2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

AI Driven Advanced Debugging in SoC Design Verification

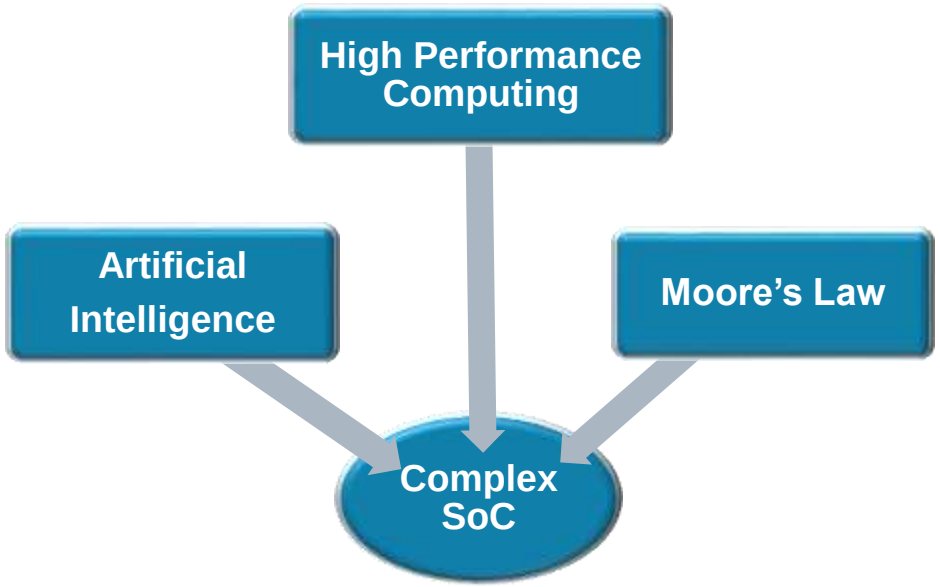
Shreya Jayateerth Joshi, Poonam M Shettar, Sanjoy Saha,
Alok Kumar, Sunil Shrirangrao Kashide, Garima Srivastava

Samsung Semiconductor India Research

Agenda

- INTRODUCTION : DESIGN VERIFICATION CHALLENGES
- STATE OF THE ART AI-ML TECHNIQUES
- PROPOSED SOLUTION
- PRE-SIMULATION FRAMEWORK
- POST-SIMULATION FRAMEWORK
- TRANSFORMER ENCODER ARCHITECTURE
- DEEP LEARNING MODELS
- LLM LOG ANALYSIS AND INTERACTIVE DEBUGGER
- INTEGRATION CHALLENGES
- RESULTS AND CONCLUSION

Introduction : Design Verification Challenges

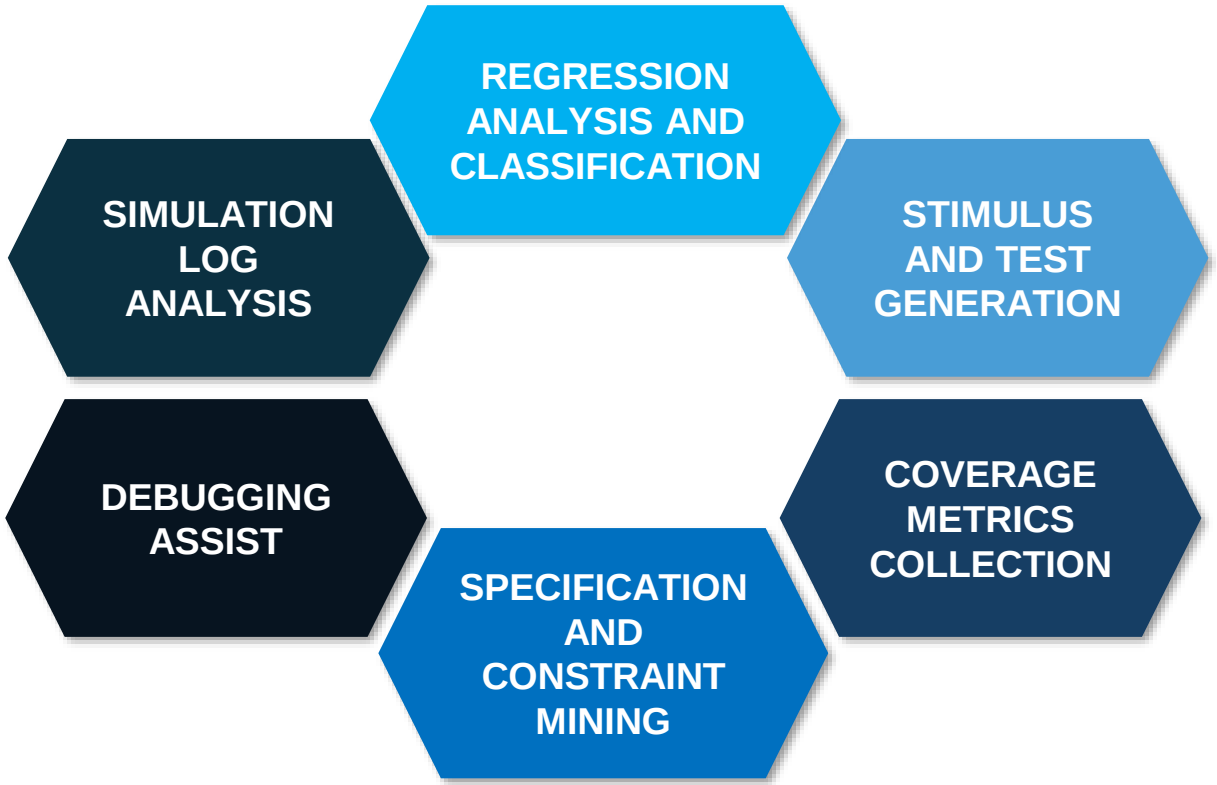


Chip size and complexity ↑ → No. of signals to be analyzed ↑ → Manual Debugging time ↑

Simulation runtime ↑ → Simulation log and dump size ↑

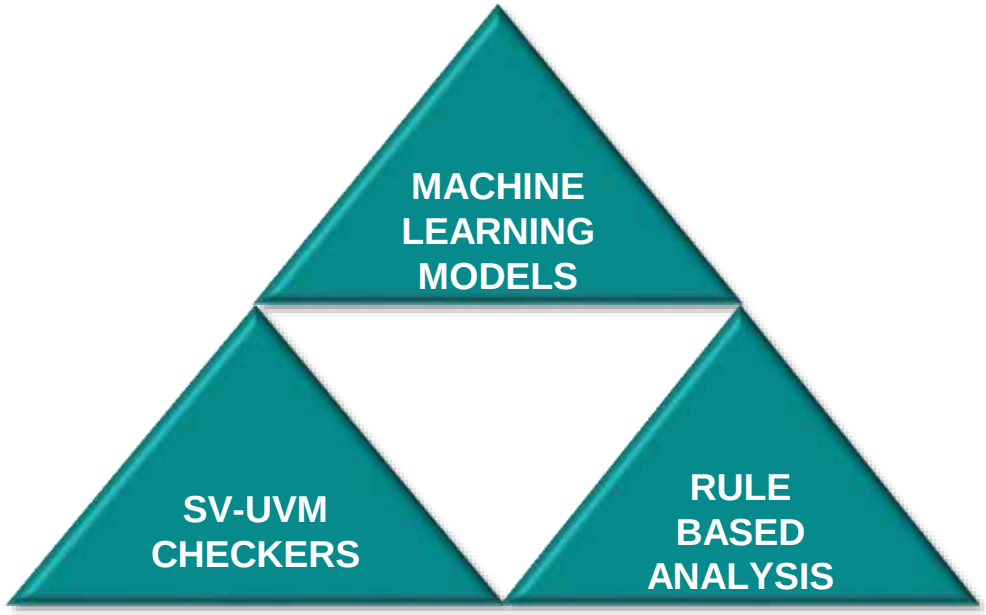
Number of Iterations ↑ → DV Effort ↑ → Chip life cycle ↑

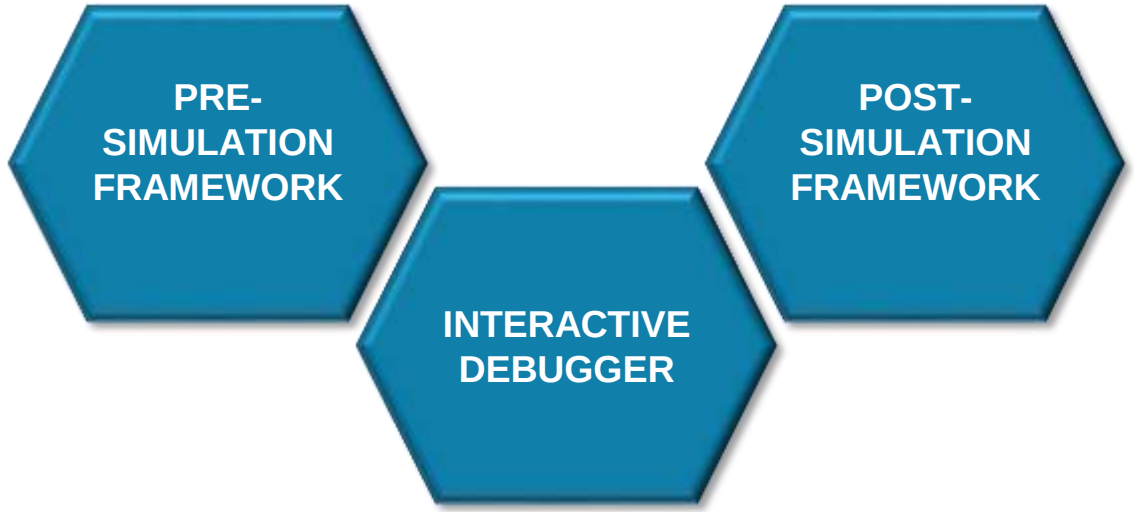
State of the art AI-ML techniques



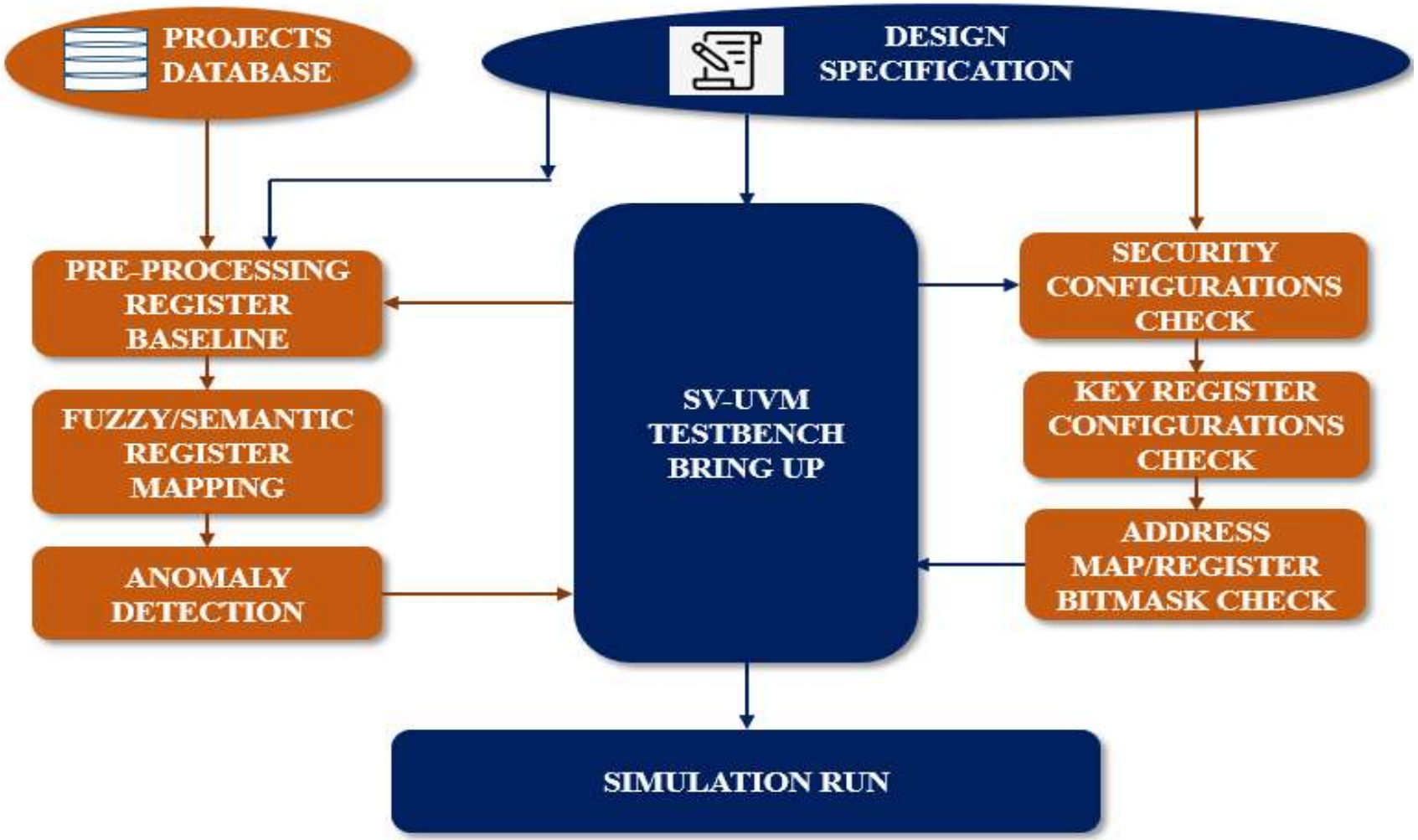
- Tested on open source designs , not complex industry level design
- Log analysis and not dump analysis
- Constrained Random Verification TB
- Debug assistance needs intervention by verification engineer

Proposed Solution

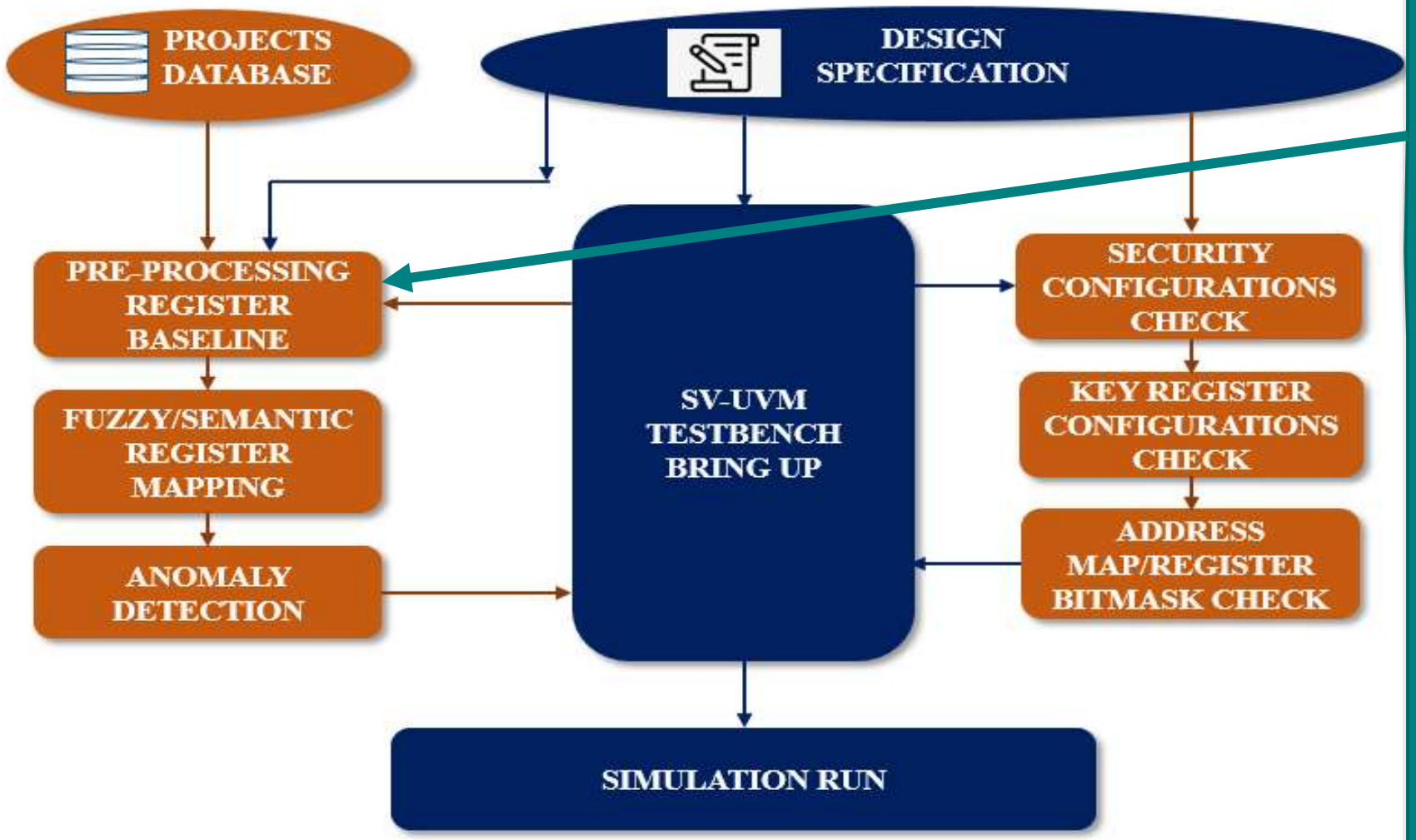




Pre-simulation Framework



Pre-simulation Framework

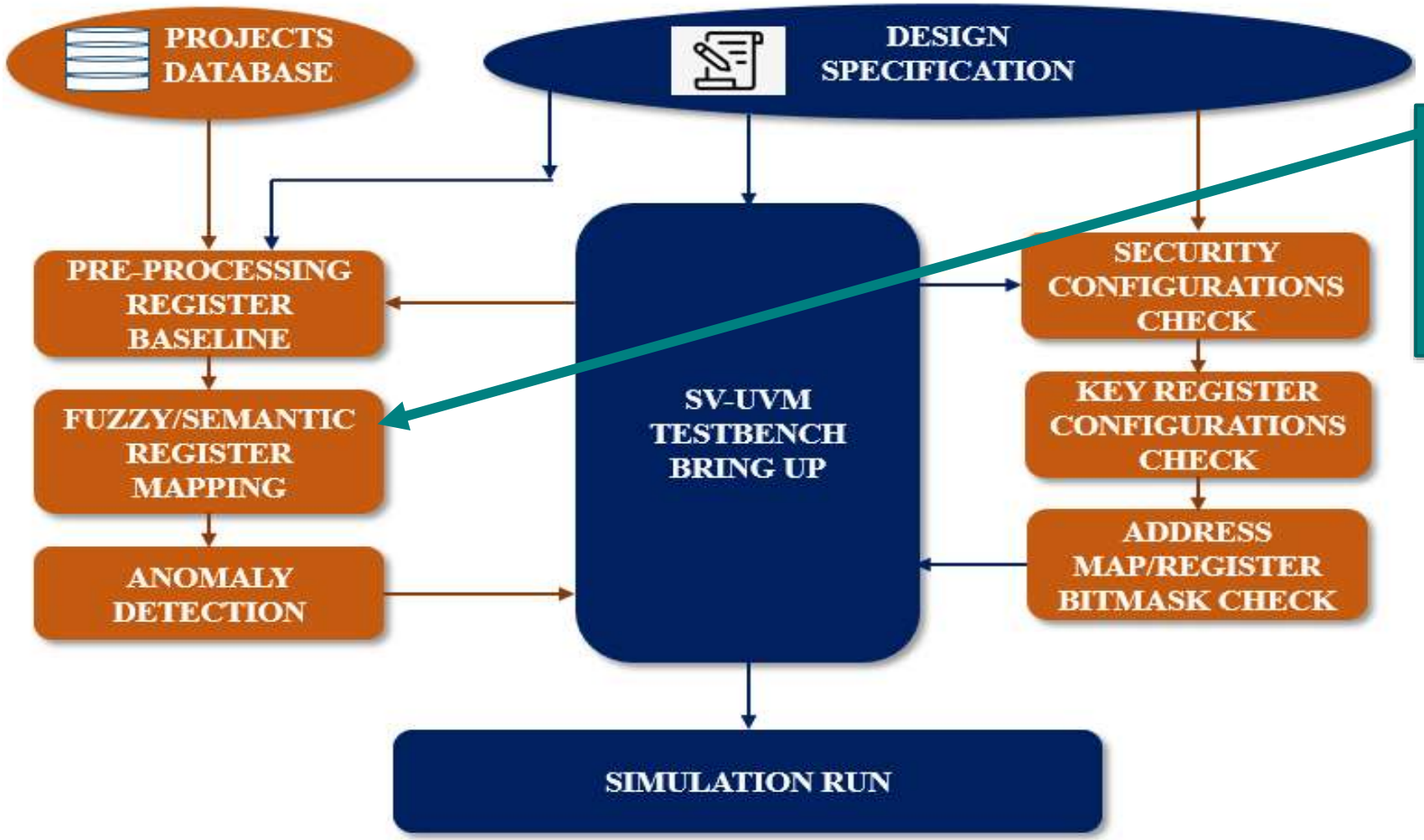


Past projects testbench register configurations = $(P_1, P_2 \ldots P_n)$
It is used to create a statistical baseline model **B = {P, V_{mode}, V_{set}}**

P = $\int_{i=0}^{i=n} \frac{P_i}{n}$,presence ratio (frequency of occurrence across projects) ,
V_{mode} = Mode Value (frequently used initialization value),
V_{set} = Value Set (set of all observed values)

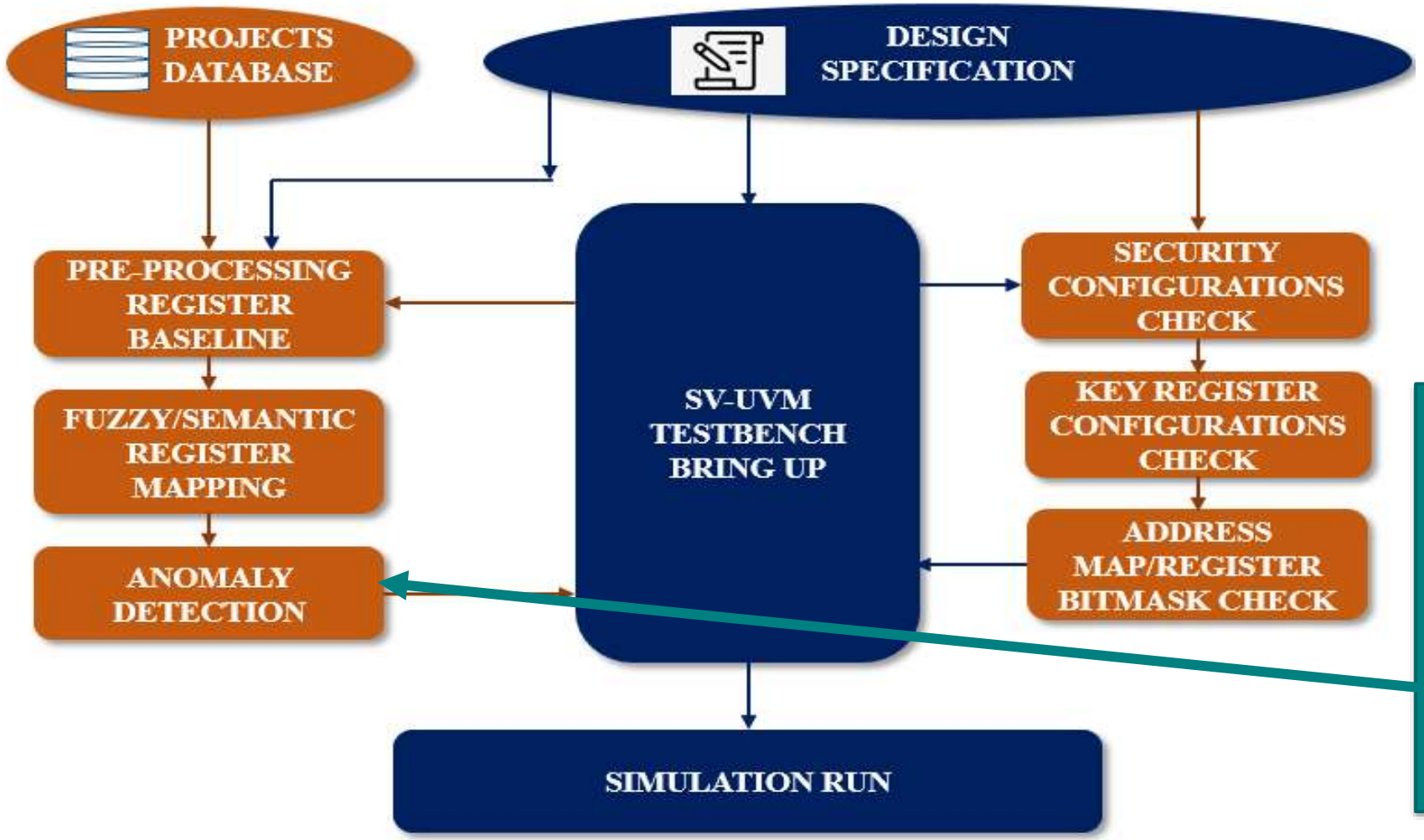
Mismatch/additional register configurations are filtered out by rule based check by comparing $P_{current}$ and $(P_1, P_2 \ldots P_n)$

Pre-simulation Framework



It filters out configuration with register/field renaming considering $P_{current}$ and $(P_1, P_2 \ldots P_n)$. A similarity ratio ≥ **0.85** flagged as probable matches.

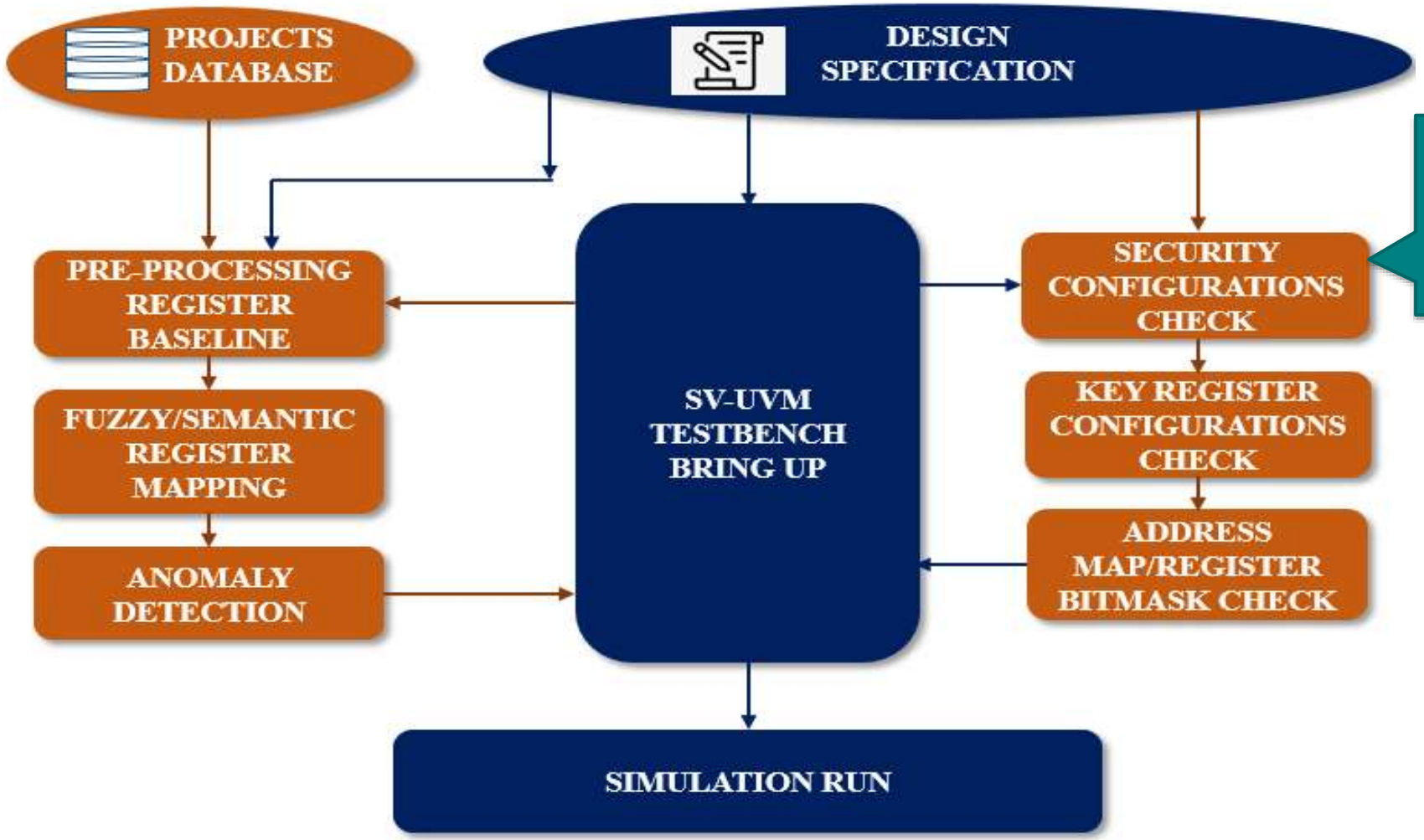
Pre-simulation Framework



Anomaly detection - Isolation Forest ML model processes $P_{current}$ and $(P_1, P_2 \ldots P_n)$

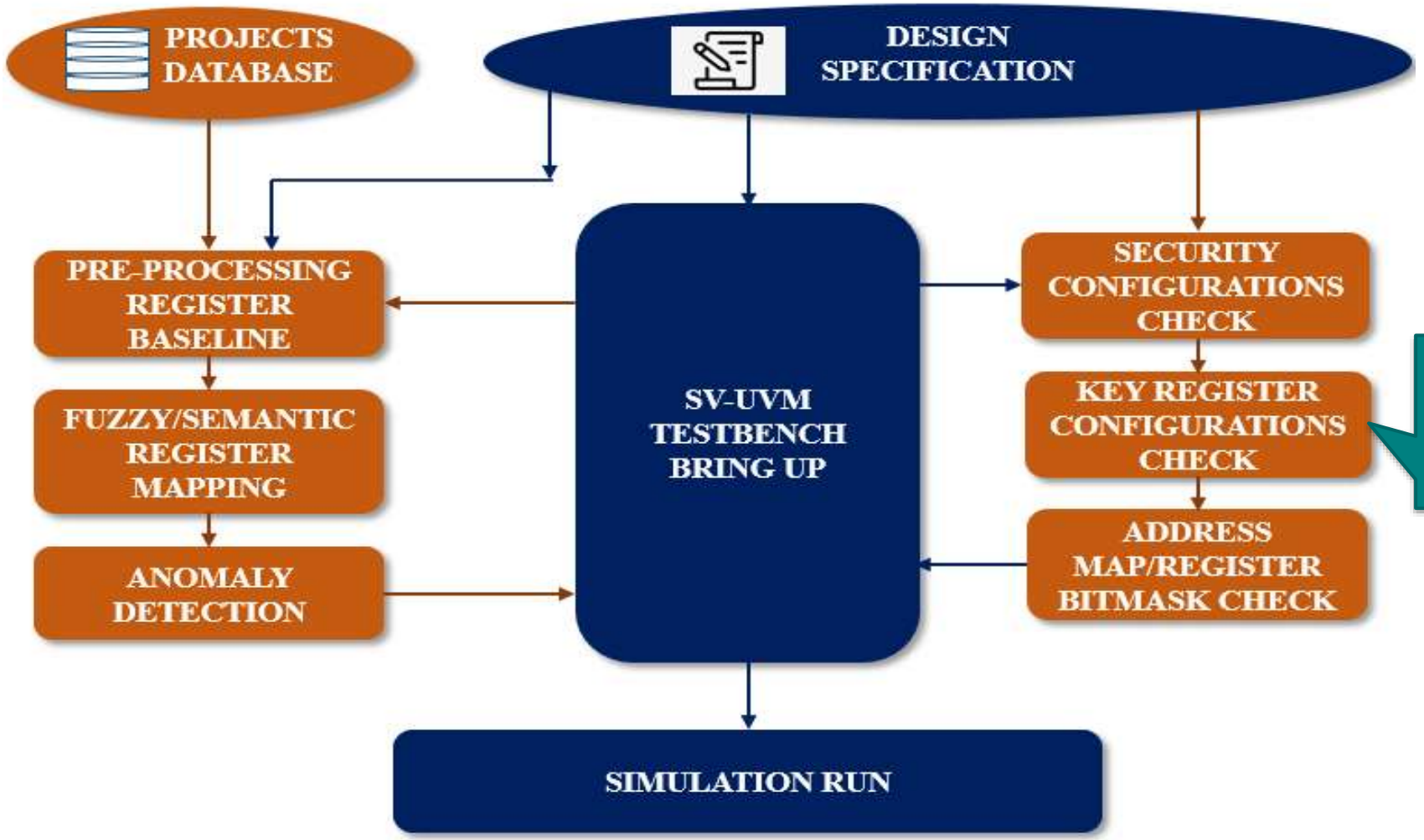
A fixed-length feature vector $\mathbf{X_i = [v_{i1}, v_{i2}, .. v_{in}]}$, where each dimension corresponds to a baseline register and i is project index.

Pre-simulation Framework



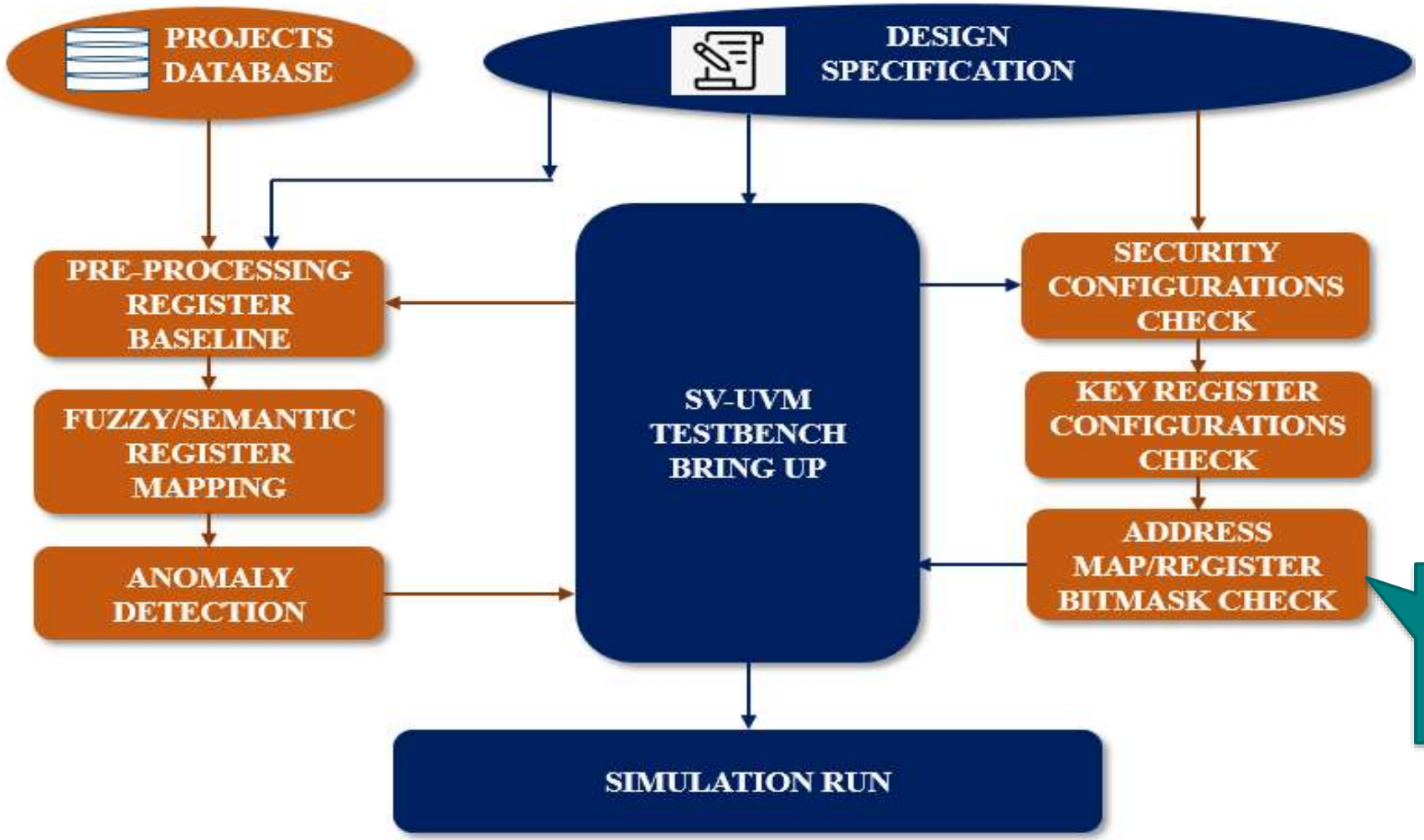
Security configurations of all $P_{current}$ register configurations are checked by referring design specification

Pre-simulation Framework



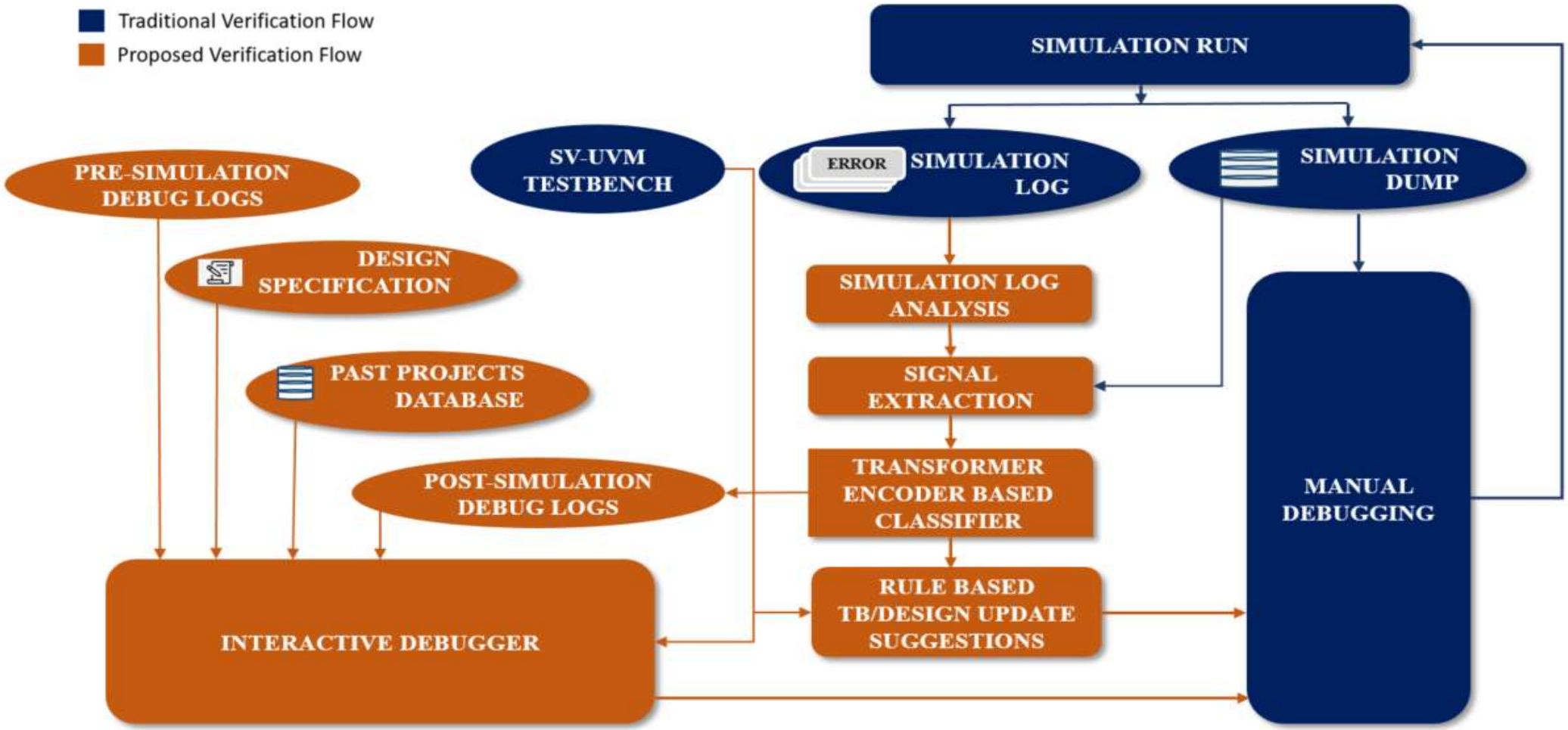
Key register configurations from $(P_1, P_2 \ldots P_n)$ are validated in $P_{current}$

Pre-simulation Framework

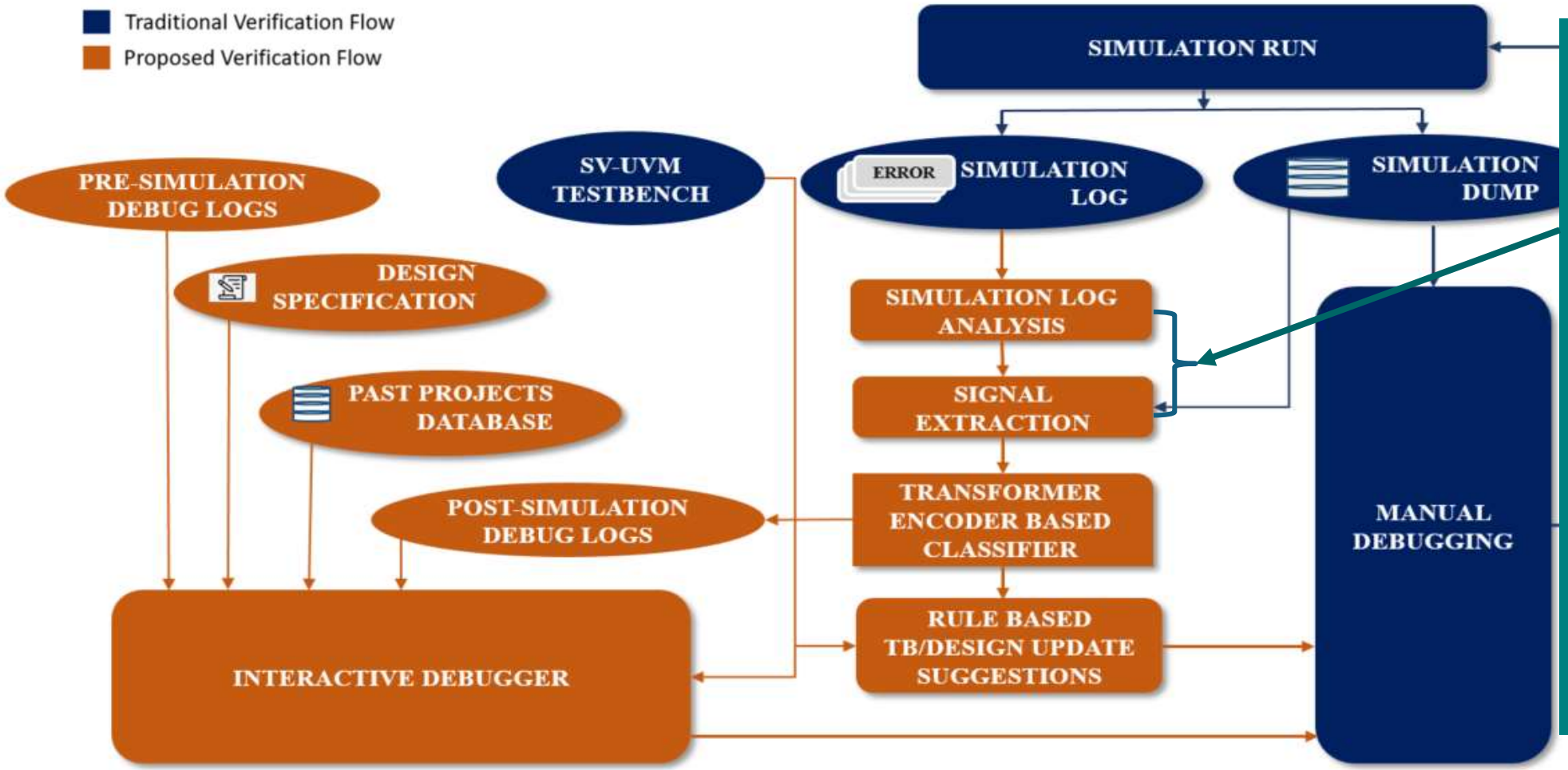


Reserved bitmask from design specification is used to validate $P_{current}$ register configurations

Post-simulation Framework



Post-simulation Framework

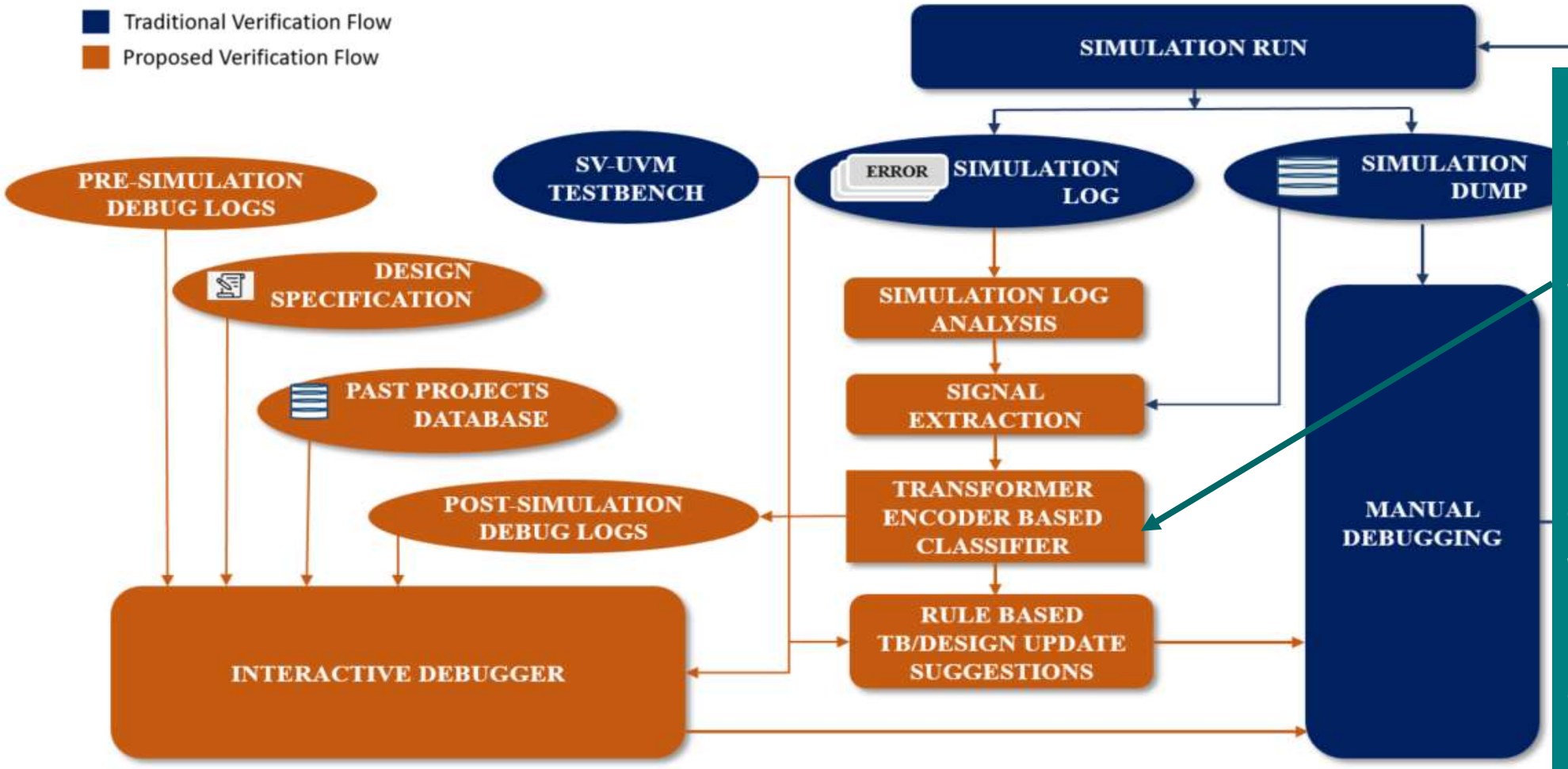


Simulation Log Analysis and Signal extraction:

i. The test progress as per UVM reporting is mapped to the signal flow path (extracted from bus architecture) to gauge the port(s) whose signals need to be extracted for further debug.

ii. The signals from this port(s) are extracted from the simulation dump in CSV format.

Post-simulation Framework



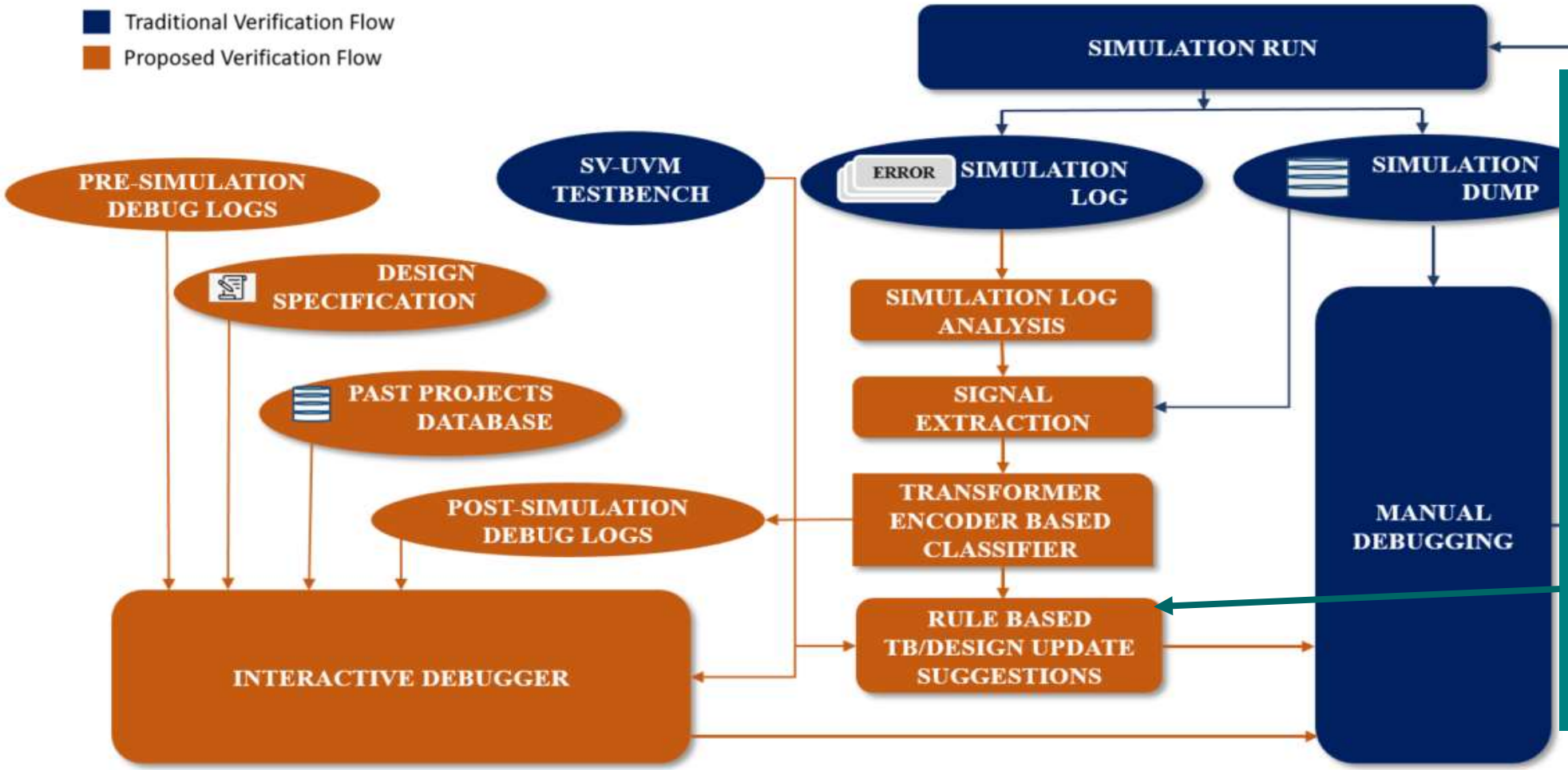
Pre-processing for Transformer encoder Classifier:

i. Valid transaction windows are picked from simulation dump and packetized into arrays.

ii. These arrays are reformatted and padded to maintain uniform window length.

iii. Preprocessed simulation dump data is fed to the transformer encoder.

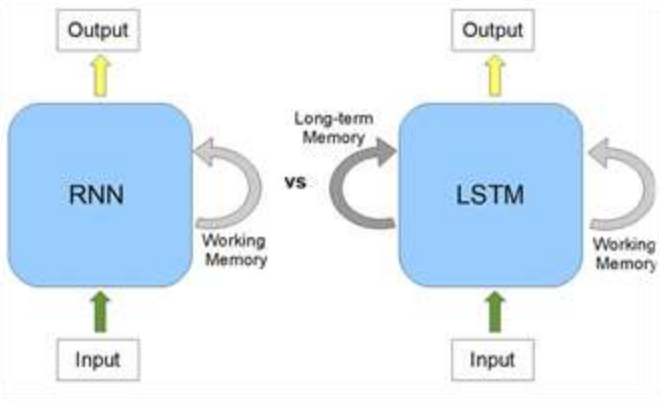
Post-simulation Framework

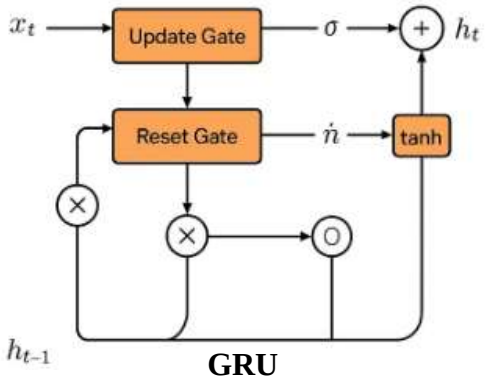


Rule-based / Historical debug suggestion:
i. Violations highlighted by ML model can be mapped to a missing signal port or an incorrect TB configuration (task and line no.)
ii. Historical debug data can be accessed using past projects debug database to suggest debugging solutions for similar issues.

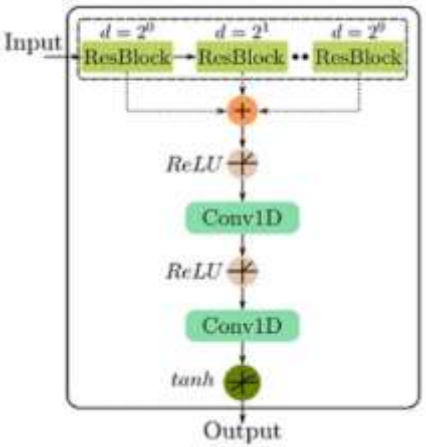
Deep Learning Models

- Recurrent Neural Network (RNN)
- Long Short-term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- Temporal Convolutional Network (TCN)
- Hybrid CNN-GRU Model
- Transformer Encoder

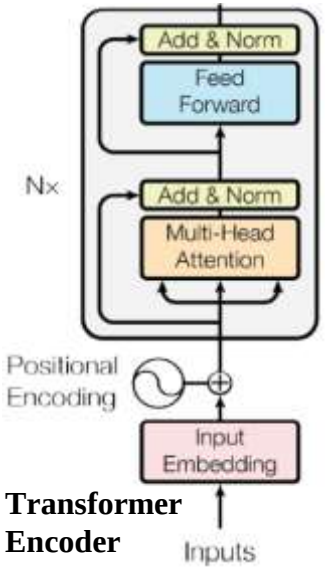




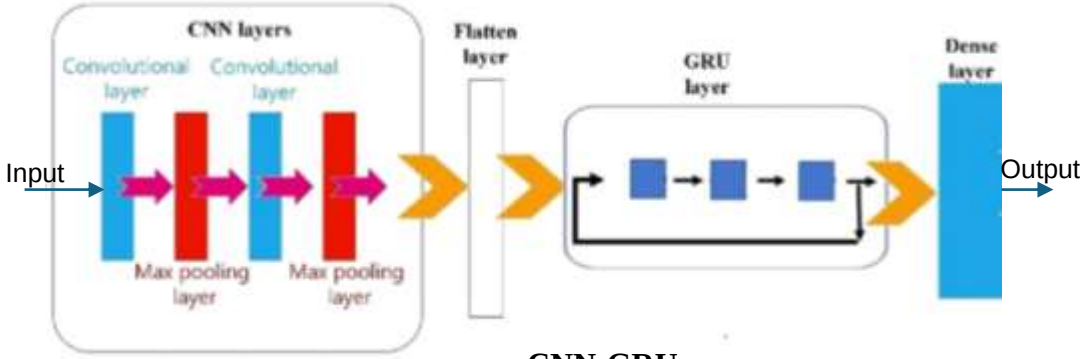
GRU



TCN



Transformer Encoder

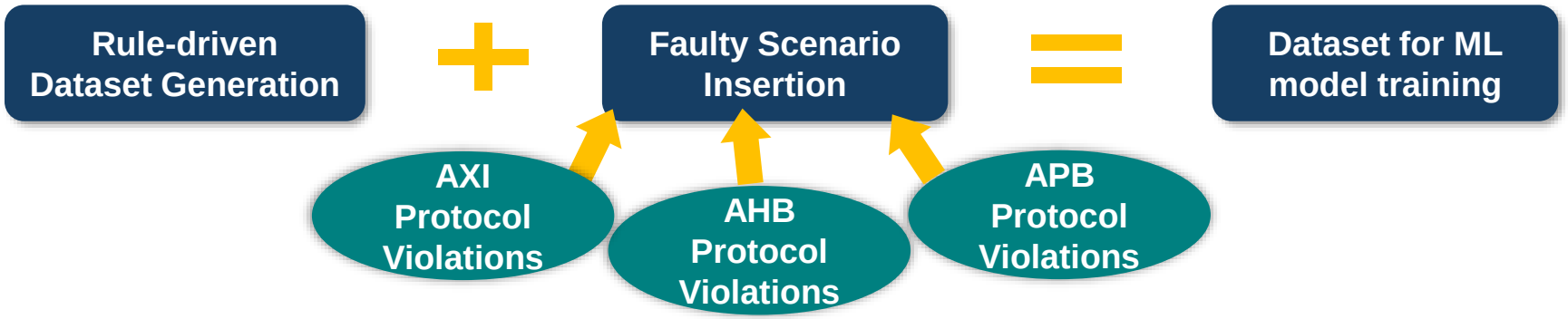


CNN-GRU

Source: www.researchgate.net

Deep Learning Models

- **Dataset generation and pre-processing for ML model training**

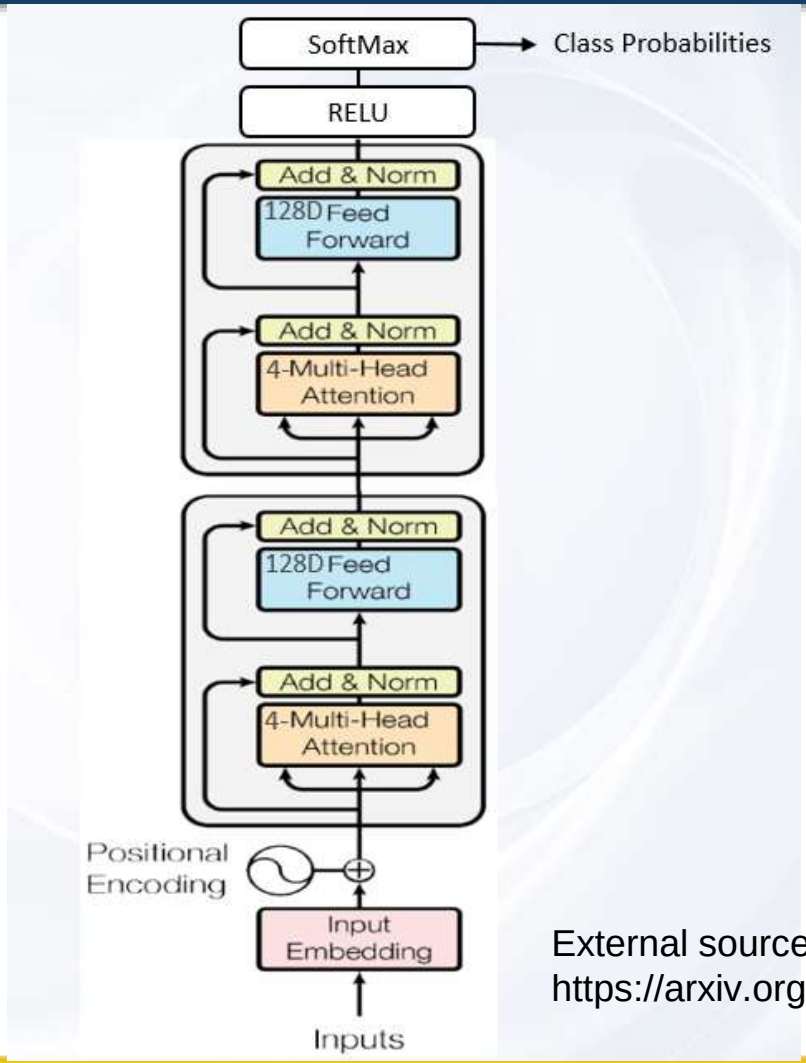


- **Toolchains and Libraries**

Pandas Library | Scikit-learn Toolkit | Pytorch Library | Regex Library | Tkinter Library

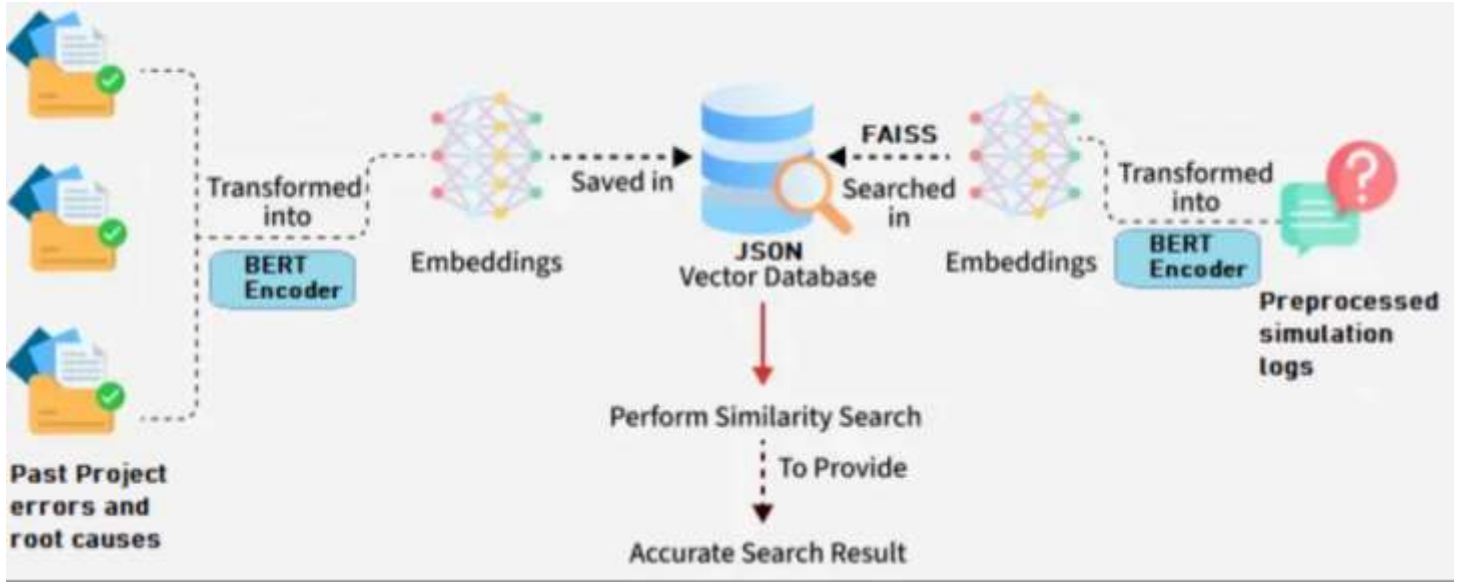
Transformer Encoder Architecture

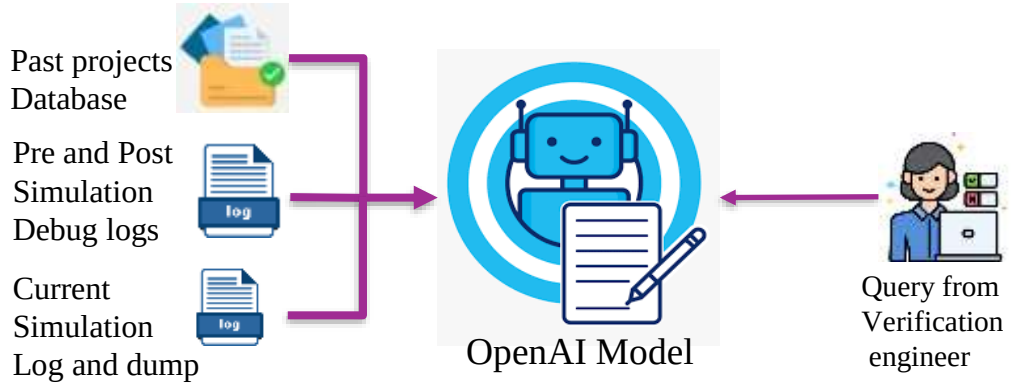
- The architecture consists of two encoder blocks, each containing 4-head multi-head self-attention layer (head dimension = 32, total attention dimension = 128) followed by a position-wise feed-forward network (inner dimension = 128) and ReLU activation.
- Both the attention and feed-forward sub-layers are wrapped with skip/residual connections, allowing the network to preserve protocol features while learning global dependencies.
- Dropout layers were added to prevent overfitting. Adam optimizer was for training transformer encoder.
- SoftMax activation function was used for the final layer.



External source:
https://arxiv.org/pdf/1706.03762

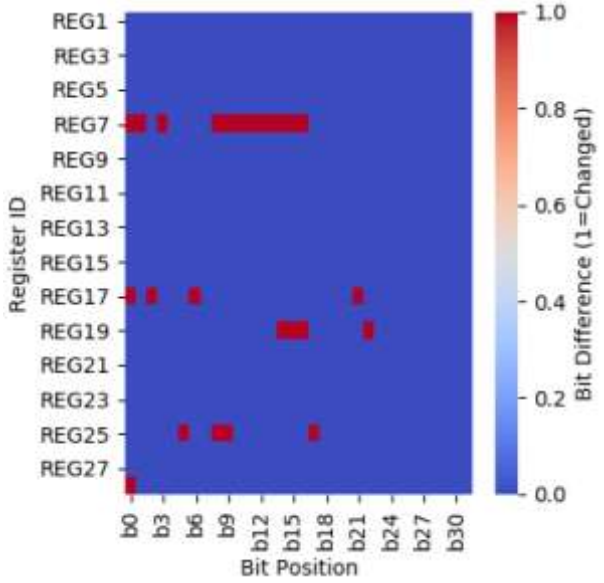
LLM Log analysis and Interactive Debugger



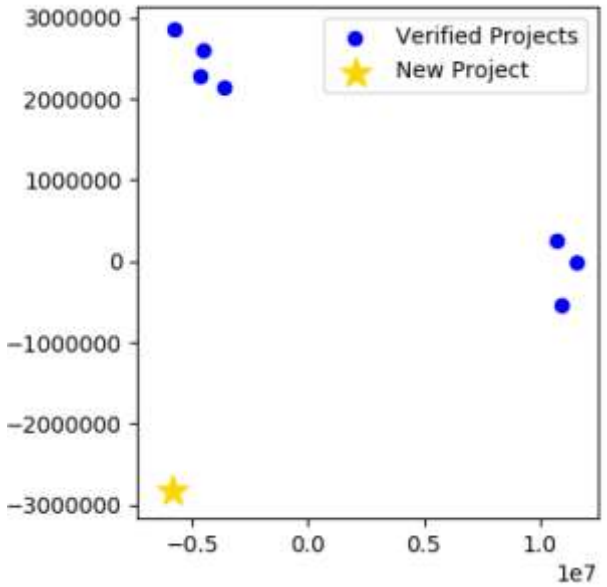


External Source:
https://share.google/4yQF98ZUN3aWTVeGi

Results : Pre-Simulation Framework

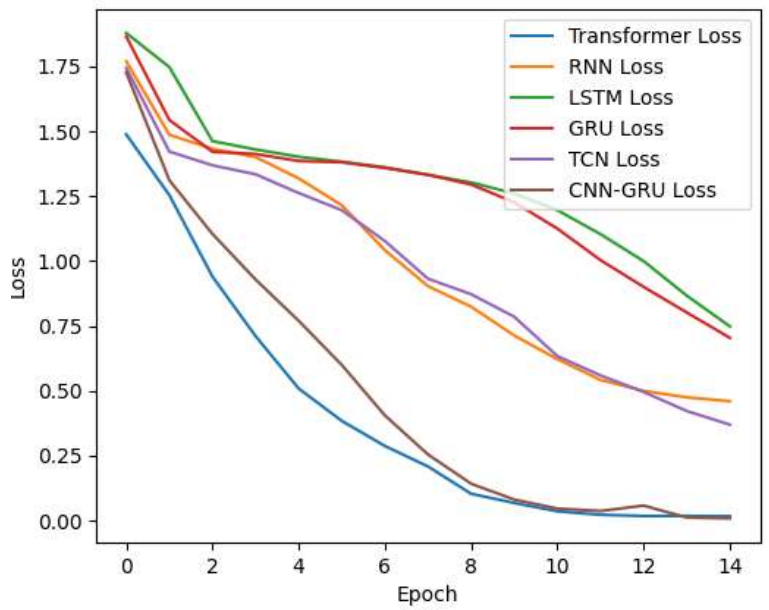


Register Bit-level deviations (new vs baseline)

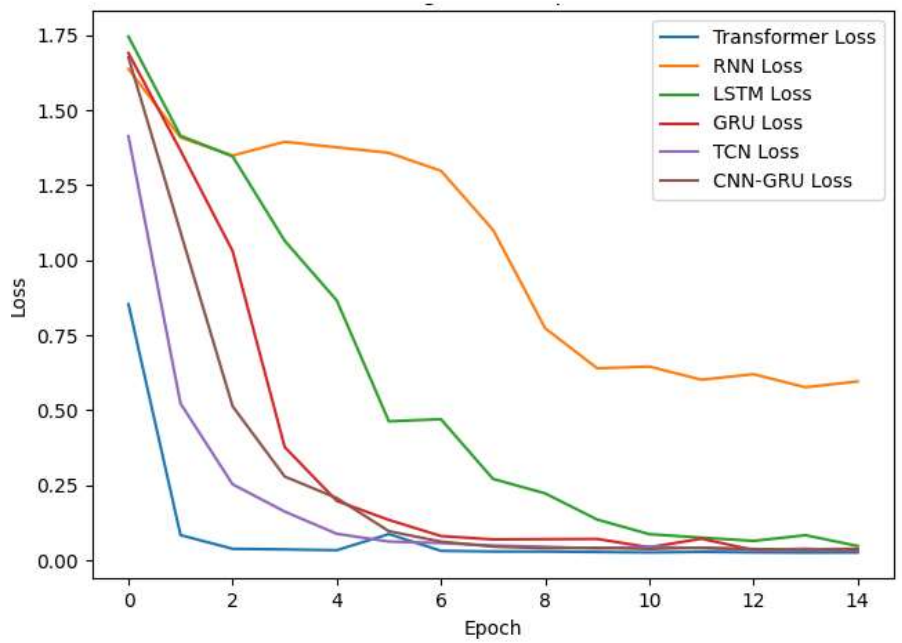


Anomaly Map (Isolation Forest + PCA)

Results : Post-Simulation Framework

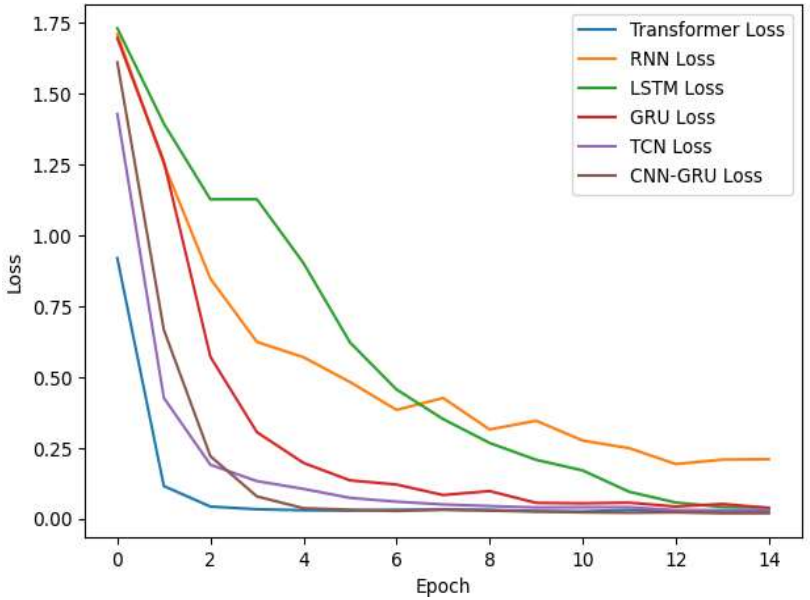


Loss across epochs APB

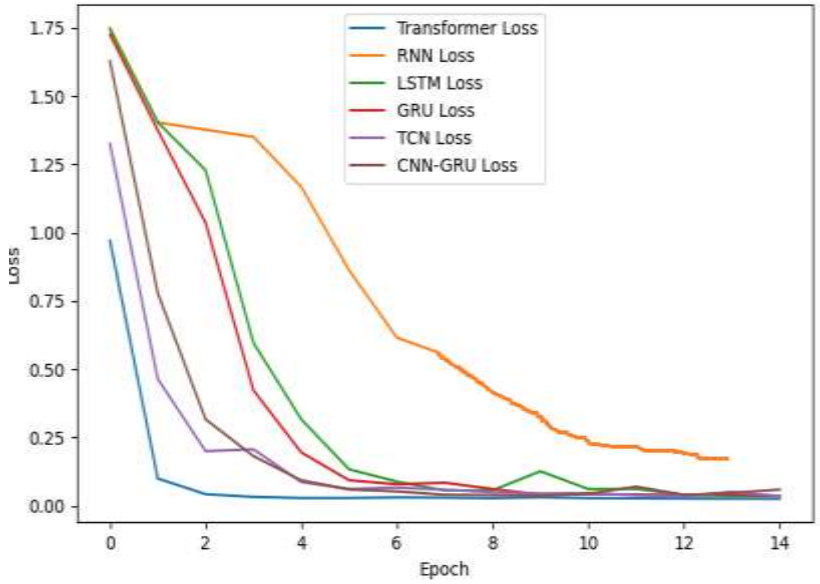


Loss across epochs AHB

Results : Post-Simulation Framework



Loss across epochs AXI-read channel



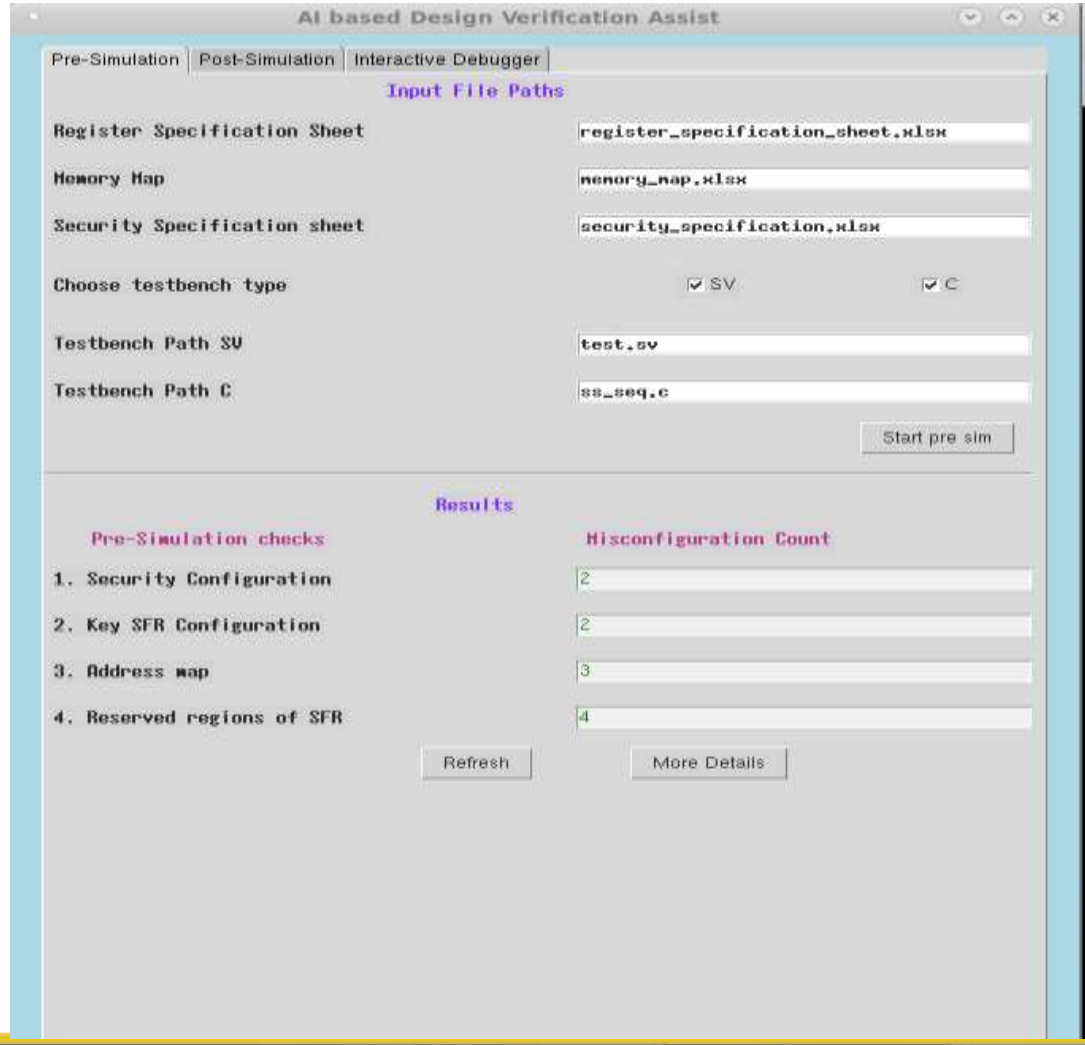
Loss across epochs AXI-write channel

Results : ML Models Performance Metrics

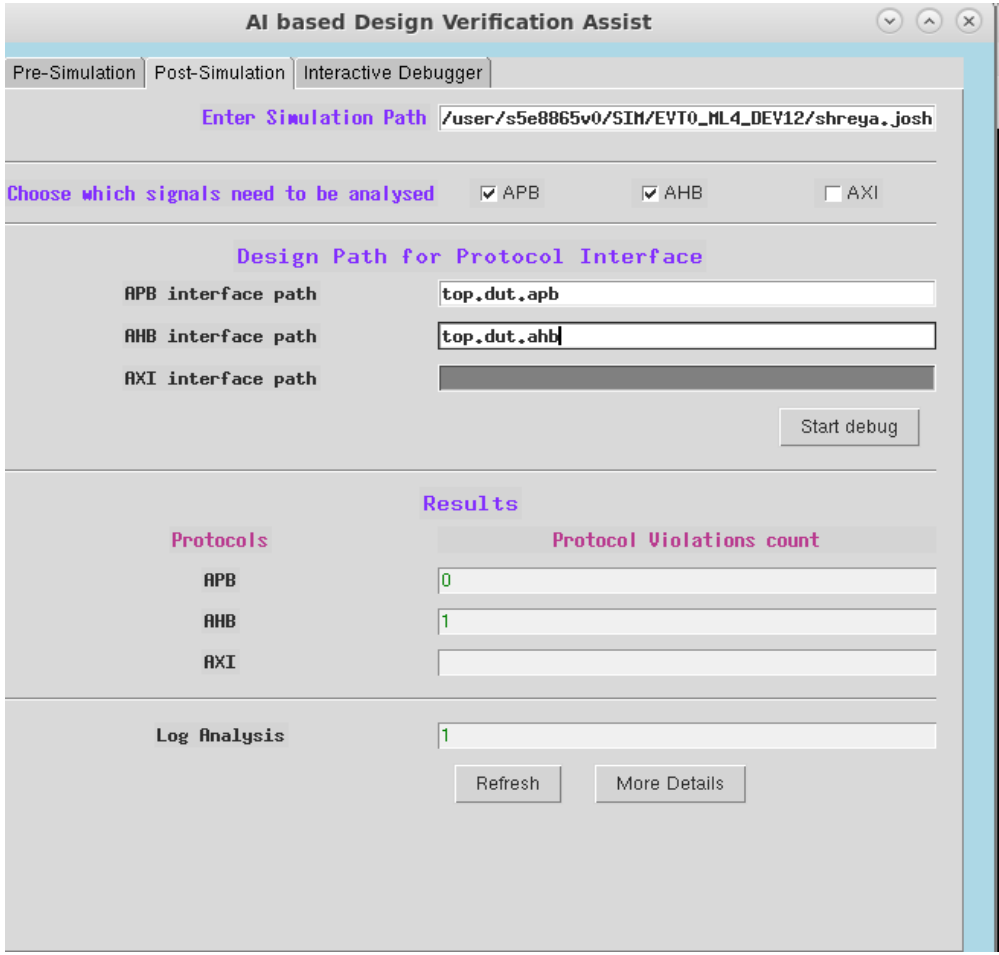
Protocol	Algorithm	RNN	LSTM	GRU	TCN	CNN-GRU	Transformer-Encoder
APB	Accuracy (%)	95.50	97.65	96.38	92.69	95.68	98.50
	Precision (%)	95.82	97.76	96.23	84.08	97.34	98.67
	F1-Score	93.74	97.77	96.05	84.85	92.60	98.48
AHB	Accuracy (%)	95.17	96.21	96.30	96.67	97.45	97.89
	Precision (%)	91.27	91.29	91.45	92.45	92.62	92.71
	F1-Score	93.18	93.20	93.24	94.56	94.73	94.84
AXI-Read Channel	Accuracy (%)	92.00	97.10	97.50	97.98	98.15	98.32
	Precision (%)	91.13	97.23	97.65	97.99	98.32	98.84
	F1-Score	91.95	96.54	96.63	98.01	98.43	98.76
AXI-Write Channel	Accuracy (%)	91.32	96.17	96.16	98.05	99.17	99.50
	Precision (%)	91.11	97.05	97.12	97.54	99.15	99.56
	F1-Score	91.56	95.89	95.91	98.16	99.16	99.51

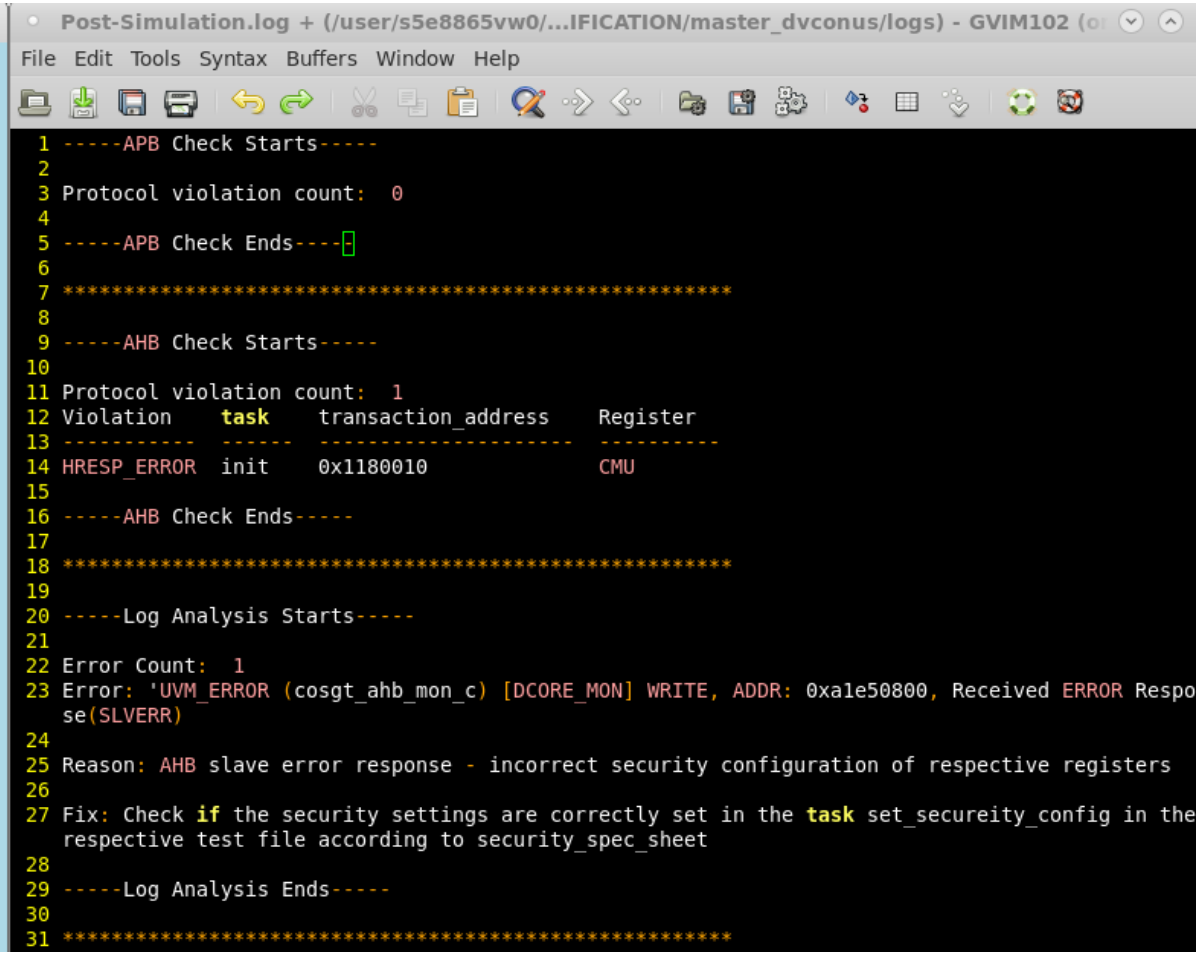
Transformer Encoder outperformed other ML models

Results : Pre Simulation Debugger GUI

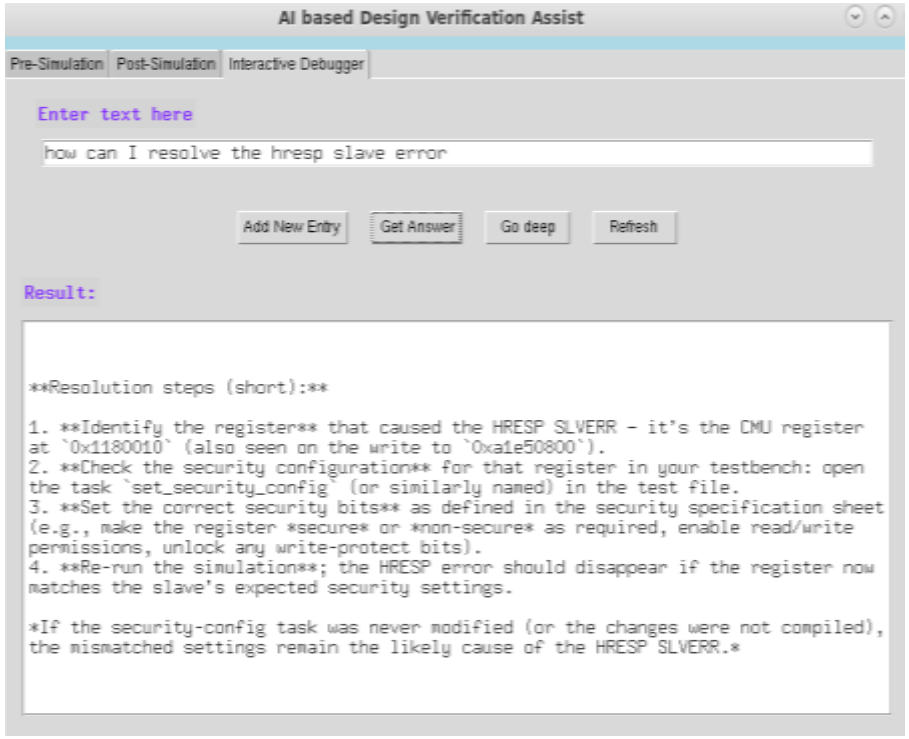


Results : Post Simulation Debugger GUI

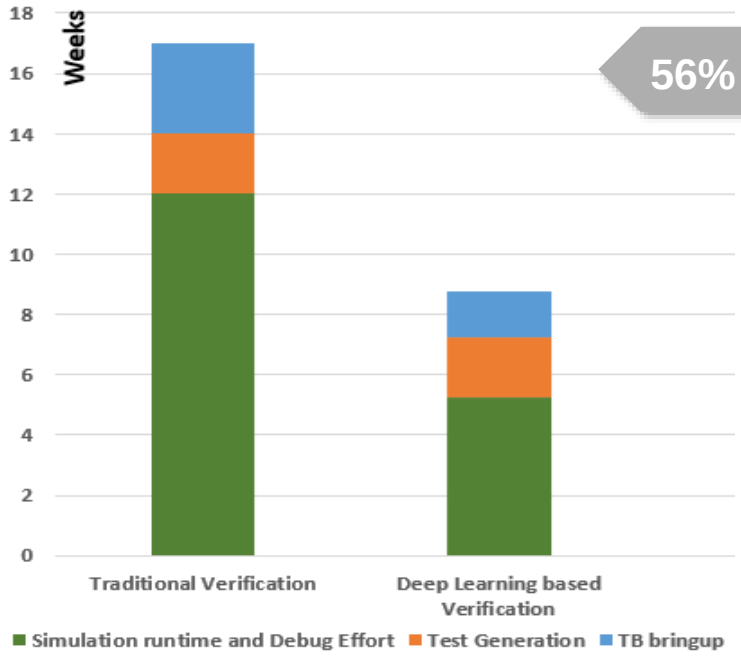




Results



Interactive Debugger GUI



56% reduction in Verification effort

Verification Effort Analysis

Conclusion and Future Scope

- **Extended Debug support : Bus architecture to dictionary**

- **Simultaneous debug :Verisium Python APIs**

- **Design Specification Analysis : Feature changes**

- **Improvise Interactive Debugger support: JIRA APIs**

- **To tackle ML model hallucination in complex SoC**

THANK YOU!