PROPOSAL

TOPIC:
LAYOUT VERSES SCHEMATIC (LVS) VISUALIZATION

MENTOR:

- JESSE CIRIMELLI-LOW
- MATTHEW GUTHAUS

APPLICANT:
MAHNOOR ISMAIL

Table of content

1. INTRODUCTION:

1.1 Project Title:

Advanced OpenRAM LVS Mismatch Visualisation and Debugging Interface

1.2 Problem:

The problem we are trying to solve is the difficult and time-consuming process of identifying and debugging Layout vs. Schematic (LVS) mismatches in VLSI design when using the Magic layout editor and OpenRAM framework. Mismatches between the physical layout and the schematic can lead to incorrect functioning of the integrated circuit, which makes it crucial for designers to identify and resolve these issues effectively.

1.4 Background:

In the current state, users working with the Magic layout editor and OpenRAM framework lack a dedicated user-friendly interface for viewing and debugging LVS mismatches. The identification and debugging process often involves manual inspection of layout and schematic files, which can be complex and time-consuming. This inefficiency slows down the overall design process and may lead to overlooked errors in the final design.

1.5 Solution:

The proposed solution is to develop an advanced, interactive Python-based GUI that efficiently visualizes LVS mismatches between layout and schematic representations. The interface will parse LVS result files, highlight mismatched nets, and visually represent extra nets in the schematic graph if any. This will enable designers to easily identify and resolve LVS mismatches, leading to an enhanced user experience and faster design process.

2. Project Goals:

1. Develop an advanced, interactive GUI for efficient visualization of LVS mismatches.
2. Streamline the LVS debugging process.
3. Enhance the overall user experience for designers working with OpenRAM.

2.1 Project Objectives:

1. Provide an efficient visualization interface that simplifies the LVS debugging process.
2. Increase productivity and satisfaction for designers working with OpenRAM.
3. Encourage wider adoption of the OpenRAM framework in the ASIC design community.

2.2 Expected Deliverable:

1. A fully functional, advanced LVS mismatch visualization and debugging interface.
2. Comprehensive documentation on the usage, integration, and maintenance of the interface with existing OpenRAM tools.

3. Implementation Plan:

3.1 Project Methodology:

To achieve the project objectives, I will collaborate closely with mentors and engage with the OpenRAM community. Regular meetings, progress updates, and feedback sessions will be scheduled to ensure the project stays on track and meets the community's needs and expectations.

3.2 Project Technical Elements:

1. Conduct a thorough analysis of user requirements and existing LVS debugging processes.
2. Design the GUI layout and user experience, focusing on efficient visualization and interaction for LVS mismatches.
3. Develop a robust Python-based parser for LVS result files, capable of handling complex cases.

4. Implement advanced visualization features that highlight mismatched nets and visually represent extra nets in the schematic graph.
5. Conduct extensive testing and validation of the interface with real-world use cases.
6. Develop comprehensive documentation, including usage guides, integration instructions, and maintenance procedures.

3.3 Challenges and Proposed Solutions:

1. Parsing complex JSON output from Netgen and accurately mapping it to layout and schematic graphs.

   Solution: Perform an in-depth analysis of JSON output structures and design a robust parser capable of handling various cases while efficiently mapping the mismatches to the respective graphs.

2. Designing an intuitive, user-friendly interface that effectively visualizes LVS mismatches and facilitates efficient debugging.

   Solution: Conduct user research, gather feedback, and implement iterative design improvements to ensure the interface meets user needs and provides an efficient debugging experience.

3. Ensuring seamless integration with existing OpenRAM tools and compatibility with various design cases.

   Solution: Collaborate closely with the OpenRAM community and mentors to identify compatibility requirements, and rigorously test the interface to ensure it works well with existing tools and design scenarios.

4. Implementation Plan:

4.1 Deliverable Plan and Schedule:

May 4 - 28 (Community Bonding Period):

- Get to know mentors, read documentation, and familiarise myself with the OpenRAM framework and Magic layout editor.

May 29 (Coding officially begins):

Week 1 (May 29 - June 4):

- Analyze user requirements and existing LVS debugging processes. Design the GUI layout and user experience.

Week 2-3 (June 5 - June 18):

- Develop the JSON parser for Netgen output and implement advanced visualization features.

Week 4 (June 19 - June 25):

- Integrate the interface with existing OpenRAM tools and ensure compatibility and seamless user experience.

Week 5-6 (June 26 - July 9):

- Conduct extensive testing and validation of the interface with real-world use cases. Begin preparing for the midterm evaluation.

Week 7 (July 10 - July 16):

- Gather feedback from mentors, iterate on the design, and refine the interface based on user needs and requirements.

Week 8-10 (July 17 - August 6):

- Further refine the interface, implement additional features based on feedback, and address any remaining issues.

Week 11 (August 7 - August 13):

- Develop comprehensive documentation, including usage guides, integration instructions, and maintenance procedures.

Week 12 (August 14 - August 20):

- Perform final testing, bug fixing, and polish the interface. Prepare for the final evaluation.

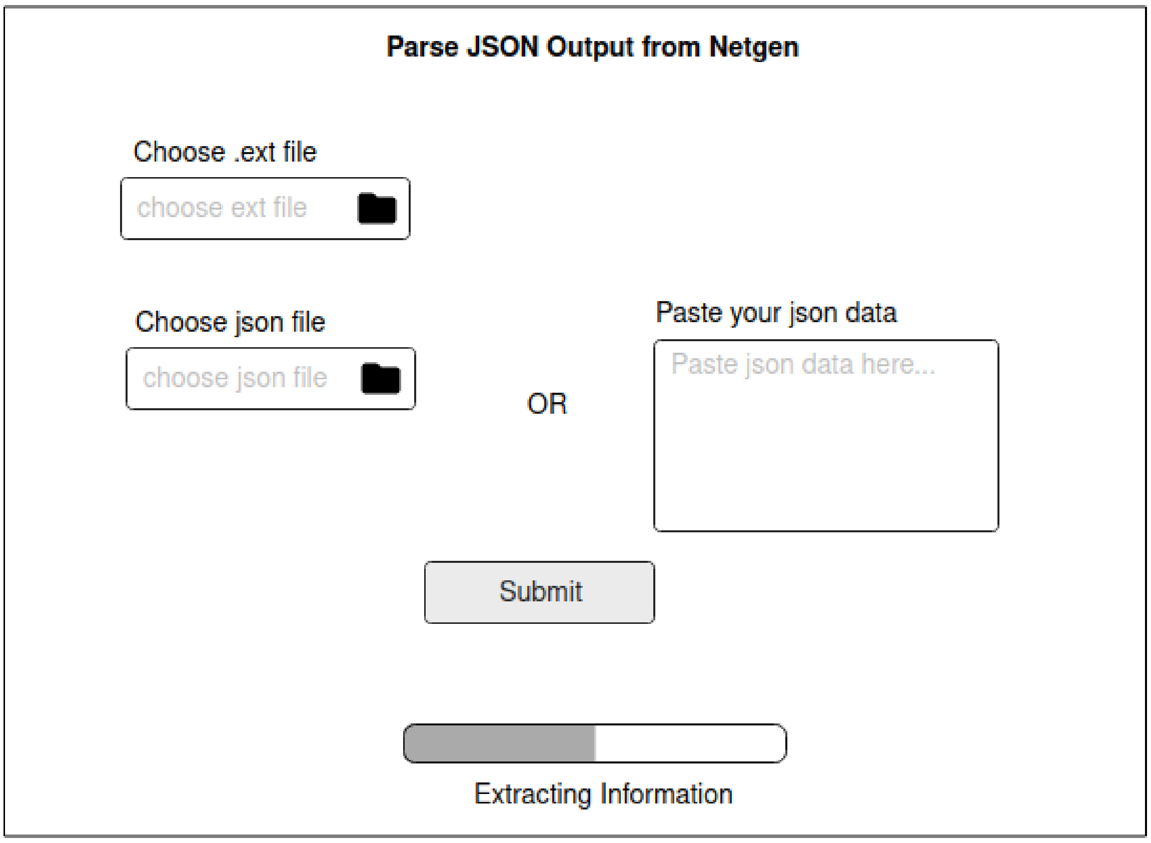
August 21 - 28 (Final week):

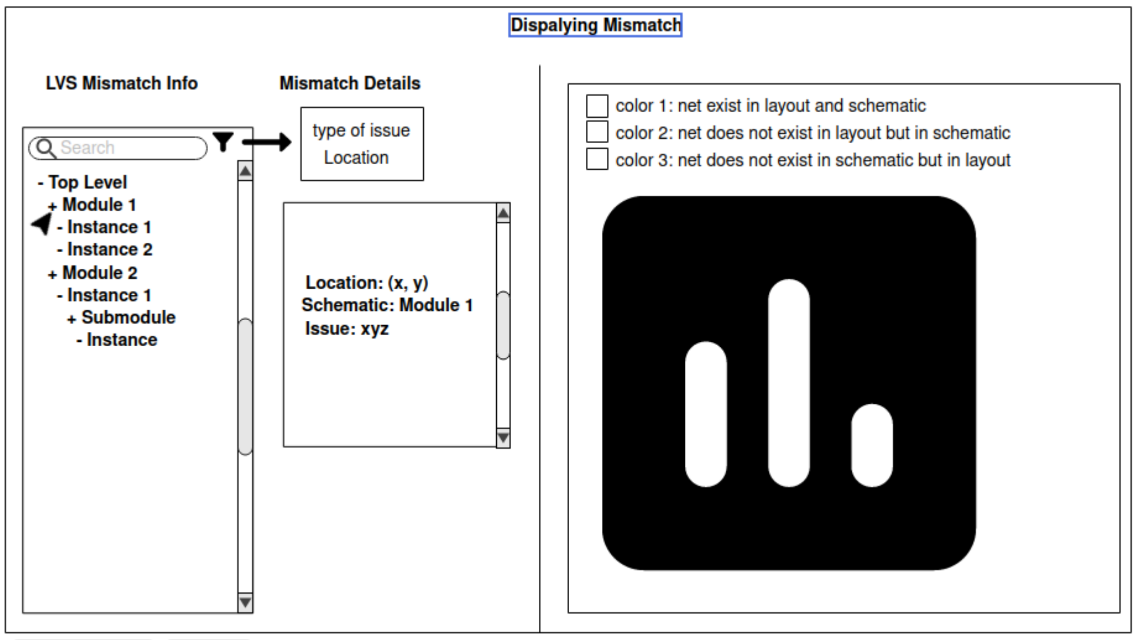
- Submit the final work product and final mentor evaluation.

5.Wireframe:

If you want to map the layout to the json file you will need to read in an additional file called a .ext file. This file is generated from the magic layout file and contains information including the coordinates of different nets. The description for that file is here. So the UI is going to need to include an input for that file as well.

According to my understanding, the rough idea presented serves to showcase the envisioned process and layout for the Python-based GUI for Efficient LVS Mismatch Visualization and Debugging.





6. BIOGRAPHICAL INFORMATION:

6.1 Education Background:

I have done Bachelors of Science in Software Engineering from Usman Institute of technology (UIT) in 2022.

6.2 Experience:

I have worked as a *Research Assistan*t in the MicroElectronics Research lab (MERL). And as an Embedded Software engineer at Intensivate.

The work i have done so far is,

1. Development of the "BURQ SUITE," an automated core verification suite based on RISC-V. It produces verification reports of regression tests performed on the core against the RISC-V Standard Golden Model Instruction Set Simulator (ISS) either Spike or Whisper.
2. Simulation of TensorFlow (TF) Lite Application on Zephyr RTOS upon Litex Vex RISC-V SoC, on top of RENODE Framework.

Currently Working On:

1. Hardware Accelerator developed in CHISEL HDL specifically for accelerating Generic Matrix Multiply (GeMM) based on Network on Chip (NoC) architecture with generic interfaces.
2. Hardware Accelerator developed in CHISEL HDL specifically for accelerating Secure Hash Algorithm (SHA-3) encryption and decryption with generic interfaces.
3. In Extended version of "BURQ SUITE", Integrated RISC-V DV and also looking into cocotb.
4. ChipShop, a cloud based GUI for accelerating SoC Design. It is based on Chipyard which is a CHISEL based SoC generator developed by UC Berkeley. ChipShop provides a cloud based web interface GUI with which users can generate SoCs with ease and prototype it on an FPGA without diving into code. Also, user's can share their designs with each other and collaborate easily with real time collaboration features. (This is being presented in the First Firesim/Chipyard workshop happening in ASPLOS'23)

6.3 Technical Strength:

- Python
- Data Structure and algorithm
- Artificial Intelligence
- TCL
- C++
- Chisel
- Scala
- Script automation
- Web development
- Html & CSS
- Javascript
- UI Designing
- Software prototyping
- Software Quality Assurance
- RISC-V ISA
- Technical report writing

6.4 Contact:

5.4.1 Full Name:

Mahnoor Ismail

5.4.2 Email:

mahnoorismail011@gmail.com

5.4.3 Github:

https://github.com/Mahnoor-ismail01

5.4.4 LinkedIn:

http://linkedin.com/in/mahnoor-ismail-b8017a179