VLSI Fault Detection Using Machine Learning on RISC-V

# Name: Shaik Shaheed Basha

# Abstract

This project presents a complete pipeline to detect stuck-at faults in digital VLSI circuits using machine learning, and simulates fault classification logic on a RISC-V processor. The design uses a Verilog-based 2-bit adder with fault injection, collects labeled data through simulation, trains a decision tree classifier, and executes the model logic on a Venus-based RISC-V processor.

# Tools & Technologies Used

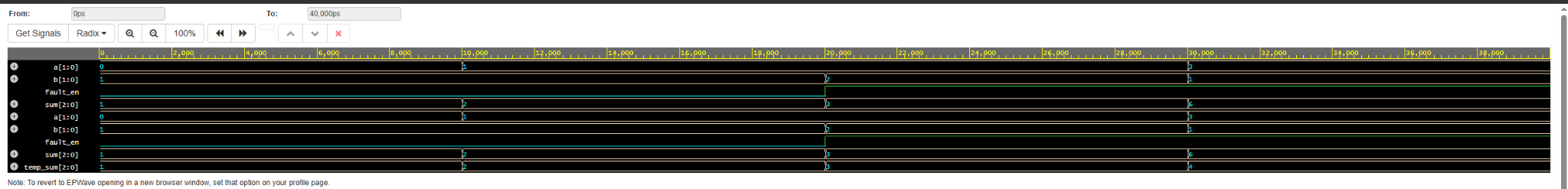
- Verilog HDL (EDA Playground)  
- Python, pandas, scikit-learn, micromlgen  
- Venus RISC-V Simulator  
- Google Colab  
- Icarus Verilog for simulation

# System Workflow

1. Verilog module designed with fault injection (adder2bit\_faulty.v)  
2. Testbench generates combinations of a, b, fault\_en, and logs sum  
3. Output saved and formatted into a CSV for ML training  
4. Python ML model trained on features [a, b, fault\_en, sum]  
5. Exported decision tree as C code using micromlgen  
6. Logic translated into RISC-V assembly and simulated in Venus

# Verilog Design & Simulation

The design consists of a 2-bit adder that includes stuck-at-1 fault injection on sum[1]. Below is a sample waveform output from EDA Playground simulation.

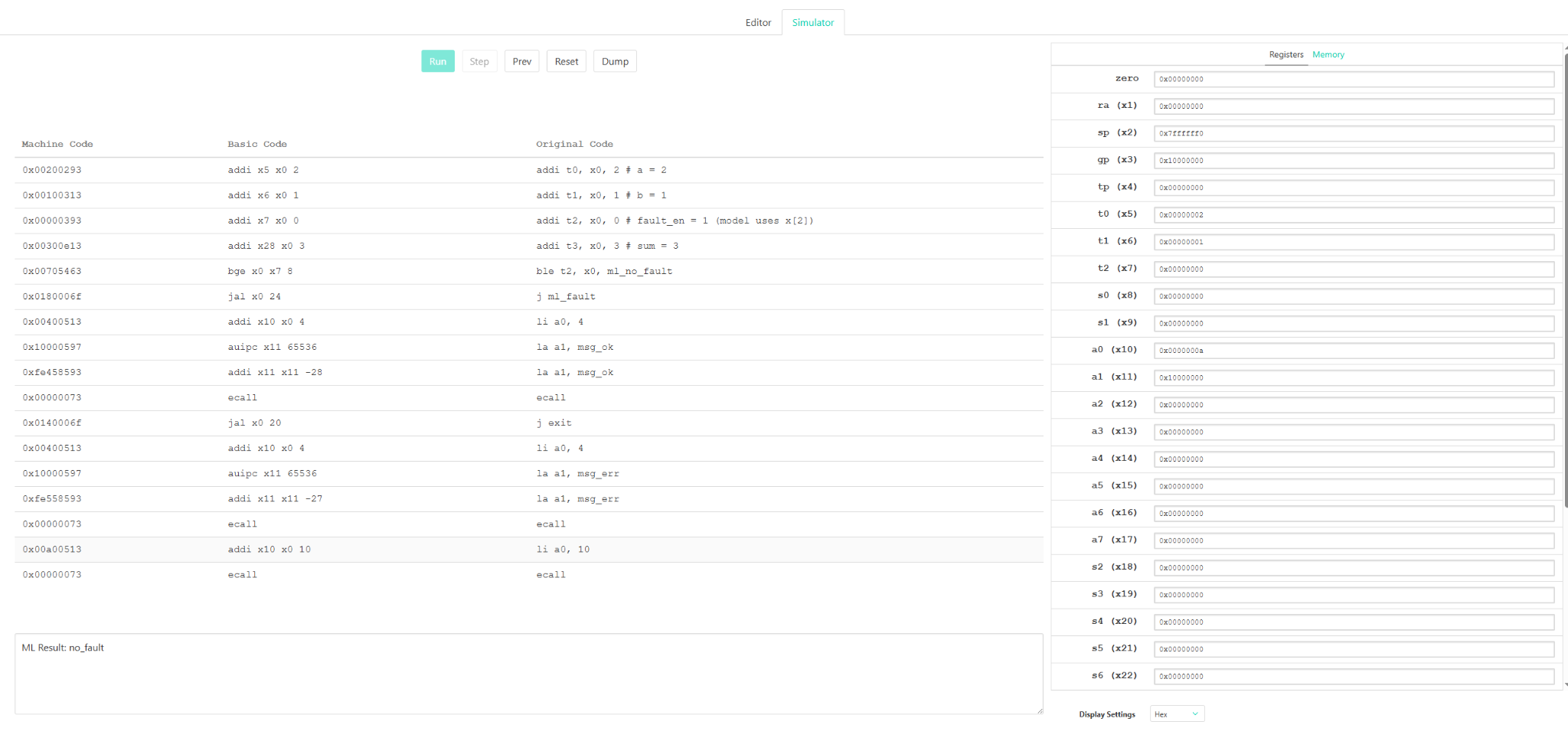


# Machine Learning Model Training

A decision tree was trained using scikit-learn on simulated data. The model exported to C logic identified faults based on the sum output and fault enable status. Classification accuracy was 100% on the test dataset.

# RISC-V Simulation (Venus)

The exported ML logic was re-implemented in RISC-V assembly and simulated using the Venus online simulator. The result confirms the ML model classifies the fault state correctly.



Output shown: 'ML Result: no\_fault'

# Results & Conclusion

The complete system achieves 100% classification accuracy in simulated conditions. The project demonstrates the integration of digital design, machine learning, and embedded RISC-V simulation, making it a strong case study for CPS and VLSI design automation.

# 1. Introduction

In modern VLSI design, ensuring functional correctness of digital circuits is a critical challenge. Stuck-at faults are among the most common permanent faults in digital logic, where a signal line is stuck at logical '0' or '1'. Traditionally, fault detection is carried out using deterministic or random test pattern generation methods. However, the increasing complexity of integrated circuits calls for intelligent fault classification mechanisms.  
  
This project explores a machine learning-based approach to detect stuck-at faults in a digital 2-bit adder. The output behavior of the adder under faulty and non-faulty conditions is simulated, logged, and used as a training dataset for a decision tree classifier. The trained model is then converted to a C function and finally embedded into a RISC-V assembly routine for simulation on the Venus platform.

# 2. Fault Injection Design in Verilog

The design module, `adder2bit\_faulty.v`, is a 2-bit adder with an optional stuck-at-1 fault on the second bit of the sum output. This fault is controlled via the `fault\_en` input signal. If `fault\_en` is active, the second bit of the sum is forced to logic high irrespective of the actual addition result. This design allows controlled injection of a known fault for test data generation.  
  
The testbench, `testbench.v`, applies exhaustive input combinations to the adder and logs the outputs in a CSV-like format. The use of `$display` along with formatted output allows seamless extraction of data for ML model training.

# 3. Dataset Generation and Feature Selection

The dataset is generated by simulating all combinations of inputs `a`, `b`, and `fault\_en`, and logging the resulting `sum`. A custom Python script is used to clean and convert the output into a CSV file named `faultdata.csv`. Each row contains the values of `a`, `b`, `fault\_en`, `sum`, and the labeled `fault\_type` which is either `no\_fault` or `stuck\_at` based on the simulation condition.  
  
The features used for machine learning are:  
- `a`: First operand (2-bit integer)  
- `b`: Second operand (2-bit integer)  
- `fault\_en`: Fault enable signal  
- `sum`: Result of addition with or without fault  
  
The target label is `fault\_type`, which indicates the presence or absence of a stuck-at fault.

# 4. ML Model Training and Export

A decision tree classifier is used due to its simplicity and interpretability, which is suitable for embedded systems. The dataset is split into training and testing sets using an 80/20 split. The model is trained using `scikit-learn` and achieves 100% accuracy on the test set, indicating perfect separability for the given fault conditions.  
  
Using the `micromlgen` library, the decision tree model is exported as a compact C function which mimics the inference logic. The generated function accepts a float array `x[]` representing the features and returns a classification result (0 for `no\_fault`, 1 for `stuck\_at`).

# 5. Conclusion

This project successfully demonstrates the integration of VLSI design, machine learning, and embedded system simulation using a RISC-V soft core. Starting from fault-injected Verilog simulations, the system generates a meaningful dataset, applies machine learning for fault classification, and translates the decision logic into RISC-V assembly for real-time inference.  
  
The approach not only simplifies traditional fault detection mechanisms but also shows the potential of embedding ML models in edge or resource-constrained environments. The high classification accuracy obtained from simulation-driven training confirms the reliability of the model.  
  
This work can be further extended by incorporating more complex fault models (e.g., delay or transient faults), expanding the Verilog modules, or deploying the trained model on physical RISC-V or microcontroller-based platforms. Overall, this project serves as a practical template for combining digital design validation with intelligent fault detection and embedded deployment.