derive mode using pre-parsed AST for module `\ZeroComparator_0x422b1f52edd46a85'.
6.11. Executing AST frontend in derive mode using pre-parsed AST for module `\LtComparator_0x422b1f52edd46a85'.
6.12. Analyzing design hierarchy..
6.13. Analyzing design hierarchy..
7. Executing OPT_CLEAN pass (remove unused cells and wires).
Warning: Ignoring module ZeroComparator_0x422b1f52edd46a85 because it contains processes (run 'proc' command first).
Warning: Ignoring module Mux_0x683fa1a418b072c9 because it contains processes (run 'proc' command first).
Warning: Ignoring module Mux_0xdd6473406d1a99a because it contains processes (run 'proc' command first).
Warning: Ignoring module Subtractor_0x422b1f52edd46a85 because it contains processes (run 'proc' command first).
Warning: Ignoring module RegEn_0x68db79c4ec1d6e5b because it contains processes (run 'proc' command first).
Warning: Ignoring module RegRst_0x9f365fdf6c8998a because it contains processes (run 'proc' command first).
Warning: Ignoring module GcdUnitCtrlRTL_0x4d0fc71ead8d3d9e because it contains processes (run 'proc' command first).
Warning: Ignoring module LtComparator_0x422b1f52edd46a85 because it contains processes (run 'proc' command first).
8. Executing RTLIL backend.
Warnings: 8 unique messages, 8 total
End of script. Logfile hash: aecda8b7e3, CPU: user 0.10s system 0.05s, MEM: 36.00 MB peak
```

a) RTL2GDS inside openroad flow. This will synthesize GCD with nangate45nm lib

Test EDA tools

📘 Icarus Verilog (Simulation)

```
iverilog -o hello hello.v (The file will be present in the cloned repo)
vvp hello
```

Expected output: `Hello, World!`

📊 GTKWave (Waveform Viewer)

To view generated `.vcd` files:

```
gtkwave dump.vcd
```

Using SLMs on Edge

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

Qualcomm Google CEDA sigda

General Info

- Default mode for uploading and using models will be **AnythingLLM**
- Snapdragon X-Elite Laptops will come with desktop app already installed and WSL configured to use the REST API for interacting with the models
- Default models will be provided, but you can bring your own if you have compiled into GGUF format

Model on-boarding

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

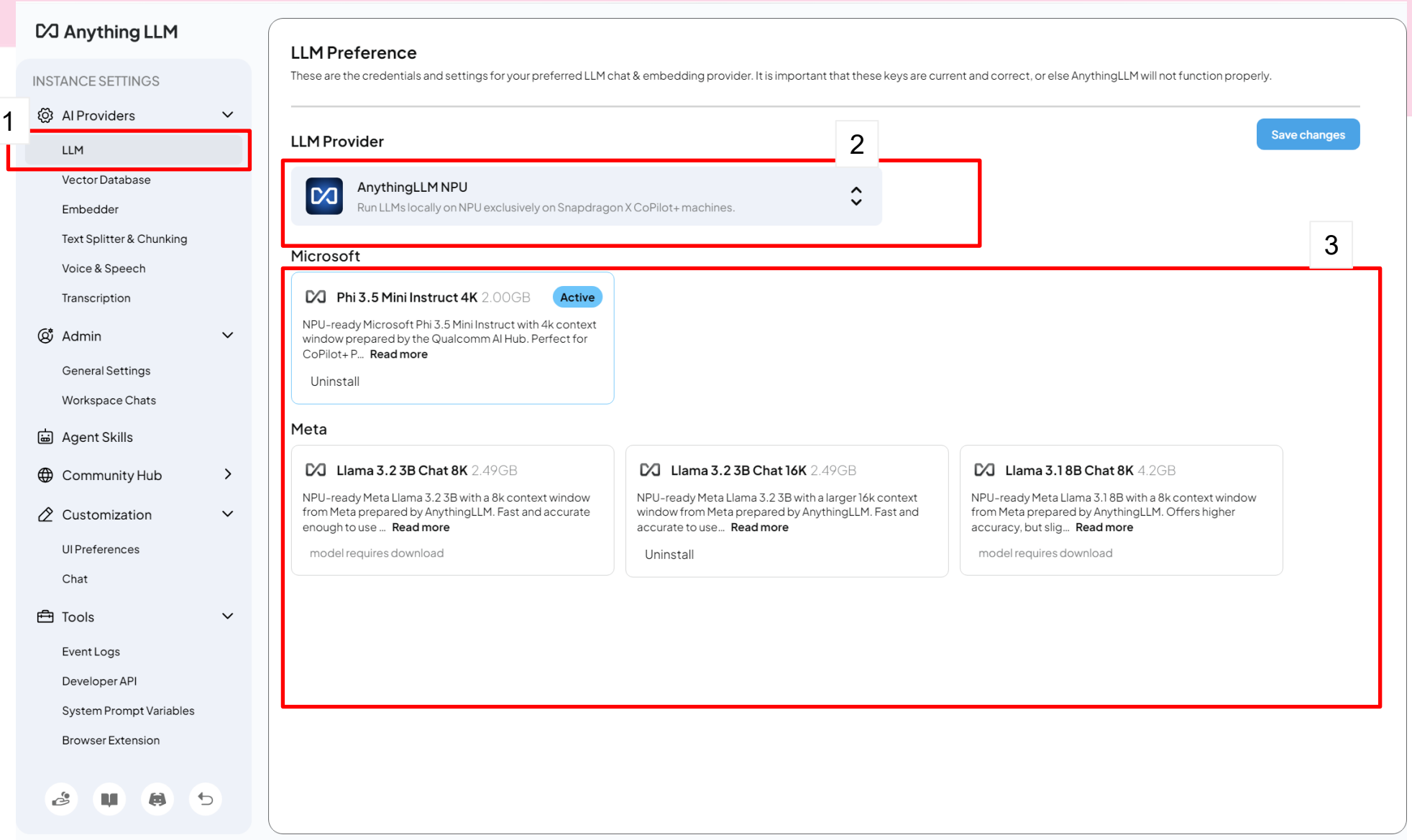
CEDA

SIGDA

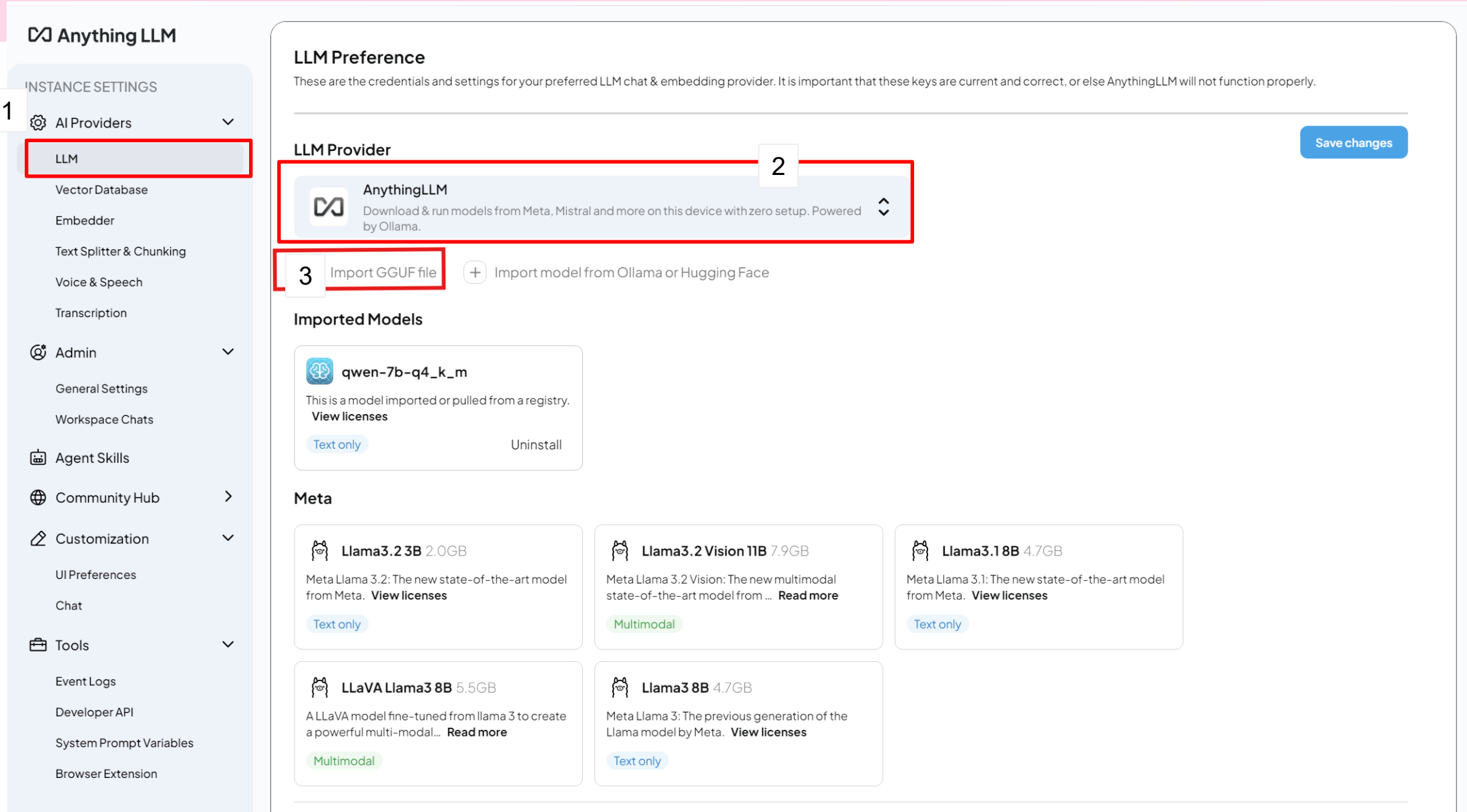
Qualcomm

Google

NPU Optimized Models



Bring your own models



Provided On-Edge Models

Model Name	Parameter Count	Context Window	NPU Optimized
Phi3.5 Mini Instruct	3.8B	4k	Yes
Llama3.2	3B	8k	Yes
Llama3.2	3B	16k	Yes
Llama3.1	8B	8k	Yes
Qwen-2.5-Instruct-Verilog-7B	7.62B	32k	No
Bring your own model	??	??	No

SLM usage

SPONSORED BY

Demo of Desktop App Features

- Basic chatbot usage
- Data connectors
- Agents

Anything LLM

+ New Workspace

GETTING STARTED 4 tasks left close

~~Create a workspace~~ ✓

Send a chat Chat

~~Embed a document~~ ✓

Set up a system prompt Set Up

Define a slash command Define

Visit Community Hub Browse

QUICK LINKS

Send Chat

Embed a Document

+ Create Workspace

EXPLORE MORE FEATURES

Custom AI Agents

Build powerful AI Agents and automations with no code.

Chat using @agent

New Build an agent flow

Slash Commands

Save time and inject prompts using custom slash commands.

Create a Slash Command

Explore on Hub

System Prompts

Modify the system prompt to customize the AI replies of a workspace.

Modify a System Prompt

New Manage prompt variables

UPDATES & ANNOUNCEMENTS

MCP Support

Import and leverage MCP tools using AnythingLLM.

AnythingLLM April 8, 2025

NVIDIA NIM Support

Unlock the power of NVIDIA NIM on Windows with RTX GPU in our latest...

AnythingLLM March 25, 2025

Community Hub Updates

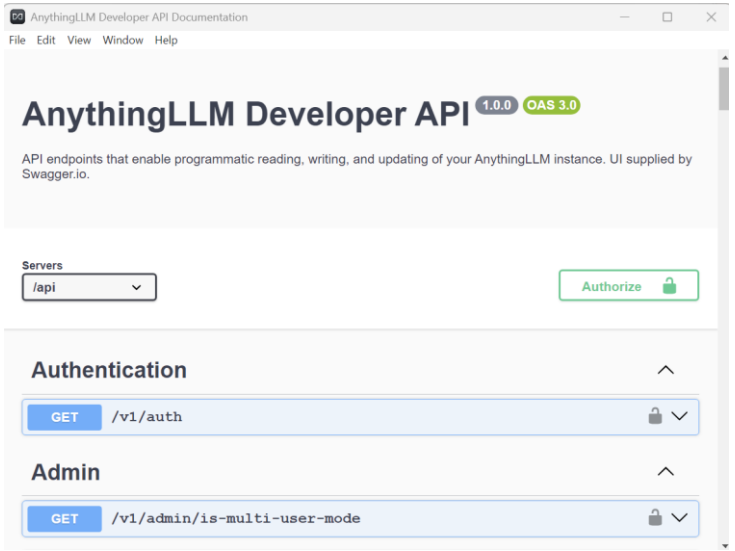
We refreshed the Community Hub with a new look and feel. Check it out!

AnythingLLM March 12, 2025

RESOURCES

Demo of Developer API

- API Access
- Basic chatbot GUI built programmatically (GitHub - thatrandomfrenchdude/simple-npu-chatbot: An NPU-accelerated chatbot running via AnythingLLM)
- Anything LLM Developer Docs is a great resource for understanding the REST API (AnythingLLM Developer API Documentation, maybe can even upload to chatbot to help you write scripts with the REST API)



v1.8.2

Anything LLM

+ New Workspace

Workspace 1

GETTING STARTED 4 tasks left close

~~Create a workspace~~ ✓

Send a chat Chat

~~Embed a document~~ ✓

Set up a system prompt Set Up

Define a slash command Define

Visit Community Hub Browse

QUICK LINKS

Send Chat

Embed a Document

+ Create Workspace

EXPLORE MORE FEATURES

Custom AI Agents

Build powerful AI Agents and automations with no code.

Chat using @agent

New Build an agent flow

Slash Commands

Save time and inject prompts using custom slash commands.

Create a Slash Command

Explore on Hub

System Prompts

Modify the system prompt to customize the AI replies of a workspace.

Modify a System Prompt

New Manage prompt variables

UPDATES & ANNOUNCEMENTS

MCP Support

Import and leverage MCP tools using AnythingLLM.

AnythingLLM April 8, 2025

NVIDIA NIM Support

Unlock the power of NVIDIA NIM on Windows with RTX GPU in our latest...

AnythingLLM March 25, 2025

Community Hub Updates

We refreshed the Community Hub with a new look and feel. Check it out!

AnythingLLM March 12, 2025

RESOURCES

11:18 PM
6/17/2025

AI

Security

Systems

EDA

Design

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation

sigda acm