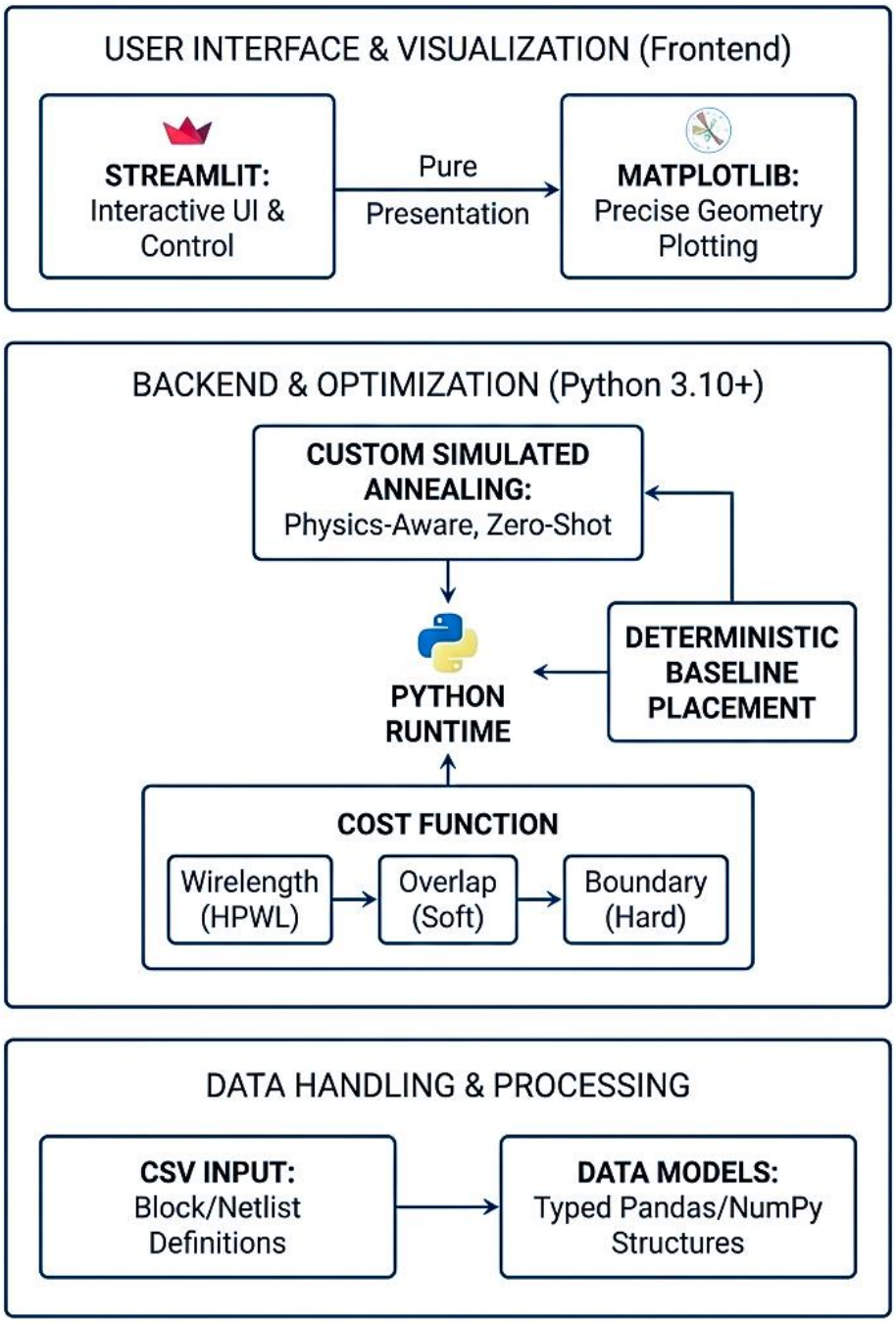
LAIOpt TECH STACK: Layout AI Optimizer

Explainable, Deterministic VLSI Macro Floorplanning



KEY DECISIONS: Explainable AI, No Training, Hybrid Deterministic

STRENGTHS: Physically Meaningful, Clean Separation, Future-Ready

LAIOpt – Technology Stack Documentation

Project Identity

Project Name: LAIOpt (Layout AI Optimizer)
Domain: Electronic Design Automation (EDA) / VLSI Physical Design
Problem Area: Early-Stage Macro Floorplanning
Core Function:
Deterministic and explainable macro placement optimization using a physics-aware Simulated Annealing framework.

1. Core Language & Runtime

Language: Python 3.10+

Why Python:
- Standard for academic EDA research
- Ideal for algorithmic prototyping
- Strong readability and modularity
- Enables rapid experimentation and correctness verification

2. User Interface & Visualization

Frontend Framework: Streamlit

Why Streamlit:
- Rapid UI development
- No frontend framework overhead
- Ideal for hackathons
- Focus remains on optimization logic

Role in LAIOpt:
- Edit block and netlist data
- Trigger optimization
- Compare baseline vs optimized layouts
- Display quantitative metrics

Visualization Engine: Matplotlib

Why Matplotlib:
- Coordinate-accurate plotting
- Deterministic rendering
- Precise geometry control
- Industry-acceptable visualization

Used to render:
• Die boundary
• Macro blocks (true dimensions)
• Heat-based coloring
• Baseline vs optimized layouts

3. Backend Optimization Engine

Algorithm: Custom Simulated Annealing

Why Simulated Annealing:
• Handles NP-Hard problems
• Escapes local minima
• No training data required
• Works immediately on new designs

Baseline Placement:
• Deterministic
• Legal (no overlaps, no boundary violations)
• Uses true block dimensions
• Mirrors professional EDA flows

4. Cost Function Design

Implemented Cost Terms:
• Wirelength (HPWL – Manhattan distance)
• Soft overlap penalty (area-based)
• Hard boundary violation penalty

Design Philosophy:
• Physically meaningful
• No fake optimization claims
• Honest abstraction level

Power & Heat:
• Modeled as metadata
• Used for visualization
• Reserved for future extensions

5. Data Handling

Input Format: CSV

Why CSV:
• Transparent and simple
• Editable by non-programmers
• Common in EDA workflows

Core Data Models:
• Block
• Net
• Die

Each model enforces physical validity.

6. System Architecture

Layered Architecture:
User Input
→ Typed Models
→ Baseline Placement
→ Simulated Annealing
→ Cost Evaluation
→ Final Placement
→ Visualization

Principles:
• UI and backend fully decoupled
• No hidden state
• Fully testable backend

7. Folder-Level Technology Mapping

frontend/ – Streamlit UI and visualization
backend/core/ – Models, baseline, cost, SA engine
backend/adapters/ – CSV loaders and serializers
data/ – Input and output data
tests/ – Unit tests

8. Dependencies

Runtime Dependencies:
• streamlit
• pandas
• numpy
• matplotlib

Purpose:
Streamlit – UI
Pandas – Data handling
NumPy – Numerical computation
Matplotlib – Visualization

9. Key Engineering Decisions

Why Not Reinforcement Learning:

- Requires large datasets
- Long training time
- Poor explainability

Chosen Approach:

- Deterministic baseline + stochastic SA
- Faster convergence
- Stable and defensible results

10. Project Strengths

- No fake AI claims
- Physically meaningful outputs
- Clean modular architecture
- Future-ready design

11. Future Extensions

- Thermal-aware optimization
- Power delivery constraints
- Hierarchical floorplanning
- DEF/GDSII export

12. Final Summary

LAIOpt is an explainable, deterministic macro floorplanner demonstrating how physical constraints and connectivity guide early-stage VLSI layout optimization.