CHIP FORGE

Decentralized Semiconductor Design Ecosystem

Transforming Hardware Innovation Through AI-Powered Collaborative Intelligence

A Bittensor Subnet Technical Architecture

Version 1.0

October 2025

ChipForge: A Project of Tatsu

Table of Contents

Chipforge Platform **4**

Platform Components 5

System Architecture 5

REST API Endpoints 5

Database Section 6

File Storage System 6

S3 Integration 6

System Workflows 7

Design and Score Submission Flow 7

Batch Distribution Algorithm 9

Error Recovery 9

EDA Server Integration 9

Evaluation Pipeline 10

Performance Characteristics 10

Validator Integration 10

Scoring Methodology 11

Score Normalization 11

Dynamic Scoring Formula 12

Detailed Metric Evaluation 13

Security Features 15

Performance & Scalability 16

Monitoring & Observability 16

List of Figures

Figure 1 ChipForge Platform 4
Figure 2 Design Submission Flow 7
Figure 3 Score Submission Flow 8
Figure 4 Evaluation Pipeline 9
Figure 5 Detailed Metric Evaluation 13

List of Tables

Table 1 Score Normalization 11

Chipforge Platform

The Chipforge Platform coordinates distributed hardware design evaluation across a decentralized network. Miners submit hardware designs in response to challenges, while validators independently assess these submissions using industry-standard EDA tools.

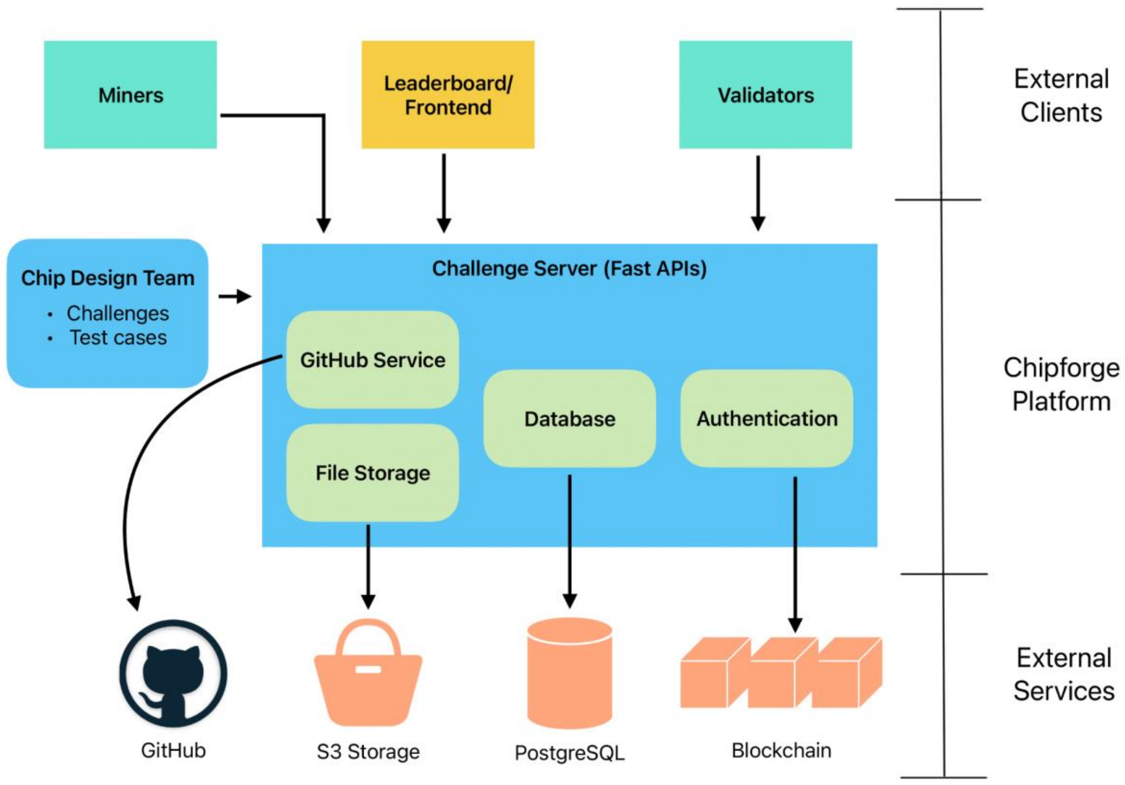

Figure 1 ChipForge Platform

Platform Components

Challenge Server - The central orchestration system managing challenge lifecycles, submission tracking, batch distribution to validators, score aggregation, and S3 storage operations.

Chip Design Team - Creates hardware challenges representing state-of-the-art circuit design problems with comprehensive test suites containing millions of instructions.

EDA Server - A FastAPI application built around open-source EDA tools (OpenLane, Verilator) that analyzes design functionality, area, timing, and power consumption.

System Architecture

The platform uses a microservice architecture with separate REST interfaces for miners and validators, designed for fault tolerance and continuous operation.

REST API Endpoints

Miner Endpoints

Active Challenge (/active) - Get currently available challenges

Challenge Info (/info) - Retrieve challenge details, time remaining, and file locations

Generate submissions id (/generate-submission-id) - for solution submission

Submit Solution (/submit) - Upload hardware design with cryptographic authentication

Check Submission (/status) - Track evaluation progress and retrieve scores

Validator Endpoints

Test Cases (/testcases/download) - Download validator-specific test suites

Current Batch (/batch/current) - Retrieve current submission batch for evaluation

Submit Score (/submit_score) - Submit evaluation results with comprehensive metrics

Database Section

The system uses PostgreSQL with five main tables:

challenges - Challenge metadata, lifecycle state, cryptographic nonces, validator requirements, and submission limits

miner_submissions - Submission metadata, file references, multi-bucket storage (miner/ validator/ archive), validation counts, and status tracking

validation_records - Individual validator evaluations, detailed score breakdown, evaluation notes, and validator-specific file paths

submission_nonces - Cryptographic nonce management with expiration tracking and rate limiting

validator_api_keys - API key lifecycle, usage tracking, audit trails, and activation/revocation history

File Storage System

S3 Integration

The S3 manager handles secure cloud storage for hardware designs:

- **Hierarchical Organization**: Designs stored with challenge ID, miner hotkey, submission attempt, and score
- **Access Control**: Time-limited pre-signed URLs for secure downloads
- **Audit Trail**: Complete logging of storage operations

System Workflows

Design and Score Submission Flow

1. Design team publishes challenge
2. Miners discover active challenges
3. Miners submit Verilog designs with signatures
4. Challenge Server validates and stores submissions
5. Submissions grouped into batches (up to 8)
6. Batches exposed to validators
7. Validators submit evaluation scores
8. Multi-validator scores aggregated
9. Leaderboard updated

Design submission flow is

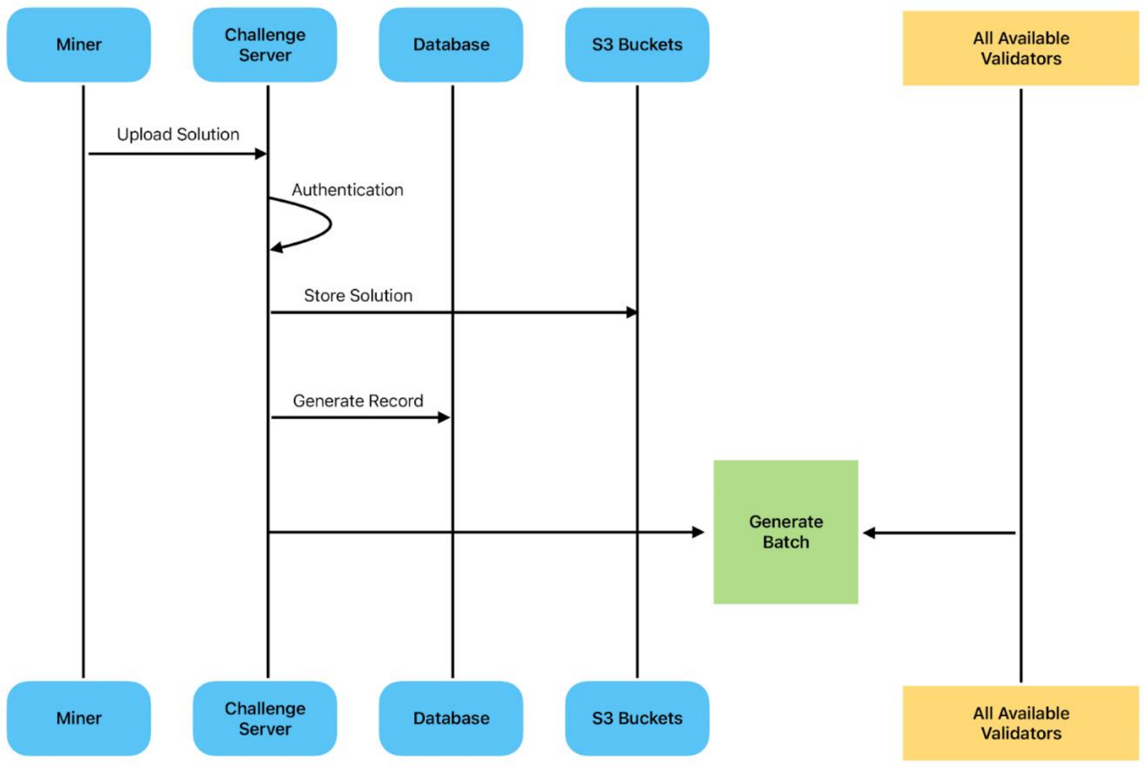

Figure 2 Design Submission Flow

Score submission flow is

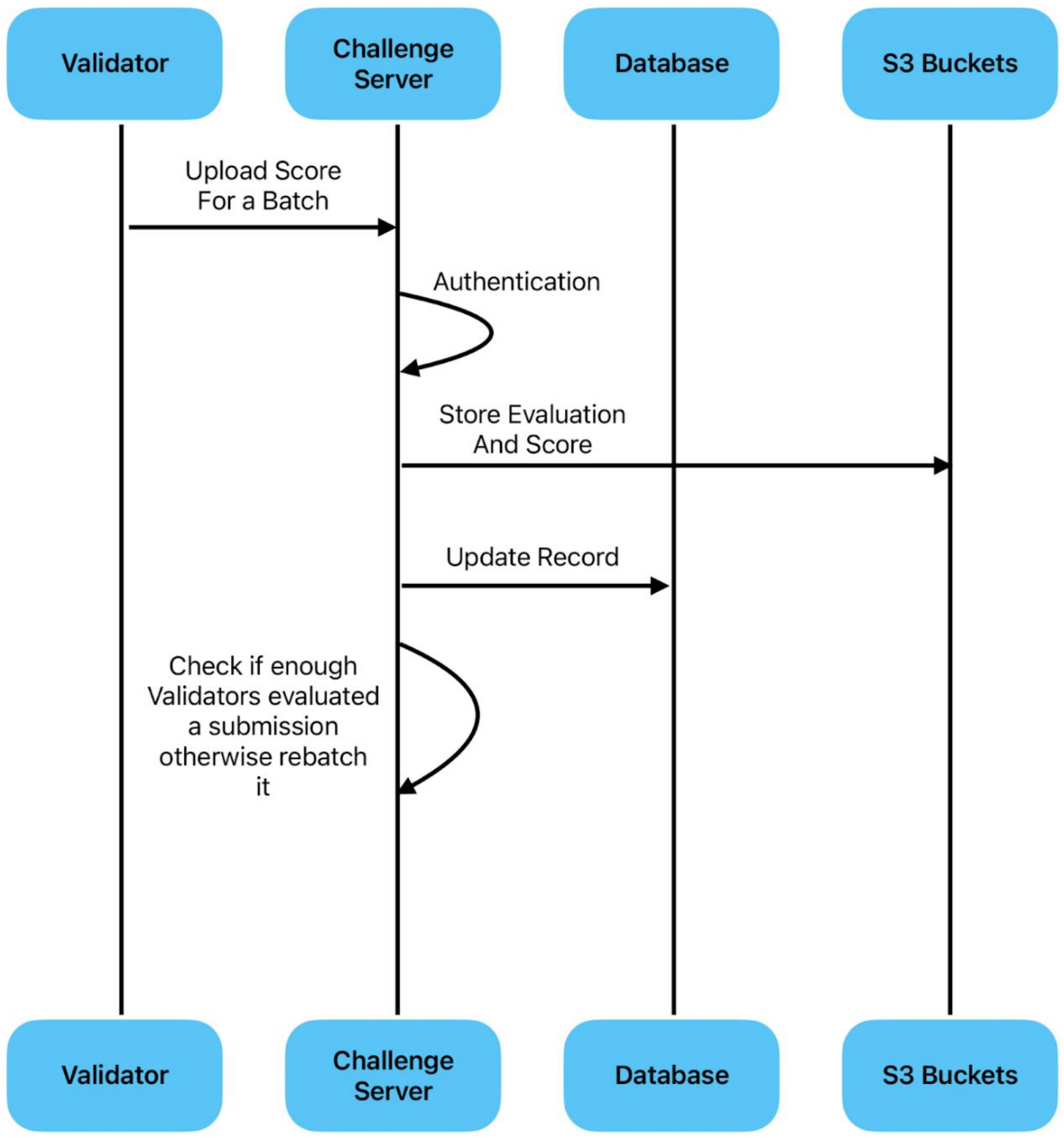

Figure 3 Score Submission Flow

Batch Distribution Algorithm

- **Batch Creation:** Groups of up to 8 submissions assembled
- **Exposure Duration:** Batches available for sufficient evaluation time
- **Multi-Validator Support:** Same submission evaluated independently by multiple validators
- **Result Aggregation:** Individual validator scores maintained separately in leaderboard

Error Recovery

- **Connection Failures:** Automatic batch redistribution and status reset
- **Evaluation Errors:** Graceful degradation with retry mechanisms

EDA Server Integration

The EDA Server provides the computational backbone for design validation through a two-stage evaluation pipeline.

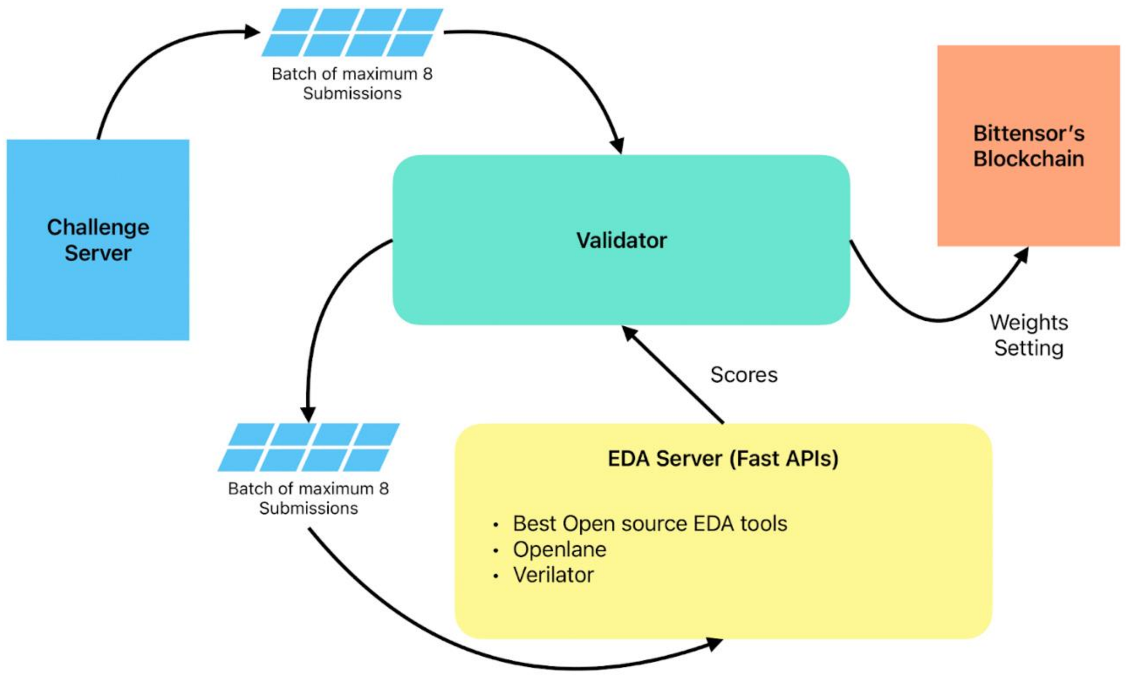

Figure 4 Evaluation Pipeline

Evaluation Pipeline

Stage 1 - Functional Verification

- Simulates Verilog design against test vectors
- Uses Verilator for verification
- Validates functional correctness
- Generates functionality score

Stage 2 - Physical Analysis

- Synthesizes design using Yosys/OpenLane
- Calculates silicon area utilization
- Determines maximum operating frequency
- Estimates power consumption
- Generates area, delay, and power scores

Score Aggregation - Combines metrics into overall score using weighted formula

Performance Characteristics

- **Parallel Processing**: 8 concurrent evaluation requests
- **Resource Requirements**: 8 vCPU, 16GB RAM
- **Throughput**: Minutes per complete evaluation

Validator Integration

The validator interfaces with both EDA Server and Challenge Server:

1. Receives evaluation metrics from EDA Server
2. Submits scores to Challenge Server
3. Identifies top-performing submissions
4. Updates blockchain weights based on performance

Scoring Methodology

The Chipforge Platform evaluates hardware designs using a comprehensive, weighted scoring function. The scoring methodology is **dynamic**, with both the weight distribution for the primary metrics and a minimum functionality threshold defined by the Chip Design Team in a challenge-specific weights.json configuration file. This allows the platform to emphasize specific design qualities, such as performance or silicon efficiency, based on the requirements of the current challenge.

Score Normalization

All individual metric scores and the final **Overall Score (**O_c**)** are normalized to the **0-1 range**, where a score of 1 represents 100% of the ideal metric performance.

Metric	Score Abbreviation	Description
Functionality	F_c	Correct execution of the instruction set.
Performance	P_c	Instructions executed per second (throughput).
Area	$Area$	Efficiency of silicon utilization (smaller is better).
Power	N/A (TBD)	Power consumption in milliwatts.

Table 1 Score Normalization

Dynamic Scoring Formula

The individual weights (W_{Func}, W_{Delay}, W_{Area}) and the minimum Functionality Threshold ($Threshold_{Func}$) are loaded from the weights.json file.

- **Constraint:** The submission must achieve a minimum Functionality Score (F_c) greater than or equal to the dynamic $Threshold_{Func}$ to receive a non-zero overall score.

$$\text{IF } F_c \geq Threshold_{Func}:$$

$$O_c = \frac{(W_{Func} \cdot F_c) + (W_{Area} \cdot Area) + (W_{Perf} \cdot P_c)}{Total\_W}$$

$$\text{IF } F_c < Threshold_{Func}:$$

$$O_c = 0$$

Note: The power metric is not currently evaluated, as it will be integrated for future challenges, such as the Neural Processing Unit (NPU) development, where power is a critical concern.

Detailed Metric Evaluation

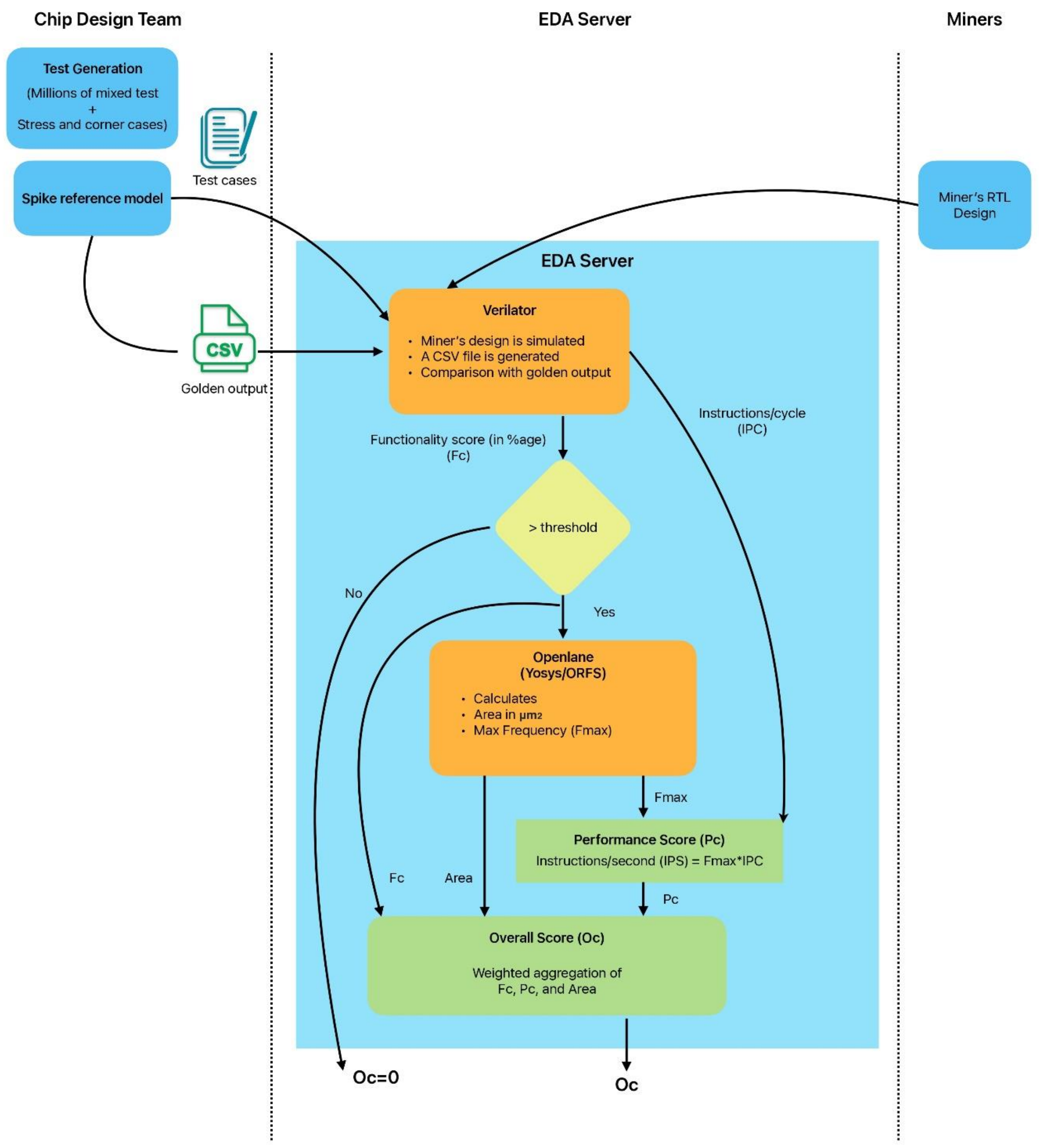

Figure 5 Detailed Metric Evaluation

1. **Functionality Score (F_c)**

   This metric measures the design's functional correctness.

   - **Test Generation & Golden Reference:** The Chip Design Team creates comprehensive test suites and runs them on the official **RISC-V ISA Simulator (SPIKE)** to generate the "golden output."
   - **Design-Under-Test (DUT) Execution:** The same test cases are executed on the miner's design using the EDA Server's **Verilator** tool for simulation.
   - **Scoring:** The F_c score is determined by comparing the DUT's results against SPIKE's golden outputs.

$$F_c = \frac{N_{correct}}{N_{total}}$$

Where $N_{correct}$ is the number of instructions that matched the golden result, and N_{total} is the total number of executed instructions. F_c is a value between 0 and 1.

2. **Performance Score (P_c)**

   Performance is quantified as **Instructions Per Second (IPS)**, representing the processing throughput.

   - **IPC & Max Clock Speed ($Fmax$):** The processor's **Instructions Per Cycle (IPC)** is measured using performance tests. The processor's maximum clock speed, referred to as $Fmax$ (Max Frequency), is determined through timing analysis using **OpenLane**.
   - Performance (IPS): The throughput is calculated as:

     $$IPS = IPC \times Fmax$$

   - **Normalization:** This raw IPS value is then normalized against a defined benchmark IPS to yield the 0-1 P_c score (a higher IPS results in a score closer to 1).

3. **Area Score ($Area$)**

   The $Area$ score measures the silicon efficiency of the design.

   - **Synthesis & Measurement:** The design is synthesized using **OpenLane (Yosys)** targeting the sky130nm PDK, and the total silicon area is extracted.
   - **Normalization:** The $Area$ score is **inversely proportional** to the measured area (a smaller area yields a score closer to 1). This score is normalized against a target area defined by the challenge.

4. **Power Consumption Score (TBD)**

   This metric is currently under development. Once integrated, the $Score_{Power}$ will be inversely proportional to the measured power consumption, normalized against a target power consumption.

Security Features

Authentication System

- **Cryptographic Signatures:** Ed25519 signature verification (Bittensor compatible)
- **Blockchain Integration:** Hotkey registration validation
- **Role Verification:** Distinguishes miners from validators
- **Rate Limiting:** Throttling on sensitive endpoints
- **Replay Prevention:** Nonce-based request validation
- **Unique Submission IDs:** Prevents duplicate processing

Access Control

- **API Key Management:** Secure validator authentication
- **File Access:** Time-limited S3 URLs
- **Admin Functions:** Password-protected operations

Performance & Scalability

Connection Pool Management

- **Database Connections:** Async connection pooling
- **Resource Monitoring:** Automatic cleanup and memory management

Caching Strategy

- **Query Optimization:** Strategic indexing and query planning
- **Real-time Updates:** Incremental data refresh

Scalability

- **Horizontal Scaling:** Stateless API design
- **Batch Processing:** Asynchronous task handling
- **Storage Partitioning:** S3 bucket segmentation

Monitoring & Observability

- **Structured Logging:** Comprehensive audit trails with process tracking
- **Error Tracking:** Detailed exception reporting