Training on
IP & SoC Functional Verification Methodology
Using UVM

LAB
Random Variable
Constraint Programming Advanced

Objectives

This lab is an extension of the previous one, so it also goes through the concept of random variables and constraints, but increasing the complexity. It shows how using random variables and constraints to generate a complete desire transactions and how to use random arrays to perform several transactions. New features from the constraint block are showed in this lab also.

Global Explanation

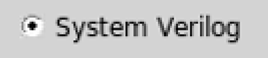
This lab will keep the same types, structures and classes from the previous one. Inheriting from the *base_labseq* (Classe already seen), the students will create a class to perform a complete addition sequence, in other words, a set of transactions that will, sequentially, perform an addition step.

After that, the students will create another class responsible to handle with a random array of the previous class, using constraint features.

It is important to remark that the data size was decreased to make easy the constraint actions visualization.

Instructions

Follow instructions given in “aedv_training_labs_intructions_for_questa.pdf”.
Open the file:
<SANDBOX>/labs-Xdays/labNN-systemverilog_constrained_random_generation_advanced /lab.sv
Select



● System Verilog

- Run the simulation

Step 0 – Analyse the random list generation

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_constrained_random_generation_advanced /lab_prog.sv

Search for “LAB-TODO-STEP-0”

- Question:
  - What does the else action do?

```
assert (tr.randomize() with {in_data inside {[0:’h81]};
                                        action == ACC})
```

Step 1 – “Default” constraint

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_constrained_random_generation_advanced /project_utils_pkg.sv

Search for “LAB-TODO-STEP-1a”

a) Add the constraint to get the action as access register by default.
b) Re-run
- Question:
  - Did you get an error? Why? (Tip: Step 0)

Search for LAB-TODO-STEP-1-b

c) Add the *soft* keyword before the statement
- Question:
  - The error was fixed? Why?
  - When you randomize a transaction without specifying the action, what do you get?

Step 2 – Complete the Addition Sequence

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_constrained_random_generation_advanced /lab_prog.sv

Search for “LAB-TODO-STEP-2a”

a) Analyse the writing and reading transactions.
- Question:
  - Is it possible to use a *foreach* loop?
  - Modify if you prefer.

Search for “LAB-TODO-STEP-2b”

b) Write a transaction to write the *cin* register (addr == ‘h03) (Tip: Look how to write the data1 and data2 register)

Search for “LAB-TODO-STEP-2c”

c) Write a transaction to read the *cout* register (addr == ‘h05)(Tip: Look how to read the dresult register)

Search for “LAB-TODO-STEP-2d”

d) Replace the random list generation with a addition sequence. (Remember the previous lab)

Extra Step 1 – Create a sequence of sequence

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_constrained_random_generation_advanced /lab_progsv

Search for "LAB-TODO-EXTRA-STEP-1a"

a) Limit the random size of the dynamic array using a constraint.

- Question:
  - o How to get a constant (non-random) size?

Search for "LAB-TODO-EXTRA-STEP-1b" and "LAB-TODO-EXTRA-STEP-1c"

b) Create the body task

c) For each *adds_dyn* element, instantiate an *addition_labseq* object (Use *foreach* loop)

d) Randomize each element and call the its *body* task (Yet inside the *foreach* loop)

Search for "LAB-TODO-EXTRA-STEP-1d"

e) Instantiate the *addition_dyn_seq* and drive all sequence in its array.

f) Re-run

Search for "LAB-TODO-EXTRA-STEP-1e"

g) Replace the dynamic array with queue

Extra Step 2 – Distribution constraint

Search for "LAB-TODO-EXTRA-STEP-2a"

a) Uncomment the constrain block that limits the result, re-run

- Question:
  - o On the distribution table, how many times do the 7 result appears?
  - o More than other values? Why?

Search for "LAB-TODO-EXTRA-STEP-2b"

b) Replace the single value statement with an interval ([**0**:**3**] := **20**, for example) **,** re-run

- Question:
  - o Which is the interval that most appears?
  - o How is the distribution inside the intervals?

Search for "LAB-TODO-EXTRA-STEP-2b"

c) Replace the ":=" operator with the ":/" one, re-run

- Question:
  - o How many times the 3 result appears? In percentage?
  - o Edit the *size_array* constraint from the *addition_dyn_arrays_labseq* to perform more interactions. The 3 value frequency is more close from the 5% now?