AI enabled coverage convergence acceleration for high-performance Visual Analytics core

Azhar Ahammad, Engineer Staff
Akshay Jain, Engineer Senior Staff
Arun Magha Raja, Engineer Senior

Qualcomm India Private Limited

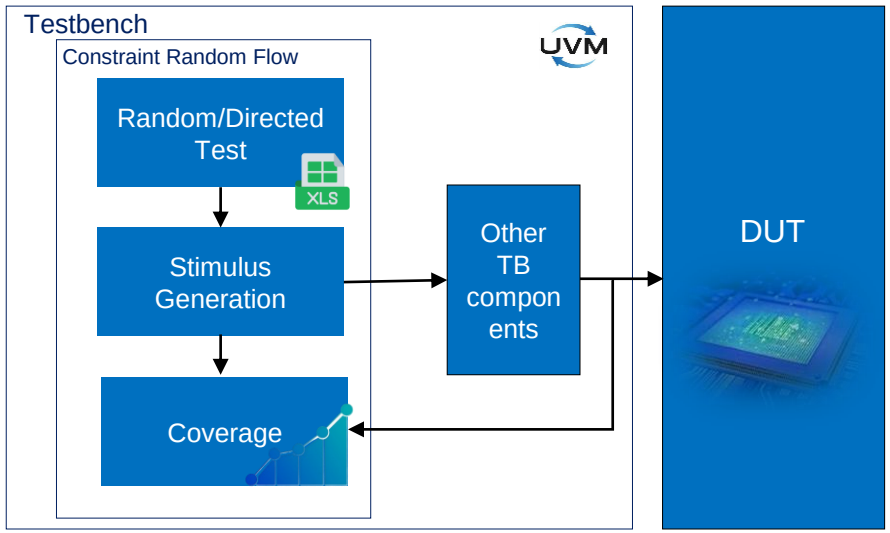
Agenda

- Introduction
- Problem statement
- Proposed solution
- Result
- Summary/Conclusion

Introduction

Introduction

- Importance of Visual analytic IP
- DV process for a new design involves:
  - Creating new testbench
  - Creating an exhaustive/robust functional coverage model
  - Multiple levels of debugs
  - Functional coverage closure for approx. 10k-80k bins
- ".Xls" based constraint random stimulus generation flow for directed CRV testing.

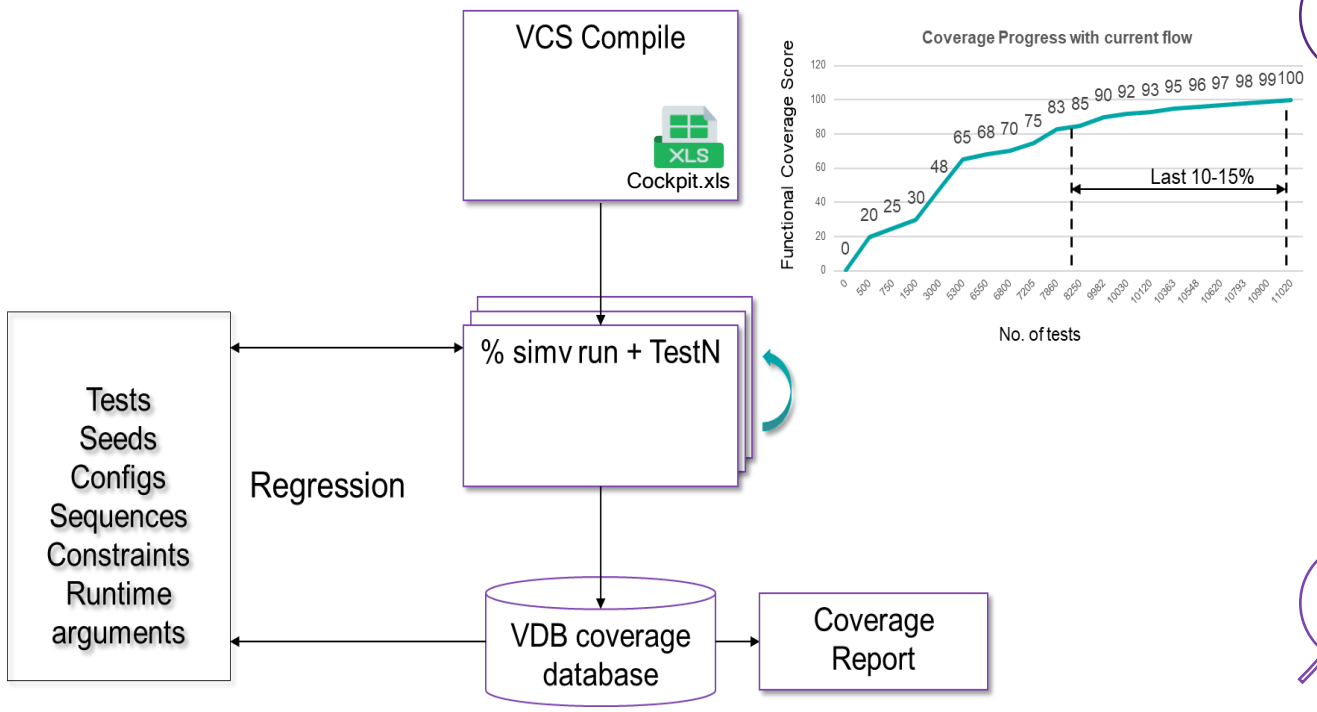


Standard Testbench flow

Visual Analytic IP Applications

Problem Statement

Problem Statement



Typical Regression flow

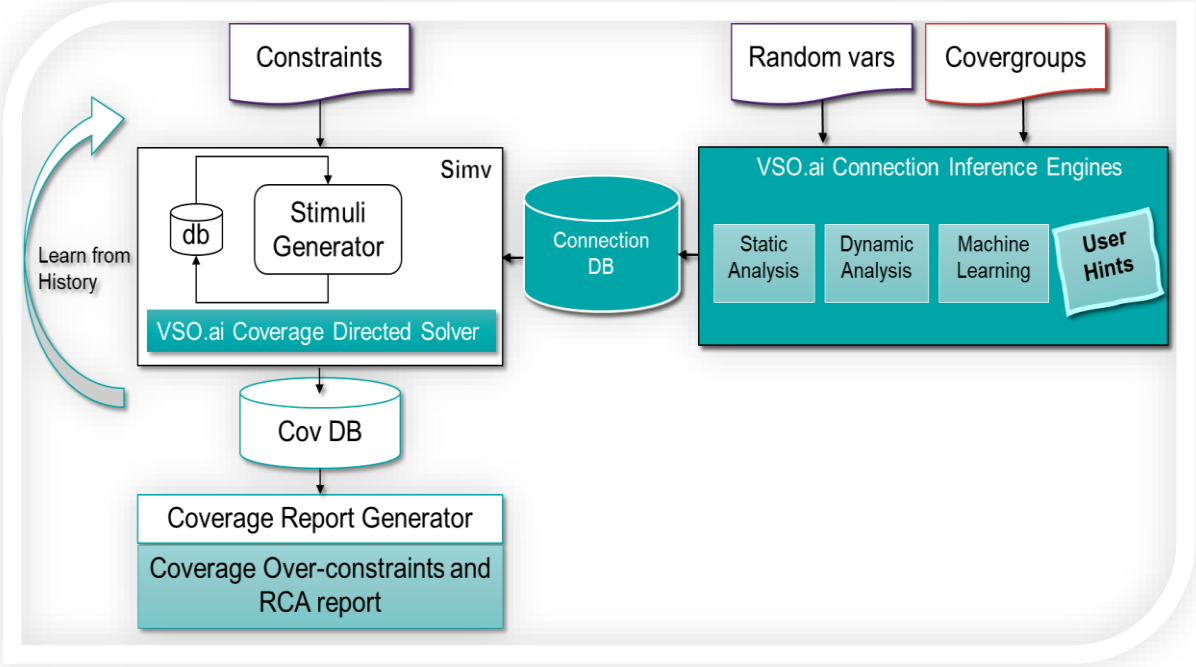
- Challenges in covering Last 10-15% bins
- Less controllability on the stimulus leads to redundant test
- Number of verification cycles to meet the coverage goals
- Reduce xls flow dependencies
- Challenges in achieving delta coverages

Requirement for a coverage accelerator method that requires minimal modifications to the testbench and the testbench setup files

Proposed Solution

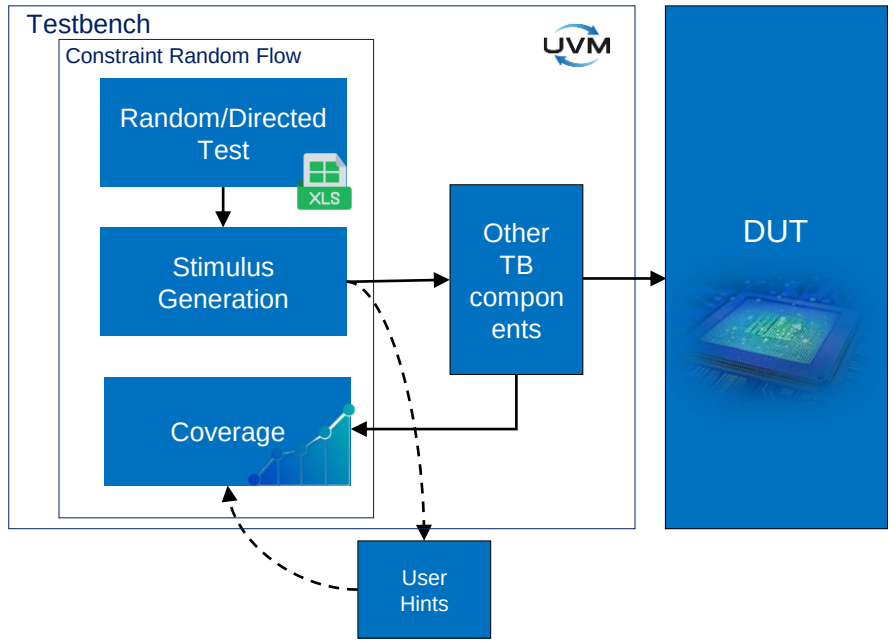
Proposed Solution

- Integration of Verification space optimization(VSO.ai)



VSO.ai working

- Usage of user hint file

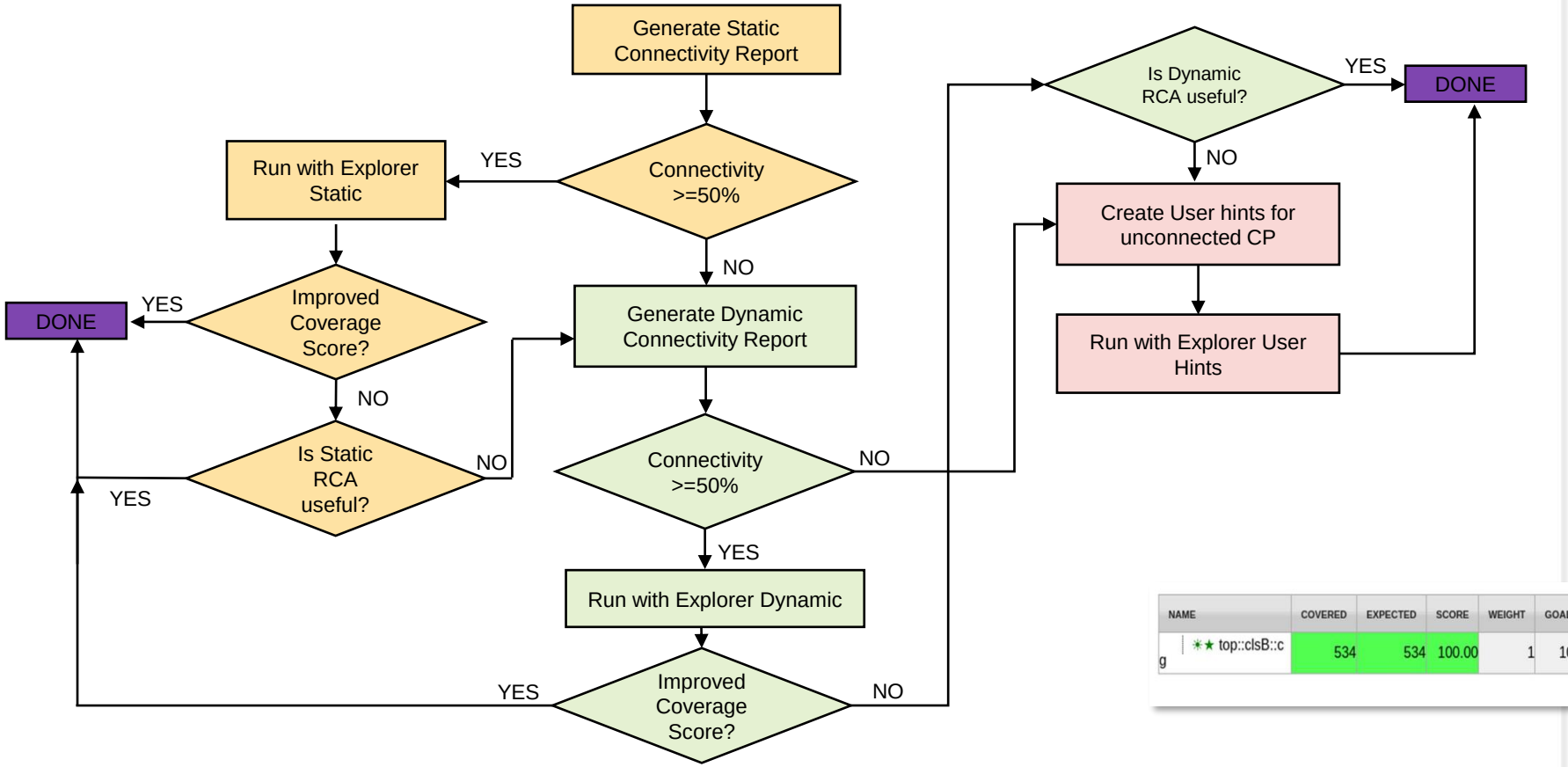


VSO.ai and user hint Integrated with TB

Infer Coverage Connectivity & Target connections in Constraint solver

User hints : hints for constraint random variables in classes connected to a coverpoint to achieve covergroup closure.

Proposed Solution



VSO.ai Connection Inference Engines Flow Chart

NAME	COVERED	EXPECTED	SCORE	WEIGHT	GOAL	AT LEAST	PER INSTANCE	AUTO BIN MAX	PRINT MISSING	COMMENT	CCEX CONNECTED	CCEX TARGETED	RCA(s)
✳★ top::clsB::cg	534	534	100.00	1	100	1	0	64	64		Yes	Yes	0.00

Connectivity report

Better connectivity → Faster coverage closure

Proposed Solution

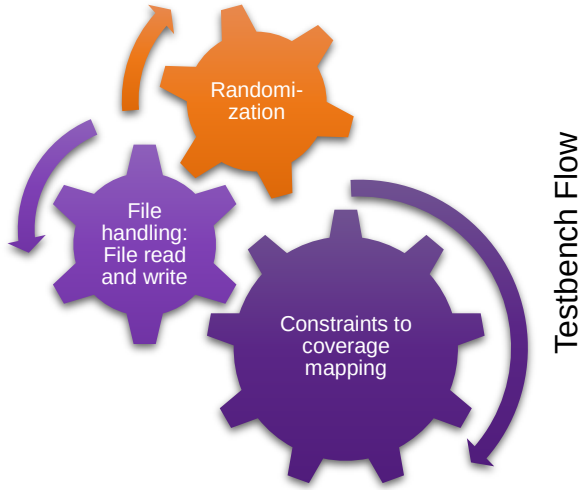
```
module top;
 string file_name = "cmd.in";

 class clsA;
   int fd;
   rand bit r_type1;
   rand bit r_type2;
   rand bit [2:0] r_cache;
   rand bit r_user;

   function void open();
     fd = $fopen(file_name, "w");
   endfunction

   function void close();
     $fclose(fd);
   endfunction

   function void write();
     $fdisplay(fd, "%0d,%0d,%0d,%0d", r_type1, r_type2, r_cache, r_user);
   endfunction
 endclass

 class clsB;
   int fd;
   bit r_type1;
   bit r_type2;
   bit [2:0] r_cache;
   bit r_user;

   covergroup cg;
      cp_type1: coverpoint r_type1;
      cp_type2: coverpoint r_type2;
      cp_cache: coverpoint r_cache;
      cp_user: coverpoint r_user;
      cr_all: cross cp_type1, cp_type2, cp_cache, cp_user;
   endgroup

   function new();
     cg = new;
   endfunction
```

- File open
- Write data to file

```
   function void open();
     fd = $fopen(file_name, "r");
   endfunction

   function void close();
      $fclose(fd);
    endfunction

   function void read();
     assert($fscanf(fd, "%0d,%0d,%0d,%0d", r_type1, r_type2, r_cache, r_user) == 4)
      else fatal
   endfunction

   function void sample();
      cg.sample();
   endfunction
 endclass

clsA clsA_inst = new;
clsB clsB_inst = new;
int num_iters = 512;

 initial begin
    /* randomize and write */
    clsA_inst.open();
    repeat(num_iters) begin
        clsA_inst.randomize();     // randomize
        clsA_inst.write();         //  write to a file
    end
    clsA_inst.close();

    /* read and sample */
    clsB_inst.open();
    repeat(num_iters) begin
        clsB_inst.read();          // read from file
        clsB_inst.sample();        // sample
    end
    clsB_inst.close();

 end
endmodule
```

- Reads data from File
- Sample data

Example code snippet : VSO.ai usage




```
target="$unit::clsB::cg.cp_type1" scope="$unit"  class="clsA" constraint="r_type1";
target="$unit::clsB::cg.cp_type2" scope="$unit"  class="clsA" constraint="r_type2";
target="$unit::clsB::cg.cp_cache"scope="$unit"   class="clsA" constraint="r_cache";
target="$unit::clsB::cg.cp_user"  scope="$unit"  class="clsA" constraint="r_user"  ;
```

Target

source

user hints file : “project_name.tsv”

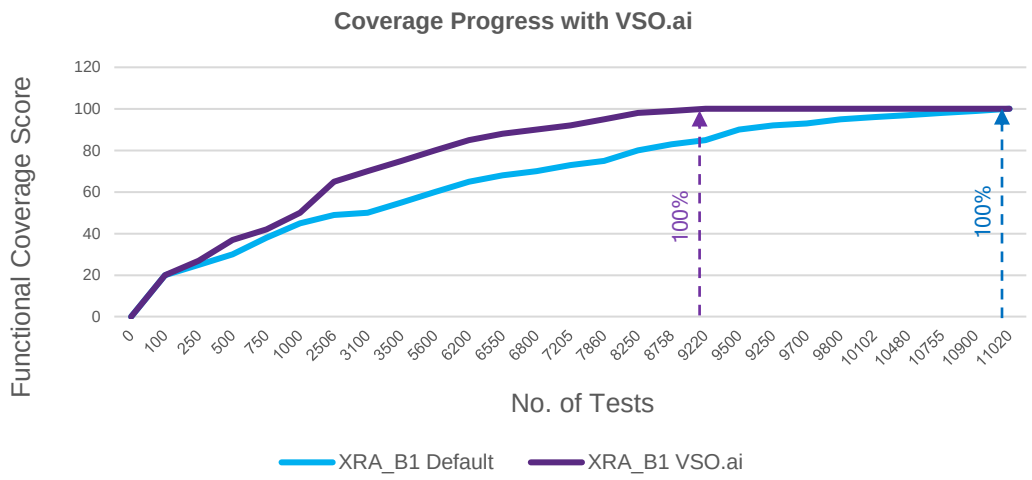
Connectivity : Map the constraints to coverage space to generate the stimulus automatically

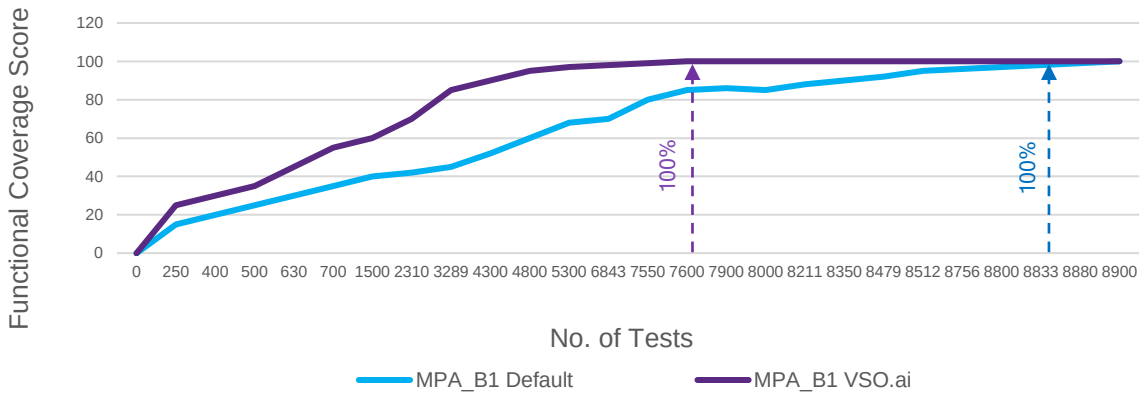
Result

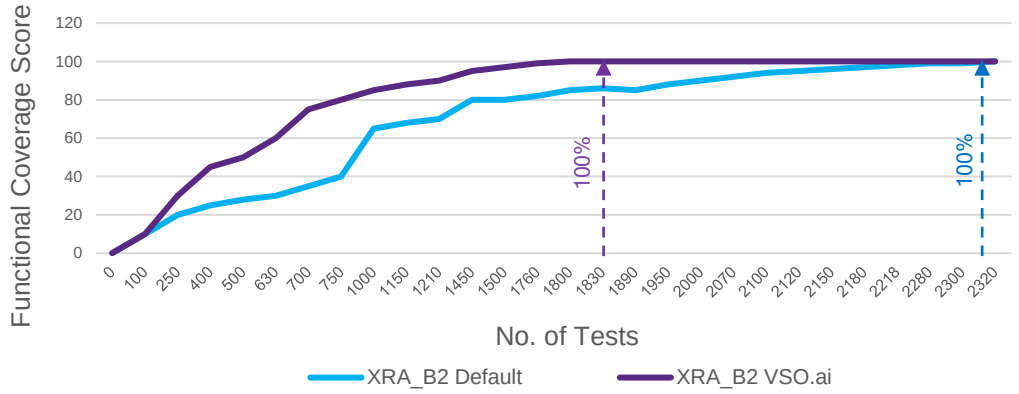
Result

VSO.ai: Faster, Higher Coverage Closure

Block Name	Total No of bins	Total tests		Total reseeds (tests+reseed)	
		Legacy	With vso	Legacy	With vso
XRA_B1	16k	1419	53	11023	9100 (17.5%⬇)
XRA_B2	4k	73	18	2320	1850 (20%⬇)
MPA_B1	9.5k	1580	71	8900	7661 (14%⬇)





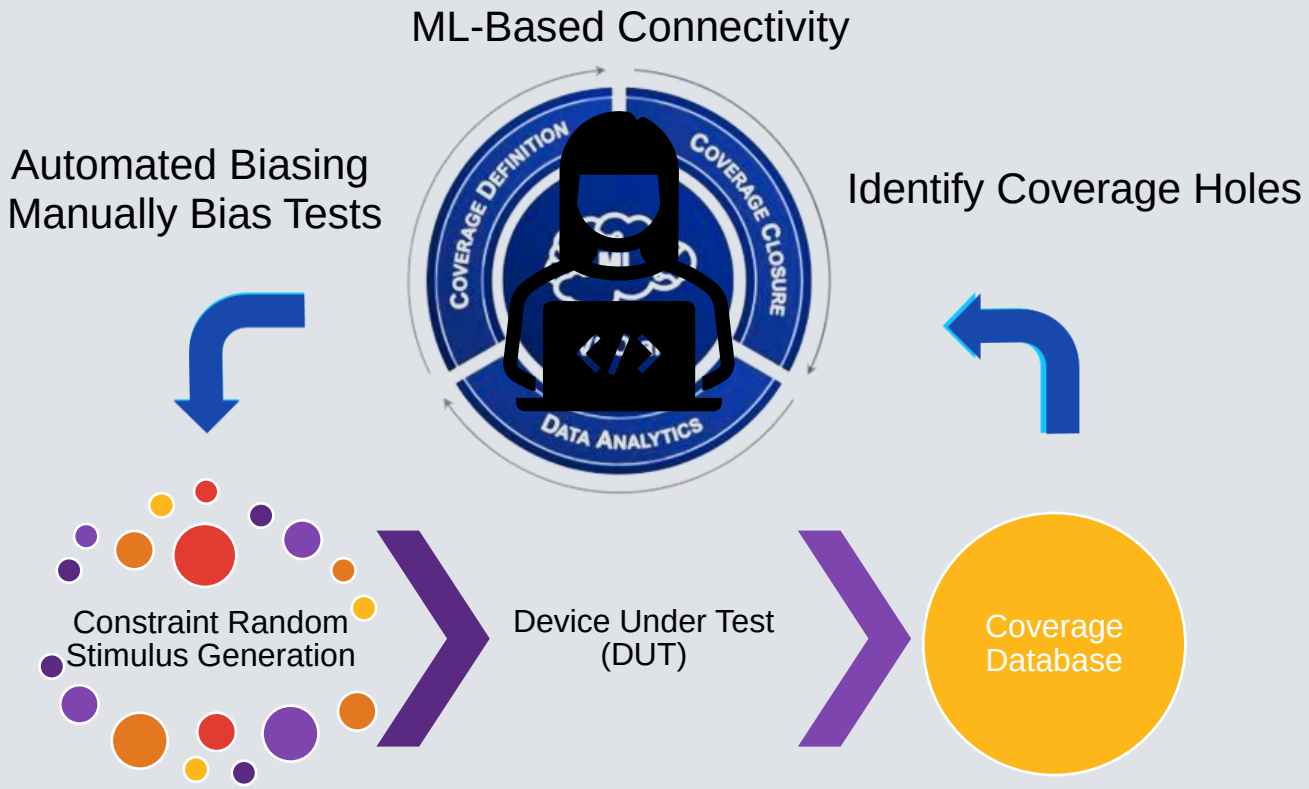


Proven results in block level coverage closure – ~14-20% reduction in tests as well as machines

Summary/Conclusion

Summary/Conclusion

AI-Driven Coverage Closure & Analytics



Today's Coverage Closure Flow

Faster Coverage Closure: *ML-Based Connectivity Engines & Coverage Directed Solvers to Target Coverage Holes*

Advanced Analytics & Diagnostics: *Root Cause Analysis to Identify Unreachable Coverage*

Learning from history: *Reading the previous simulation vdb for inter-simulation; additional option available for reading external merged vdb*

No TB changes : *Plug and play method;*
No changes in the existing constraints/sequences;
Changes are only in Makefile and env variables

This method enables the DV and DE engineers to focus their efforts on fixing the bugs rather than searching for them


THANK YOU
Our Technology, Your Innovation™