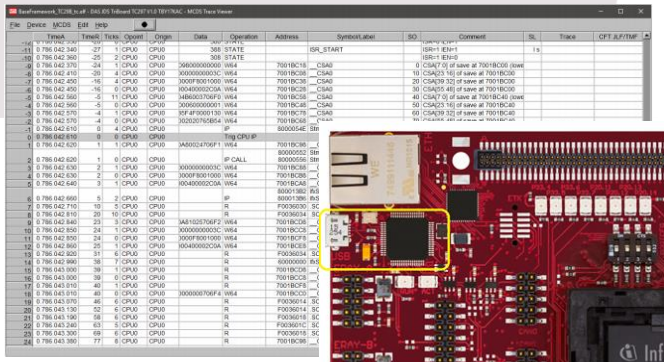
Using PSS: An Automotive Case Study

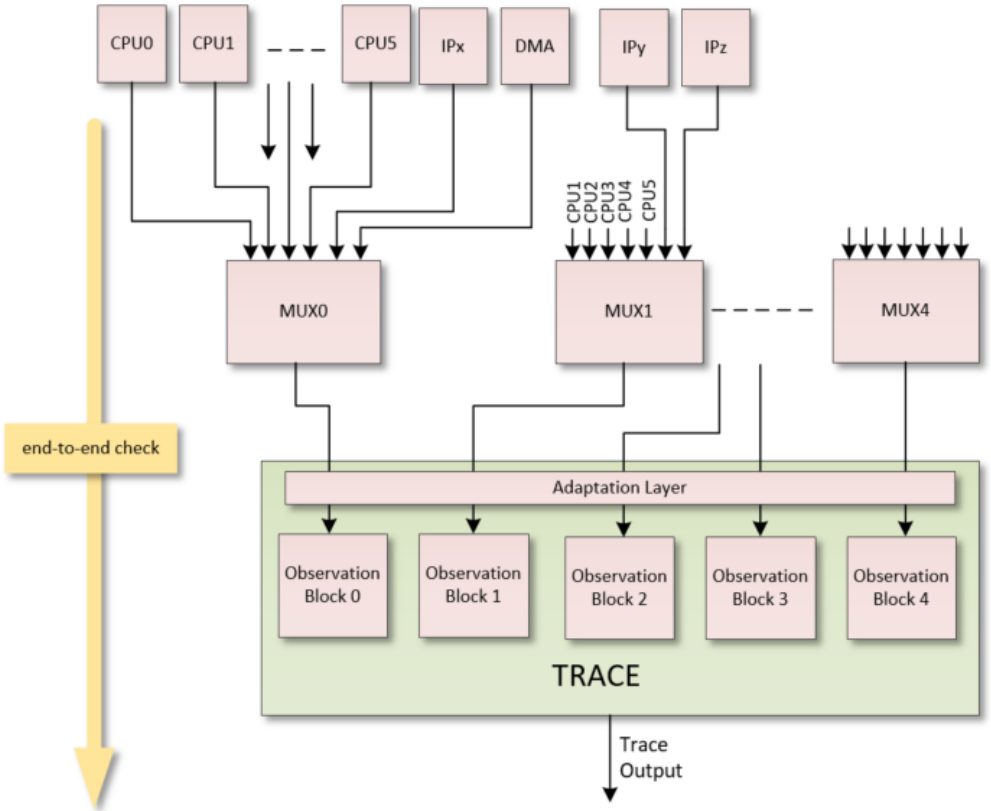
Ben Sutton (IFX)
Leo Matturi (IFX)
Kumar Shankar Dass Dilip (TVS)
Gopalakrishnan Balasubramanian (TVS)

Infineon Technologies UK Ltd.
Test and Verification Solutions Ltd

Motivation

- The AURIX™ microcontroller family is targeted at Automotive applications, both for Powertrain and Safety. It contains a comprehensive on-chip trace solution to aid debug.
- The TRACE IP supports concurrent trace of many different on-chip sources – via a complex multiplexing infrastructure.
- Functional verification of correct integration at subsystem and SOC level is a big challenge





1. A lot of system connectivity – varies across all family derivatives

2. End to End verification of Trace paths
   - TRACE IP contains adaptation logic to handle multiple trace source protocols
   - Integration of all sources with the TRACE IP is at subsystem/SOC level.

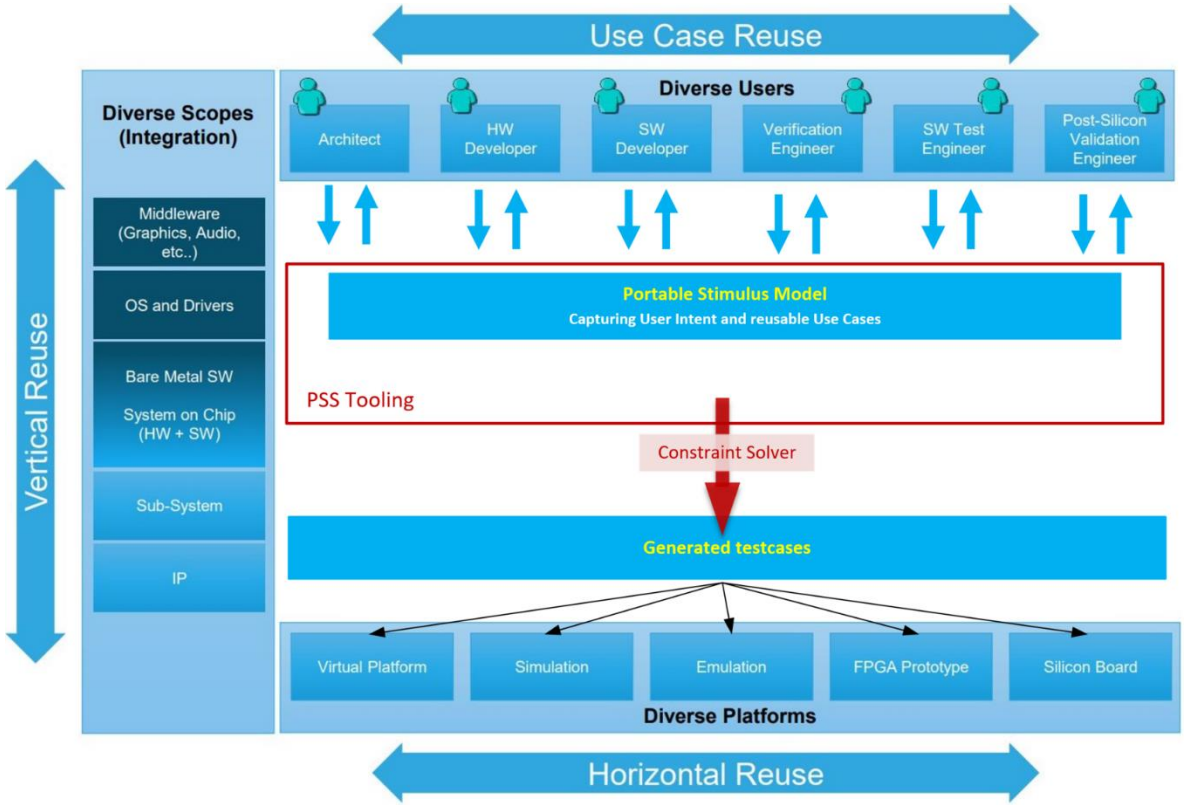
   - A big integration challenge to ensure TRACE works with all sources

Challenge

- Existing methodology comprises platform tests manually coded in C, coupled with scripted checking to verify trace output.
- Disadvantages of this approach:
  - Maintenance and reuse of these test suites across multiple chip derivatives is both time consuming and expensive.
  - Test programmer has to manage system resources – avoiding conflicts between code running on multiple CPUs in parallel
  - Legacy scripting checking flow is hard to maintain and not scalable.

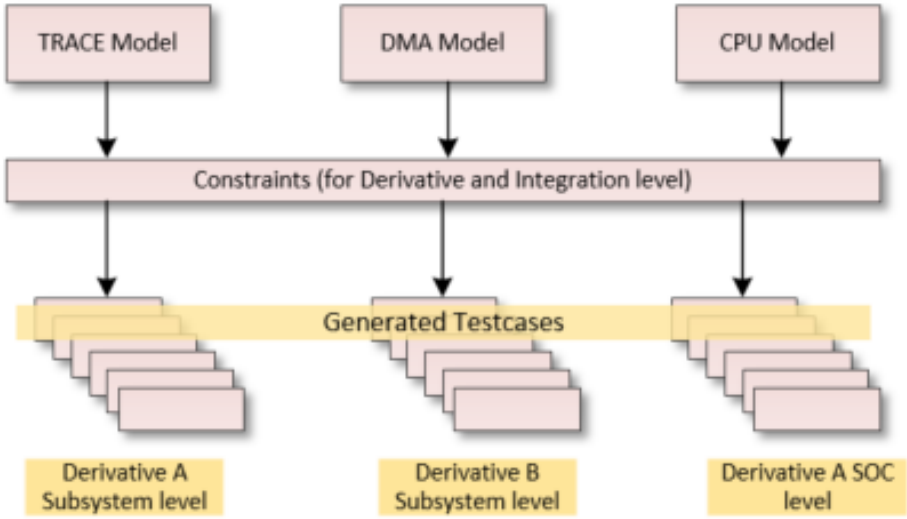
- Need to improve efficiency for test generation and reuse – investigated testcase automation solutions.
- The Accellera Portable Test & Stimulus Standard (PSS) gives a framework to define your high level verification intent in a model (stimulus and test scenarios).
- Verification engine generates testcases targeting different integration levels, platforms and chip derivatives.

- Variety of PSS tools available
- Chose Cadence Perspec System Verifier PSS tool – already in use at Infineon.
- Intention was to automate our testcase generation for reuse in:
  - Different derivatives.
  - Different integration levels (Subsystem, SOC).
  - Different platforms (simulation, emulation, validation).



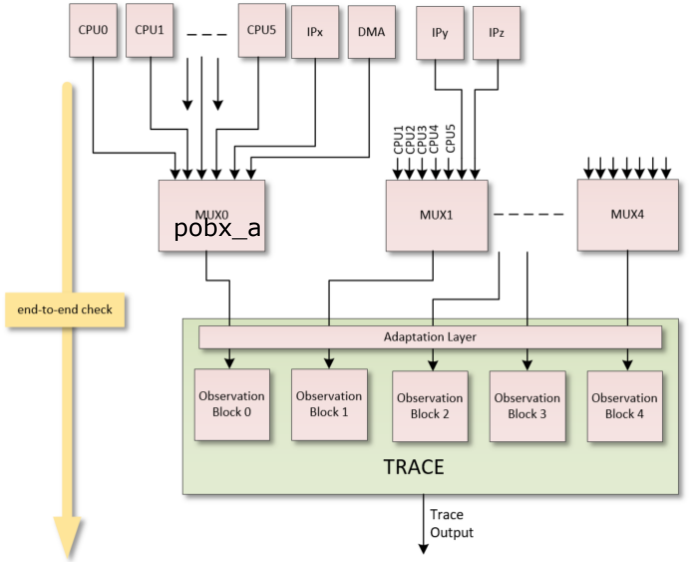
Solution

- Created a model of the TRACE functionality.
- Written in Perspec SLN (System-Level Notation), one of the bases for Accellera PSS.
- Existing models for other key system IP, such as the CPUs and DMA already exist at Infineon. Can be used in conjunction with the new TRACE model to provide trace stimuli.



- Example constraint code in the TRACE model:

```
#define MCDS_POBX_SRC_LIST CPU0,CPU1,CPU2,CPU3,CPU4,CPU5,LMU0,OLDA,OTGB,OTGBM,NONE;

#define MCDS_POBY_SRC_LIST CPU1,CPU2,CPU3,CPU4,CPU5,OTGBM,NONE;

#define MCDS_POBZ_SRC_LIST CPU0,NONE;

#define MCDS_CPUx_SRC_LIST CPU0,CPU1,CPU2,CPU3,CPU4,CPU5;

action pobx_a like ob_base {

    constraint ob_typ == OB_POBX;

    constraint src != NONE => trc_in.idx == ob_typ.as_a(uint);

    constraint tmux_out.ob_typ == OB_POBX;

    constraint src in [MCDS_POBX_SRC_LIST];

  };
```




```
constraint (src     in [MCDS_CPUx_SRC_LIST]) =>  tmux_out.tcmux_typ in [CPU_DTRACE, CPU_MEMSLAVE];
```

TRACE Model – Implementation example

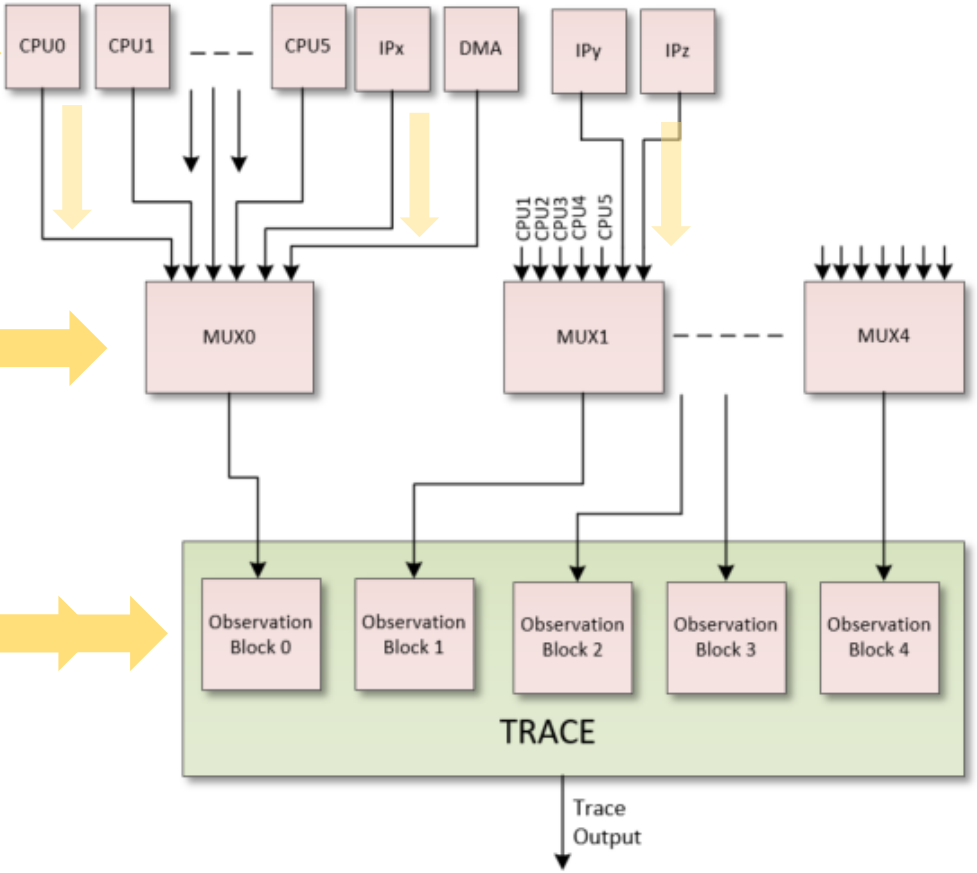
3 The existing IP models are constrained to generate appropriate traffic for our generated scenario

1 A test is generated choosing a random trace source for all Observation Block (OB) MUXs, CPU[0-5, DMA, etc.]

2 An appropriate TRACE configuration is programmed

constraints

constraints



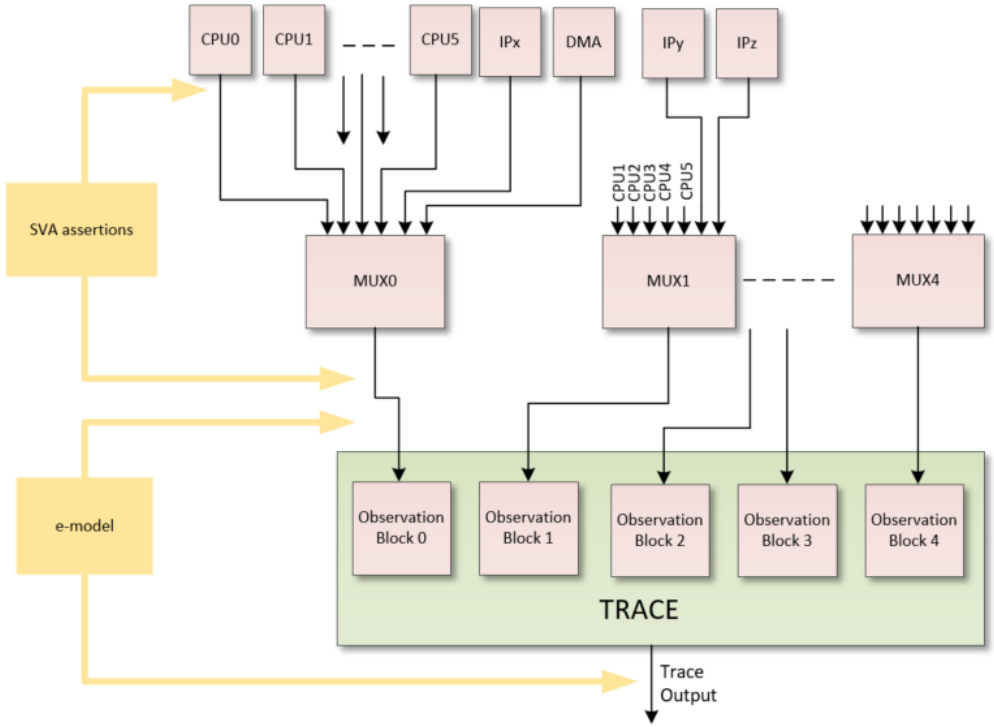
MUX0
Select CPU0 trace source
Program TRACE OB0 registers for a CPU program tracing configuration
Instruct CPU model to generate CPU0 traffic

MUX1
Select DMA trace source
Program TRACE OB1 registers for a DMA data tracing configuration
Instruct DMA model to initiate transfers

Checking Flow

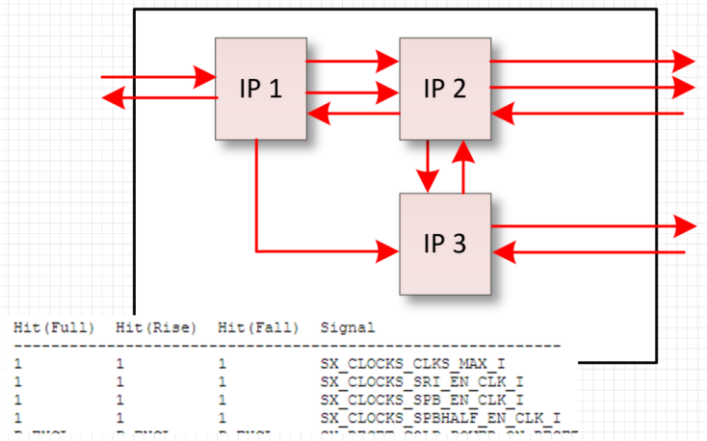
- Easily generate hundreds of tests but legacy scripting based verification flow is not scalable.
- SVA Assertions are used to prove trace connectivity
- The TRACE IP testbench environment is a constrained random environment utilising Specman ***e***, with a reference model enabling end-end verification of the TRACE flow.
- The IP reference model is instantiated at the integration level in passive mode to verify the trace flow through the TRACE IP.

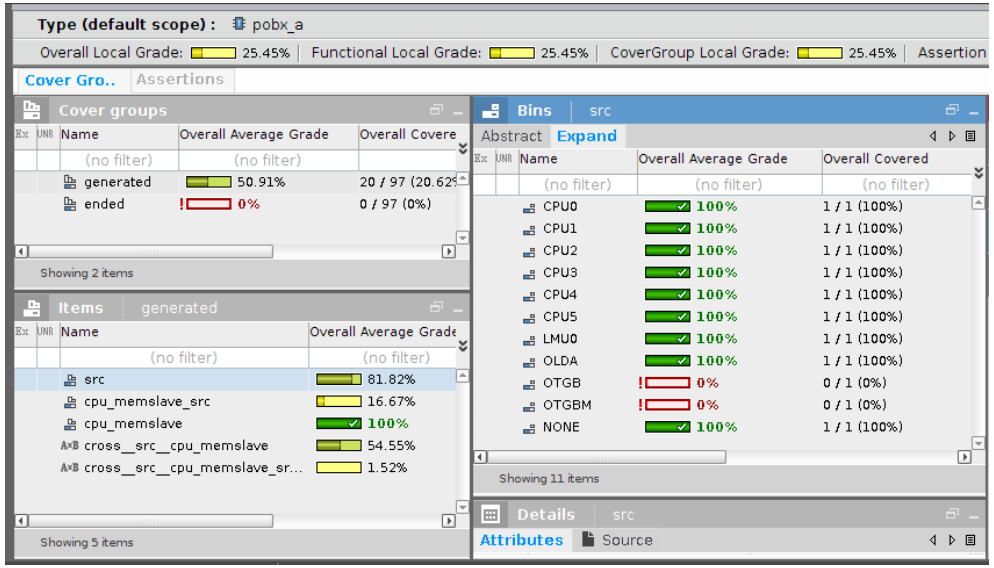
- Other PSS models in our integration verification include checking, however the TRACE checking is independent.
- Existing Verification IP model was available for reuse – saving effort.
- Verification IP model is simulated at integration level, potentially highlighting wrong assumptions made in IP level modelling.



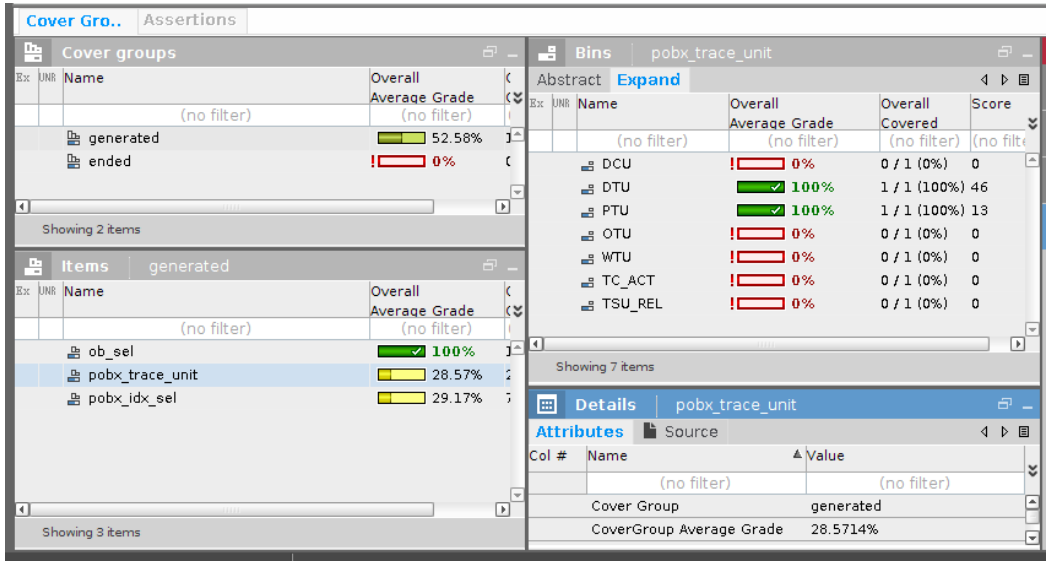
Coverage

- Basic metric for integration verification is Design Implementation Toggle Coverage, ensuring all IP interfaces have been exercised.
- This only provides a measure of activation of the *implemented* features/connectivity and does not identify those missing.
- On top we specify Functional Coverage, ensuring system features are exercised and interesting scenarios covered.
- The PSS tooling provides Generation Functional Coverage, allowing us to know before regressing what will be stimulated and hence if all required scenarios will be covered by a generated test suite.
- This Generation Coverage is then converted into Runtime Coverage





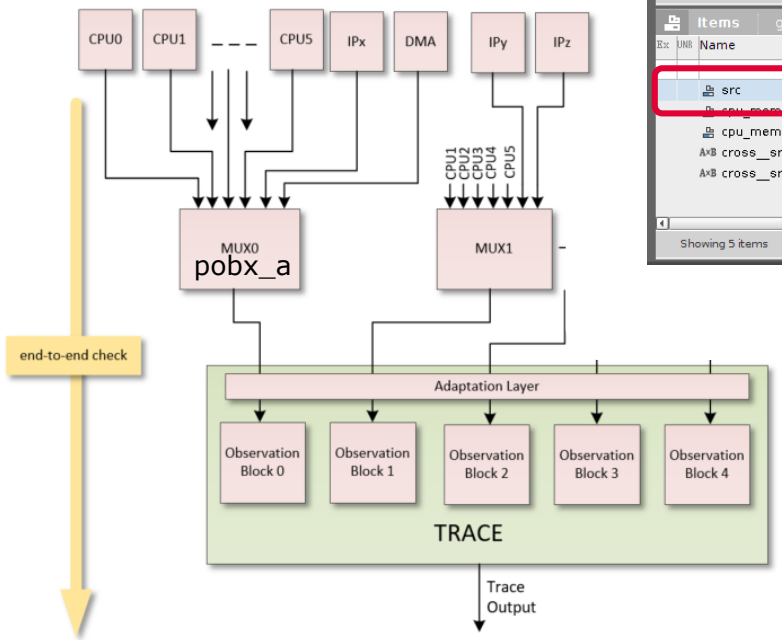
Trace path Functional Coverage

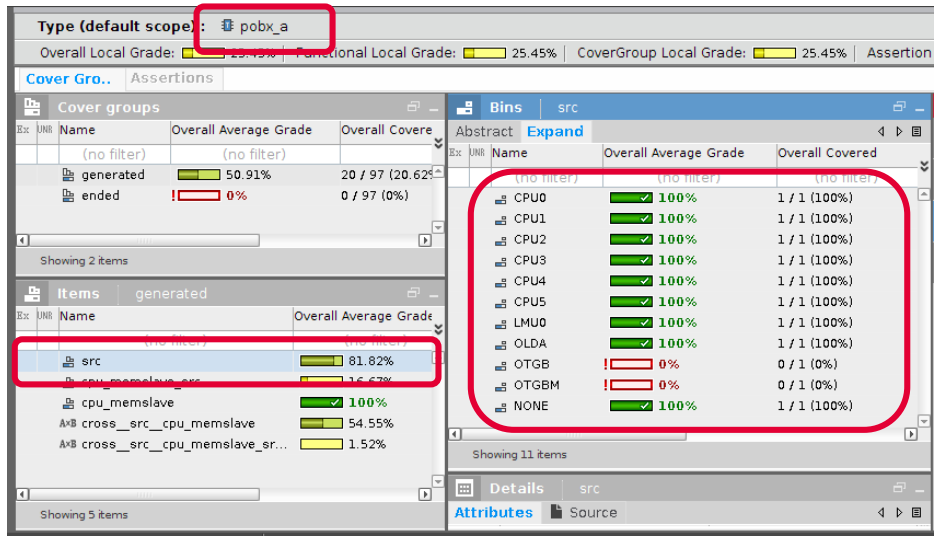


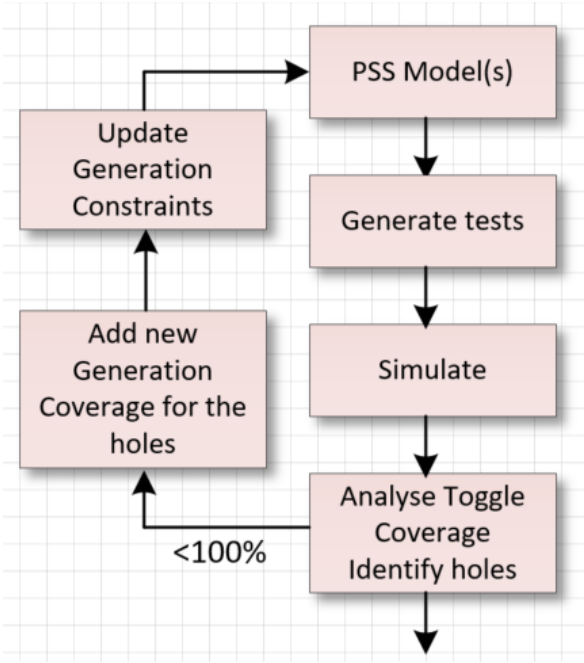
TRACE Configuration Functional Coverage

Coverage

- Toggle coverage can only be measured in simulation but Generation Functional Coverage (at interface level) gives a good idea of expected toggle coverage.







- An iterative development loop allows stimuli generation to be updated for improved toggle coverage.
- e.g.
  - toggle coverage shows the error signals are not toggling
  - Add error generation to the model and add generation coverage of this
  - Then we will know in advance of regressing whether error signal toggle coverage will be 100%

Conclusions

- Development of the PSS model flow has shown real benefits, including quick setup of new product derivatives and reduced workload for test maintenance.
- Very quick to generate 100’s of tests and associated functional coverage shows completeness.
- Easy to generate a lot of stimuli but we need robust checkers and coverage - Reuse of IP checkers enabled this

- Real benefit reusing existing PSS models of other IPs – can come from other subsystems or SOC level.
- Developers don’t need to understand full functionality of the models being utilised.

- Future Work
  - As new trace sources are developed, their accompanying PSS models can be reused for stimuli in TRACE integration verification with minimal overhead.