2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Using Automation to Close the Loop Between Functional Requirements and Their Verification

Brian Craw – Cypress Semiconductor
David Crutchfield – Cypress Semiconductor
Martin Oberkoenig - Cypress Semiconductor
Markus Heigl - Cypress Semiconductor
Martin O'Keeffe - Cypress Semiconductor

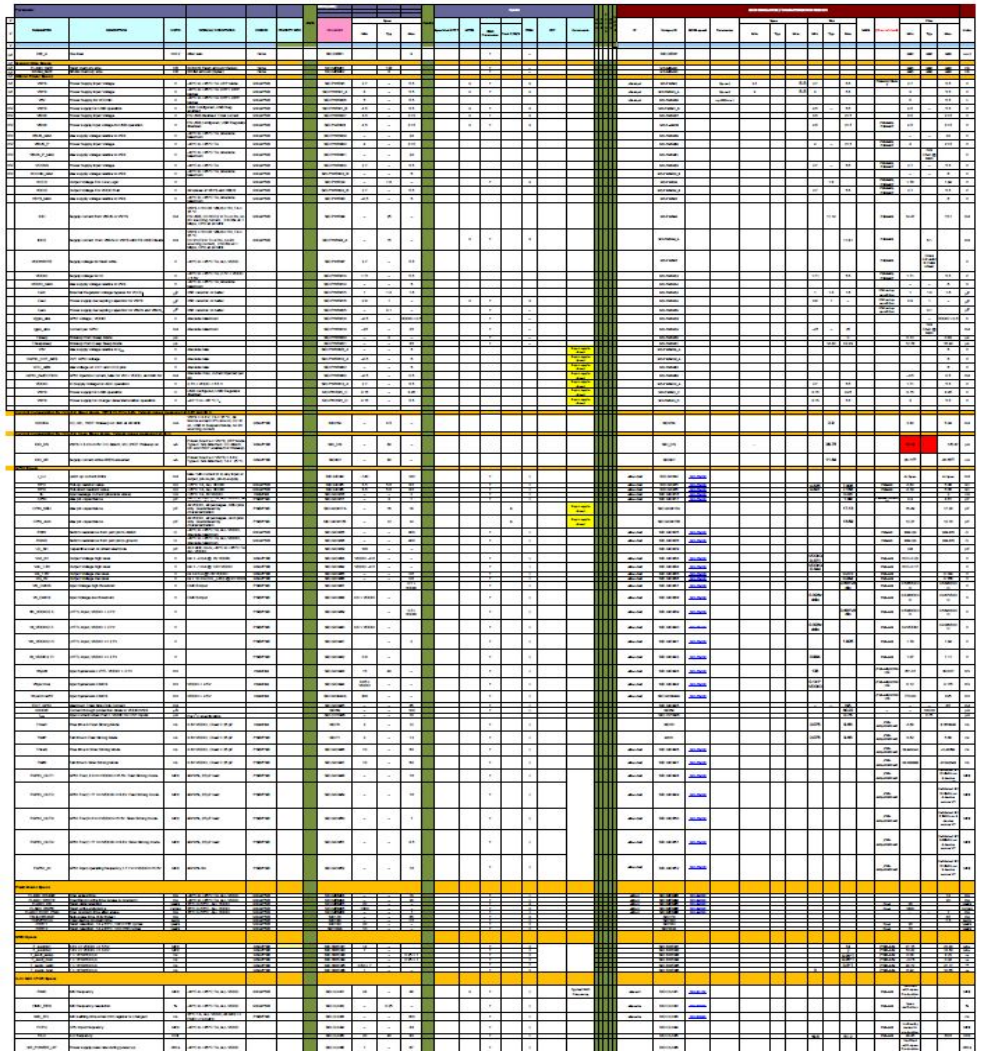
2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- The problems with manual requirements tracking
- The Flow and Tools
  - Requirements Management Tool (RM Tool)
  - Coverage Code Generator (CovGen)
  - Verification Management System (VMS)
- Conclusions

Manual Processes are Error Prone

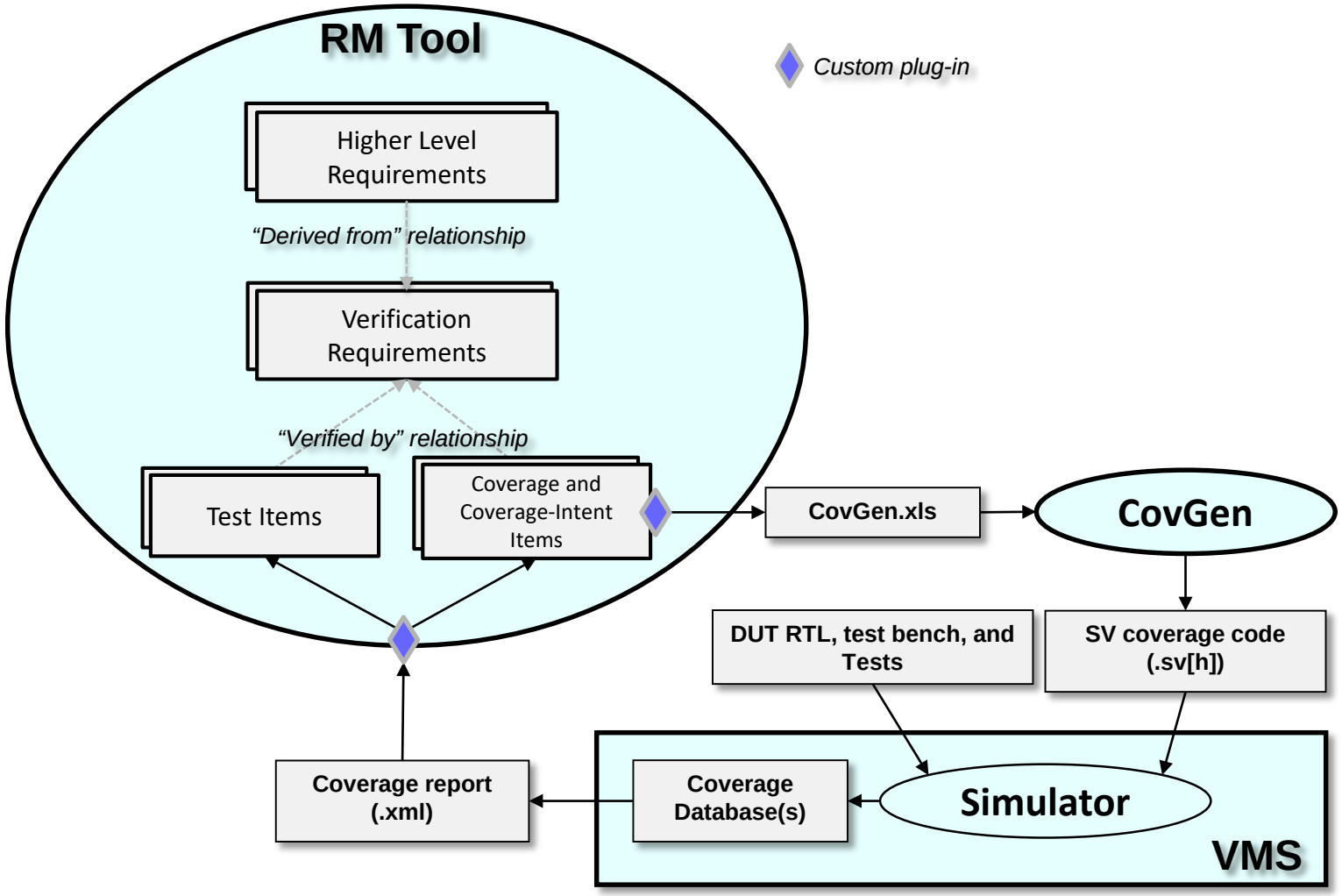
- Tracking spreadsheets are overly complex and therefore under-utilized
- Possible misses on requirement changes
- Incomplete coverage metric to requirement mapping can be hard to detect



ISO 26262 Compliance

- Quality standard required for Automotive products
- Requires a high level of traceability throughout the development process to guarantee top-level safety requirements have been implemented and verified down to the lowest level
- Difficult to fully achieve without the use of a Requirements Management tool and some level of automation.

The Flow

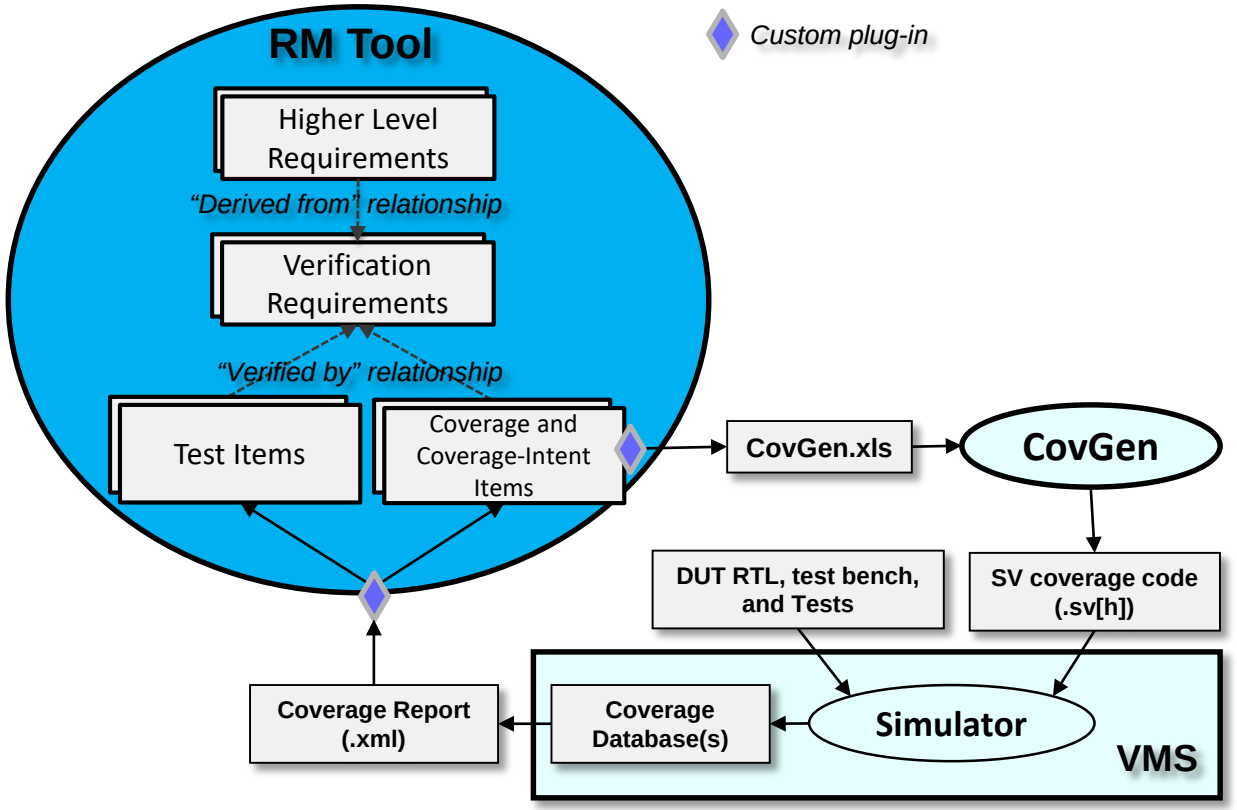

RM Tool
Custom plug-in
Higher Level Requirements
"Derived from" relationship
Verification Requirements
"Verified by" relationship
Test Items
Coverage and Coverage-Intent Items
CovGen.xls
CovGen
DUT RTL, test bench, and Tests
SV coverage code (.sv[h])
Coverage report (.xml)
Coverage Database(s)
Simulator
VMS

Requirements Management Tool

- Features

– Usable across business disciplines

  - Marketing, Design, Verification, etc.
  - Every group uses the same tool
  - Want to avoid exporting/importing of data from/to different tools

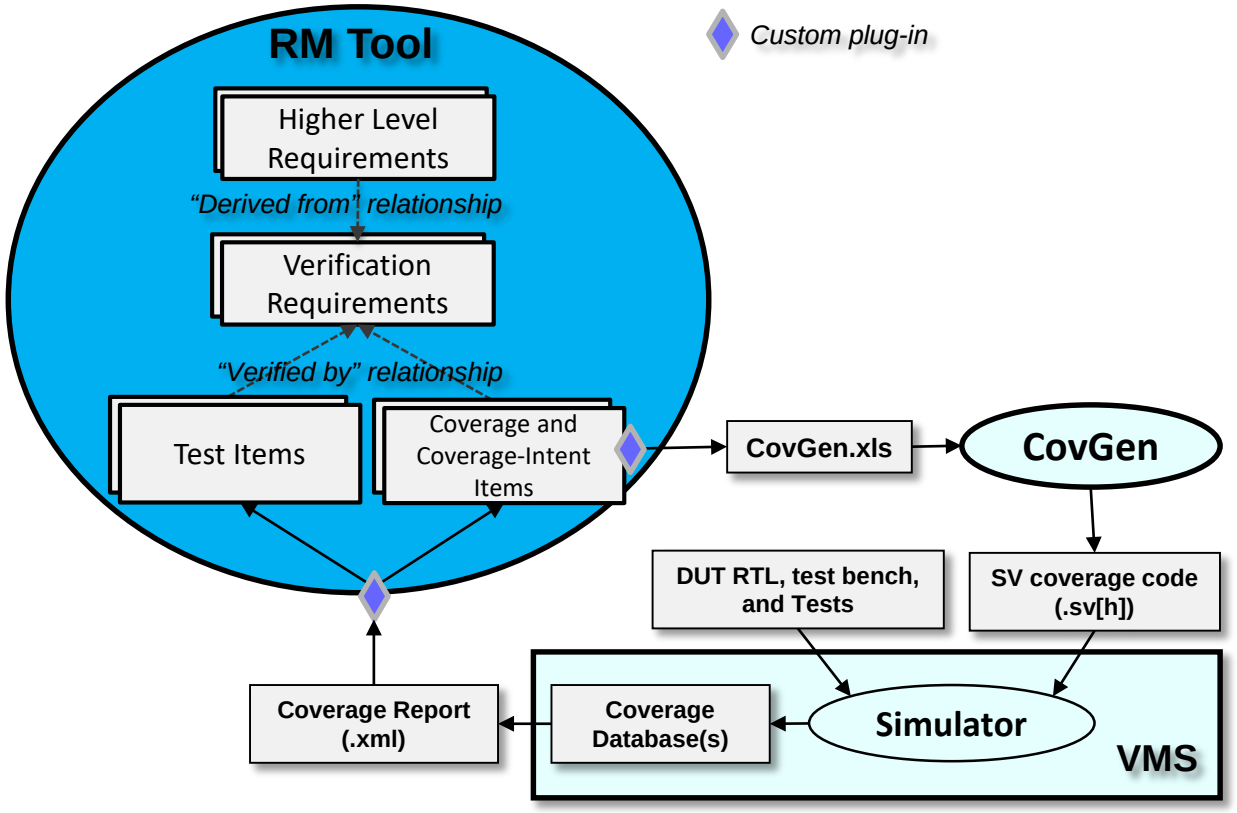


Requirements Management Tool

- Features

– Usable across geographies

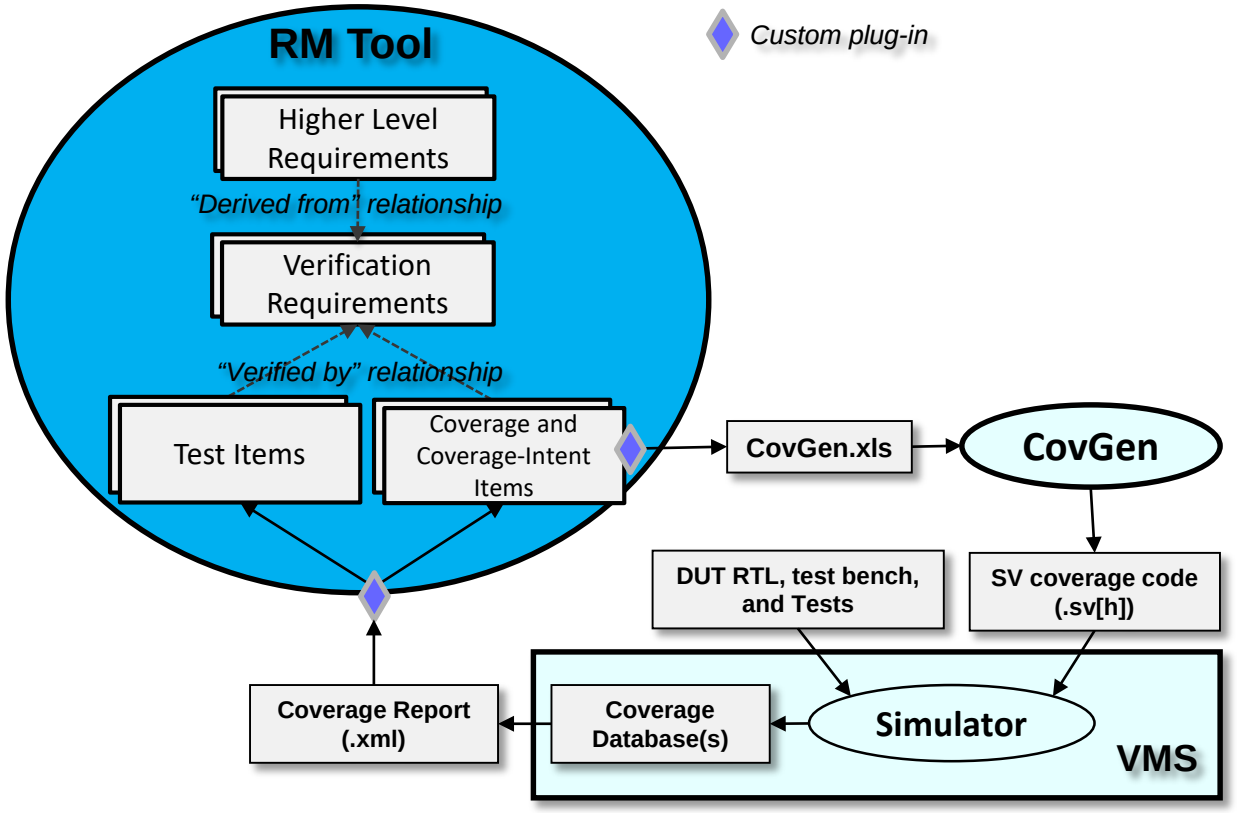
  - Centralized repository of data
  - No nightly syncing of data



Requirements Management Tool

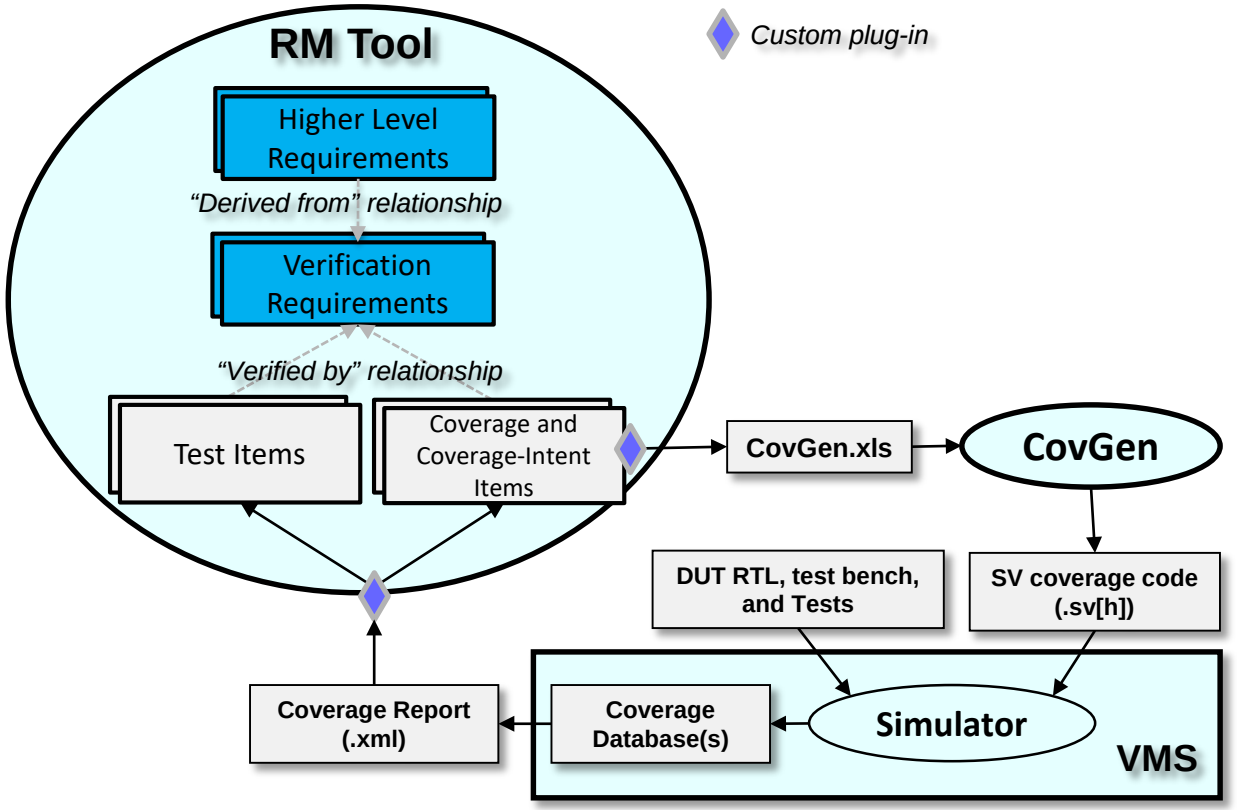
- Features

– Customizable
  - Can create custom items and workflows to track more than just requirements such as functional coverage state and test status
  - Embedded scripting for custom report generation
  - Available API for advanced extensions



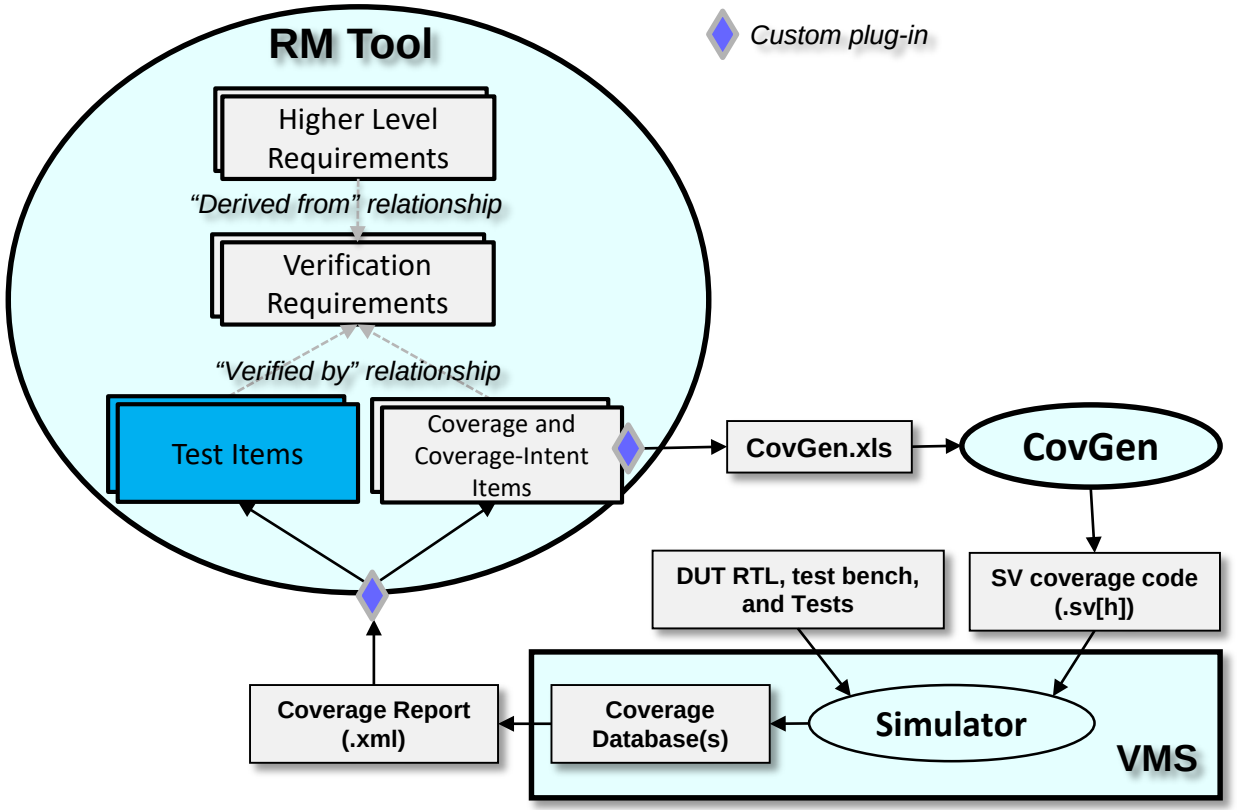
Requirements Management Tool

- The Requirements
  - High Level Requirements
    - Customer, Marketing, Architecture
  - Verification Requirements
    - Derived from higher levels



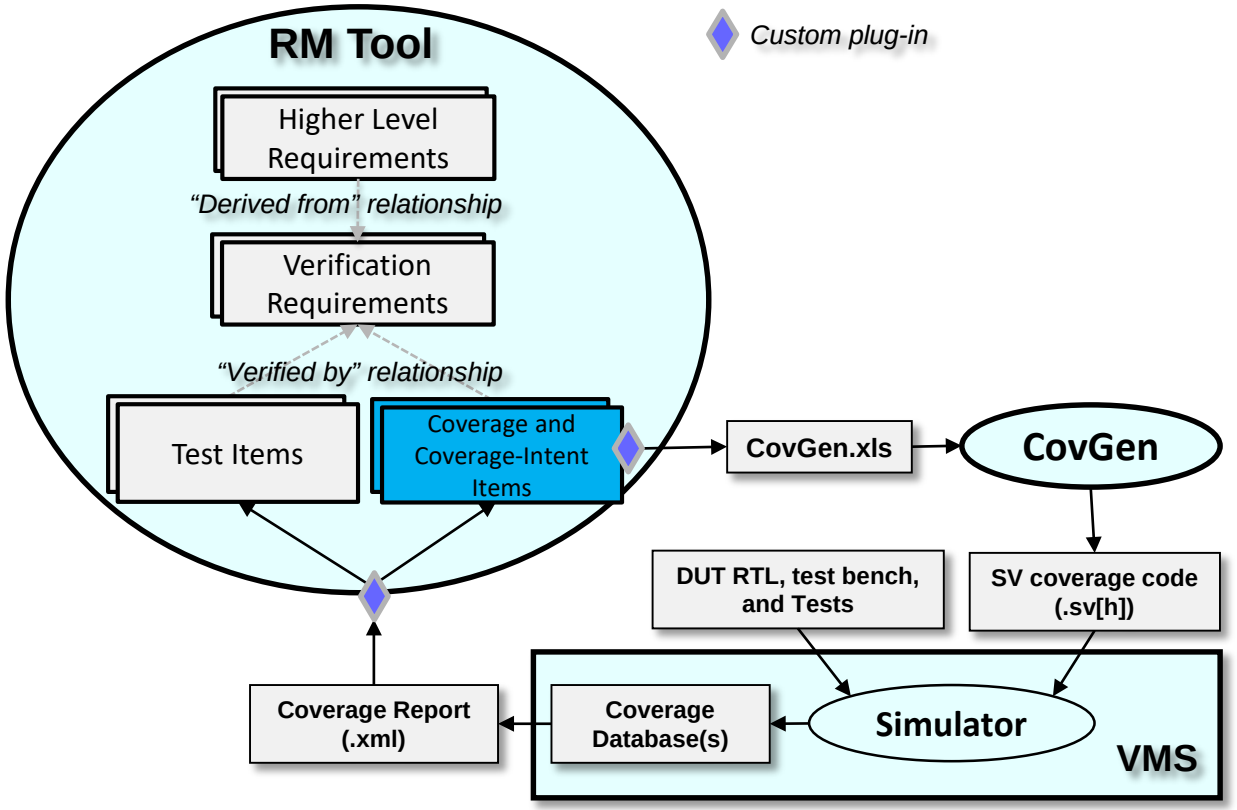
Requirements Management Tool

- Test Items
  - Describe functional tests
  - Have a “Verify” relationship to Verification Requirements
  - Can be used to create a Test Plan



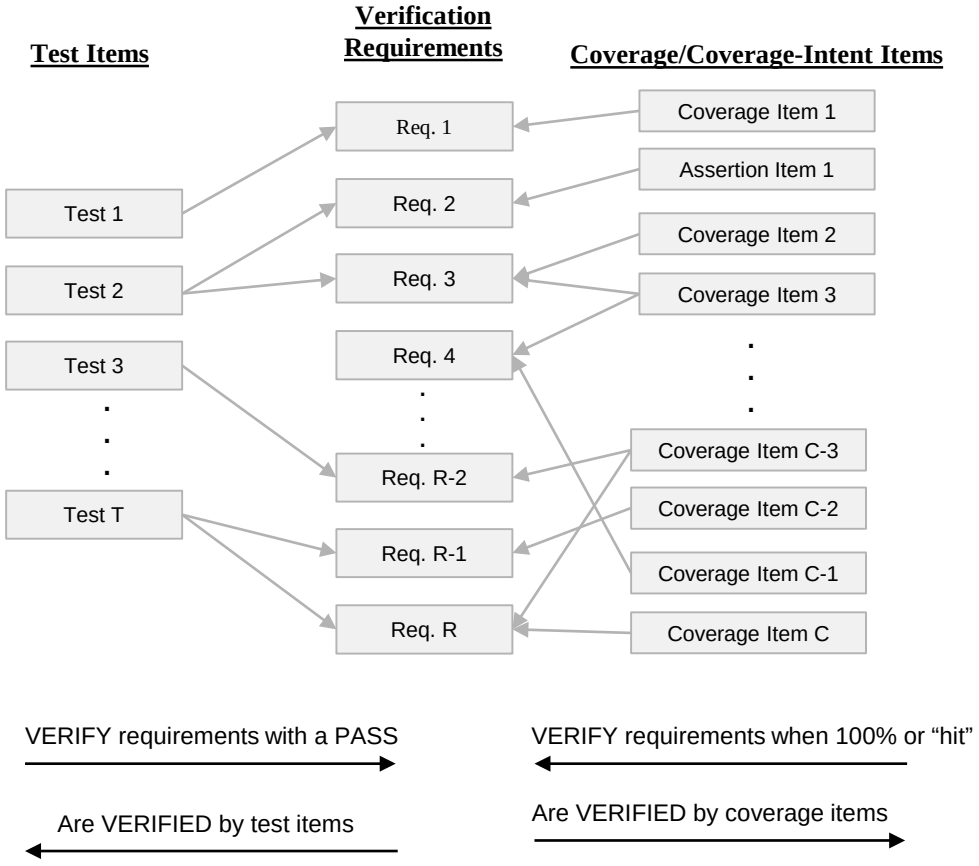
Requirements Mangement Tool

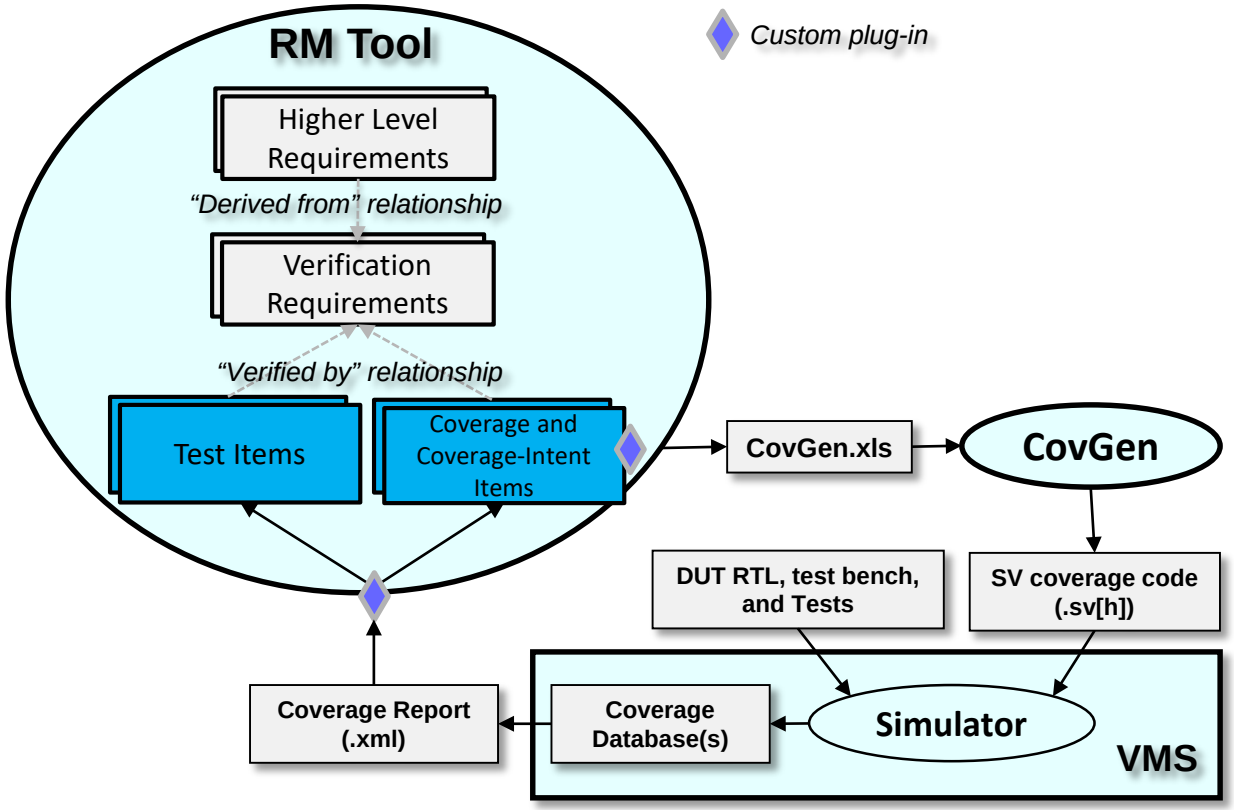
- Coverage and Coverage-Intent Items
  - Describe SystemVerilog coverage constructs: Coverpoints, Crosses, and Assertions
  - Have a “Verify” relationship to Verification Requirements
  - These items are updated with regression results via a custom plug-in



Requirements Mangement Tool

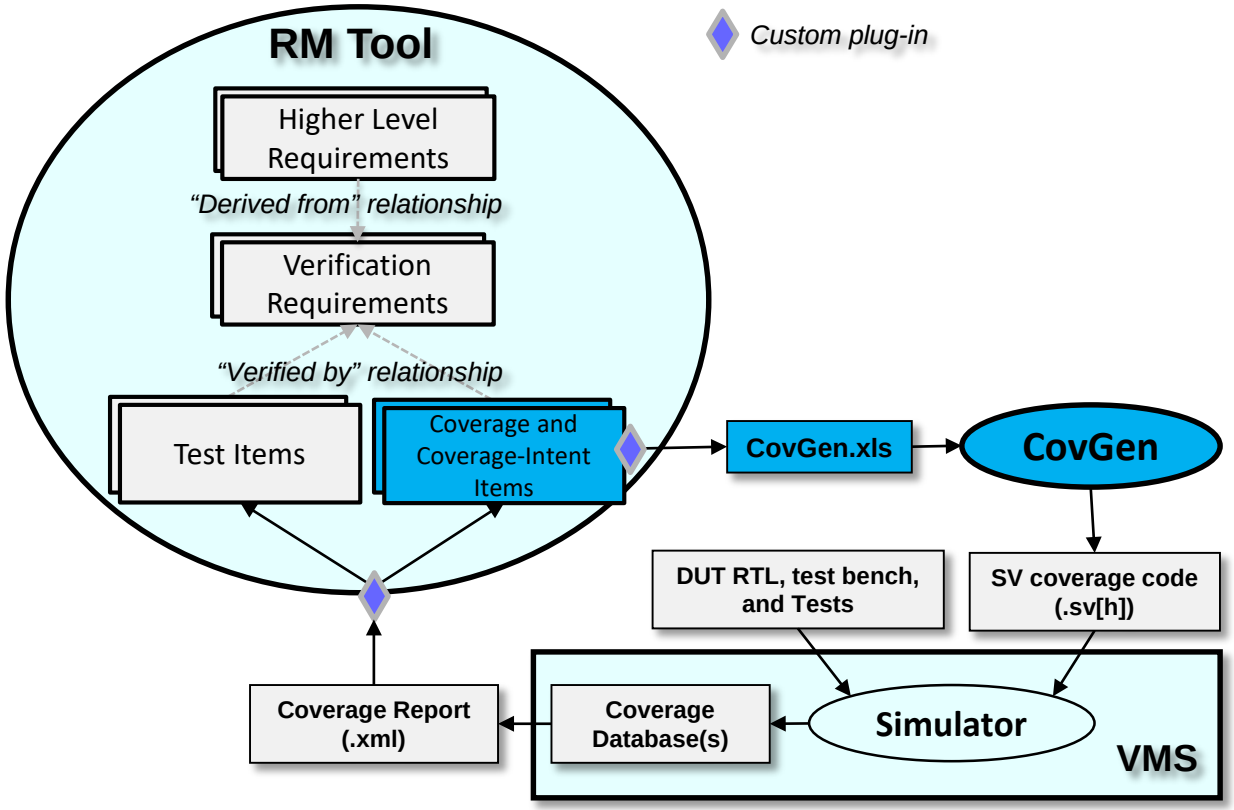
- How are Requirements Verified?





Coverage Generator (CovGen)

- In-house tool to generate SystemVerilog code from a coverage intent specification
  - Reads in an XLS file and outputs SV code
  - Custom plug-in for RM tool written to export Coverage-Intent items into XLS format
  - Each Coverage-Intent item corresponds to a line in the XLS and represents a Coverpoint or Cross
  - Special keywords are used to describe various types of bins



Coverage Generator (CovGen)

- Benefits
  - Ensures correct SV syntax
  - Maintains naming conventions
    - Important that the names match between RM Tool and test bench so back-annotation is successful
  - Easier to review intent in a spreadsheet format or in the RM Tool than the SystemVerilog code itself

Coverage Intent XLS Format

CG_EXAMPLE												
COMMENT	WGT											
CovGen example	1											
TYPE	CATEGORY	REGISTER	FIELD	BIT_FIELD	QTY	NAME	WGT	BINS	SAMPLE EVENT	CP1	CP2	COMMENT
CP	Var		ENABLE	1	2	cp_enable	1	AUTO				
CP	TYPE= CHAN_MODE_T	MODE	MODE	7:0	1	cp_mode_mode	1	TRANS = SINGLE => DUAL				Captures the transtion from single channel to dual channel mode
CP	Reg	CTL	CHAN1	2:0	1	cp_ctl_chan1	1	WALKONES	chan1_en			
CP	Reg	CTL	CHAN2	5:3	1	cp_ctl_chan2	1	MIN=3				
CS						cs_cp_ctl_chan1_cp_ctl _chan2	1	IGNORE {CP1=MIN & CP2=7}		cp_ctl_chan1	cp_ctl_chan2	

Example Coverage Generator Output

Covergroup

```
covergroup SAMPLE_cg;
  type_option.weight = 1;
  type_option.comment = "CovGen example";
        .
        .
        .
endgroup : SAMPLE_cg;
```

Coverpoint with transition bin

```
 cp_mode_mode: coverpoint CHAN_MODE_T'(mode.peek("MODE")) iff (sample_cp_mode_mode)
{
    // Captures the transtion from single channel to dual channel mode
   type_option.weight = 1;
    bins bin_single_fb_dual = (SINGLE => DUAL);
 }
```

Cross with ignore_bins

```
cs_cp_ctl_chan1_cp_ctl_chan2: cross cp_ctl_chan1, cp_ctl_chan2  {
  type_option.weight = 1;
  ignore_bins bin_cp1_0_cp2_7 = (binsof(cp_ctl_chan1) intersect {0} &&
                                 binsof(cp_ctl_chan2) intersect {7});
}
```

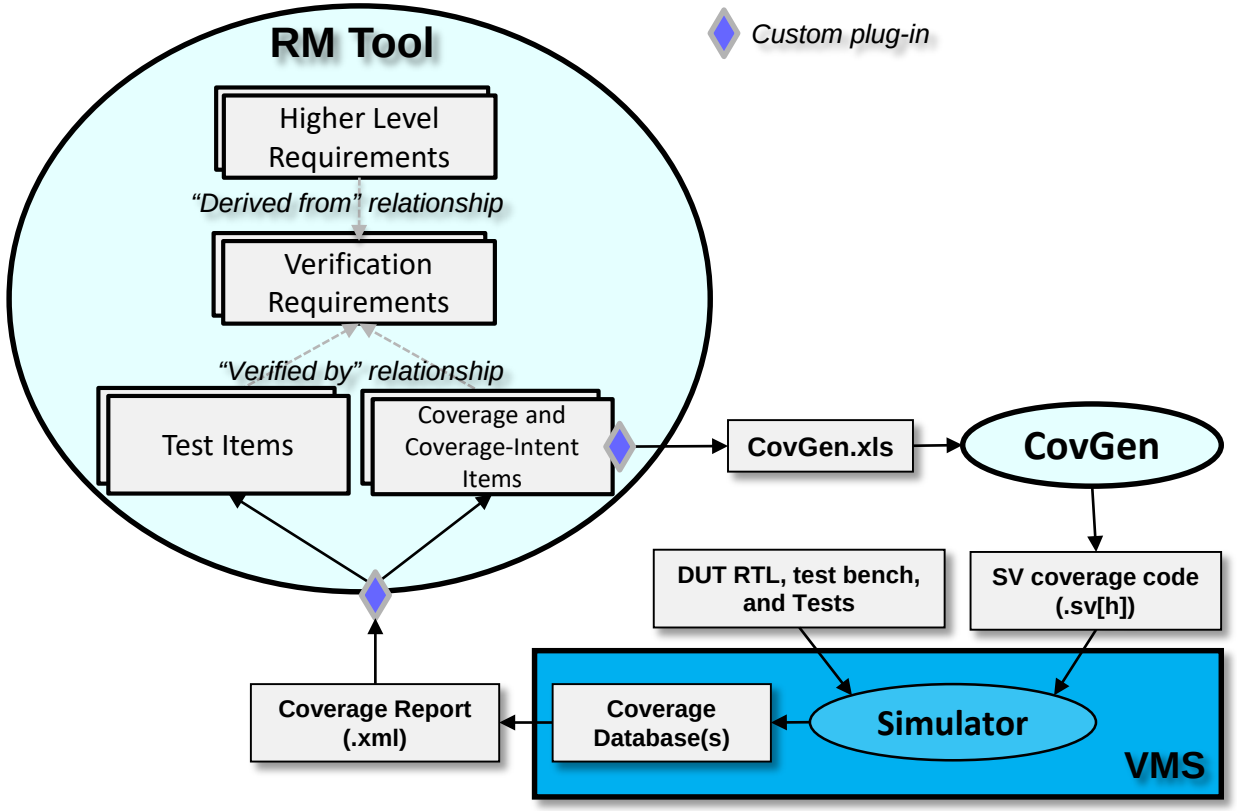
Coverpoint with walking one bins

```
cp_ctl_chan1: coverpoint bit_3_t'(ctl.peek("CHAN1")) iff (chan1_en) {
  type_option.weight = 1;
   bins bin_walkone_0 = {3'b001};
   bins bin_walkone_1 = {3'b010};
   bins bin_walkone_2 = {3'b100};
 }
```

Coverpoint with auto generated bins

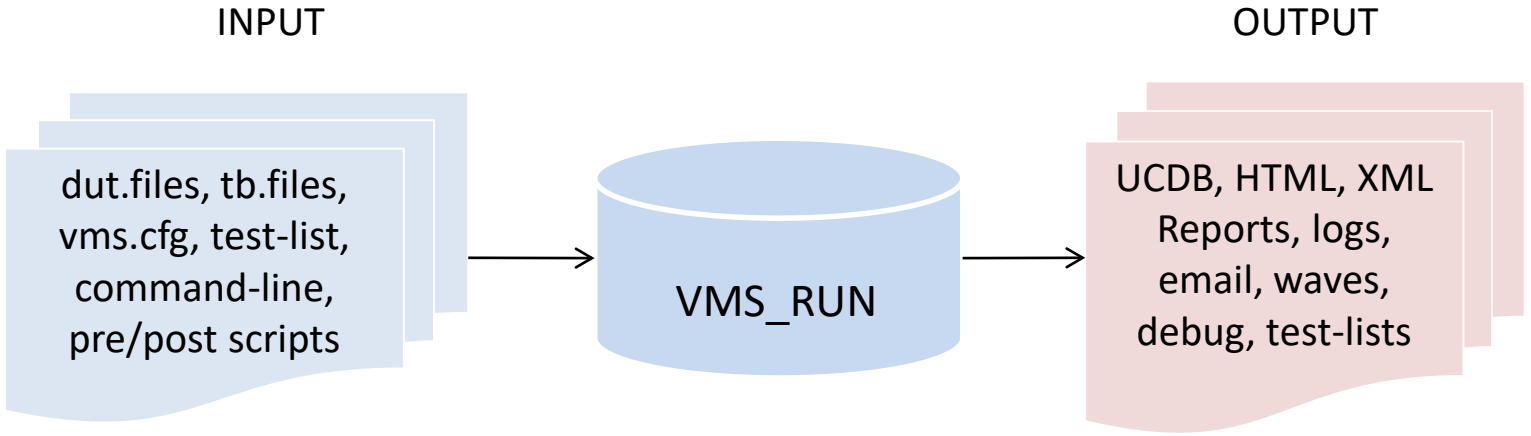
```
cp_enable: coverpoint ENABLE iff (sample_cp_enable) {
  type_option.weight = 2;
  bins bin_0 = {0};
  bins bin_1 = {1};
}
```

Verification Management System



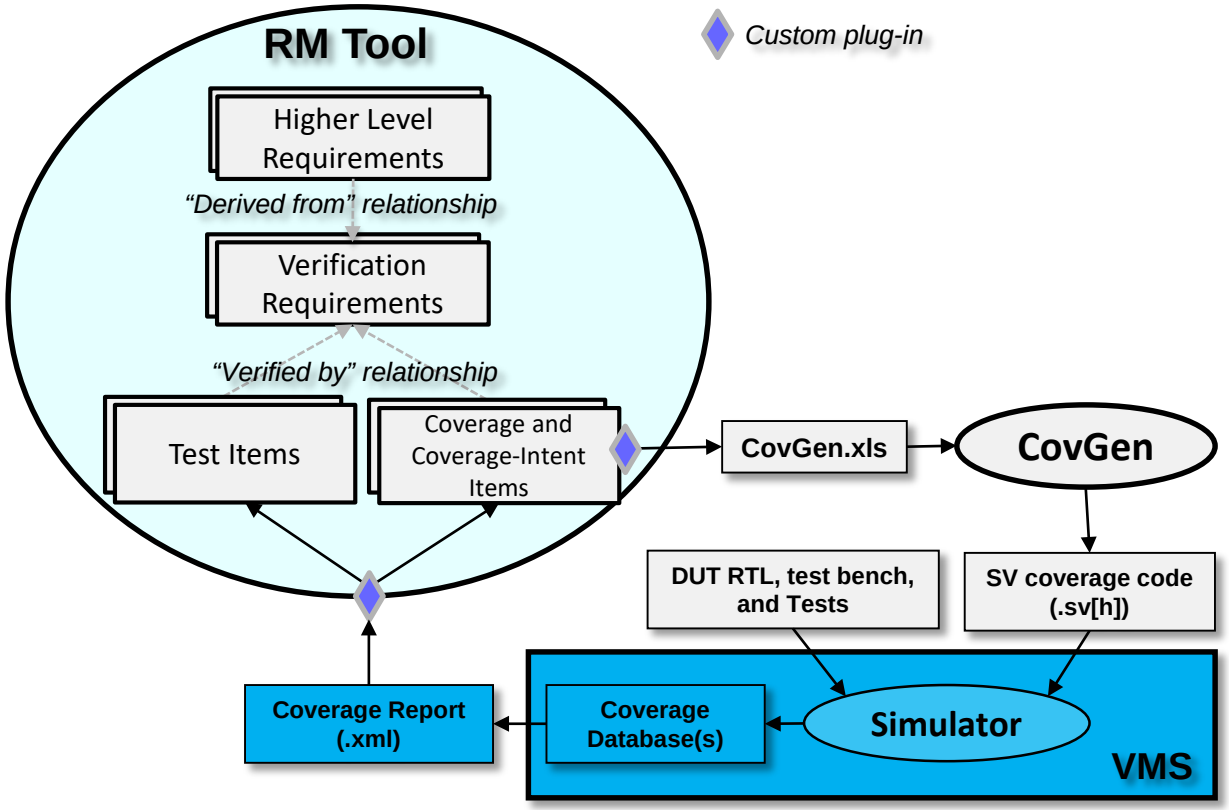
Verification Management System

- VMS – Verification Management System
- Established a standard approach to:
  - Design and test bench organization
  - Regression management
  - Regression status / coverage collection



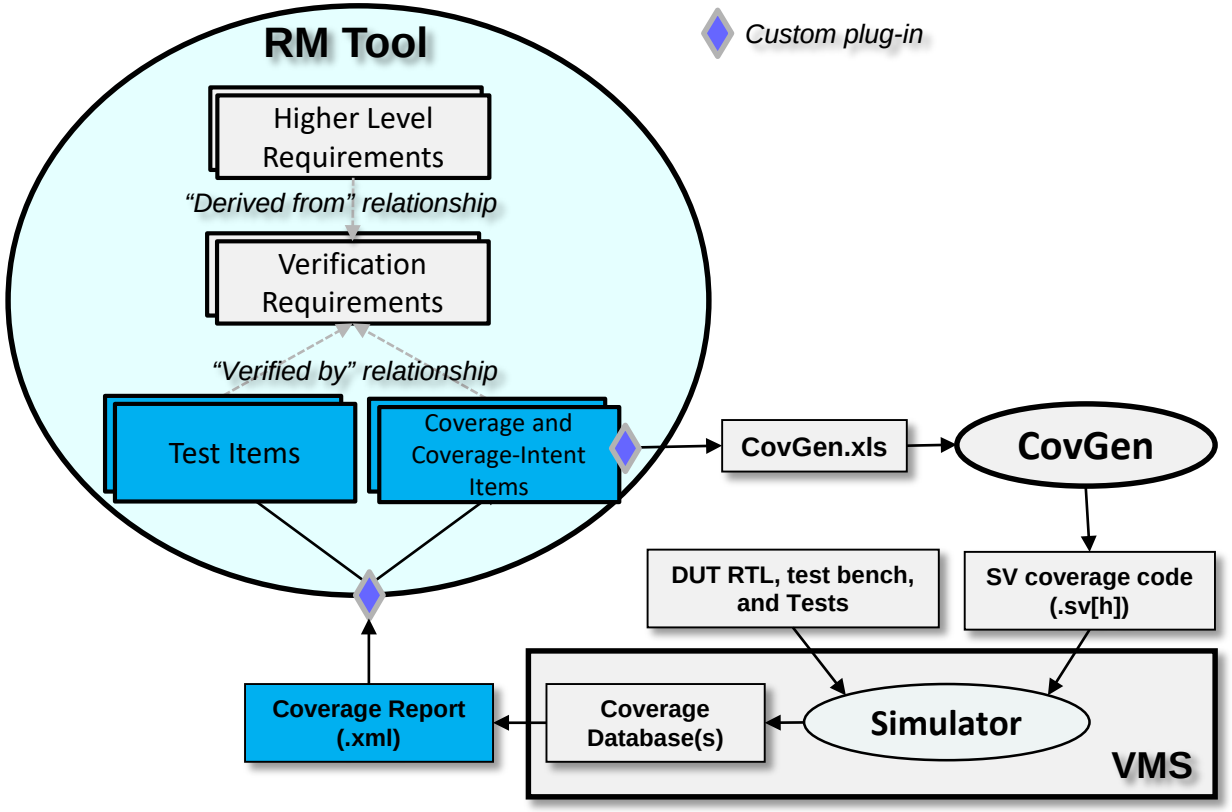
Verification Management System

- VMS will produce coverage databases for all simulations and merge them into a single file
- Vendor coverage database access utilities are used to generate an XML report containing all coverage metrics



Coverage Results Import

- Custom plug-in written to import coverage metrics report
- Coverage Intent Items are updated with the coverage results.
  - i.e. percent bin hits, assertion fires
- Test Items are updated with test results.
  - i.e. pass/fail



Conclusions

- The automation and traceability provided by this flow allow us to efficiently achieve the quality demanded by both industry and internal standards
- Current State: Development nearly complete. Pilot projects due to start soon
- Future Work: Plan to broaden the flow to benefit other aspects of development such as mixed-signal IP development and verification, validation, document generation, etc