2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 28 - MARCH 3, 2022

Fnob: Command-Line Dynamic Random Generator

Haoxiang Hu, Tuo Wang

Meta Infra ASIC

accellera
SYSTEMS INITIATIVE

Intuition

- Say we have a random variable in TB:
  - Need different value of same random type?
  - Need different random type?
  - Need different value/type within the test on-the-fly?
  - Need different combination of random type/value?
- Any easy way to do it?

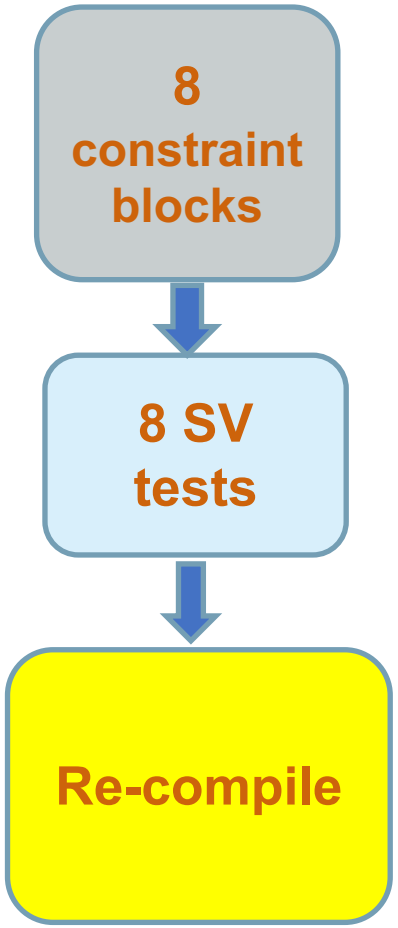
Before

When one variable and 8 testplan items

```
rand logic [11:0] src_id_x;
```

Test 0	Src_id_x within [2:8]
Test 1	Src_id_x within [3:9]
Test 2	Src_id_x within [2:9]
Test 3	Src_id_x == 2
Test 4	Src_id_x within [3:8]
Test 5	Src_id_x 70%==2, 30%==3:9
Test 6	Src_id_x 70%==9, 30%==2:8
Test 7	Src_id_x == 9

Constraint 0	{src_id_x inside {[2:8]};}
Constraint 1	{src_id_x inside {[3:9]};}
Constraint 2	{src_id_x inside {[2:9]};}
Constraint 3	{src_id_x == 2;}
Constraint 4	{src_id_x inside {[3:8]};}
Constraint 5	{src_id_x dist {2:=70, [3:9]:=30};}
Constraint 6	{src_id_x dist {9:=70, [2:8]:=30};}
Constraint 7	{src_id_x == 9;}

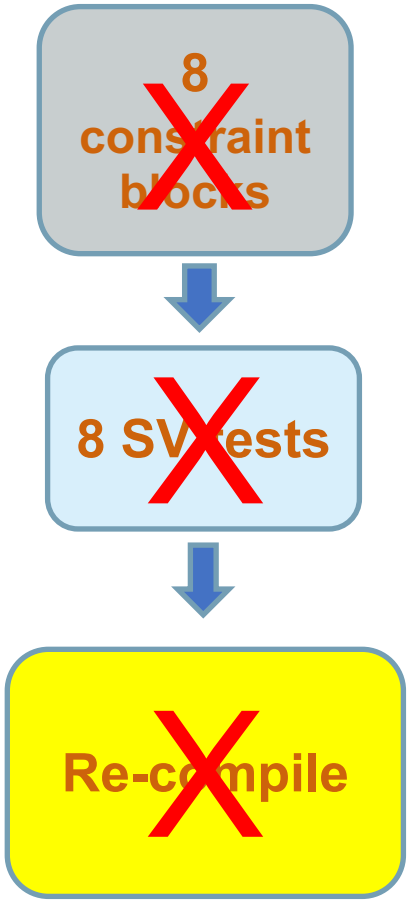


After When one variable and 8 testplan items

```
fnob_pkg::fnob src_id_x_fnob;
m_fnob_const = new(" src_id_x_fnob ", FNOB_CONST, '{"val":'{0}});
```

Test 0	Src_id_x within [2:8]
Test 1	Src_id_x within [3:9]
Test 2	Src_id_x within [2:9]
Test 3	Src_id_x == 2
Test 4	Src_id_x within [3:8]
Test 5	Src_id_x 70%==2, 30%==3:9
Test 6	Src_id_x 70%==9, 30%==2:8
Test 7	Src_id_x == 9

CLI 0	+uvm_set_config_string=*,src_id_x_fnob,unif:2:8
CLI 1	+uvm_set_config_string=*,src_id_x_fnob,unif:3:9
CLI 2	+uvm_set_config_string=*,src_id_x_fnob,unif:2:9
CLI 3	+uvm_set_config_string=*,src_id_x_fnob,constant:2
CLI 4	+uvm_set_config_string=*,src_id_x_fnob,unif:3:8
CLI 5	+uvm_set_config_string=*,src_id_x_fnob,dist:2:2:3:9_7:3
CLI 6	+uvm_set_config_string=*,src_id_x_fnob,dist:9:9:2:8_7:3
CLI 7	+uvm_set_config_string=*,src_id_x_fnob,constant:9



What's Fnob

Random Number Generator

- In-line/CLI override
- Simpler & faster

Solution

- Built-in coverage
- Mix random

Developer/User Community

- Developer friendly

APIs

Declaration — fnob_pkg::fnob m_fnob_dist;

Instantiation — m_fnob_dist= new(<FNOB_NAME>, <FNOB_TYPE>, <FNOB_PARAM>)

m_fnob_dist = new(
"m_fnob_dist", **FNOB_DIST**,
"{"**val**":{10,25,150,160,0,5},
"**prob**":'{2,6,6}});

Equivalent to:
m_fnob_dist dist {[10:25]:=2,
[150:160]:=6,
[0:5]:=6};

Generation — m_fnob_dist.gen()

Override

- +uvm_set_config_string=*,m_fnob_dist,dist:0:10:100:1000_5:1
- uvm_config_db#(string)::set(null,"*", "m_fnob_dist", "unif:10:200");

OVERRIDE

Two types of override

In-line Override

- UVM config_db based.
- Re-compile needed.
- Support multiple override per test.

Example:

```
uvm_config_db#(string)::set(null, "*", "m_fnob_unif", "unif:0:10");
```

Command Line Override(CLI)

- UVM config_db CLI based.
- No recompile needed.
- Override once per test.

Example:

```
+uvm_set_config_string=\*,m_fnob_unif,unif:10:100
```

Random Types

FNOB_CONST
constant value

FNOB_UNIF
uniform random value

FNOB_C_UNIF
cyclic uniform random value

FNOB_IN_LIST
value from defined list, equivalent to “inside”

FNOB_DIST
random value based on distribution; equivalent to “dist”

FNOB_PATN
value following fixed pattern as defined

FNOB_LOG
random value from logarithmic distribution

FNOB_C_PATN
cyclic value from fixed pattern

FNOB_NORM
random value from normal distribution

FNOB_INV_NORM
random value from inverse normal distribution

And more by developers …

Fnob Coverage

Consecutive cover-group:

For Fnob types that has fixed length of params

```
//-----------------------------------
covergroup fnob_cg;
  option.per_instance = 1;
  option.goal         = 100;
  option.comment      = "fnob_cg";

  fnob_rand_cg:coverpoint m_gen{
    bins val_min = {m_unif_min};
    bins val_max = {m_unif_max};
    bins val_mid = {m_unif_mid};
  }

endgroup // fnob_cg
```

Non-consecutive cover-group:

For Fnob types that has variable length of params

```
covergroup fnob_pattern_cg(int val) with function sample(int cp);
  option.per_instance = 1;
  option.goal         = 100;
  option.comment      = "fnob_cg";

  fnob_rand_cg:coverpoint cp{
    bins val_all = {val};
  }

endgroup
fnob_pattern_cg cg[];

//-----------------------------------------------
function new(string name="", T params[$]);

  super.new(name);

  m_vals = params;
  cg = new[m_vals.size()];

  for (int ii=0; ii<m_vals.size(); ii++) begin
    cg[ii] = new(m_vals[ii]);
  end

endfunction // new
```

Fnob Multi

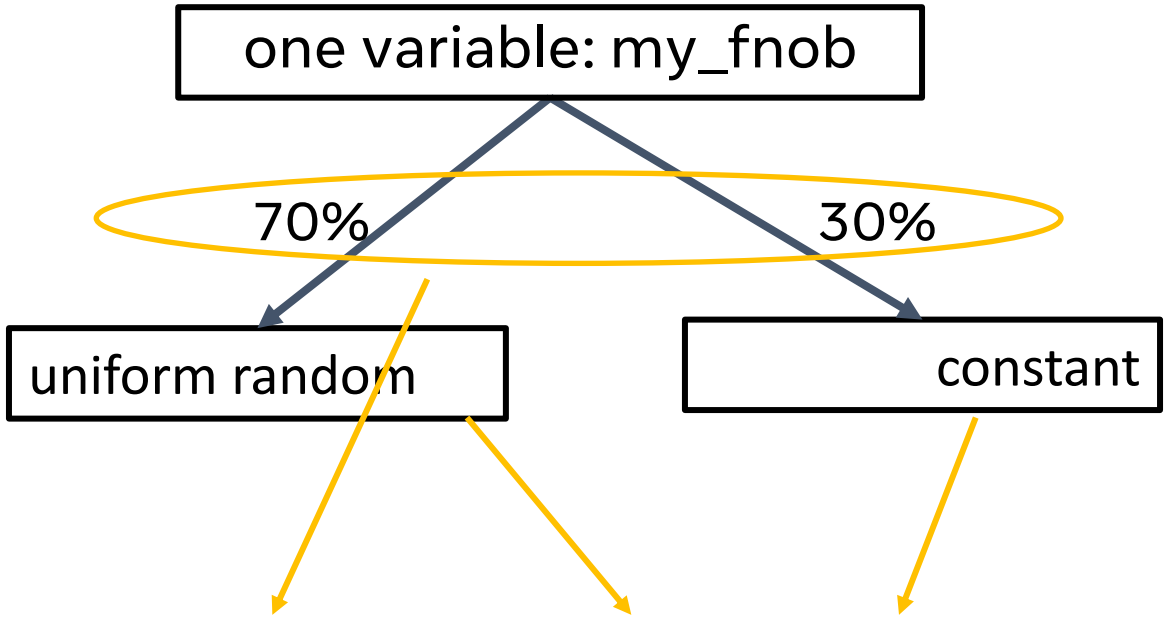
We have single type of random per variable:

- uniform random
- constant
- distribution
- pattern

.....

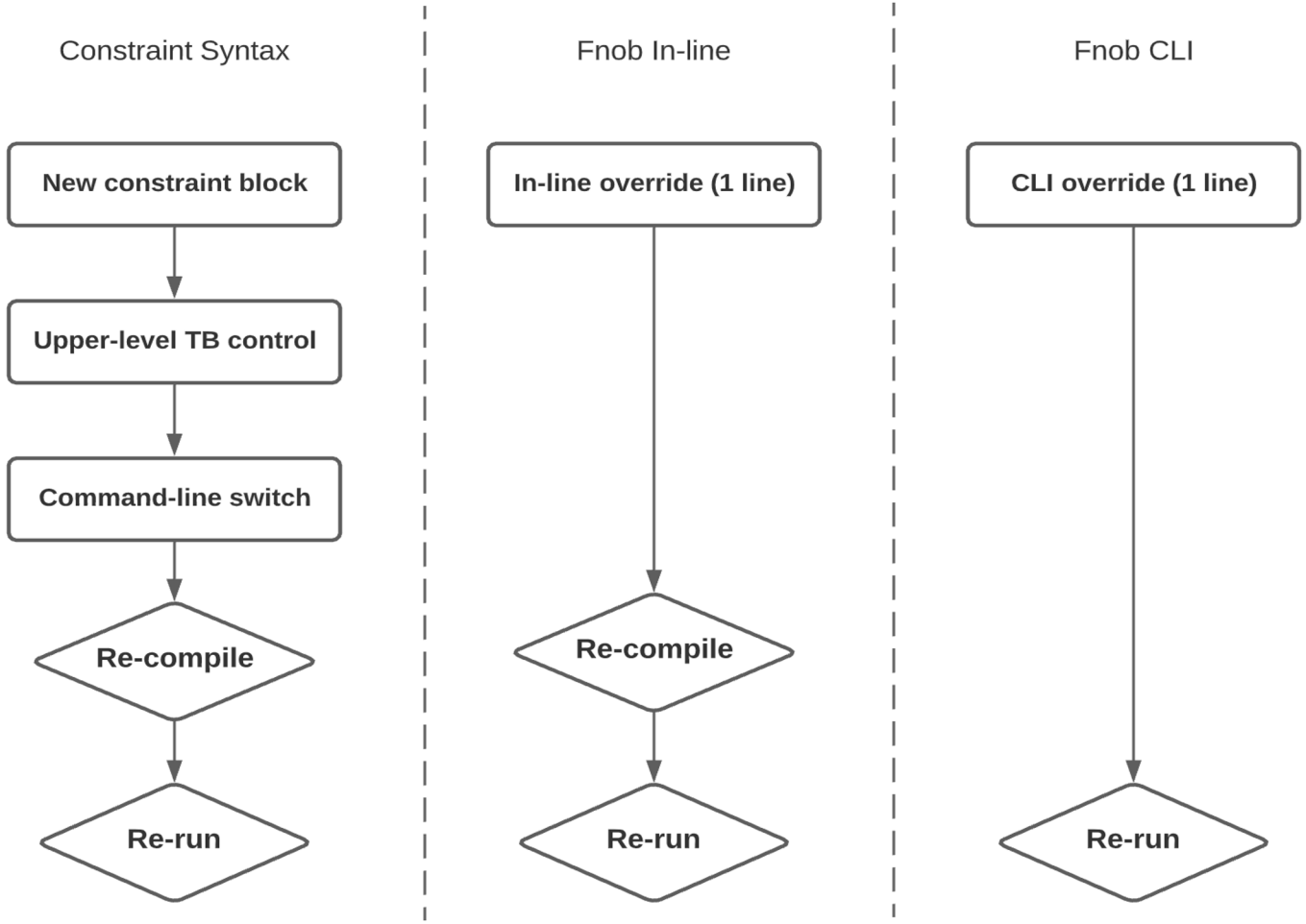
What if we want to mix them within same test?

Example: 70% uniformly fast; 30% constant slow;




```
m_delay_fnob = fnob#(bit[63:0])::new_multi("m_delay_fnob","(dist:0:0:1:1_7:3)(unif:5:20)(constant:100)")
```

Flow Reduction



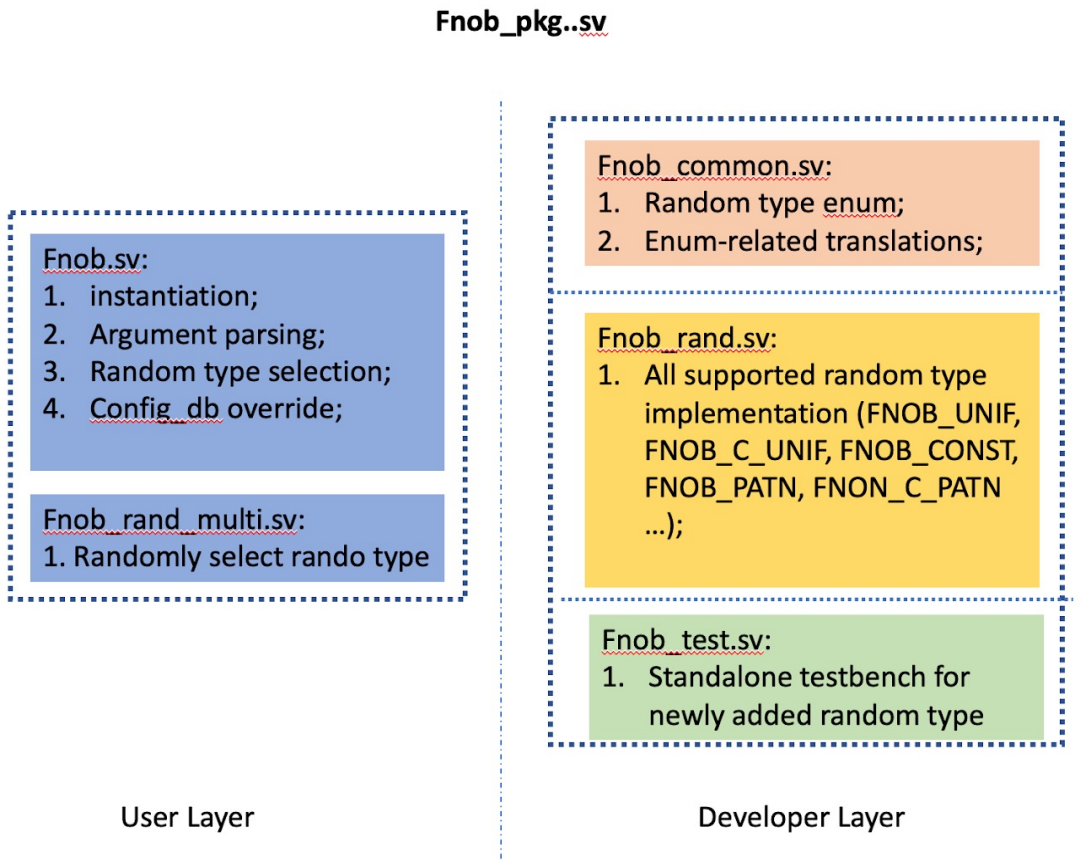
- Reduce extra coding;
- Save compile time;
- Speed-up test dev;

Performance Improvement

- “Constraint” traverse entire value space by size of data type;
- “Fnob” traverse limited value space by pre-defined system function;

Random Keyword	Performance Test				
	Number of Call	*Constraint: vendor 1*	*Fnob: vendor 1*	*Constraint: vendor 2*	*Fnob: vendor 2*
inside	100,000,000	127s	5s	117s	1s
dist	100,000,000	69s	4s	119s	2s
constant	100,000,000	63s	4s	116s	3s
normal	100,000,000	68s	8s	118s	2s

Fnob Developer Interface



Questions