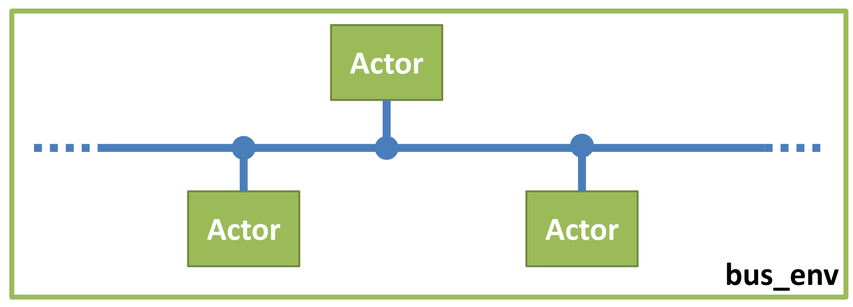
Randomizing UVM Config DB Parameters

Jeremy Ridgeway, Avago Technologies, Inc., Fort Collins, CO

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

HELP! I want full control over my config parameter!

```
class bus_env extends uvm_env;
  int num_actors;
  bus_actor_agent actors[];
  function void build_phase(uvm_phase phase);
    uvm_config_db#(int)::get(this, "", "num_actors", num_actors);
  endfunction
endclass
```



1. Randomized the non-rand parameter
2. Set a constant value on command-line
3. Actually, I'd really like to be able to set the random constraint on the command-line. Can you do that?

Option #1: Instrument parameter to use UVM Config DB access OR Randomization

```
rand int num_actors;
constraint valid { num_actors inside {[1:10]}; }
function void build_phase(uvm_phase phase);
  int rand_num_actors;
  if(uvm_config_db#(int)::get(this, "", "rand_num_actors", rand_num_actors))
    uvm_config_db#(int)::get(this, "", "num_actors", num_actors);
  else
    randomize(num_actors);
endfunction
```

- Time-consuming
- Must instrument *each* config parameter as necessary
- Modify/remove/add constraints through class inheritance and UVM Factory override

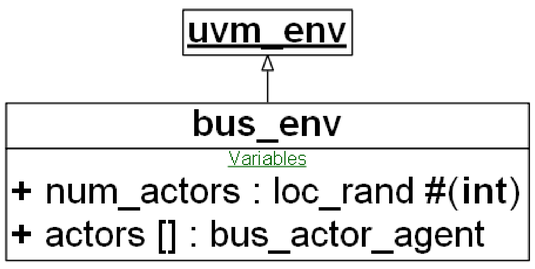
Option #2: Add Random-layer to UVM Config DB

```
loc_rand#(int) num_actors;
function void build_phase(uvm_phase phase);
  rand_config_db#(int)::get(this, "", "num_actors", num_actors);
endfunction
```

- Up-front cost to implement loc_rand and rand_config_db
- Re-usable for *every* randomize-able parameter type
- Re-usable for *every* project.

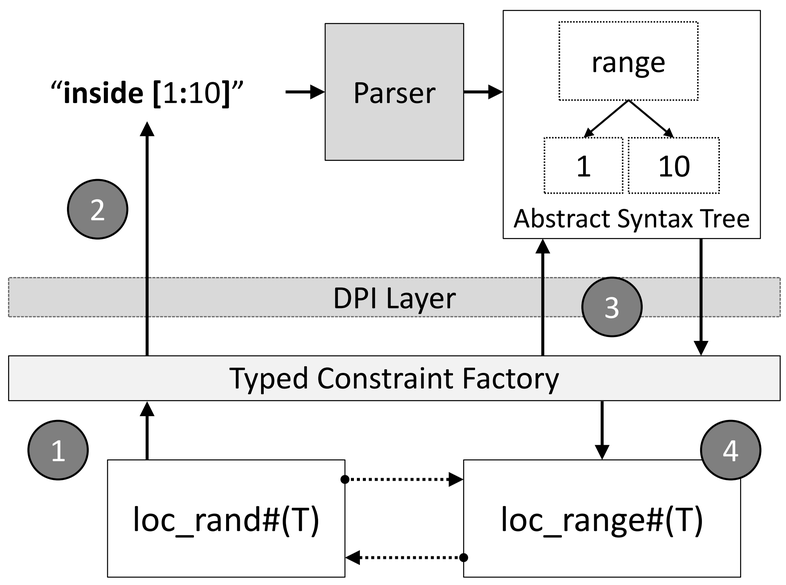
Interchangeable Constraints: loc_rand

Type-parameterized random container class is instantiated in place of 'rand' member.

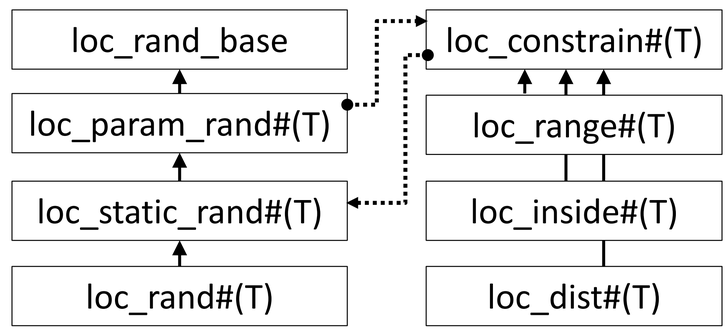


Function loc_rand.push() performs:

1. Call pkt.length.push("inside [1:10]").
2. New parser instantiated to process string and generate abstract syntax tree (AST).
3. Conversion from AST to SystemVerilog constraint using generalized containers.
4. Well-formed constrained is returned as a reference to loc_rand instance.



SystemVerilog class hierarchy for supported constraints:



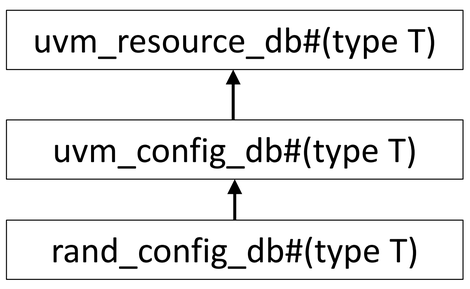
Random class containers can be instantiated on-the-fly.

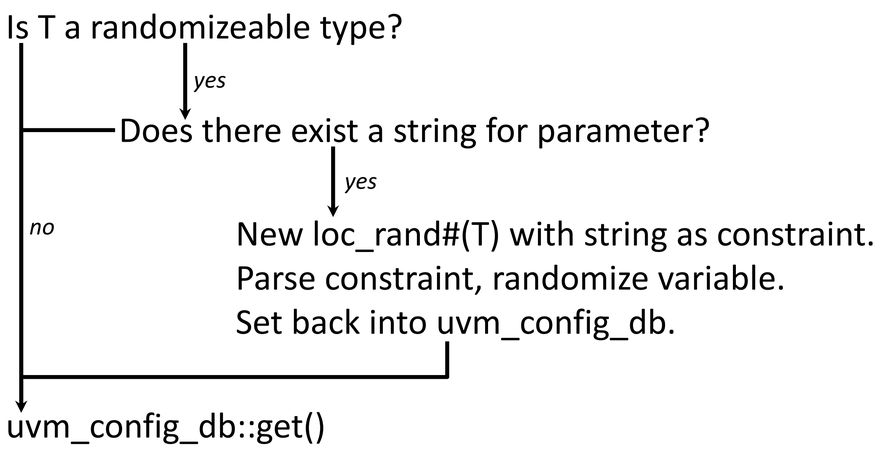
Constraints are changed with a string supplied via:
API, UVM Config DB, and Command-line Access

All that is required is the parameterized data-type: T.

Random Config DB Layer: config_rand_db

Type-parameterized random layer is extended from uvm_config_db overriding the get() function.





Now, get() checks for corresponding string type on parameter. If it exists and is a constraint, a new random variable class is instantiated, randomized, and value set in the UVM config DB. The rand_config_db class ensures randomization occurs exactly once.

With a simple change to config DB class name, to access the new random convenience layer, all get() accesses with randomize-able SystemVerilog types inherit random capability.

Constant on the command-line (or directly in config DB): 2

```
> simcmd +uvm_set_config_int= \
        '*.bus_env,num_actors,3'
> simcmd +uvm_set_config_string=\
        '*.bus_env,num_actors,3'
```

Random on the command-line (or directly in config DB): 1 3

```
> simcmd +uvm_set_config_string= \
        '*.bus_env,num_actors,inside [1:10]'
> simcmd +uvm_set_config_string=\
        '*.bus_env,num_actors,dist {3:=5, 5:=10, 7:=3}'
```