

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

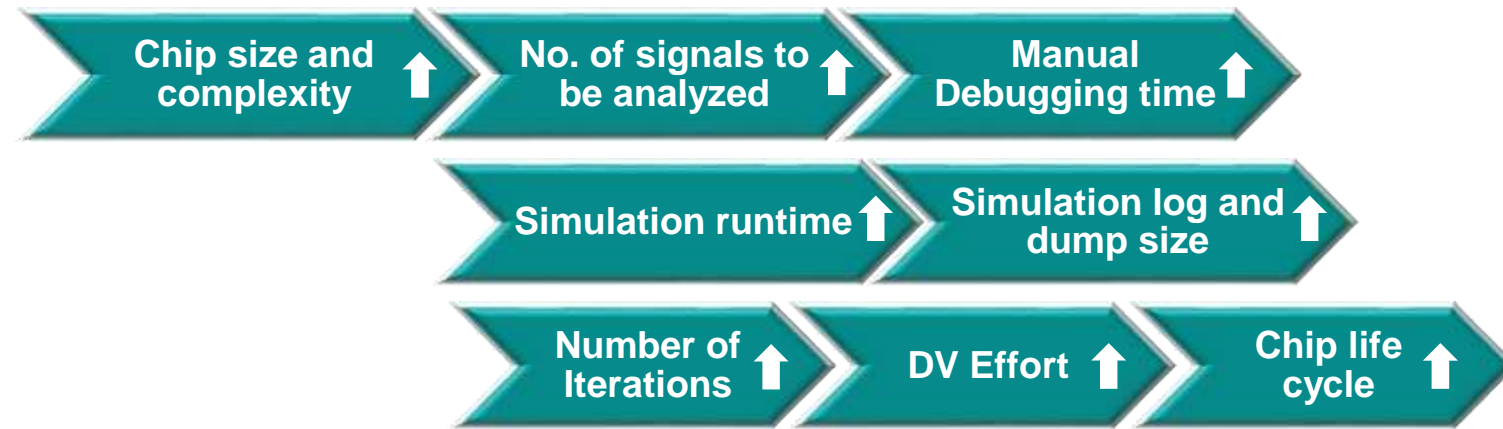
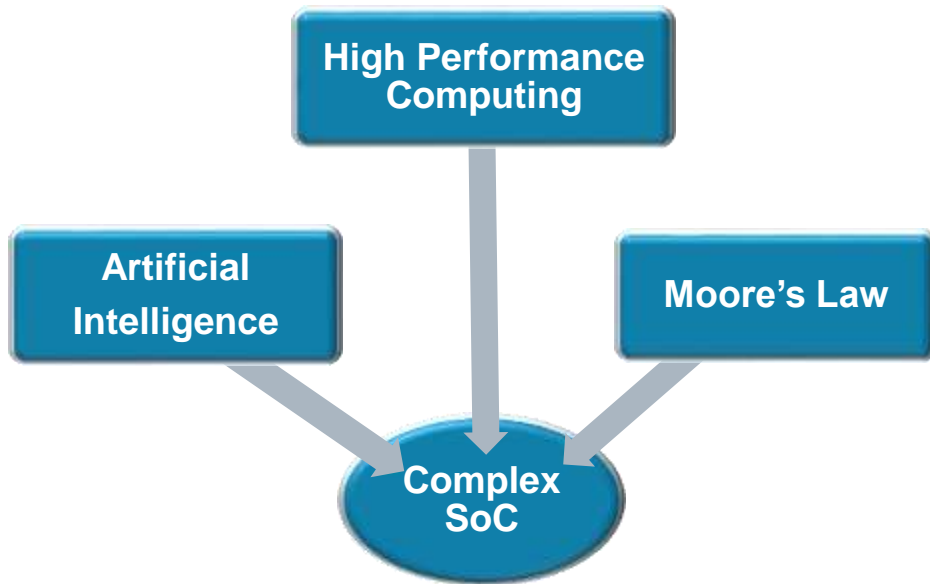
DEEP LEARNING MODELS

LLM LOG ANALYSIS AND INTERACTIVE DEBUGGER

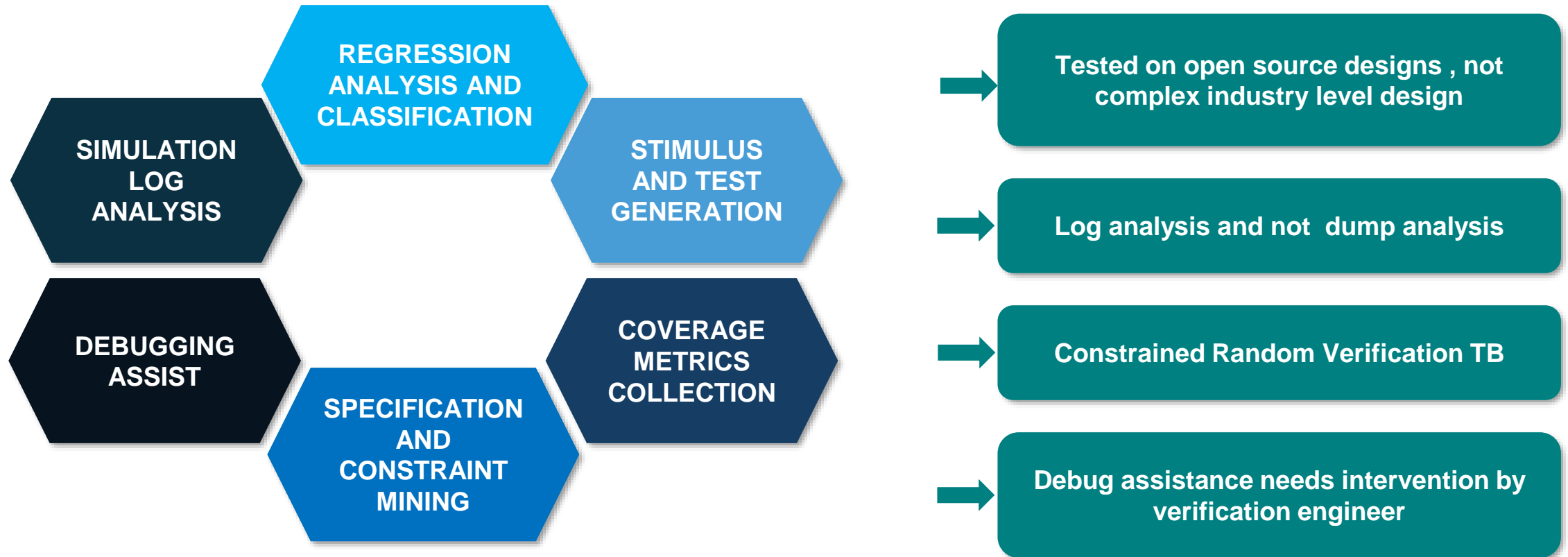
INTEGRATION CHALLENGES

RESULTS AND CONCLUSION

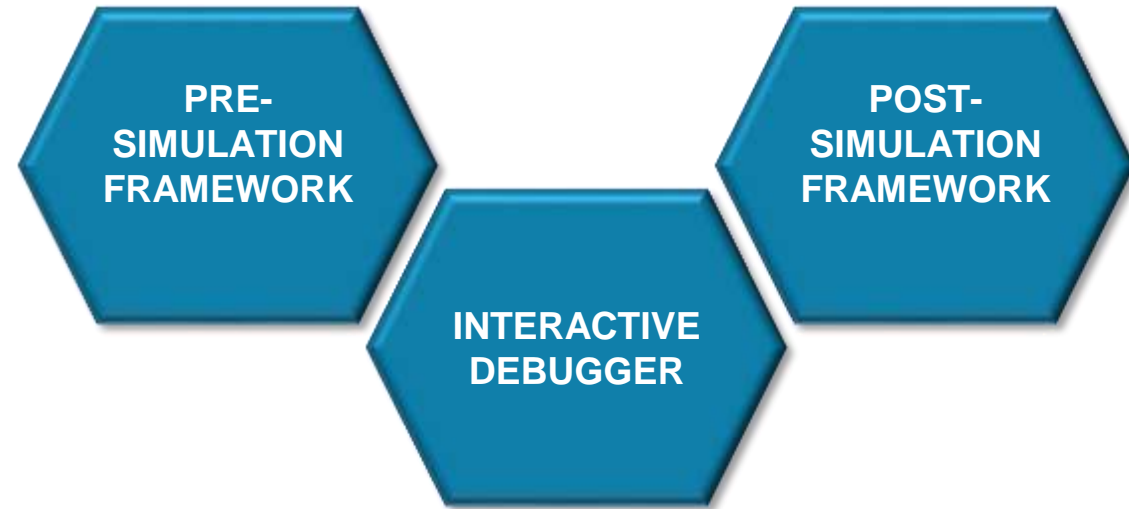
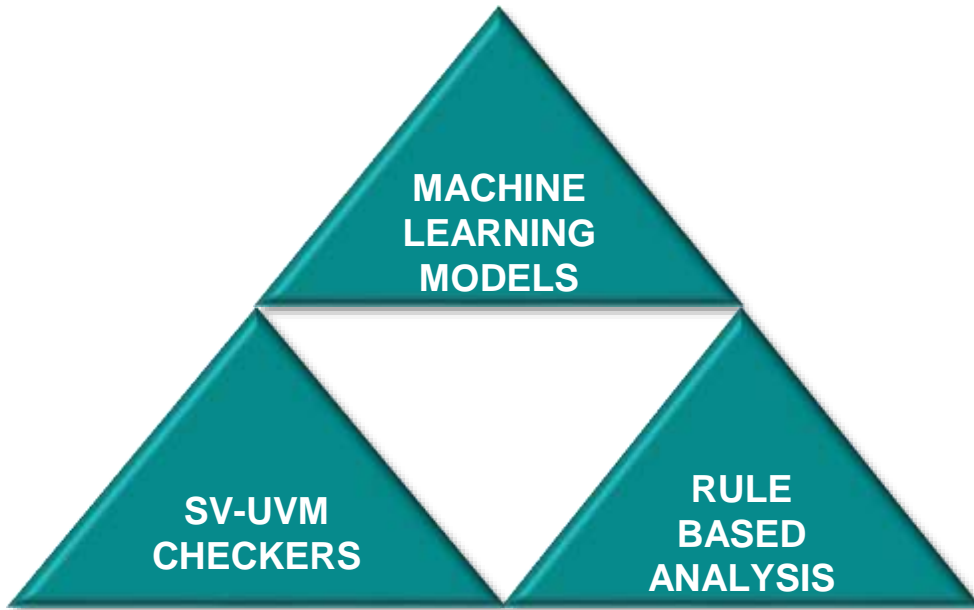
# Introduction : Design Verification Challenges



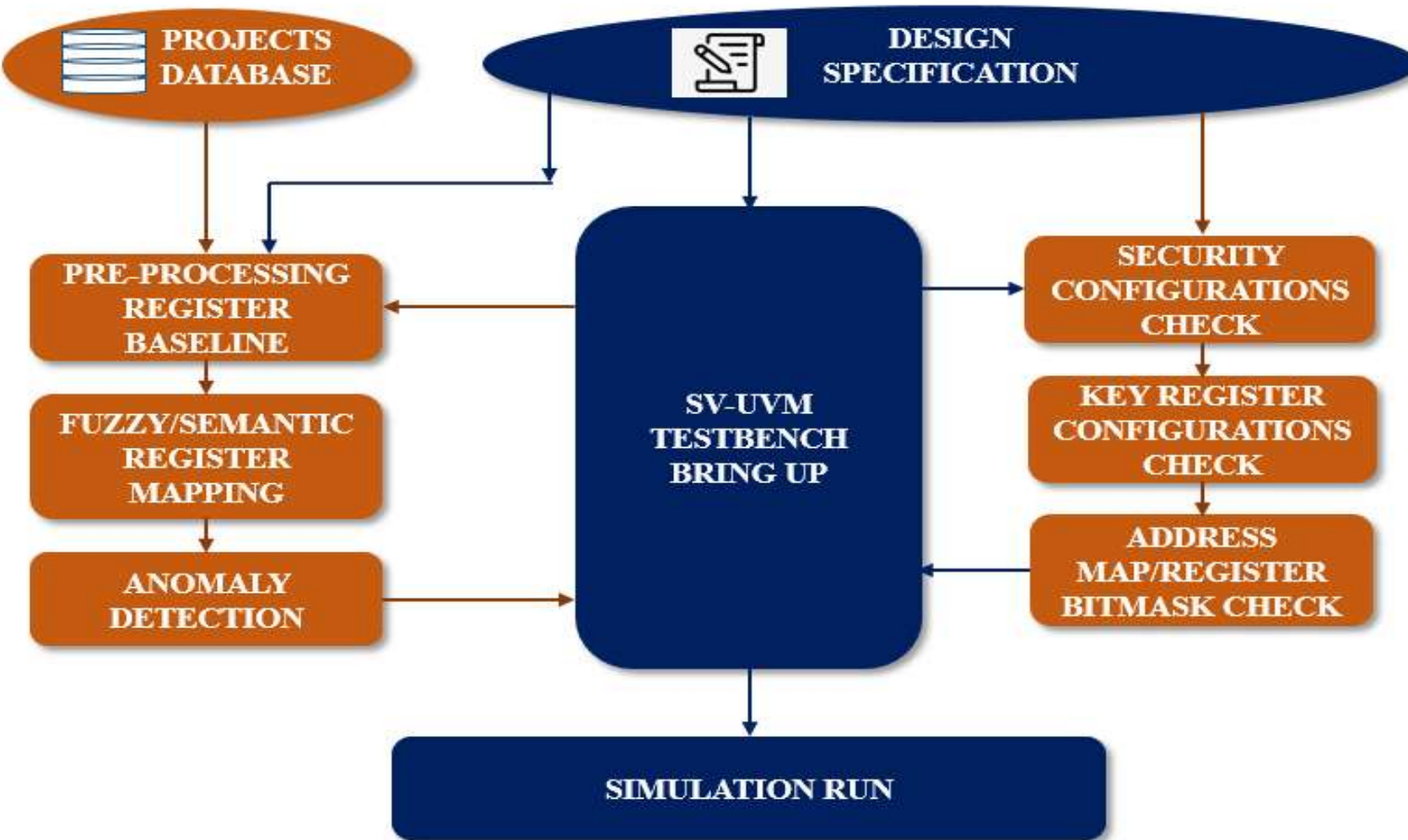
# State of the art AI-ML techniques



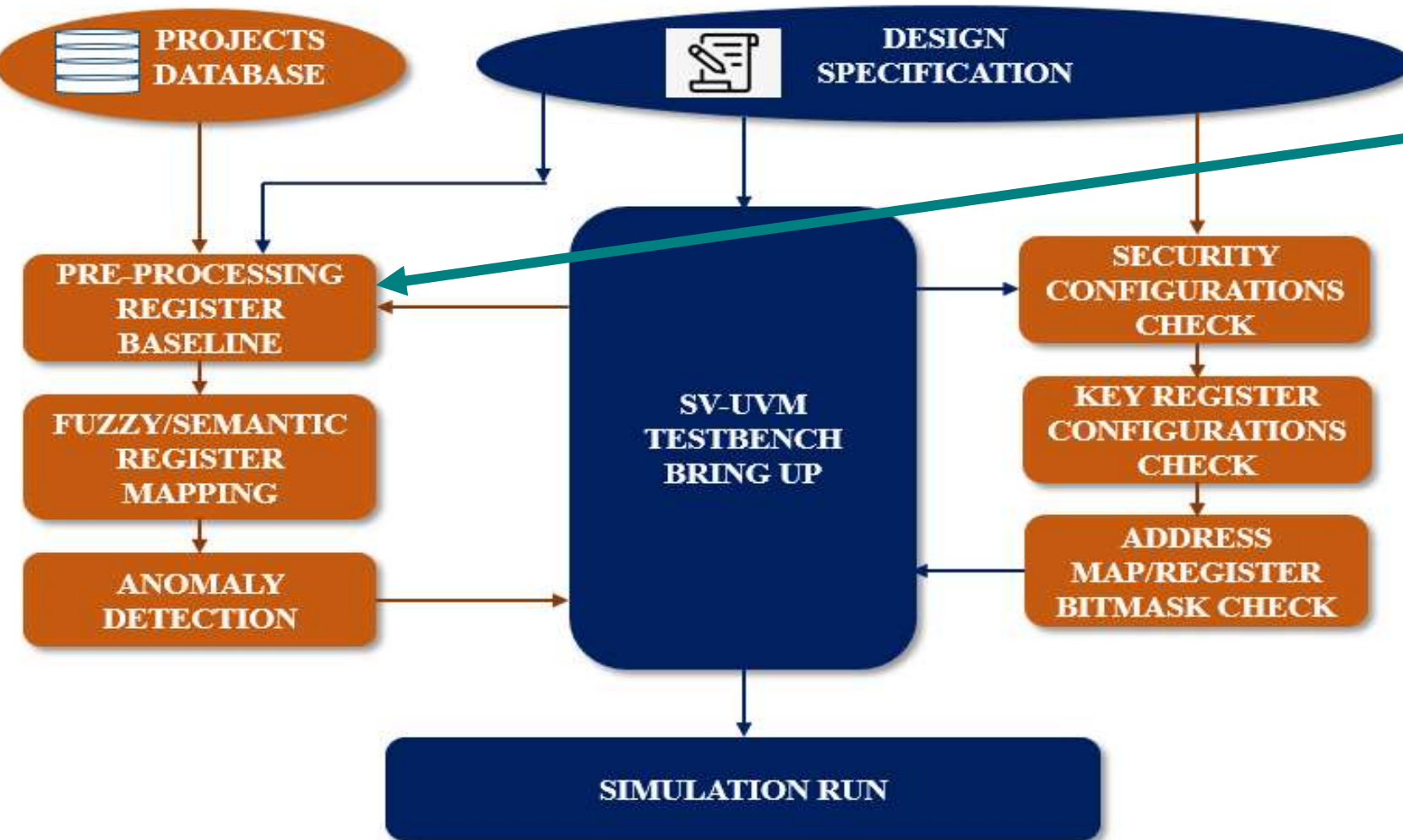
# Proposed Solution



# Pre-simulation Framework



# Pre-simulation Framework



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

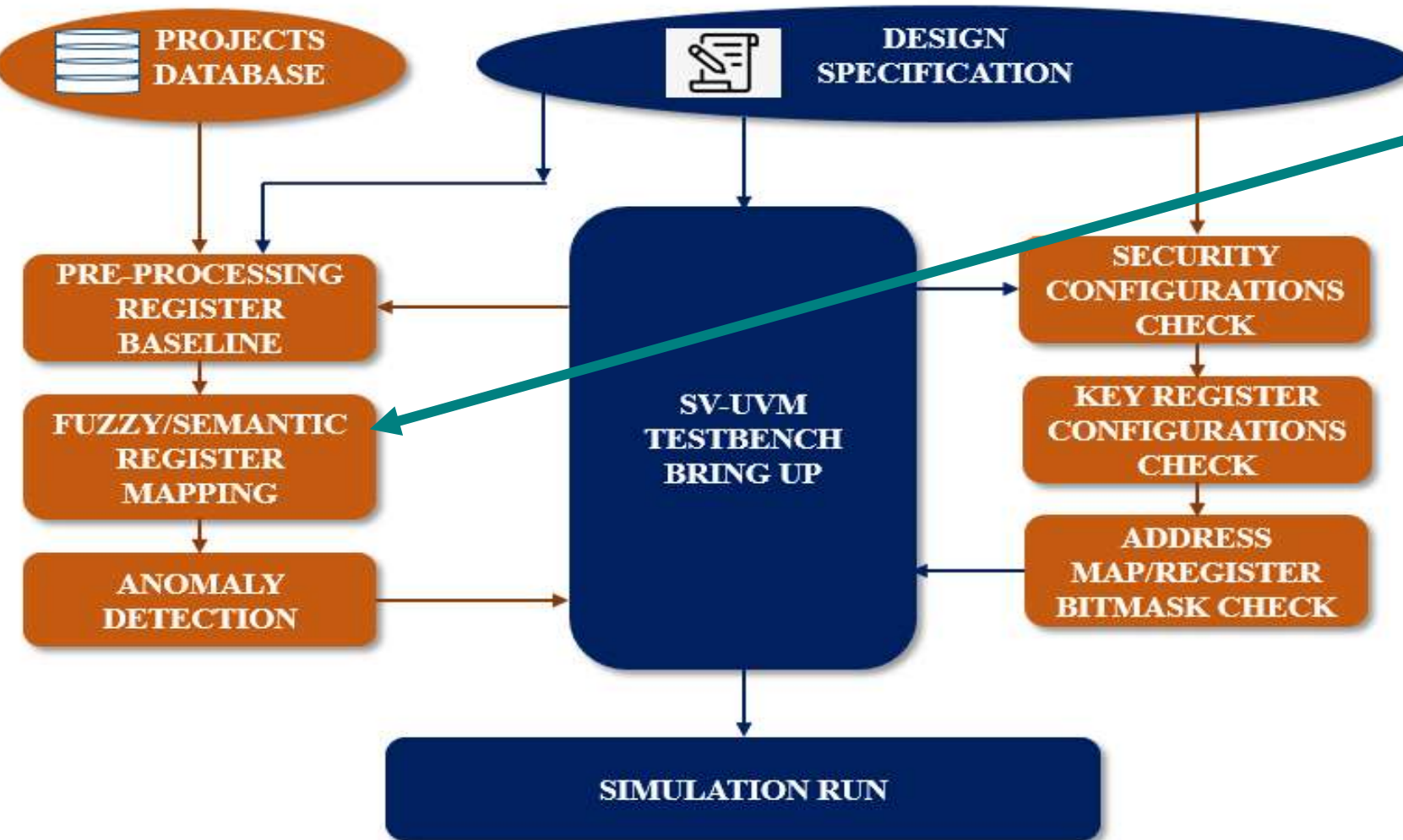
$P = \int_{i=0}^{i=n} \frac{P_i}{n}$ , presence ratio (frequency of occurrence across projects),

$V_{mode}$  = Mode Value (frequently used initialization value),

$V_{set}$  = Value Set (set of all observed values)

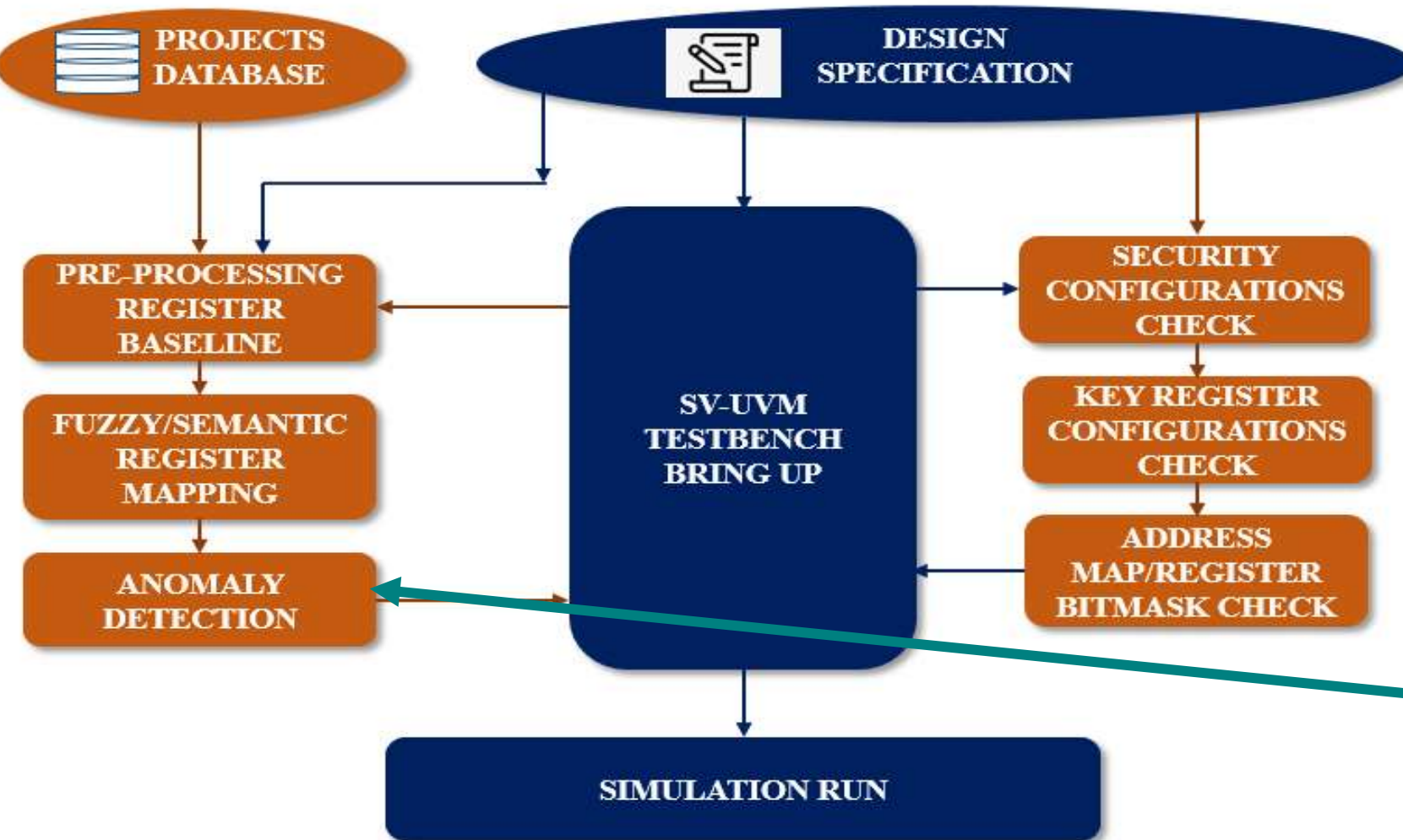
Mismatch/additional register configurations are filtered out by rule based check by comparing  $P_{current}$  and  $(P_1, P_2 \dots P_n)$

# Pre-simulation Framework



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

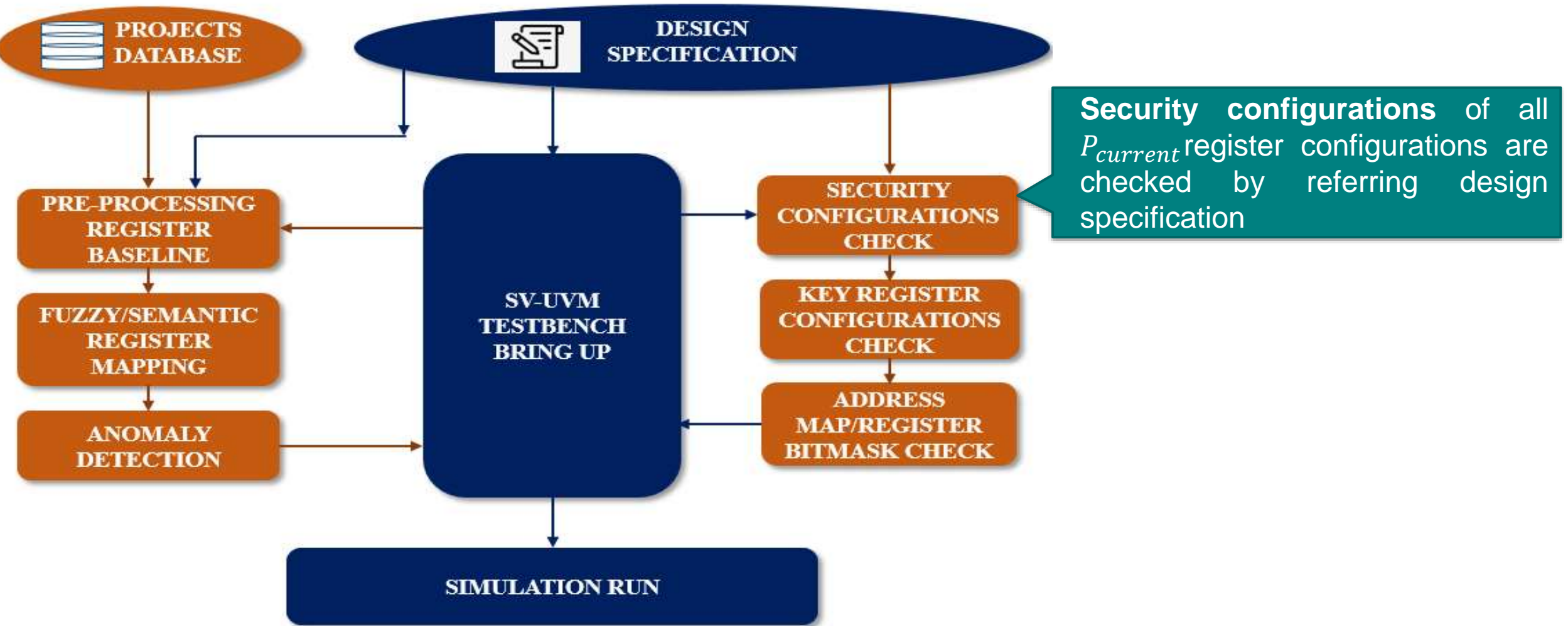
# Pre-simulation Framework



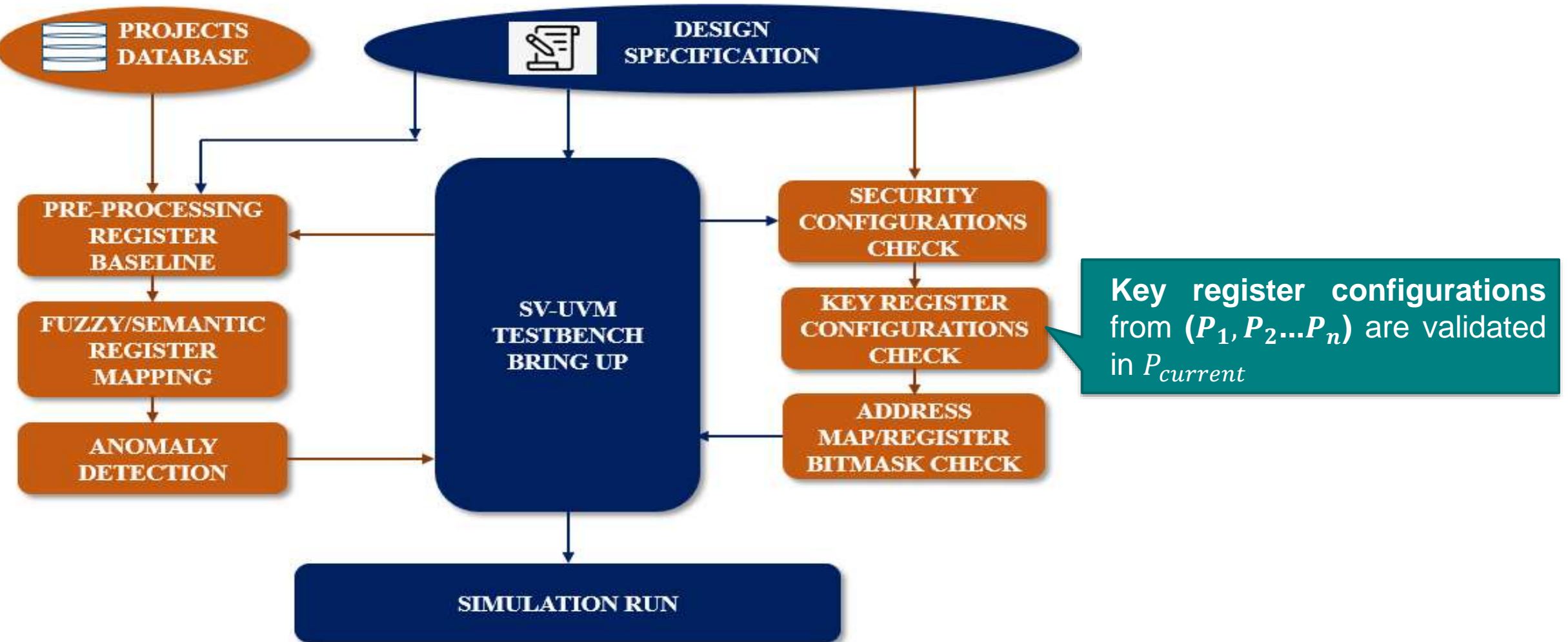
**Anomaly detection** - Isolation Forest ML model processes  $P_{current}$  and  $(P_1, P_2 \dots P_n)$

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

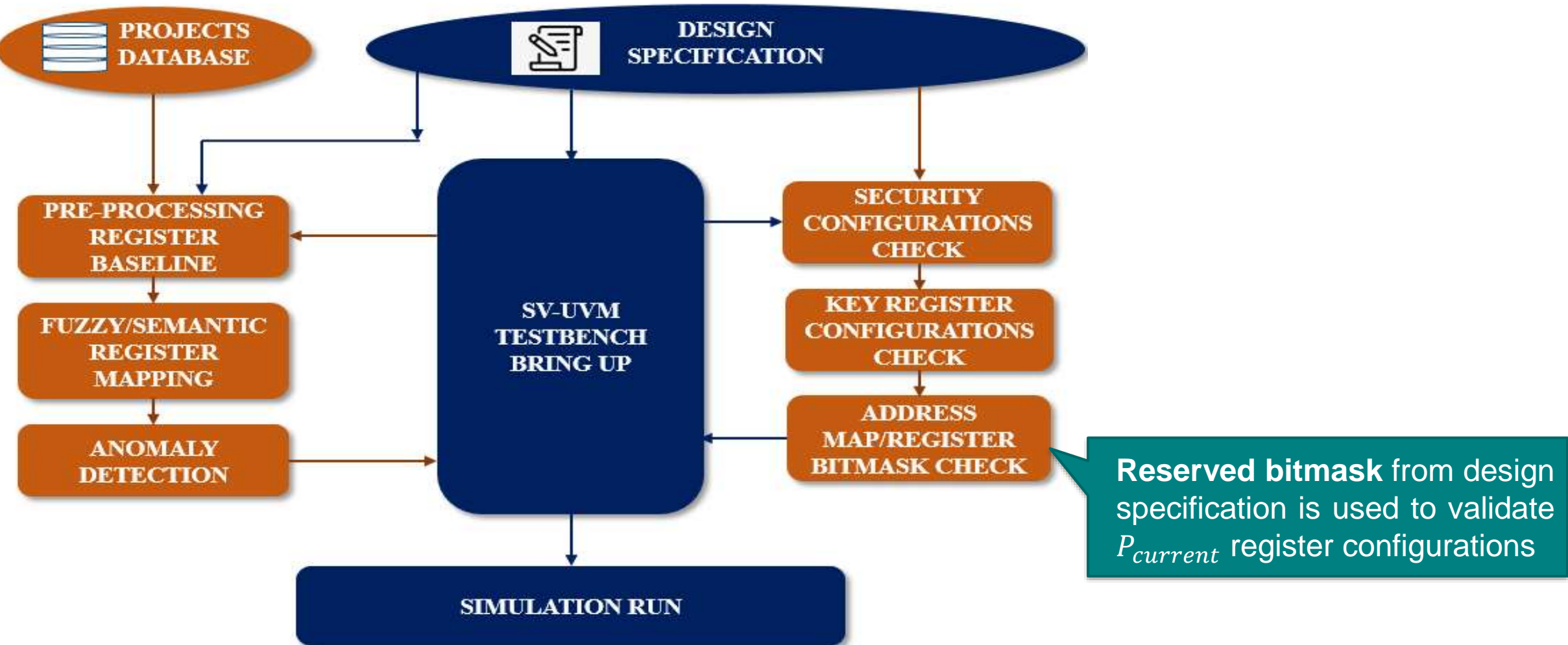
# Pre-simulation Framework



# Pre-simulation Framework

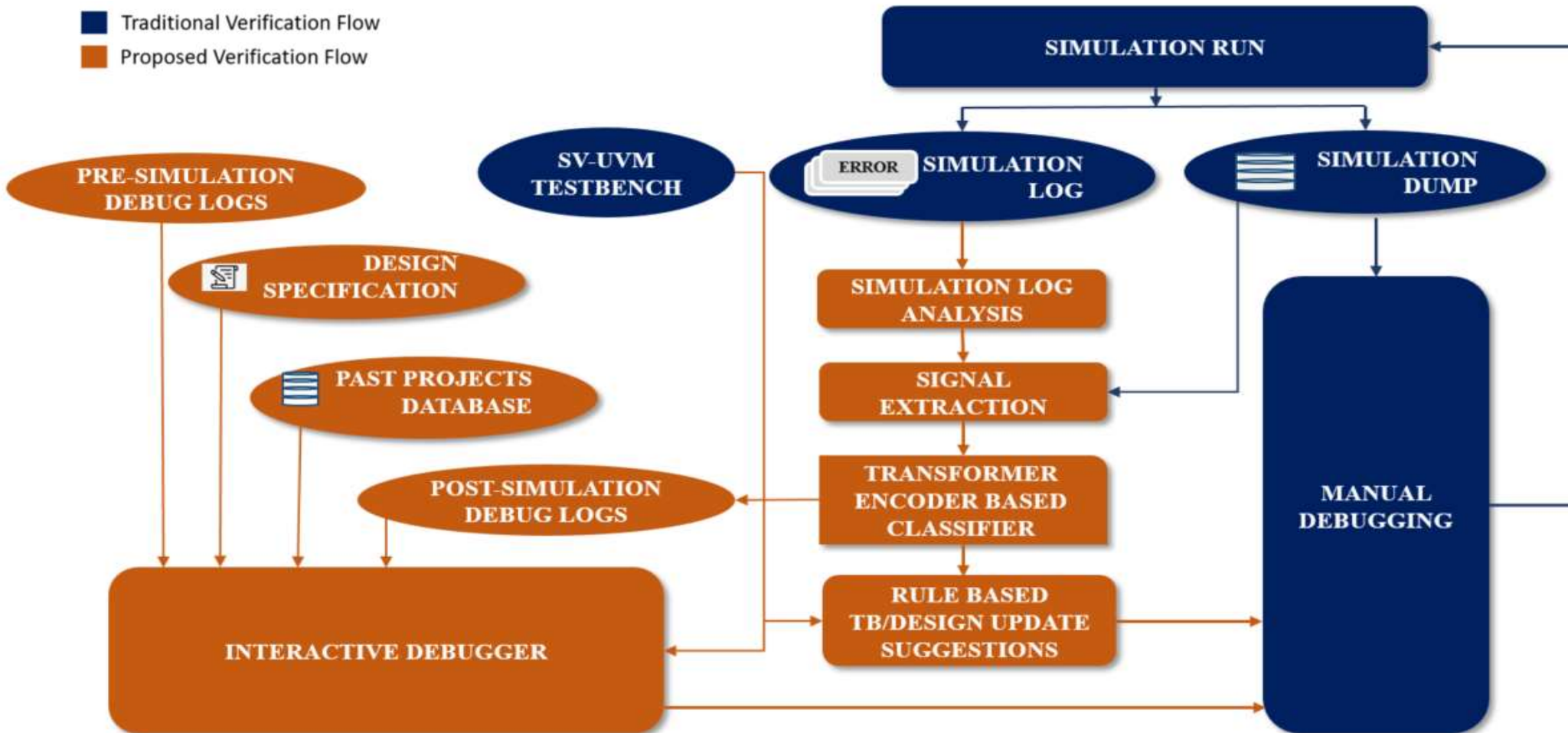


# Pre-simulation Framework



# Post-simulation Framework

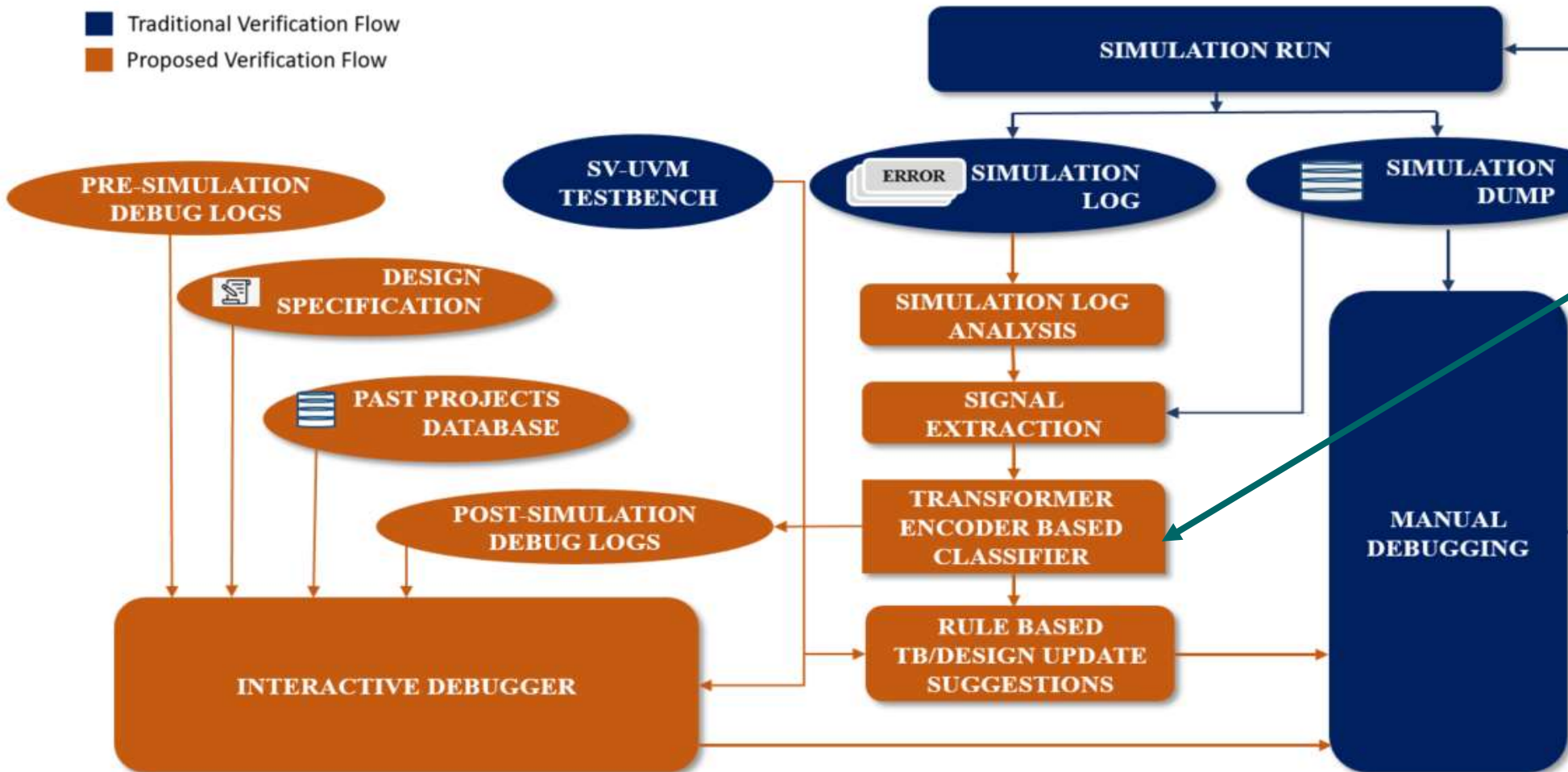
■ Traditional Verification Flow  
■ Proposed Verification Flow





# Post-simulation Framework

■ Traditional Verification Flow  
■ Proposed Verification Flow

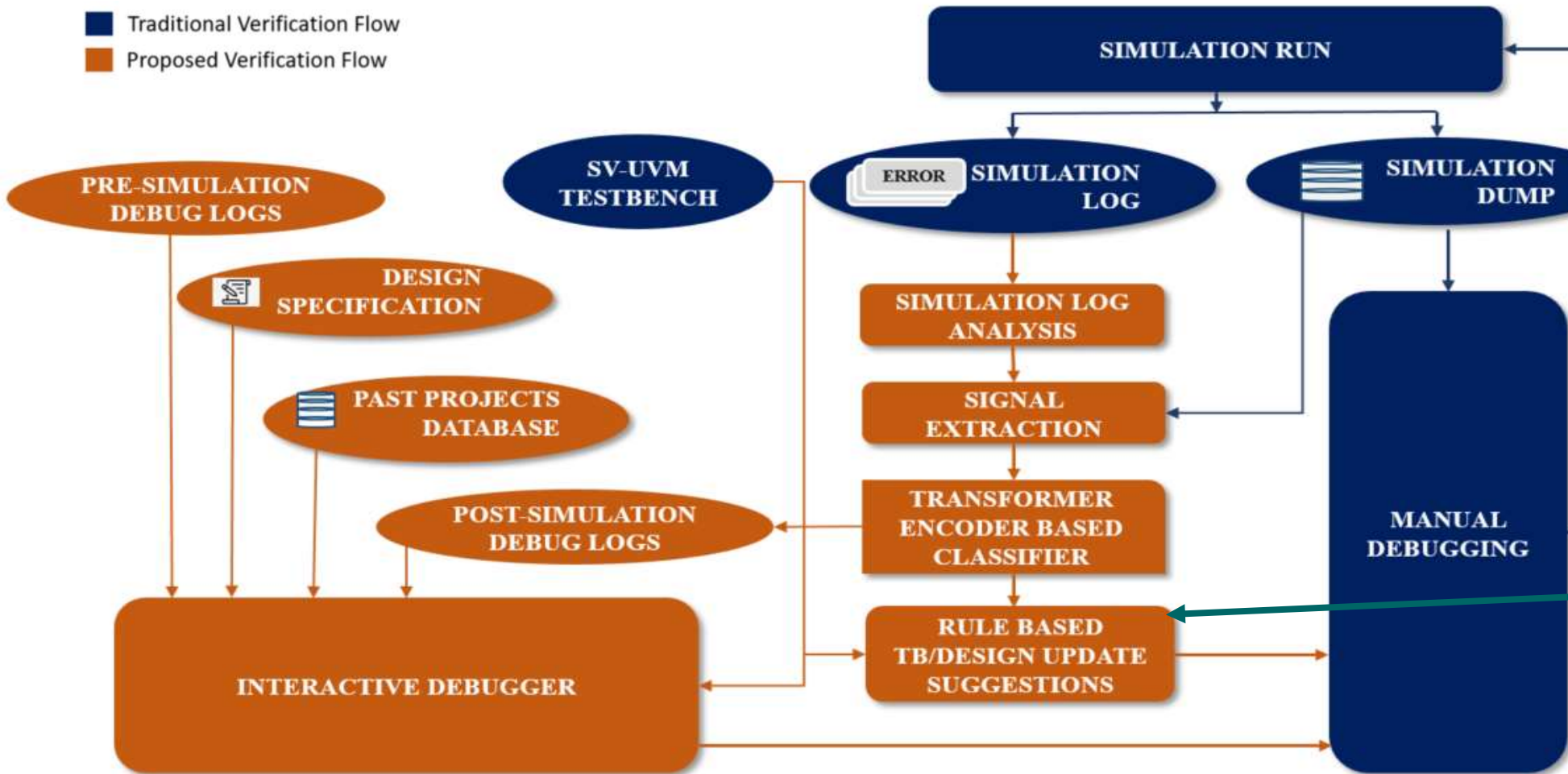


## Pre-processing for Transformer encoder Classifier:

- 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

■ Traditional Verification Flow  
■ Proposed Verification Flow



**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)

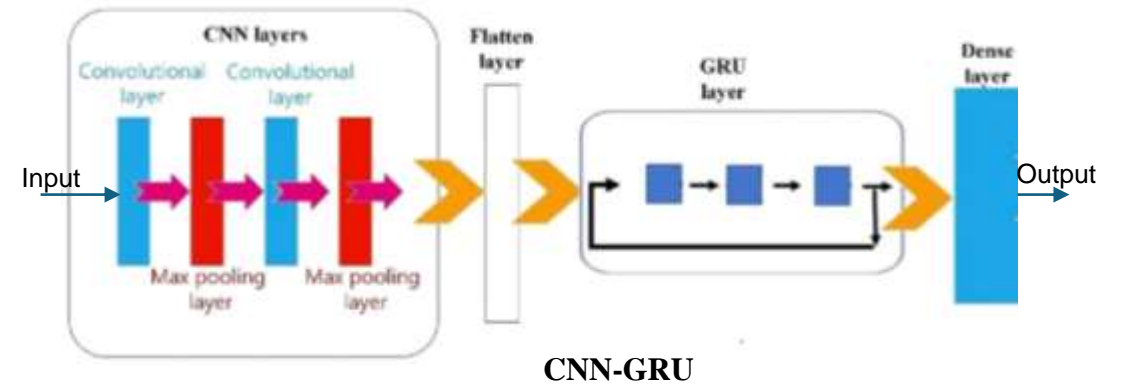
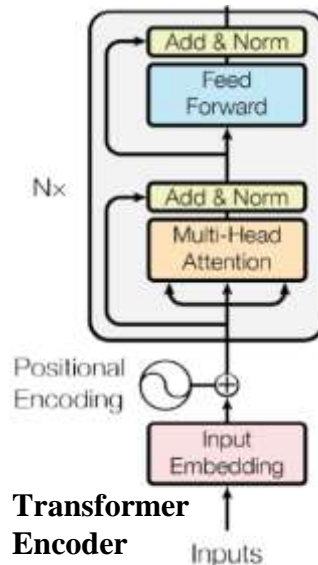
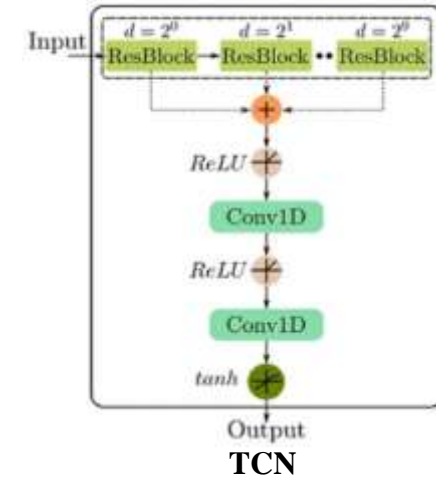
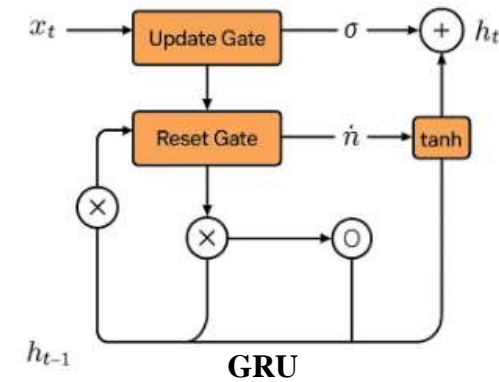
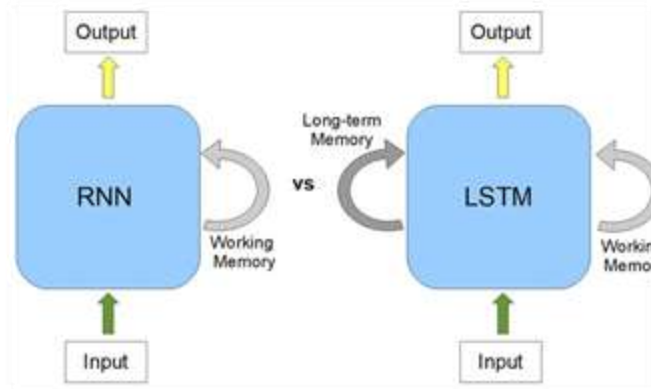
Long Short-term Memory (LSTM)

Gated Recurrent Unit (GRU)

Temporal Convolutional Network (TCN)

Hybrid CNN-GRU Model

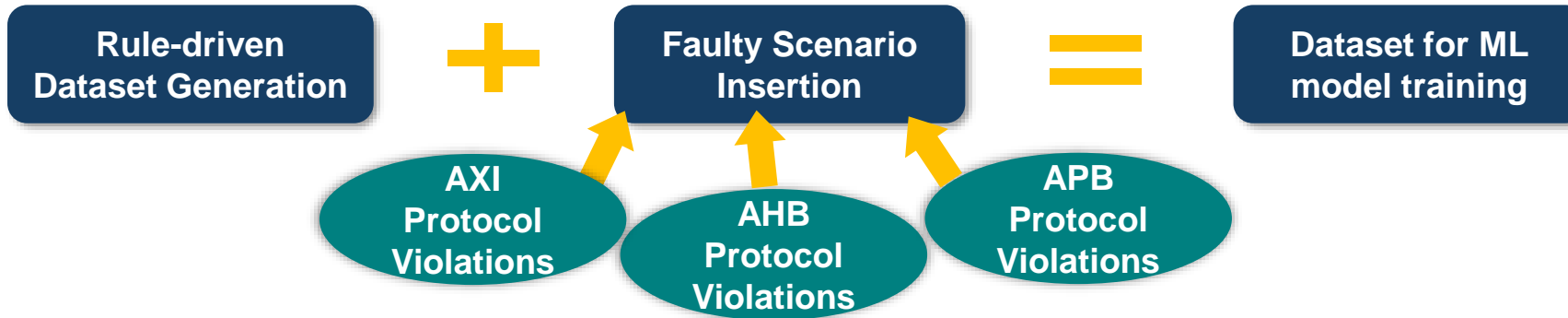
Transformer Encoder



Source: [www.researchgate.net](http://www.researchgate.net)

# Deep Learning Models

- Dataset generation and pre-processing for ML model training

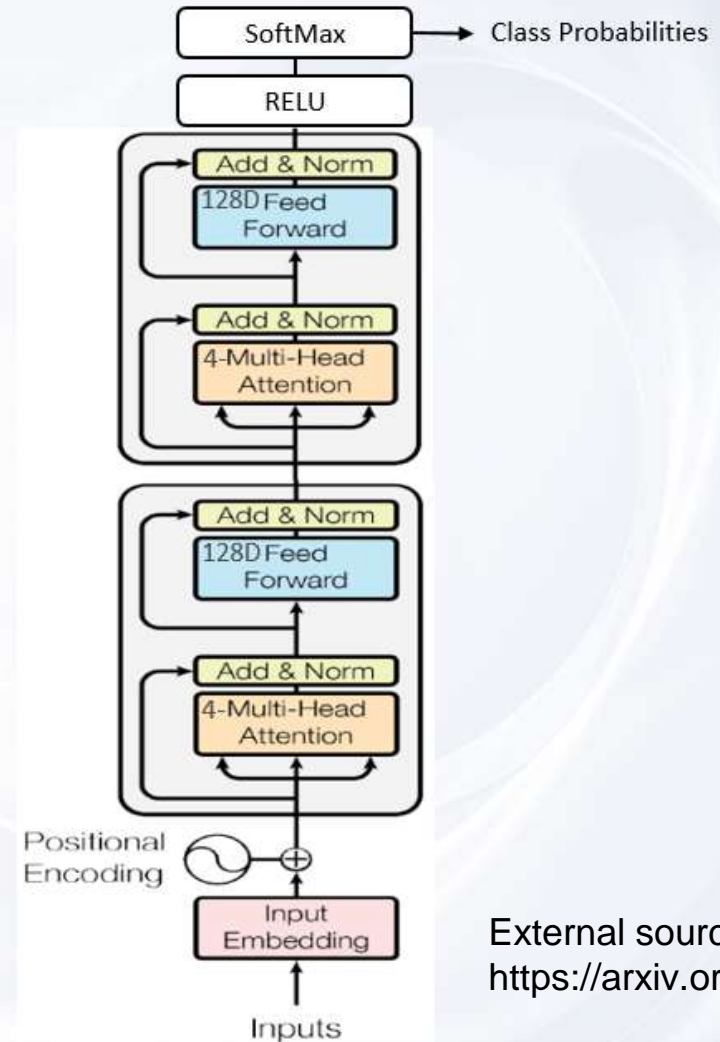


- Toolchains and Libraries



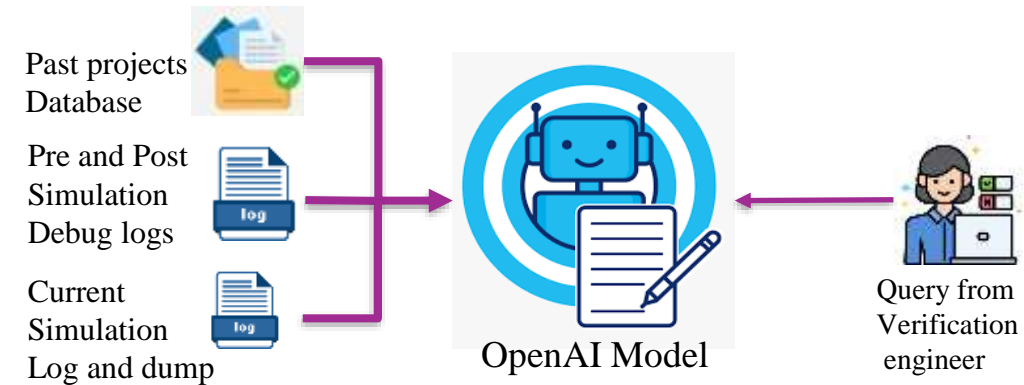
# Transformer Encoder Architecture

- 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.



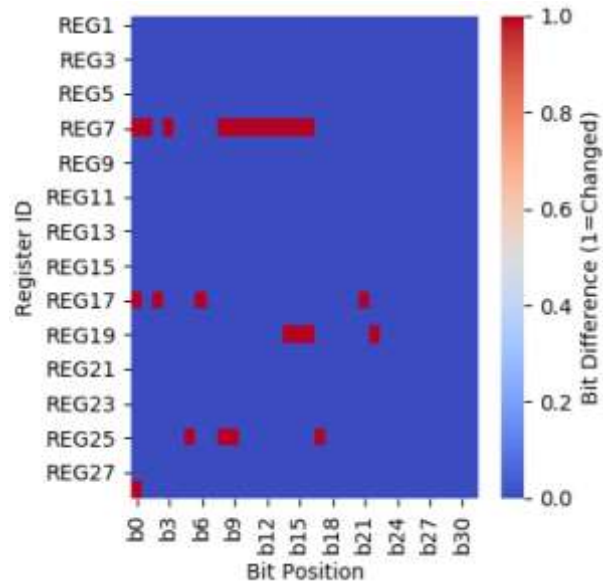
External source:  
<https://arxiv.org/pdf/1706.03762>

# LLM Log analysis and Interactive Debugger

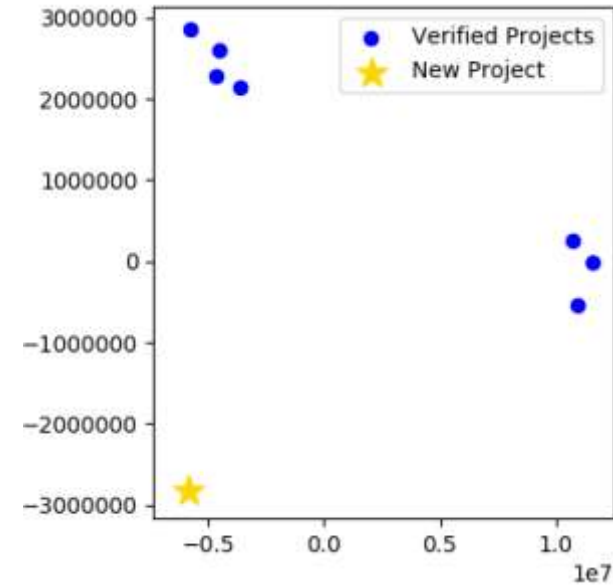


External Source:  
<https://share.google/4yQF98ZUN3aWTVeGi>

# Results : Pre-Simulation Framework

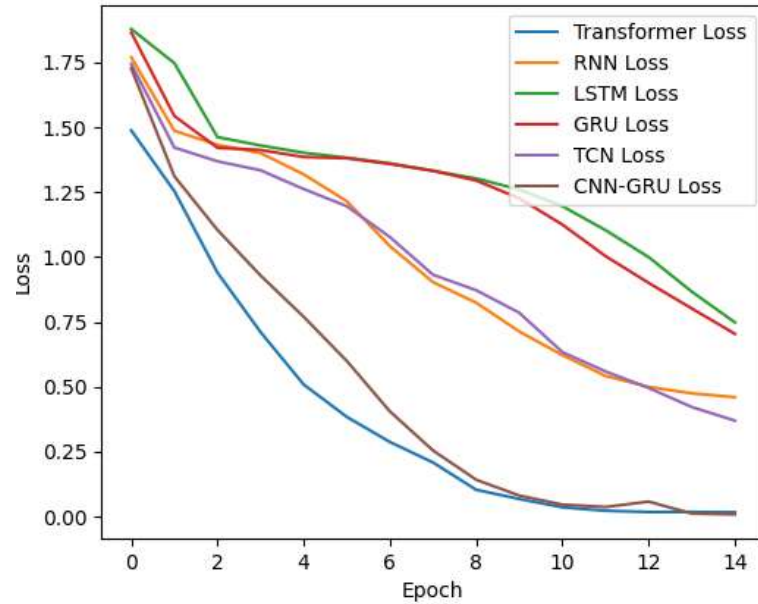


Register Bit-level deviations (new vs baseline)

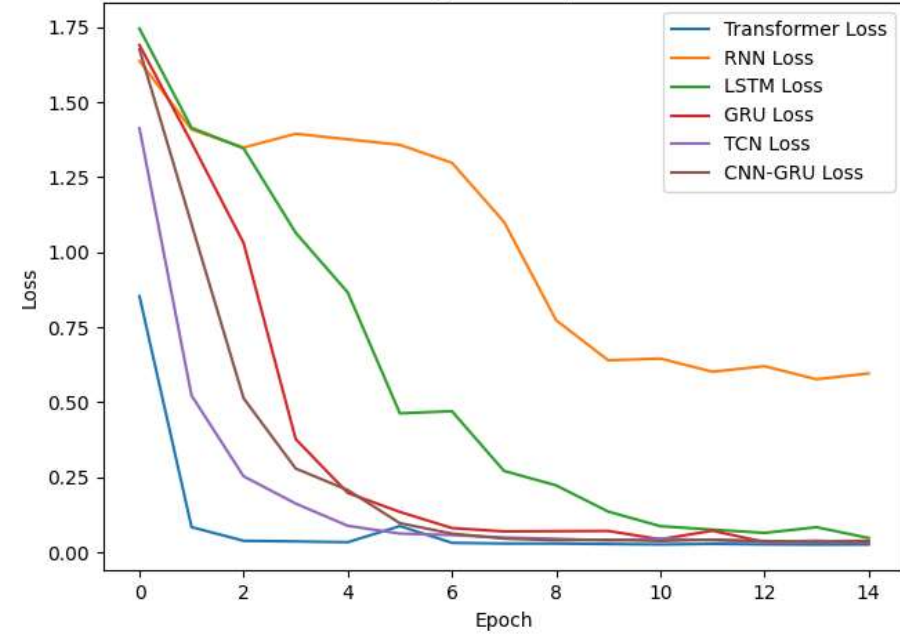


Anomaly Map (Isolation Forest + PCA)

# Results : Post-Simulation Framework

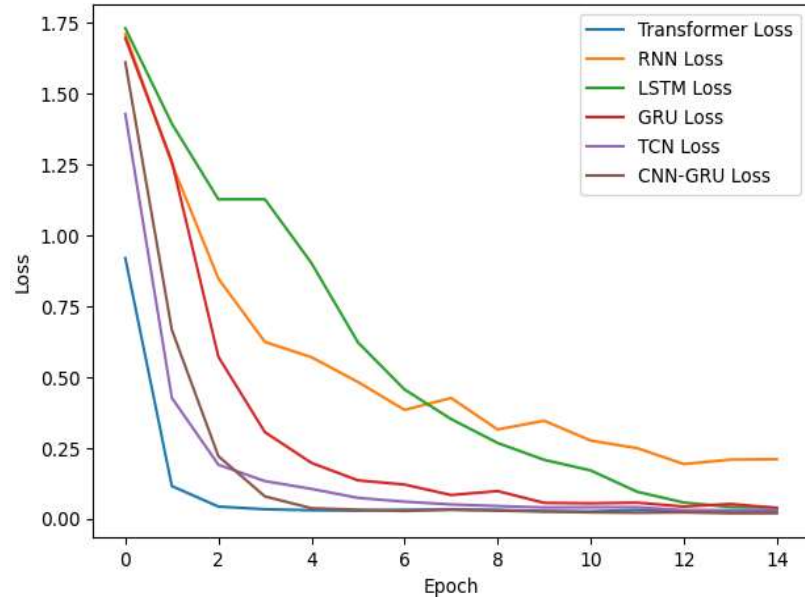


Loss across epochs APB

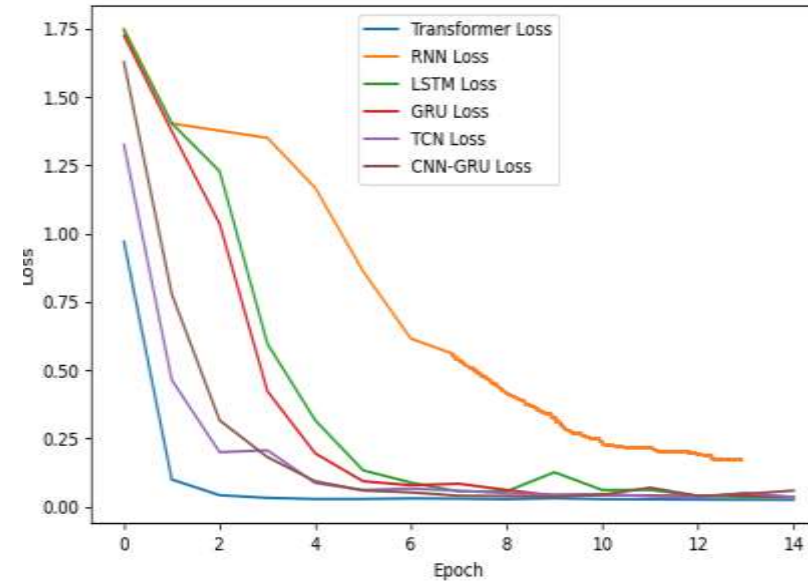


Loss across epochs AHB

# 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

The image displays the 'AI based Design Verification Assist' GUI and a terminal window showing the 'Pre-Simulation.log' output.

**GUI Screenshot:**

- Input File Paths:**
  - Register Specification Sheet: register\_specification\_sheet.xlsx
  - Memory Map: memory\_map.xlsx
  - Security Specification sheet: security\_specification.xlsx
  - Choose testbench type:  SV,  C
  - Testbench Path SV: test.sv
  - Testbench Path C: ss\_seq.c
  - Start pre sim button
- Results:**
  - Pre-Simulation checks:**
    1. Security Configuration
    2. Key SFR Configuration
    3. Address map
    4. Reserved regions of SFR
  - Misconfiguration Count:**
    - 1. Security Configuration: 2
    - 2. Key SFR Configuration: 2
    - 3. Address map: 3
    - 4. Reserved regions of SFR: 4
  - Refresh and More Details buttons

**Pre-Simulation.log Output:**

```
4 Misconfigured-bit Description
5 -----
6 SS31 Security Indicator
7 SS4 Control security
8
9 -----Security Configuration Check Ended-----
10
11 *****
12
13 -----Key SFR configuration check Starts-----
14
15 Misconfiguration Count: 2
16 Base-Register Register-Offset Value
17 -----
18 PMU 0x3a20 0x200
19 APBIF 0x101c 0x3
20
21 -----Key SFR configuration check Ends-----
22
23 *****
24
25 -----Address map Misconfiguration check starts-----
26
27 Misconfiguration count: 3
28 Base-Register
29 -----
30 PMU
31 CMU
32 APBIF
33
34 -----Address map Misconfiguration check Ends-----
35
36 *****
37
38 -----Reserved Regions of SFR check Starts-----
39
40 Write to Reserved region counts: 4
41 Base-Register Register-offset Reserved-bits written
42 -----
43 PMU 0x11862190 0x2
44 PMU 0x11862290 0x2
45 PMU 0x11862b10 0x2
46 PMU 0x11862b90 0x2
47
48 -----Reserved Regions of SFR check Ends-----
```

# Results : Post Simulation Debugger GUI

The screenshot displays the 'AI based Design Verification Assist' GUI with three tabs: 'Pre-Simulation', 'Post-Simulation', and 'Interactive Debugger'. The 'Post-Simulation' tab is active, showing the simulation path and signal analysis options. The 'Design Path for Protocol Interface' section contains input fields for APB, AHB, and AXI interface paths. The 'Results' section shows a table of protocol violations, and the 'Log Analysis' section shows a count of 1 violation.

Protocols	Protocol Violations count
APB	0
AHB	1
AXI	

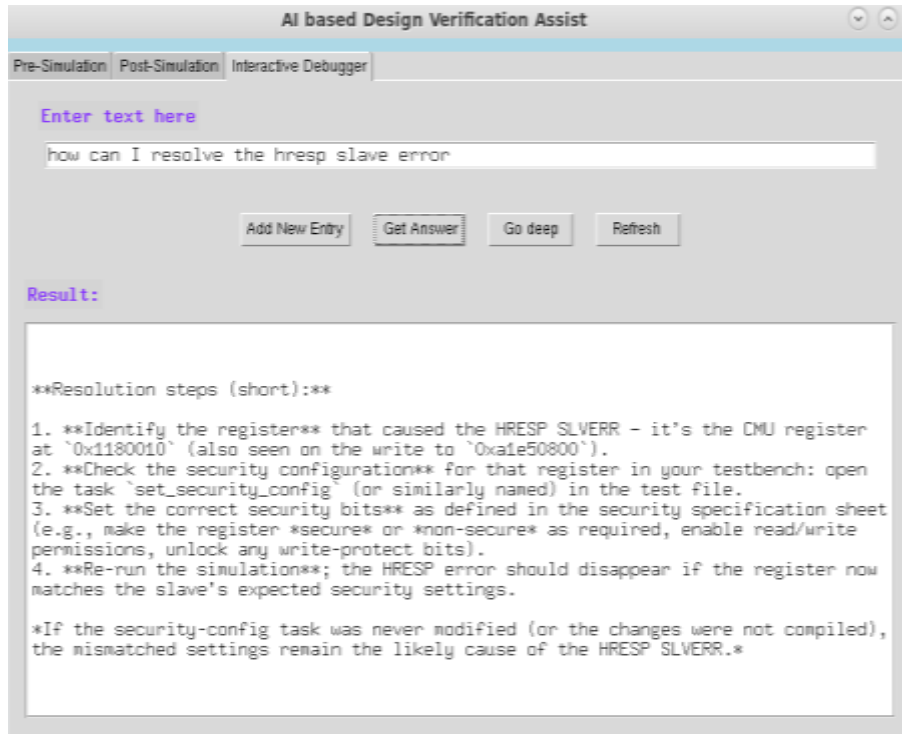
Log Analysis: 1

Buttons: Refresh, More Details

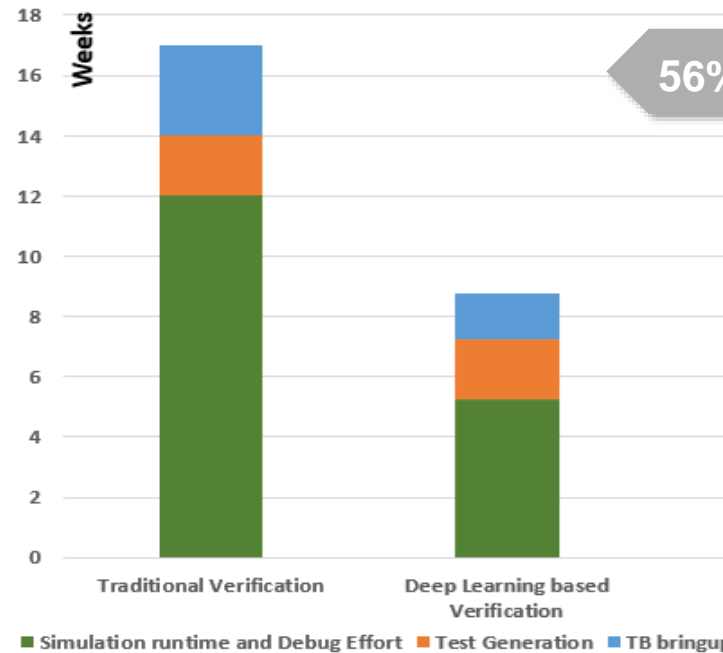
The log window on the right shows the following output:

```
1 -----APB Check Starts-----
2
3 Protocol violation count: 0
4
5 -----APB Check Ends-----
6
7 *****
8
9 -----AHB Check Starts-----
10
11 Protocol violation count: 1
12 Violation    task    transaction_address    Register
13 -----
14 HRESP_ERROR  init    0x1180010    CMU
15
16 -----AHB Check Ends-----
17
18 *****
19
20 -----Log Analysis Starts-----
21
22 Error Count: 1
23 Error: 'UVM_ERROR (cosgt_ahb_mon_c) [DCORE_MON] WRITE, ADDR: 0xale50800, Received ERROR Respo
se(SLVERR)
24
25 Reason: AHB slave error response - incorrect security configuration of respective registers
26
27 Fix: Check if the security settings are correctly set in the task set_security_config in the
respective test file according to security_spec_sheet
28
29 -----Log Analysis Ends-----
30
31 *****
```

# Results



Interactive Debugger GUI



Verification Effort Analysis

# Conclusion and Future Scope

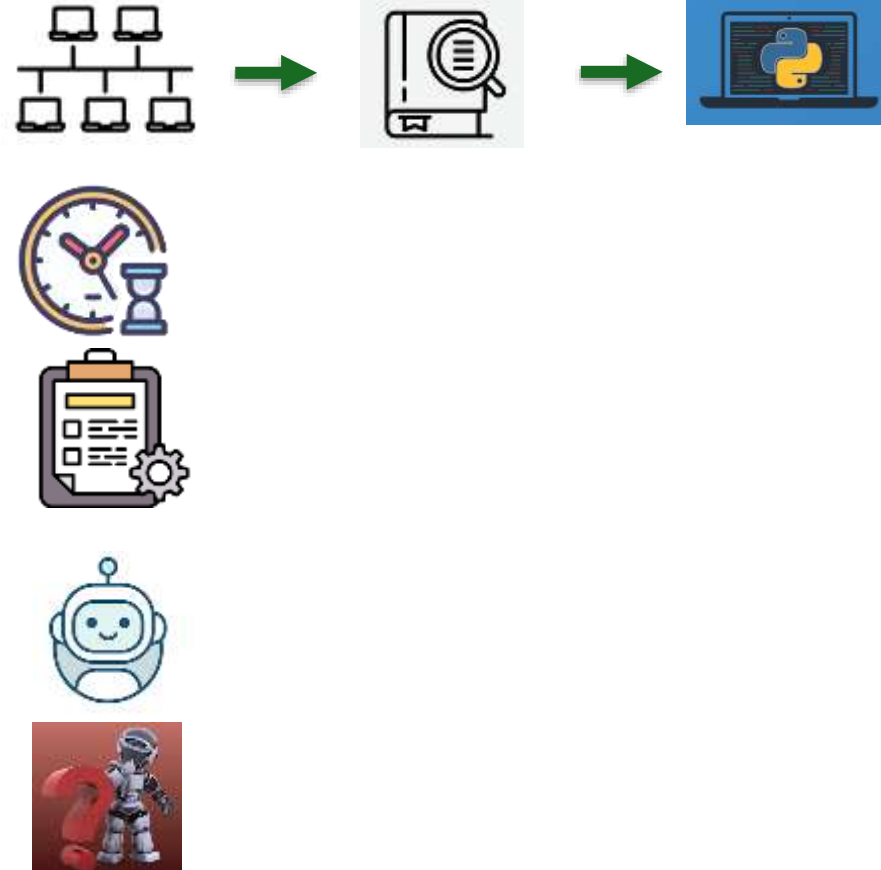
Extended Debug support :  
Bus architecture to dictionary

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!