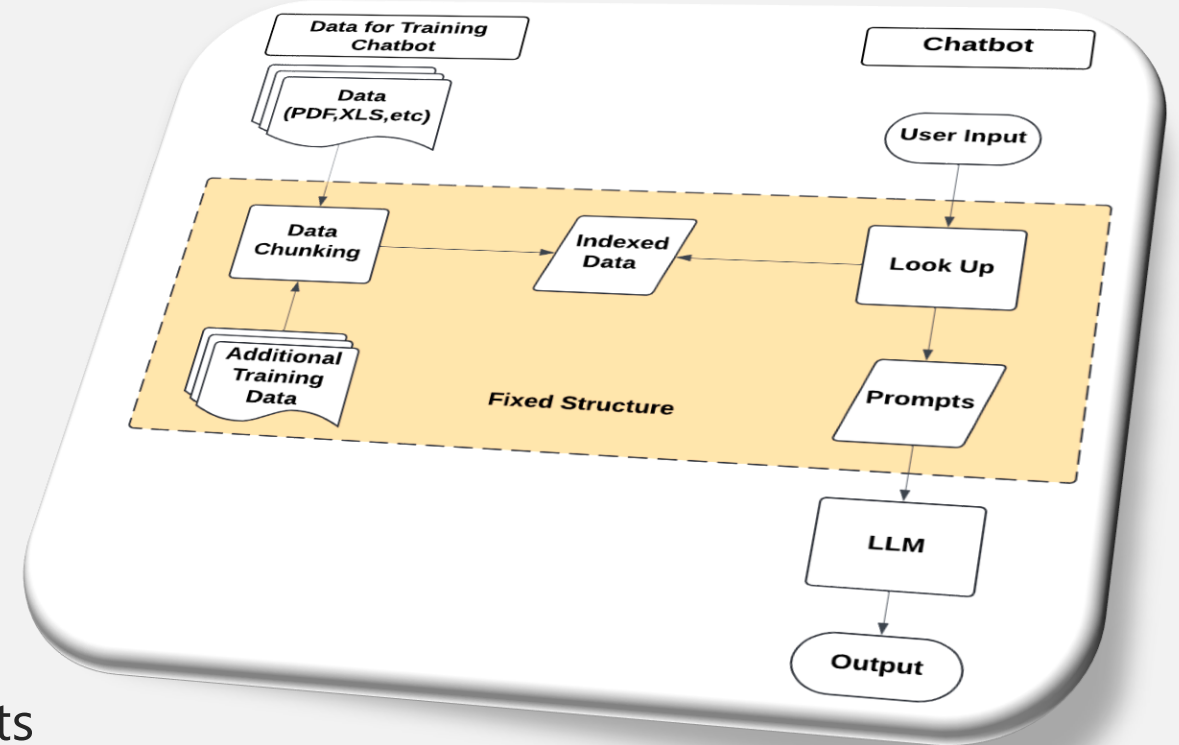


Examples of the Practical Application of AI in Design Verification

Author and Presenter: Dr. Mike Bartley, Founder/CEO, Alpinum Consulting

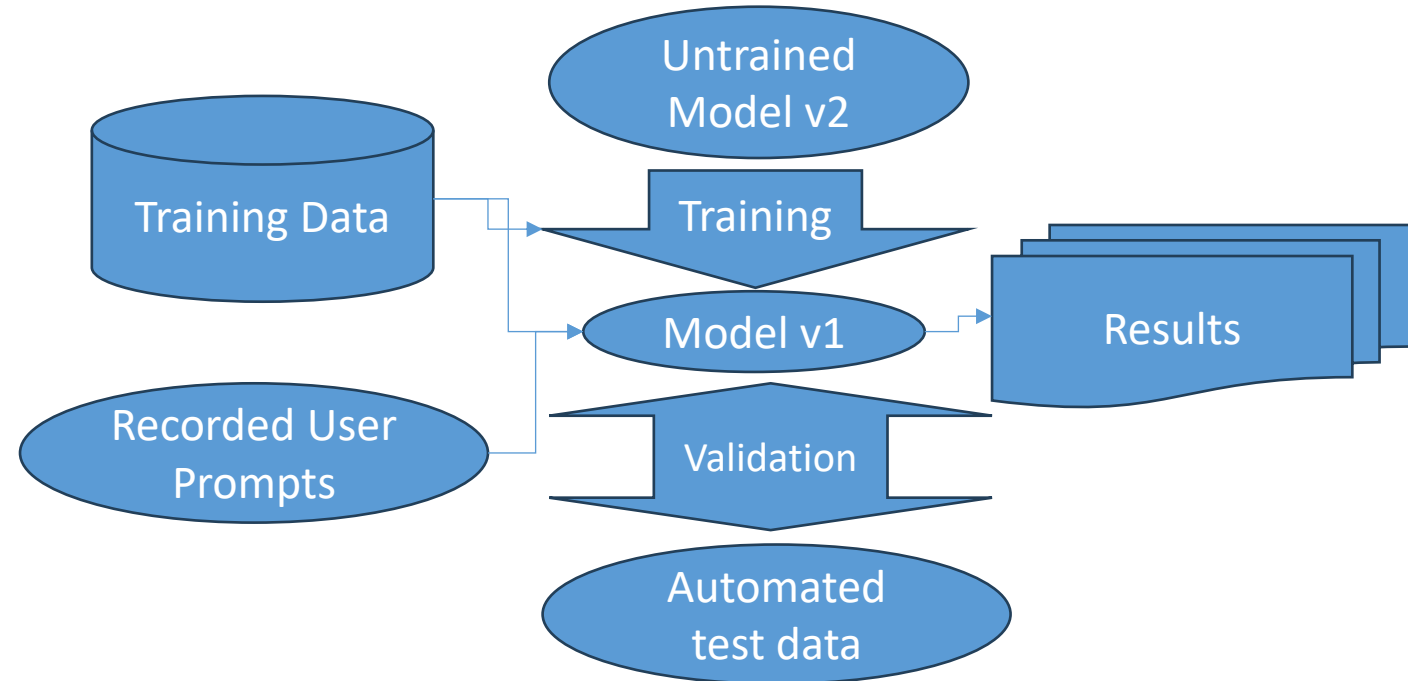
Co-author: Marmik Soni, Manager, Tessolve Semiconductor

- An Effective Company-wide AI Strategy
- AI/ML Experiments in IP Verification
- AI Ecosystem
- Collaboration Opportunity
- Tool 1: UVM AI Training Tool
- Tool 2: Assertify Tool
- Tool 3: Spec Extraction Tool
- Tool 4: RAG Assessment
- Tool 5: UVM TB Automation with Python scripts
- Tool 6: Coverage Closure (Flow)
- Summary



An Effective Company-wide AI Strategy

- Models are replaceable
 - Ensure our data is stored to re-train new model releases
- Maintaining security
 - Different set of infra for various security needs
- Centre supports innovation at the edge
 - Bot infra design/ suggestions
 - Prompt expertise
 - Model selection
 - Model adaptation
 - AI infra migration
 - Custom solution design
 - Access control
 - External tool/ vendor scrutiny



Central AI services:

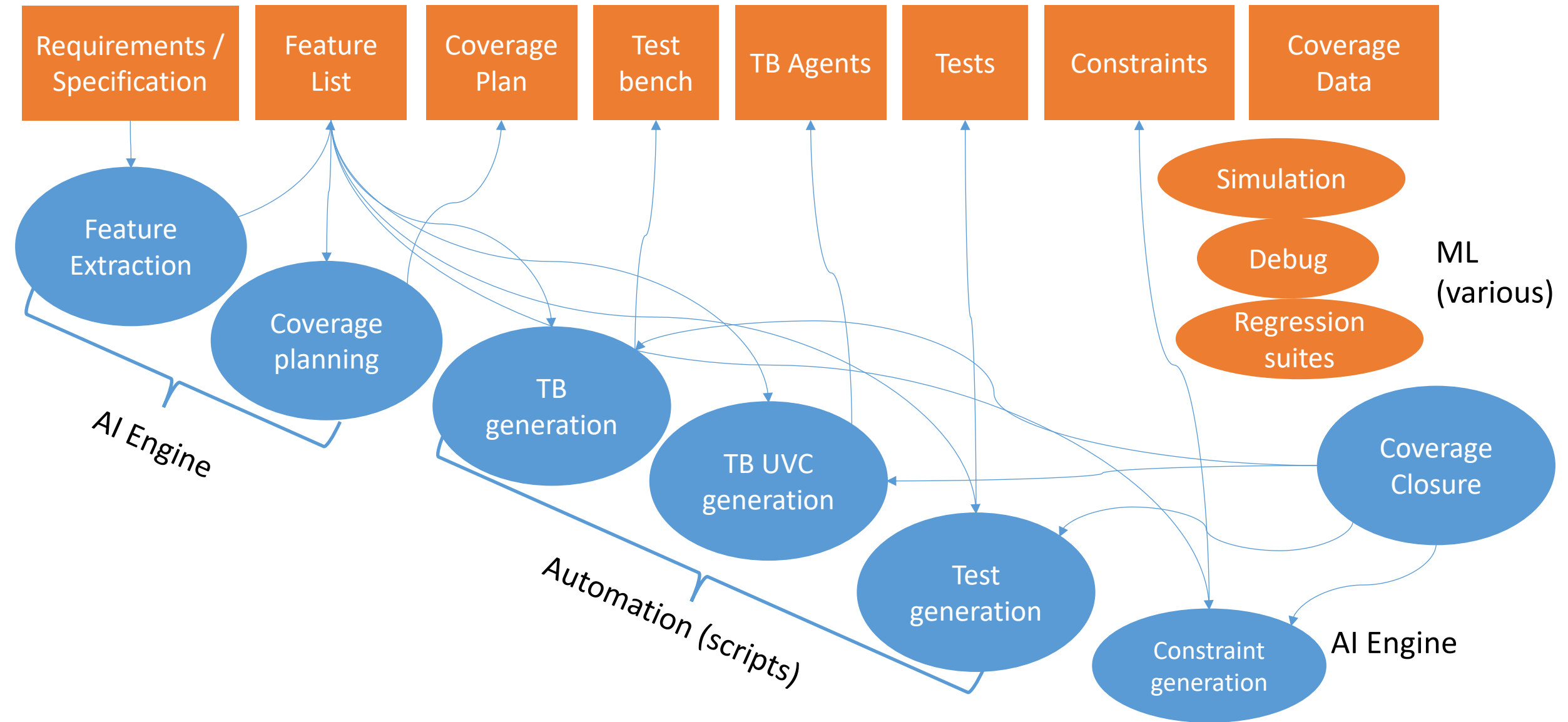
- Services/Guidance/reviews

Central SW services

- SW dev services

AI/ML Experiments in IP Verification

Verif Flow



Knowledge Bots:

- **Features:** Faster to develop & deploy
- **Use case:** With less confidential data, Project CoPilot
- **Security:** Low
- **Cost:** Low
- Reasonable accuracy,
- **Project:** UVM Chatbot, FUSA Chatbot

Build: Cloud Deploy: Cloud/ Hosting Services

- **Feature:** Faster develop & deploy
- **Use case:** Client projects, highly confidential data.
- **Security:** High
- **Cost:** High

Build: Local Deploy: Local

- **Features:** Slower to develop & slower to deploy
- **Use case:** For highly confidential data
- **Security:** High
- **Cost:** High
- **Challenge:** Less Adaptability, less Scalability, Upfront Cost

AI Apps

- DV Spec Analysis Tool
- DV AI TB Tool
- Assertify Tool
- Domain/ Project based AI models
- PDK AI Training Tool
- AI Code Copilot

Tool 1: UVM AI Training Tool

Organized Content

- Follows planned flow

Quick Topics

- Most important topics' explanation

MCQs

- Test, Evaluation, weak concept suggestions

FAQs

- Clarifies on common important doubts

Q&A

- ANY question support

About

- Tool briefing for effective use

Benefits:

- Self guided and self paced flow
- Analytics & custom topic suggestions
- Ask anything about UVM
- Improved accessibility
- Reduced training time
- Scalable for other domains

Tool 1: UVM AI Training Tool

Welcome, marmik soni!

Logout

Navigation

Organized Content

Quick Topics

Generate MCQs

Top 10 FAQs

Q&A Chat

Chat History

About



Tessolve UVM AI Training Tool

Q&A Chat



Explain UVM Sequencer please!



UVM Sequencer

Introduction

In Universal Verification Methodology (UVM), the sequencer is a key component responsible for controlling the flow sequences that generate complex stimulus and the drivers that convert these stimuli into pin-level activities.

Role of the Sequencer

Developed by AI CoE team

Tool 1: UVM AI Training Tool

Welcome, marmik soni!

Logout

Navigation

Organized Content

Quick Topics

Generate MCQs

Top 10 FAQs

Q&A Chat

Chat History

About



Tessolve UVM AI Training Tool

Generate MCQs

Select a topic

UVM Components

Generate MCQs

MCQs for UVM Components:

Question 1: What is the primary class from which the `uvm_component` class inherits?


Select an option:

- ☒ (A) `uvm_report_object`
- ☐ (B) `uvm_sequence`
- ☐ (C) `uvm_driver`
- ☐ (D) `uvm_monitor`

Question 2: Which of the following `uvm_component` classes encapsulates sub-components for sequence communication with the `uvm_driver`?

Select an option:

- ☒ (A) `uvm_subscriber`

 Developed by AI CoE team

Tool 2: Assertify Tool

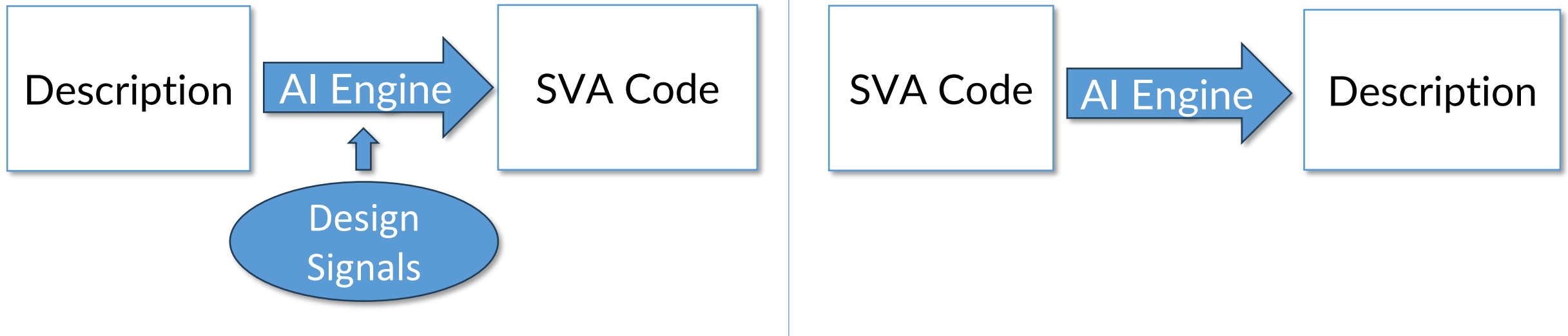
Problems around Assertions are:

- | Growing Complexity in Verification
- | Human Errors

- | Time intensive asks
- | Lack of Consistency and Skill Gap

Use case:

1. Generate Assertions from spec/ engineer's simple description using AI models.
2. Generate description from the SVA code.



Tool 2: Assertify Tool (Description -> SVA)

Description	SVA Code
<ul style="list-style-type: none">Ensure that signal a is always high at every rising edge of the clock clk.	<ul style="list-style-type: none">property a_always_high; @(posedge clk) a == 1; endproperty
<ul style="list-style-type: none">Ensure that signal_x changes its value on every clock cycle.	<ul style="list-style-type: none">property signal_x_changes; • @(posedge clk) signal_x != signal_x[1]; • endpropertysignal_x_changes_check: assert property (signal_x_changes);
<ul style="list-style-type: none">check if sig_a is down , sig_b may only rise for one cycle before the next time that sig_a is asserted.	<ul style="list-style-type: none">property sig_b_rise_after_sig_a_down; • @(posedge clk) (sig_a == 0) -> ##1 (sig_b == 1) ##1 (sig_b == 0) -> (sig_a == 1); • endpropertysig_b_rise_after_sig_a_down_check: assert property (sig_b_rise_after_sig_a_down);

Tool 2: Assertify Tool (SVA -> Description)

SVA	Description
<ul style="list-style-type: none">assert property (@(posedge clk) a -> ##1 (b throughout [*5]));	This assertion checks that whenever signal 'a' is true at a rising clock edge, signal 'b' must remain true for the next 5 consecutive clock cycles.
<ul style="list-style-type: none">assert property (@(posedge clk) disable iff (resetsn == 0) (reset == 1));	The assertion ensures that whenever the system is not in a reset state (i.e., resetsn is high) , the reset signal must be high. If resetsn is low, the check is disabled, allowing the system to remain in reset without triggering this assertion.
<ul style="list-style-type: none">@(posedge ACLK) disable iff (!ARESETn) AWVALID && AWREADY) -> ##[0:\$] (WVALID && WREADY && WLAST) [*1:\$]; endproperty	This SystemVerilog assertion checks that whenever AWVALID and AWREADY are both true at the rising edge of ACLK, then eventually WVALID, WREADY, and WLAST will all be true at the same time, at least once, within an unbounded number of clock cycles, provided ARESETn is not deasserted.

Tool 2: Assertify Tool - Signal match analysis

#	Approach	Top signal	Assertify tool input english description signal	Signal in generated SVA
1	Case mismatch	axi_awvalid	AXI_AWVALID	axi_awvalid
		axi_awready	AXI_AWREADY	axi_awready
2	Additional character	axi_awvalid	AXIVALID	axi_awvalid
		axi_awready	AXIREADY	axi_awready
3	Partial match	axi_awvalid	AXIVLD	axi_awvalid
		axi_awready	AXIRD	axi_awready
4	Partial match	axi_awvalid	valid	axi_awvalid
		axi_awready	ready	axi_awready
5	Highly mismatch	axi_awvalid	VLD	axi_awvalid
		axi_awready	RDY	axi_awready

Tool 2: Assertify Tool

Upload TOP Level PDF File:

Choose a PDF file

Drag and drop file here
Limit 1GB per file • PDF

Browse files

axi_top.pdf
36.7KB

SystemVerilog Assertion to English

SystemVerilog Assertion Code

```
systemverilog
property awvalid_protocol;
  @(posedge clk) disable iff (!rst_n)
    axi_awvalid |-> ##[1:$] axi_awready;
endproperty

assert property (awvalid_protocol);
```

Convert to English

Code Description

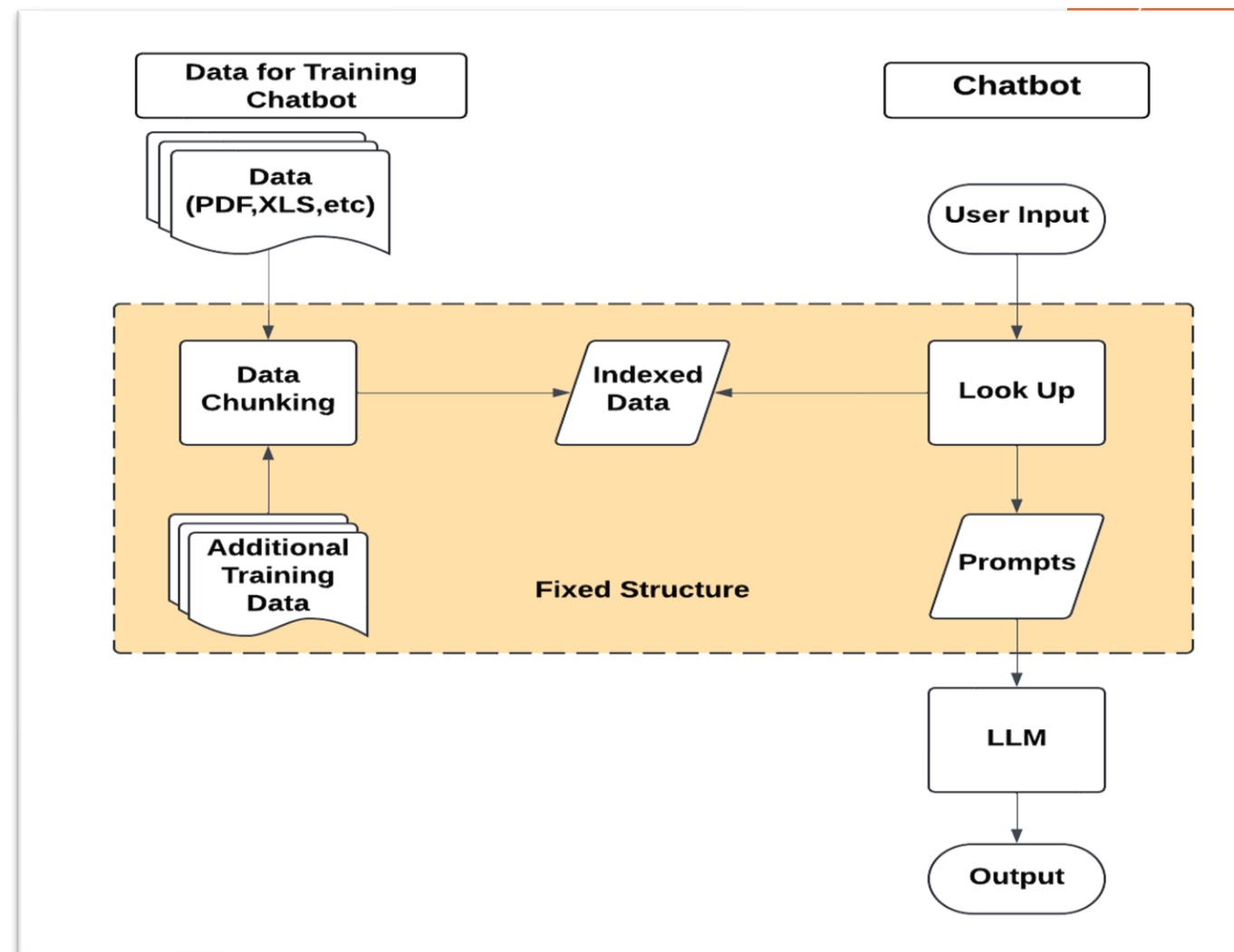
The `SystemVerilog` assertion ``awvalid_protocol`` checks that whenever the ``axi_awvalid`` signal is asserted (indicating that the write address is valid), the ``axi_awready`` signal must be asserted at some point in the future, ensuring that the write address handshake is completed. This must hold true on every positive edge of the ``clk`` signal, provided the ``rst_n`` signal is not asserted (i.e., the system is not in reset).

Tool 3: Spec Extraction Tool

- RAG (Retrieval Augmented Generation) Tool for DV Spec Extraction

- Inputs: Project Spec, Training Data,
- Prompt (Character)
- User Prompt
- LLM Intelligence
- Output

Flow



Spec PDF
Review



Model
benchmarking



Prompt and
response
recording



Filtering by
manual
validation



Excel sheet with
confidence
score

Tool 3: Spec Extraction Tool

View PDF:

AXI4_specification 1.pdf

AMBA® AXI and ACE Protocol
Specification

AXI3, AXI4, and AXI4-Lite®
ACE and ACE-Lite

ARM

Copyright © 2020, 2021, 2019, 2017 ARM. All rights reserved.
ARM114 (2020-2021) (D1171)

AMBA AXI and ACE Protocol Specification
AXI3, AXI4, and AXI4-Lite
ACE and ACE-Lite



Hello, Marmik Bhupendrakumar Soni

Select Model:

GPT-4o

o1-preview

Mistral Large

Codestral

Tessolve Chatbot v6.0



please generate detailed AXI4 feature list with different bus channels, unaligned data transfers, burst based transactions with more details on burst type, burst size and length from the uploaded pdf

MS

Tessolve Chatbot v6.0



please generate detailed AXI4 feature list with different bus channels, unaligned data transfers, burst based transactions with more details on burst type, burst size and length from the uploaded pdf



Detailed AXI4 Feature List

1. AXI4 Bus Channels

The AXI4 protocol defines several independent transaction channels to facilitate high-performance and high-frequency system designs. These channels are:

- **Read Address Channel:** Carries control information describing the nature of the data to be transferred for read operations.
- **Read Data Channel:** Transfers data from the slave to the master during read operations.
- **Write Address Channel:** Carries control information describing the nature of the data to be transferred for write operations.
- **Write Data Channel:** Transfers data from the master to the slave during write operations.
- **Write Response Channel:** Signals the completion of the write transfer from the slave to the master.

These channels allow for the separation of address/control and data phases, which helps in achieving high-frequency operation without using complex bridges.

2. Unaligned Data Transfers

The AXI protocol supports unaligned data transfers using byte strobes. This feature is crucial for accessing mixed-endian data structures in a single memory space. The protocol uses a byte-invariant endianness scheme, which ensures that any multi-byte element in a data structure:

- Uses the same continuous bytes of memory, regardless of the endianness of the data.
- The order of bytes in memory is determined by the endianness, indicating whether the first byte in memory is the most significant byte (MSB) or the least significant byte (LSB) of the element.
- Any byte transfer to a given address passes the eight bits of data on the same data bus wires to the same address location, regardless of the endianness of any data element.

3. Burst-Based Transactions

The AXI protocol is burst-based and defines three burst types:

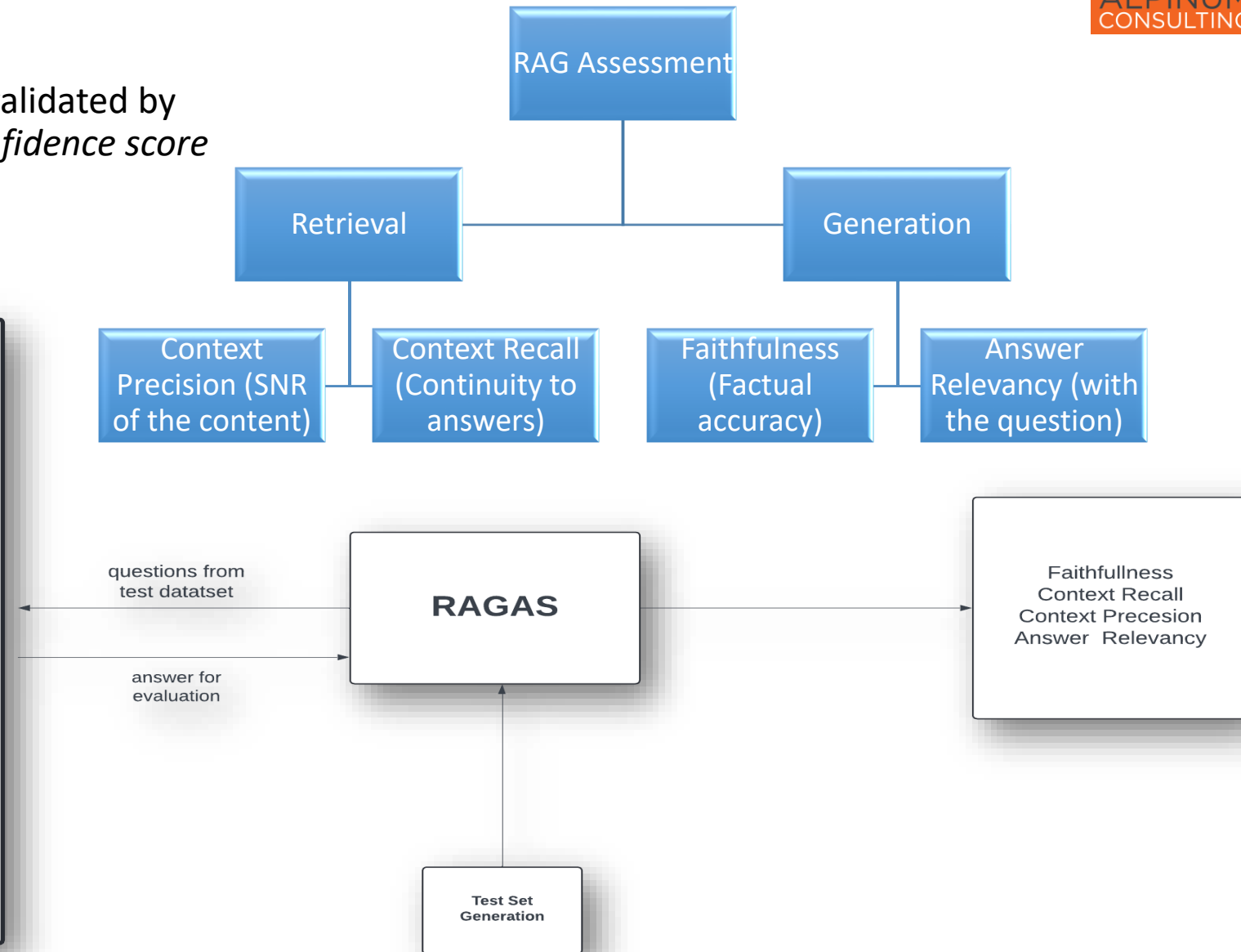
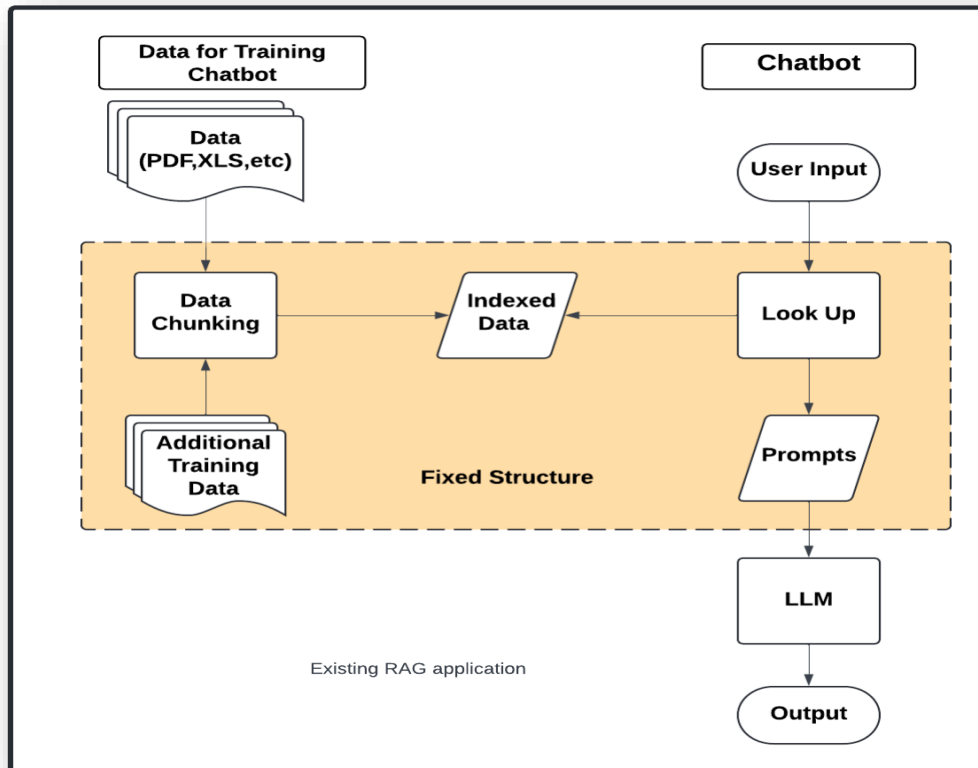
- **Fixed Burst:** The address remains constant for every transfer in the burst.
- **Incrementing Burst (INCR):** The address increments by the size of the transfer for each transfer in the burst.
- **Wrapping Burst:** The address increments like an incrementing burst but wraps around to a lower address when a wrap boundary is reached.

Burst Size



Tool 4: RAG Assessment

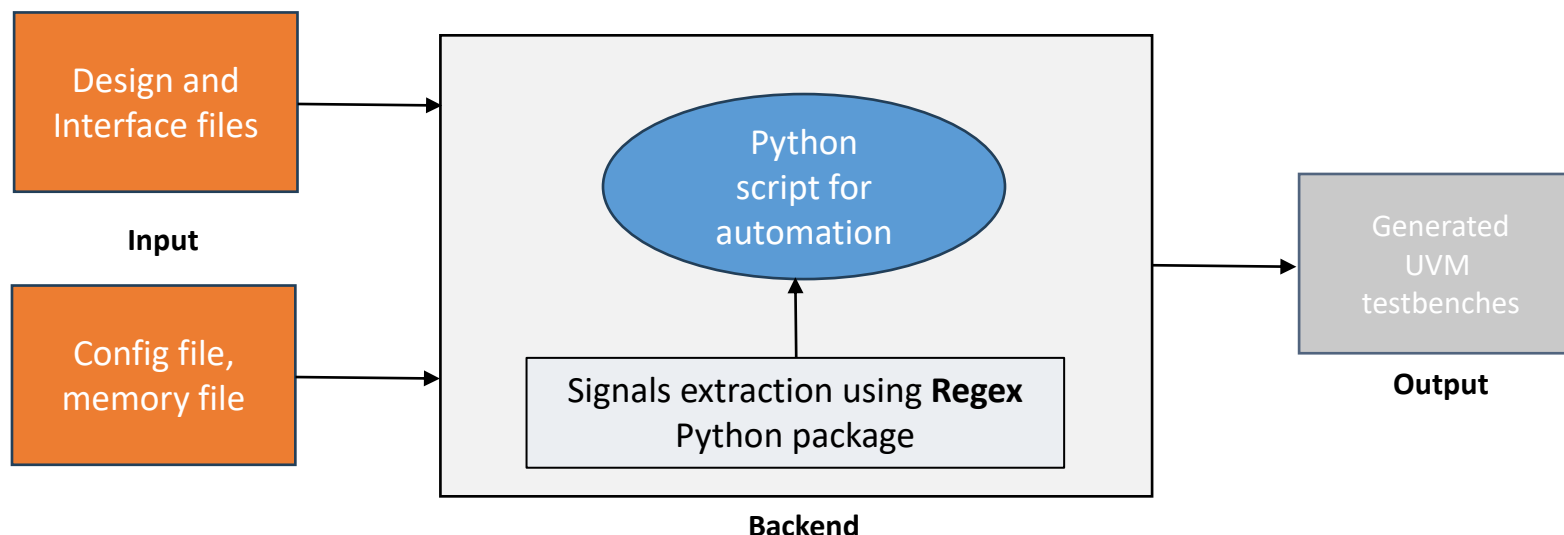
- Manual validation
 - The generated content is manually validated by the engineers and tagged with a *confidence score*



Tool 4: RAG Assessment

question	answer	contexts	ground_truth	faithfulness	answer_relevancy	context_recall	context_precision
What role do integrated circuits (ICs) play in...	Integrated circuits (ICs) are used in virtuall...	[Applications in Electronics\nThe ability to c...	Integrated circuits (ICs) are fundamental buil...	1.000000	0.946481	1.00	1.0
What are the potential future developments in ...	The future of semiconductors will likely invol...	[Future Directions\nThe future of semiconducto...	The future of semiconductors is poised for exc...	1.000000	0.992269	1.00	1.0
How does Moore's Law relate to the development...	Moore's Law predicts that the number of transi...	[The Role in Technological Advancement\nThe co...	Moore's Law predicts that the number of transi...	0.833333	0.936722	1.00	1.0

Tool 5: UVM TB Automation with Python scripts



Input

- User should provide **Design, Interface, Config (optional), Memory (optional)** files.

Backend System

- The UVM standard code will be saved in **python script**.
- Python script will extract signals from design and interface files.
- By utilizing the extracted signals, the test bench components gets created.

Output

- The UVM testbench components in **System Verilog** format will get saved.

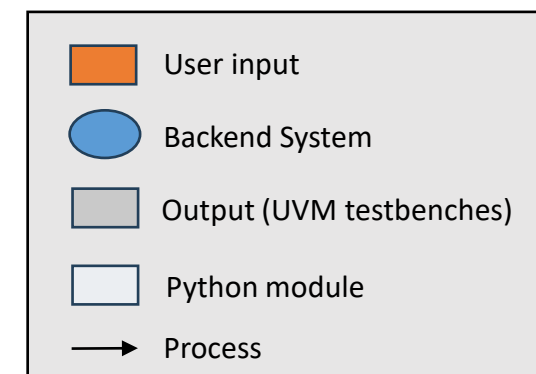
• **FIFO**

• **Single Port RAM**

• **Dual Port RAM**

• **AXI....**

TESSOLVE
A HERO ELECTRONIX VENTURE



AI for Config files, ML for parameter optimization

cādence CONNECT
EUROPE

Tool 5: UVM TB Automation with Python scripts

The screenshot displays a UVM testbench environment with two code editors and a terminal window.

Left Editor: SPRAM_monitor.sv

```
1
2 class SPRAM_monitor extends uvm_monitor;
3     `uvm_component_utils(SPRAM_monitor)
4
5     uvm_analysis_port#(SPRAM_seq_item) analysis_port;
6
7     // Virtual interface handle
8     virtual mem_if myvinf;
9
10    SPRAM_seq_item collect_port;
11
12    function new(string name = "SPRAM_monitor", uvm_component parent)
13        super.new(name, parent);
14        collect_port = new();
15        analysis_port = new("analysis_port", this);
16    endfunction :new
17
18    virtual function void build_phase(uvm_phase phase);
19        super.build_phase(phase);
20        if (!uvm_config_db#(virtual input_if)::get(this, "", "myvinf",
21            `uvm_fatal("NO myvinf ", "Virtual interface not found")
22
23        analysis_port=new("analysis_port",this);
24        collect_port=SPRAM_seq_item::type_id::create("collect_port",
25    endfunction :build_phase
26
27    virtual task run_phase(uvm_phase phase);
```

Right Editor: SPRAM_agent_passive.sv

```
1
2 class SPRAM_agent_passive extends uvm_agent;
3     SPRAM_monitor mon;
4
5     `uvm_component_utils(SPRAM_agent_passive)
6
7     function new(string name="SPRAM_agent_passive", uvm_compone
8         super.new(name, parent);
9     endfunction
10
11    virtual function void build_phase(uvm_phase phase);
12        super.build_phase(phase);
13        if(get_is_active() == UVM_PASSIVE) begin
14            mon = SPRAM_monitor :: type_id::create("mon", this)
15        end
16    endfunction
17
18 endclass
19
```

Terminal Window

```
[]
data_out_control_signal: oe
----- True
UVM test bench components for SPRAM design have been generated successfully.
>[]
data_out_control_signal: oe
----- True
UVM test bench components for SPRAM design have been generated successfully.
----- True
```

Tool 5: : UVM TB Automation - Benefits

- ~90% time reduction in the efforts needed for UVM TB generation
- Higher accuracy
- Consistent process
- Less chances of human errors

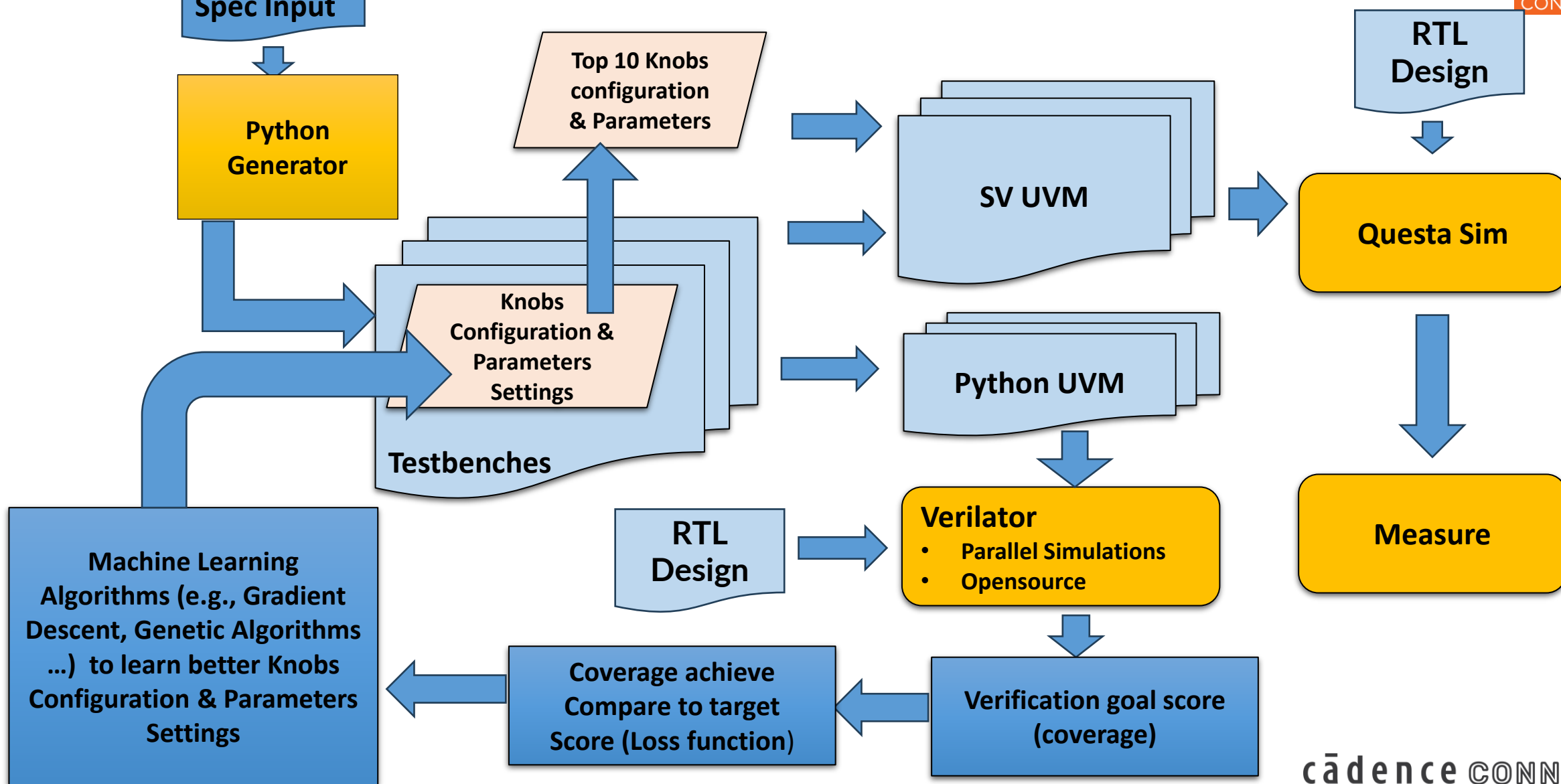
Tool 5: UVM TB Automation - Challenges

- New designs may bring unseen challenges

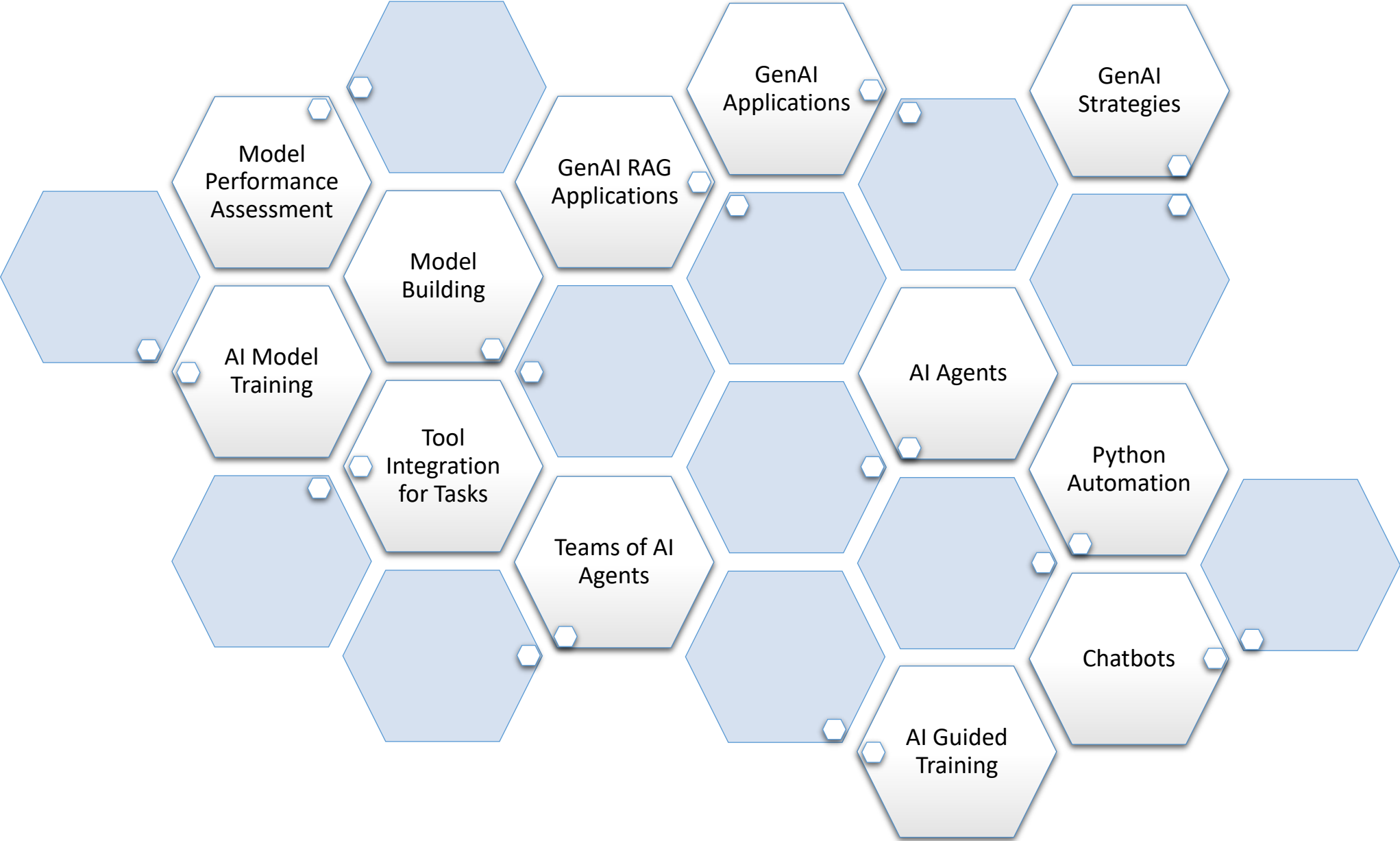
Tool 5: UVM TB Automation - Roadmap

- AI to generate config file
- Generated TB is completely in control by parameters
 - Use of AI to optimize parameters

Tool 6: Coverage Closure (Flow)



Collaboration with AI Research





Register Extraction



Test Plan Generation



Testcases



Specification Analysis



Coverage Planning



Assertions



Cover points



Testbench
Automation



Feature Description



Unit Test Bench
Generation



AI Agents for Tasks



Tool Integration &
Automated Validation

Thank You!

Any questions, please reach out

Mike Bartley, Founder/CEO,
Alpinum Consulting
mike@alpinumconsulting.com
+44 (0) 7796 307958

Visit our website



Book a call



LinkedIn



Marmik Soni, Technical Manager,
Tessolve Semiconductor Pvt Ltd.
marmikbhupendrakumar.soni@tessolve.com
+91 90339 67780

[Marmik Soni](#) | [LinkedIn](#)