A SMART GENERATION OF DESIGN ATTRIBUTES
FOR VERIFICATION CLOSURE USING SPECMAN

Meirav Nitzan
Xilinx Inc.
meiravn@xilinx.com

Yael Kinderman
Cadence Design Systems Inc
yaelk@cadence.com

Efrat Gavish
Cadence Design Systems Inc
efratg@cadence.com

The problem:

Design attributes (parameters) affect the way the design behaves

- It is necessary to test the design with all relevant parameter values
- Need to make sure various combinations of parameters are checked, sometimes exhaustively
WHY NOT TEST ALL COMBINATIONS?

Not scalable.

- E.g:
  - a design with 20-30 parameters
  - each parameter may have 2 to 10 values
  - the exhaustive permutation set could reach hundreds of thousands in size, or even more.

- A huge amount of simulation time
- Covering all permutations for a bit more complex designs may not be feasible

Efficiency - turnaround time for running a regression is too long

- If a bug fix takes a week to fully verify -> a lot of idle time for designers.

Functionally, there is no functional justification to cross all values of all parameters with each other.
SOLUTION REQUIREMENT

1. Automated random generation of parameters sets, considering the following:
   • All parameters legal values and constraints
   • Dependencies between parameters

2. Avoiding repetitions of parameters sets
   • Each parameter set => re-run of the regression test suite

3. Flexibility to define either full parameters sets generation, as well as smaller parameters sets
   • Can be useful for defining nightly regressions, specific feature testing etc.

4. Proper coverage of the parameters space needs to be verified

Requirement 4 is solved by creating a config class in an OVM/UVM environment, with proper coverage model, accessible to all components
PARAMETERS GENERATION - OUTLINE

- Specific parameters constraints for a full/sub regression suite
- Parameters definition & constraints
- Parameters vectors for regression runs
- Pre Simulation Step
- Simulation and coverage collection
- Verification Testbench
- Simulator
- Simulation results
- DUT
CASE STUDY– A PARAMETERIZED DUAL PORT RAM DESIGN

Sub-Regression definitions:
BRAM_MODE is TDP
READ_WIDTH_A is 18 or 36
READ_WIDTH_B is 18 or 36
WRITE_WIDTH_A is 18 or 36
WRITE_WIDTH_B is 18 or 36
Exercise ALL combinations of:
READ_WIDTH_A x READ_WIDTH_B
WRITE_WIDTH_A x WRITE_WIDTH_B
Randomize all other parameters
COMPARING SIMPLE SYSTEM-VERILOG RANDOMIZATION TO SPECMAN BASED SOLUTION

• Running SystemVerilog randomization until 100% coverage of the sub-regression requirements were reached, yielded a significantly varied number of parameter sets – from 7 to 26

• Running the Specman Based solution with the same coverage requirements yielded 4 to 6 parameter sets

• Why the difference?
  • The Specman based solution first exhaustively generates the first cross, while randomizing other parameters. The in only needs to generate values for the second cross not generated already
  • SV randomization cannot specially consider the combinations we care about, hence reaching them may take a varying number of cycles