

Interchangeable SystemVerilog Random Constraints

Jeremy Ridgeway
LSI Corporation, Inc.

March 24, 2014
SNUG Silicon Valley

Agenda

SystemVerilog Constraints

Dynamic Constraints

Overview

Implementation

Performance Review



SystemVerilog Random Constraints

- Language is declarative:
 - Symbols
 - hard-coded into constraint
 - must be resolved prior to run-time
- Blocks:
 - Operate together via Boolean AND connective
`len >= 0 && len <= 1500 && len > 1000`
 - Overridden through inheritance
- Manipulation
- Performance

```
class packet;  
    rand bit[15:0] len;  
    constraint valid_len {  
        len inside {[0:1500]};  
    }  
endclass  
  
class long_packet extends packet;  
    constraint long_len {  
        len > 1000; // AND valid_len  
    }  
endclass  
  
class error_packet extends packet;  
    constraint valid_len {  
        // override  
        len inside {[1501:1600]};  
    }  
endclass
```

SystemVerilog Random Constraints

- Language
- Blocks
- Manipulation via extension:
 - Inheritance chain is known
 - Test bench hierarchy is known
 - UVM Factory used to override in new test
 - Re-compile is performed
- Performance
 - Fast, tightly integrated in simulator
 - Anything else imparts penalty

```
class packet;  
    rand bit[15:0] len;  
    constraint valid_len {  
        len inside {[0:1500]};  
    }  
endclass  
  
class long_packet extends packet;  
    constraint long_len {  
        len > 1000; // AND valid_len  
    }  
endclass  
  
class error_packet extends packet;  
    constraint valid_len {  
        // override  
        len inside {[1501:1600]};  
    }  
endclass
```

Example from [2]

SystemVerilog Random Constraints

Questions

- Extending global/static classes?
- Changes to random variable?
 - Verification engineer
 - Designer working on a bug fix?
- Wouldn't it be nice to just set it on the command line?

```
class global_config;  
  rand sim_mode;  
  constraint valid_c {  
    dist {  
      0 := 40, // "normal"  
      1 := 50, // "err injection"  
      2 := 10  // special case  
    }  
  }  
  
  function new();  
    std::randomize(sim_mode);  
  end  
endclass  
  
// Single global instance in sim  
global_config cfg = new;
```

Constraints are Set in Code with Some Flexibility

Agenda

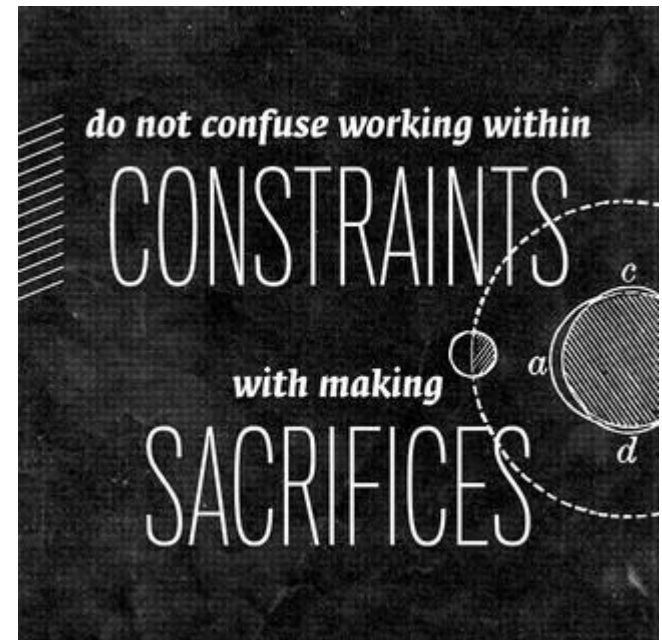
SystemVerilog Constraints

Dynamic Constraints

Overview

Implementation

Performance Review



Picture from [3]

Dynamic Random Constraints

- A string parsed at run-time
 - Associated with lvm_rand object
 - Dynamically changed via UVM Resource Database
- Symbols are:
 - Hard coded into the constraint
 - Variable/function reference illegal


```
set_constraint("", "PACKET_LEN",
               "inside [min:max]");
```
 - String composition is OK


```
set_constraint("", "PACKET_LEN",
               $sformatf("inside [%0d:%0d]",
                       min, max));
```

```
class packet;
  lvm_rand#(bit[15:0]) len;
  function new();
    len = new("PACKET_LEN", this);
    len.push("inside [0:1500]");
  endfunction // default constraint
endclass

class long_packet extends lvm_test;
  function void build_phase();
    set_constraint("", "PACKET_LEN",
                  "inside [1000:1500]");
  endfunction
endclass

class error_packet extends lvm_test;
  function void build_phase();
    set_constraint("", "PACKET_LEN",
                  "inside [1501:1600]");
  endfunction
endclass
```

Dynamic Random Constraints

- A string parsed at run-time
- Symbols
- Command line connection
 - Load into uvm_config_db


```
> simv +uvm_set_config_string=\
          "*", "PACKET_LEN", "1510"
> simv +uvm_set_config_string=\
          "*", "PACKET_LEN", \
          "dist {1:=1, 1000:=1,
          1500:=1}"
> simv +uvm_set_config_string=\
          "*", "PACKET_LEN", \
          "inside ['h3e8:15'o37_20]"
```
- Performance

```
class packet;
  lvm_rand#(bit[15:0]) len;
  function new();
    len = new("PACKET_LEN", this);
    len.push("inside [0:1500]");
  endfunction // default constraint
endclass

class long_packet extends lvm_test;
  function void build_phase();
    set_constraint("", "PACKET_LEN",
                  "inside [1000:1500]");
  endfunction
endclass

class error_packet extends lvm_test;
  function void build_phase();
    set_constraint("", "PACKET_LEN",
                  "inside [1501:1600]");
  endfunction
endclass
```

Dynamic Random Constraints

- A string parsed at run-time
- Symbols
- Command line connection
 - Load into uvm_config_db
 - Lazy set up


```
> simv +PACKET_LEN=1510
> simv +PACKET_LEN=\
    "dist {1:=1, 1000:=1,
        1500:=1}"
> simv +PACKET_LEN=\
    "inside ['h3e8:15'o37_20]"
```
- Performance

```
class packet;
  lvm_rand#(bit[15:0]) len;
  function new();
    len = new("PACKET_LEN", this);
    len.push("inside [0:1500]");
  endfunction // default constraint
endclass

class long_packet extends lvm_test;
  function void build_phase();
    set_constraint("", "PACKET_LEN",
                  "inside [1000:1500]");
  endfunction
endclass

class error_packet extends lvm_test;
  function void build_phase();
    set_constraint("", "PACKET_LEN",
                  "inside [1501:1600]");
  endfunction
endclass
```

Dynamic Random Constraints

- A string parsed at run-time
- Symbols
- Command line connection
- Performance
 - A penalty is assessed for each uvm_resource_db access
 - This penalty may become too cumbersome for simulation
 - Where and how dynamic constraints are applied are key

```
class packet;
    lvm_rand#(bit[15:0]) len;
    function new();
        len = new("PACKET_LEN", this);
        len.push("inside [0:1500]");
    endfunction // default constraint
endclass

class long_packet extends lvm_test;
    function void build_phase();
        set_constraint("", "PACKET_LEN",
            "inside [1000:1500]");
    endfunction
endclass

class error_packet extends lvm_test;
    function void build_phase();
        set_constraint("", "PACKET_LEN",
            "inside [1501:1600]");
    endfunction
endclass
```

Constraints are Optimized for Flexibility

Clocking Example

Random initial delay and customized phase

```

module tb;
  import uvm_pkg::*;
  import lvm_pkg::*;
  bit test_clk;

  lvm_rand #(int) test_clk_del;
  lvm_rand #(int) test_clk_phase_ps;

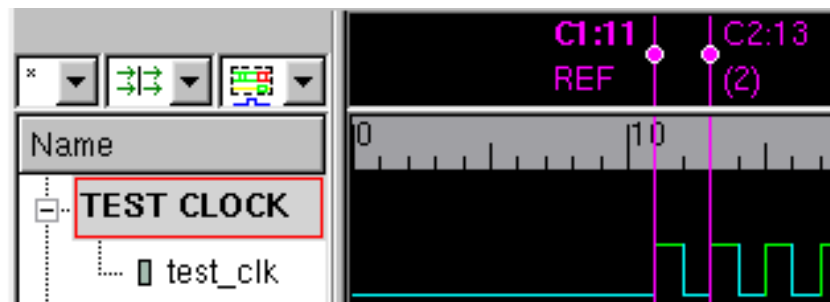
  initial begin
    test_clk_del = new("TEST_CLK_DEL");
    test_clk_phase_ps = new("TEST_CLK_PHASE_PS");
    test_clk_del.push("10");
    test_clk_phase_ps.push("1000");

    m_do_config_settings(); // copied from uvm_root
    test_clk = 0;           // initial value

    fork
      #(test_clk_del.current())
        forever #(real'(test_clk_phase_ps.current()) / 1000.0)
          test_clk = ~test_clk;

    join
  end
endmodule

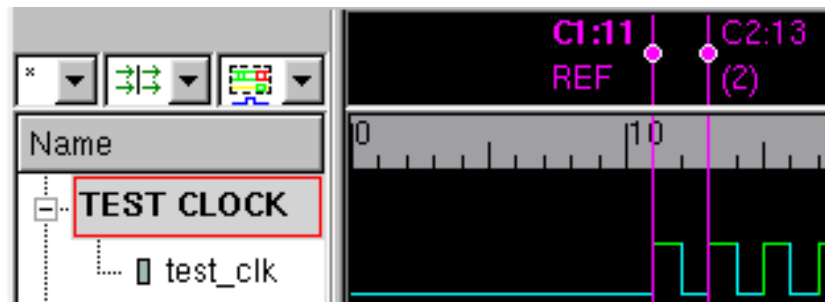
```



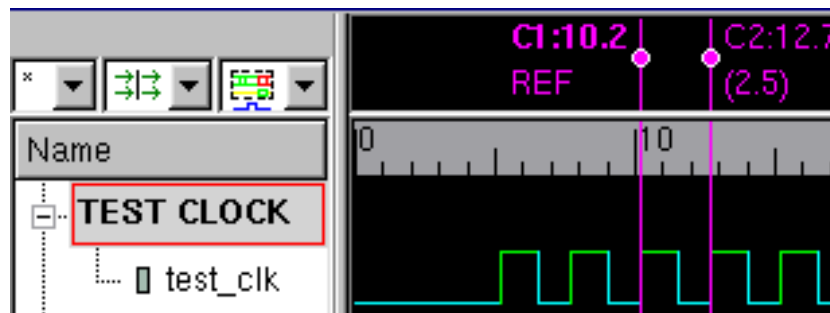
Clocking Example

Random initial delay and customized phase

➤ simv



➤ `simv +UVM_SET_CONFIG_STRING="*", "TEST_CLK_DEL", 'inside [1:5]' \`
`+UVM_SET_CONFIG_STRING="*", "TEST_CLK_PHASE_PS", '1250'`

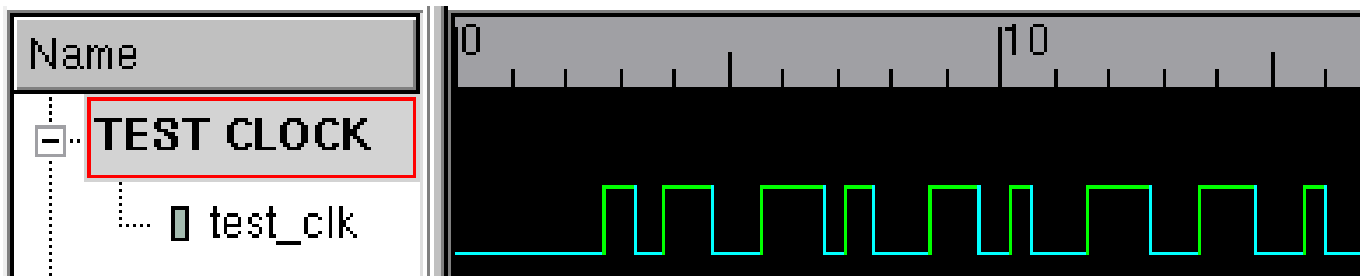


Clocking Example

Random initial delay and random phase

```
module tb;

...
fork
  #(test_clk_del.current())
  forever #(real'(test_clk_phase_ps.next()) / 1000.0)
    test_clk = ~test_clk;
join
end
endmodule
```



```
➤ simv +UVM_SET_CONFIG_STRING="*", "TEST_CLK_DEL", 'inside [1:2]' \
  +UVM_SET_CONFIG_STRING="*", "TEST_CLK_PHASE_PS", 'inside [400:1400]'
```

Agenda

SystemVerilog Constraints

Dynamic Constraints

Overview

Implementation

Performance Review



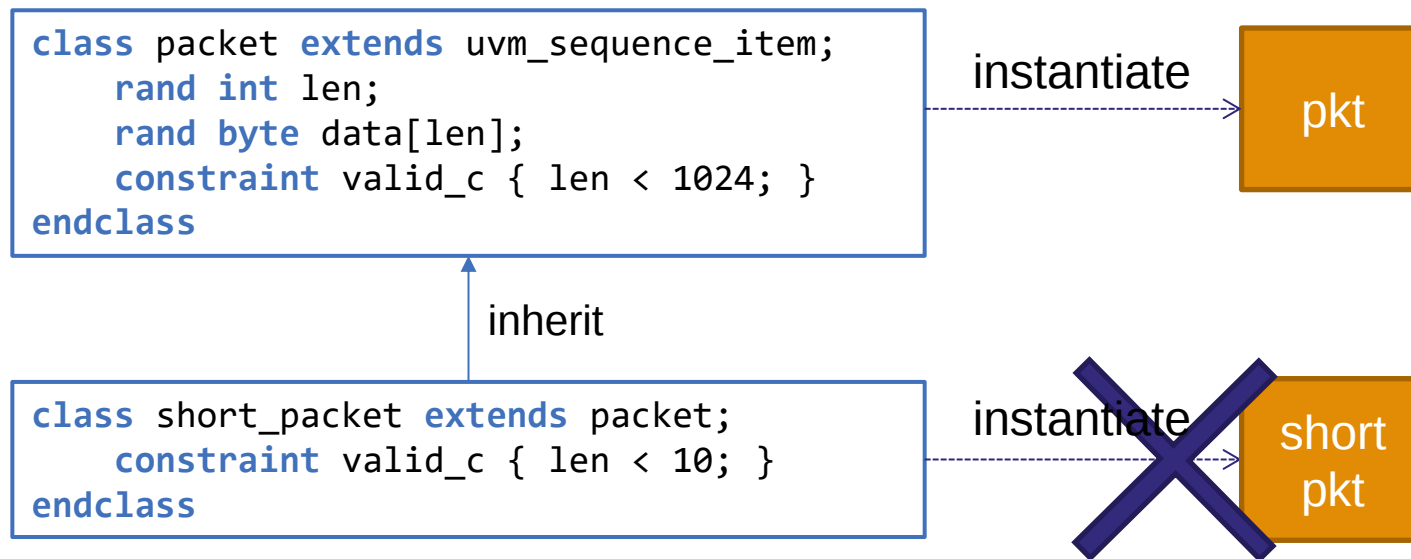
Implementation

- Container classes – SV
 - Random value
 - Constraint
- Front-end components
 - Parser – C++
 - Constraint factory – SV
- Global dictionary – UVM resource database

Constraints

Observation 1 (simple, obvious)

- SystemVerilog constraints are declarative



- Pertinent only when instantiated
 - No instance of `short_packet` = only original constraint on packet

Constraints must be instantiated to have an effect

Constraints

Observation 2 (probably only a little less obvious)

- SystemVerilog constraints work over references

```
class short_pkt_seq extends uvm_sequence;  
    rand packet pkt; // reference to an instance of type packet  
    constraint valid_c { pkt.len < 10; }  
endclass
```

- Compilation-time rules apply to the referenced class
 - pkt must be of class lineage type packet
 - len must be a public member of packet
 - len must be of type to compare to a constant int value = 10

Constraints may be separate from the randomized value

Separate the Random Value

Dynamic constraints implementation

```
class packet extends  
    uvm_sequence_item;  
    rand LengthVal len;  
    rand byte data[];  
endclass
```

```
class LengthVal;  
    rand int val;  
    constraint valid_c {  
        val < 1024;  
    }  
endclass
```

inherit

```
class LengthVal2 extends LengthVal;  
    constraint valid_c {  
        val < 10;  
    } // overriding constraint  
endclass
```

Separate the Constraint

Dynamic constraints implementation

```
class packet extends  
    uvm_sequence_item;  
    rand LengthVal len;  
    rand byte data[];  
endclass
```

```
class LengthVal;  
    rand int val;  
    LengthCon c_ref;  
endclass
```

```
class LengthCon;  
    LengthVal r_ref;  
    constraint valid_c {  
        r_ref.val < 1024;  
    }  
endclass
```

inherit

```
class LengthCon2 extends LengthCon;  
    constraint valid_c {  
        rand_ref.val < 10;  
    } // overriding constraint  
endclass
```

Container-ize

Dynamic constraints implementation

```
class packet extends
    uvm_sequence_item;
    rand MyVal#(int) len;
    rand byte data[];
endclass
```

```
class MyVal#(type T = int);
    rand T val;
    MyCon#(T) c_ref;
endclass
```

```
class MyCon#(type T = int);
    MyVal#(T) r_ref;
    int max;
    constraint valid_c {
        r_ref.val < max;
    }
endclass
```

inherit

```
class MyCon2#(type T = int)
    extends MyCon#(T);
    constraint valid_c {
        rand_ref.val < ??;
    } // overriding constraint
endclass
```

Random Container

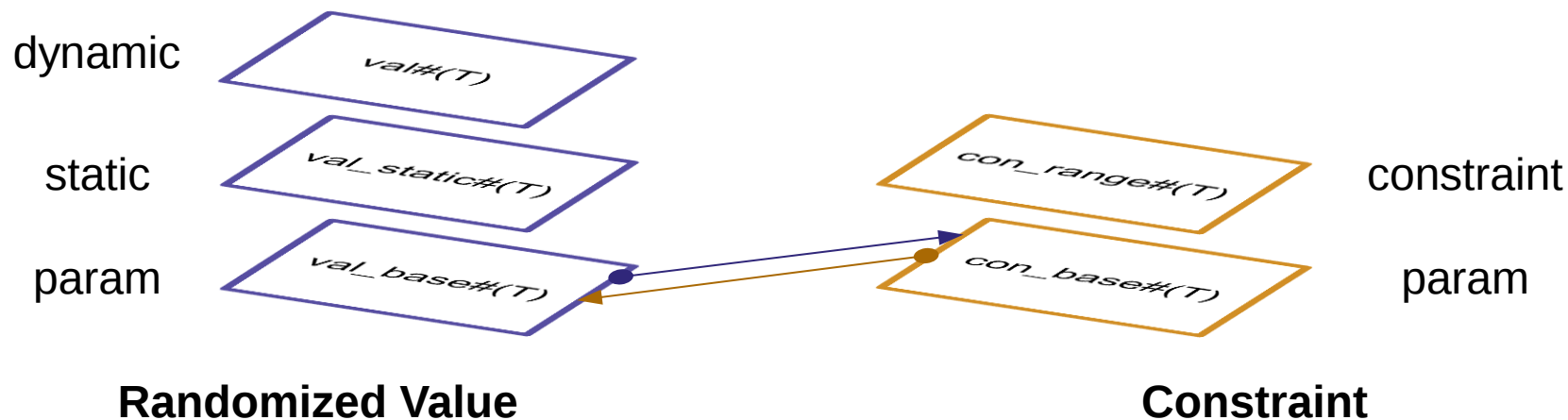
Range constraint: from min – max, inclusive

```
class MyVal#(type T = int);  
  rand T val;  
  MyCon#(T) c_ref;  
endclass
```

references

```
class MyCon#(type T = int);  
  MyVal#(T) r_ref;  
  int min, max;  
  constraint valid_c {  
    r_ref.val >= min;  
    r_ref.val <= max;  
  }  
endclass
```

Container Inheritance Stacks



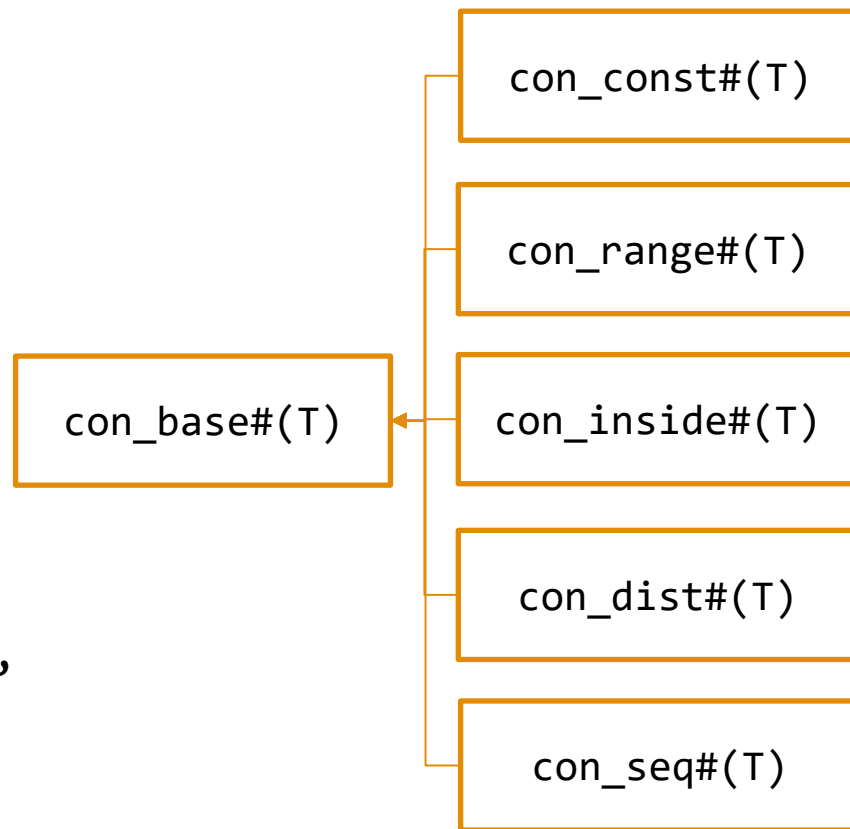
- Parameterized base class


```
rand T val
con_base#(T) c_ref;
```
- Static layer has only API
- Dynamic layer accesses global resource database
- Parameterized base class


```
val_base#(T) v_ref;
```
- Constraint layer applies single constraint type

Constraint Container Types

- Const – constant value
`val == 4`
- Range – contiguous range
`val inside [1:4]`
- Inside – value set
`val inside { 1, 2, 4 }`
- Dist – weight value set
`val dist { 1 := 10, 2:= 10,
 4 := 80 }`
- Seq – linear value sequence
`val seq [ 0, 4, 1 ]`



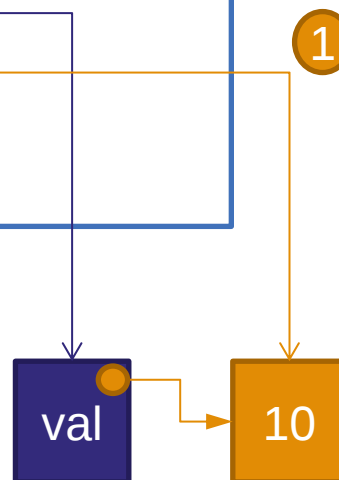
See paper for grammar

Implementation

- Container classes – SV
 - Random value
 - Constraint
- Front-end components
 - Parser – C++
 - Constraint factory – SV
- Global dictionary – UVM resource database

Front-end Processing

```
class A;
  rand MyVal#(logic [4:0]) val;
  function new;
    val = new;
    val.push("10");
  endfunction
endclass
```

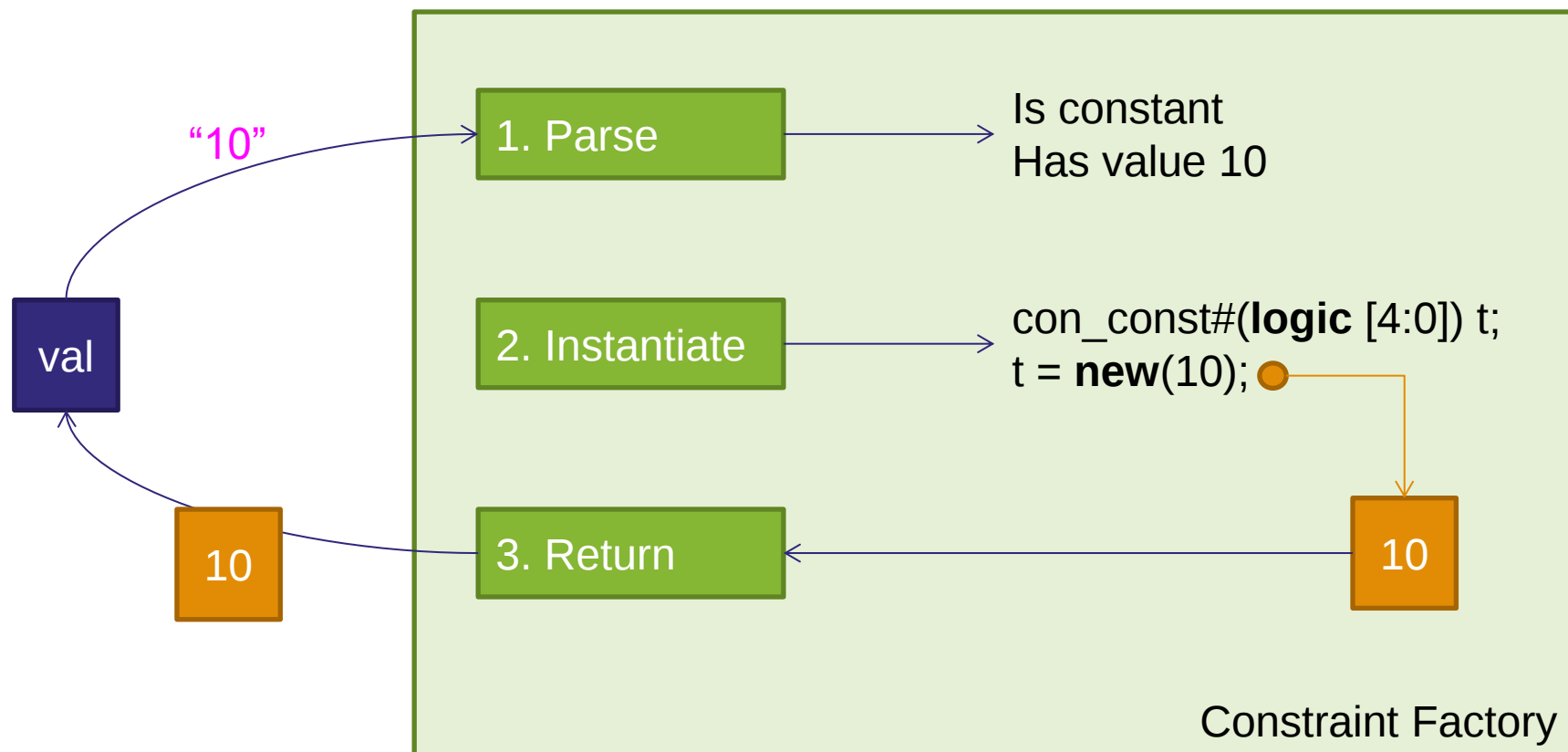


"inside [0:1024]"

- Have a class with a random value
 - 1 Want to directly put a constraint (default)
 - 2 Want to magically put a new constraint

Constraint Instantiation

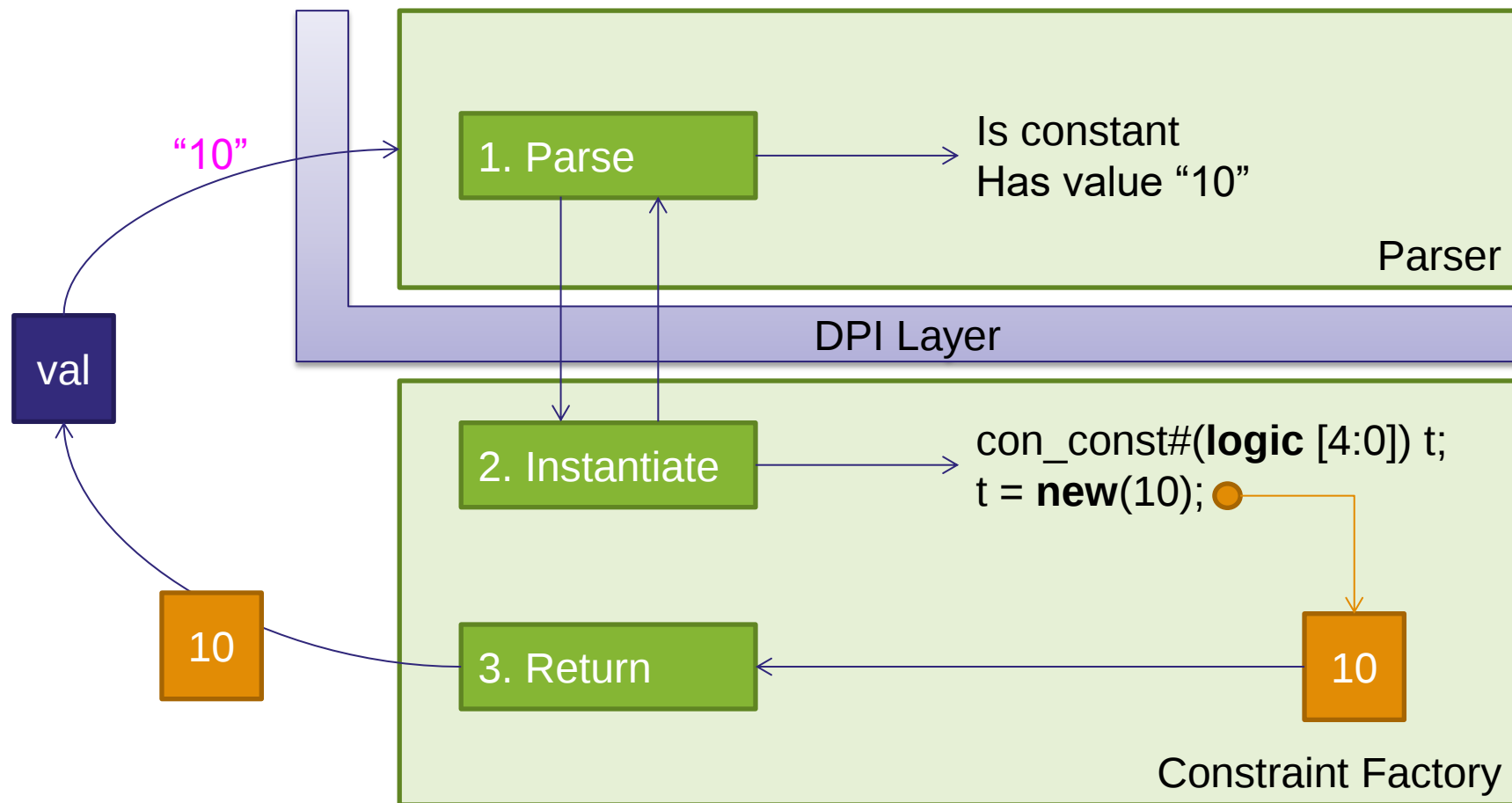
Directly through push API



- Instantiate and return must be in SystemVerilog

Constraint Instantiation

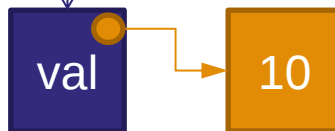
Directly through push API



- Lex/Bison generated parser

Front-end Processing

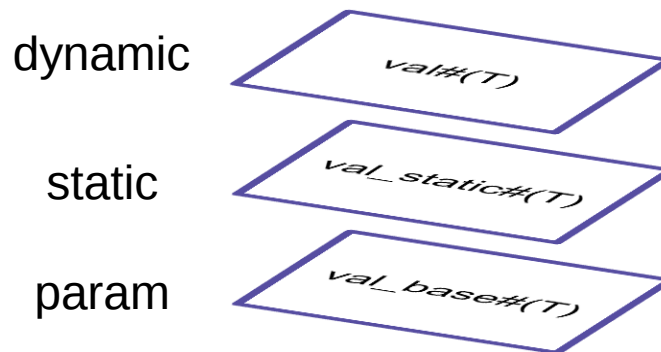
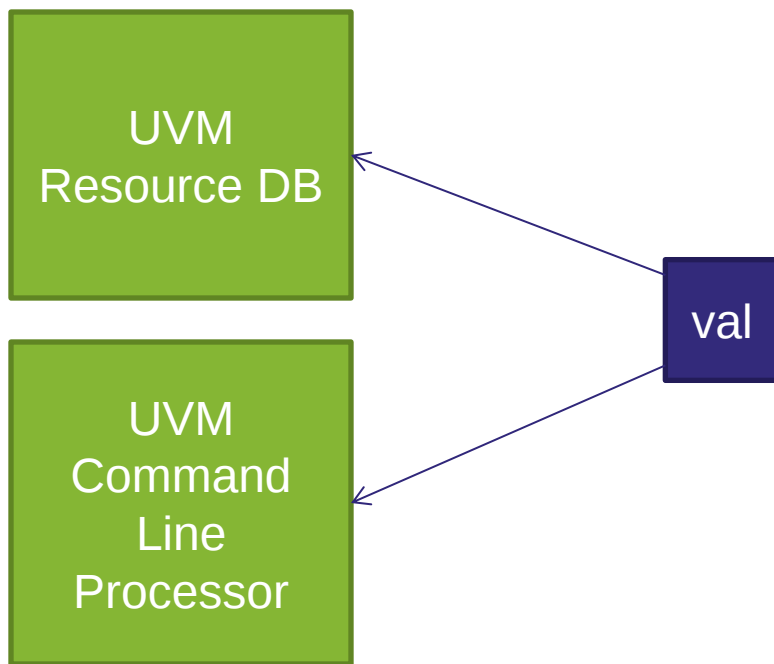
```
class A;
  rand MyVal#(logic [4:0]) val;
  function new;
    val = new;
    val.push("10");
  endfunction
endclass
```



②
"inside [0:1024]"

- Have a class with a random value
 - Want to directly put a constraint (default)
 - ② Want to magically put a new constraint

Constraint Sources

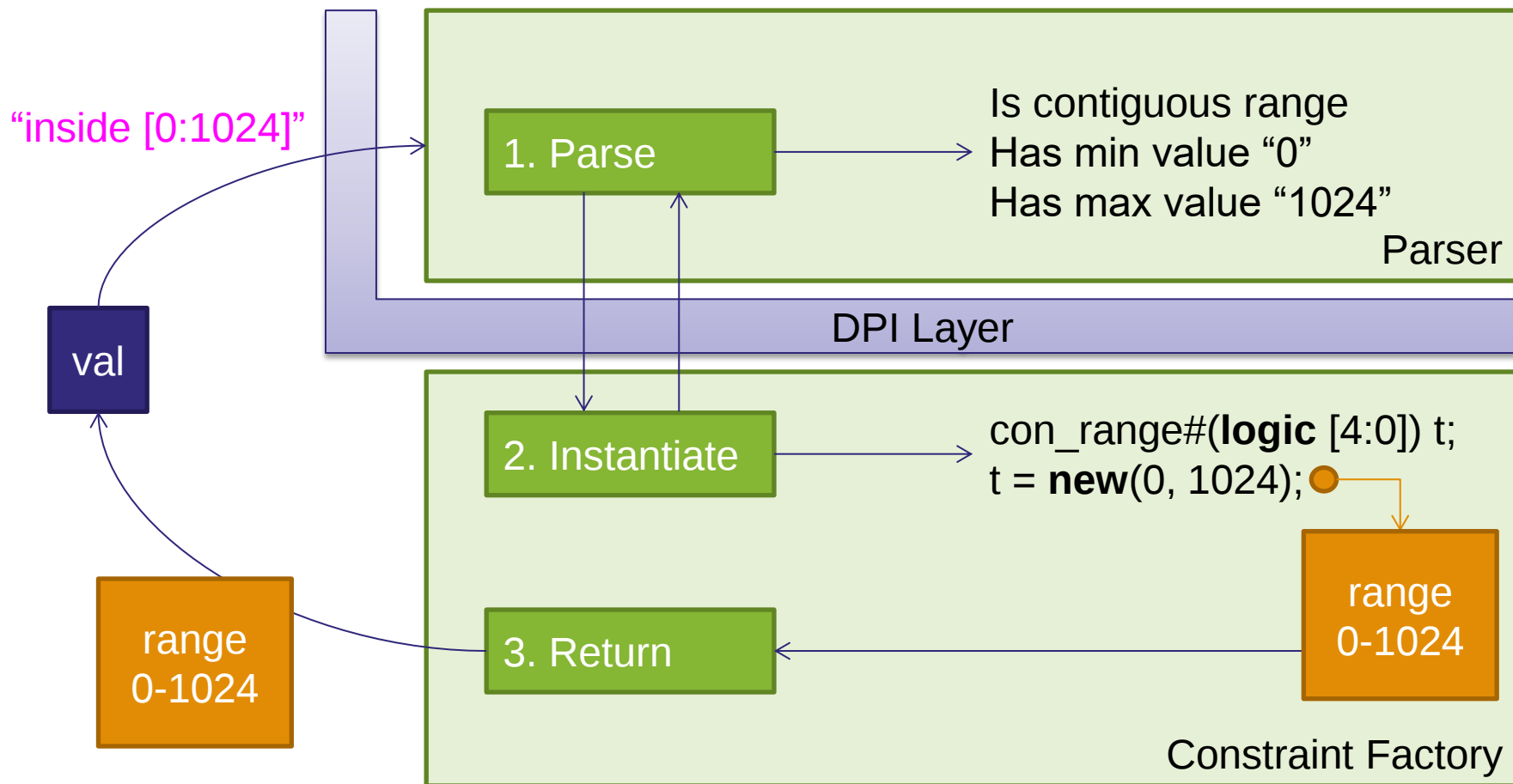


Randomized Value

- Dynamic constraints only
- Val has:
 - Name
 - Scope
- Lookup *scope.name* in global resources for:
“inside [0:1024]”

Constraint Instantiation

Dynamic via global access



- Found constraint instantiated the same as direct access

Agenda

SystemVerilog Constraints

Dynamic Constraints

Overview

Implementation

Performance Review

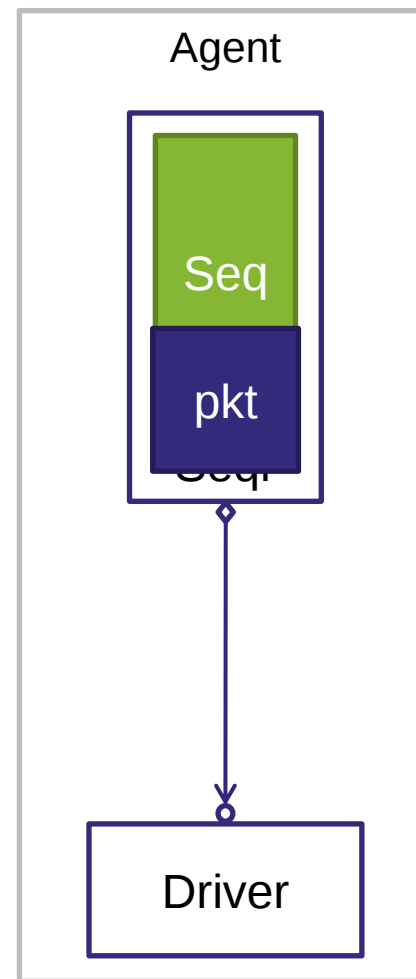


Picture from [5]

Performance Considerations

Simple test bench

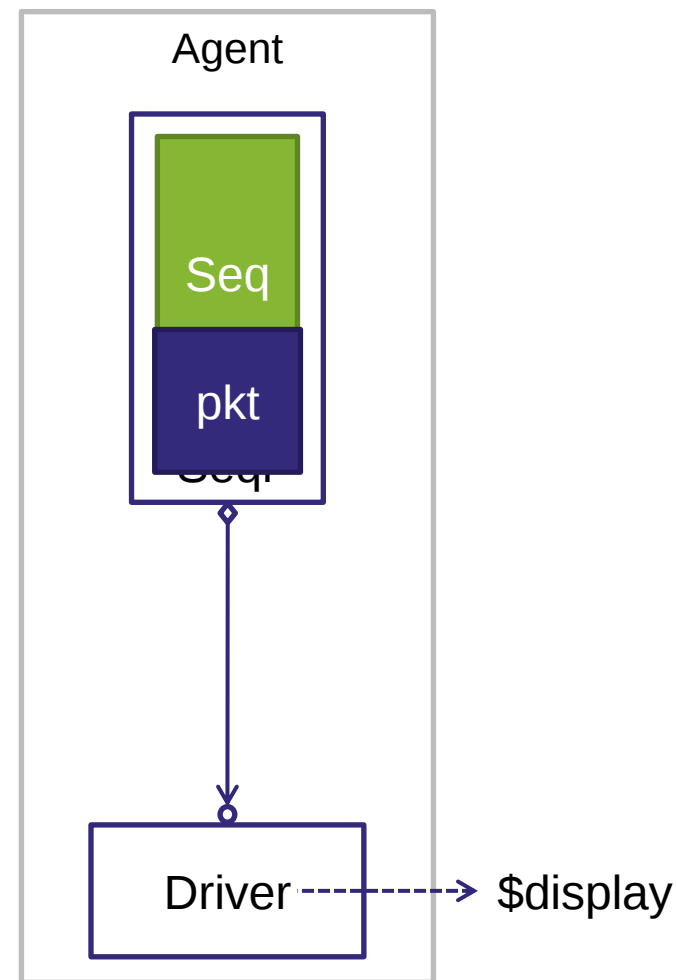
- 1 Agent
- 1 Sequence



Performance Considerations

Simple test bench

- 1 Agent
 - 500 x
 - 5,000 x
 - 50,000 x
 - 500,000 x
 - 5,000,000 x
- Tests
 - Distribution
 - Inside set
 - Inside range

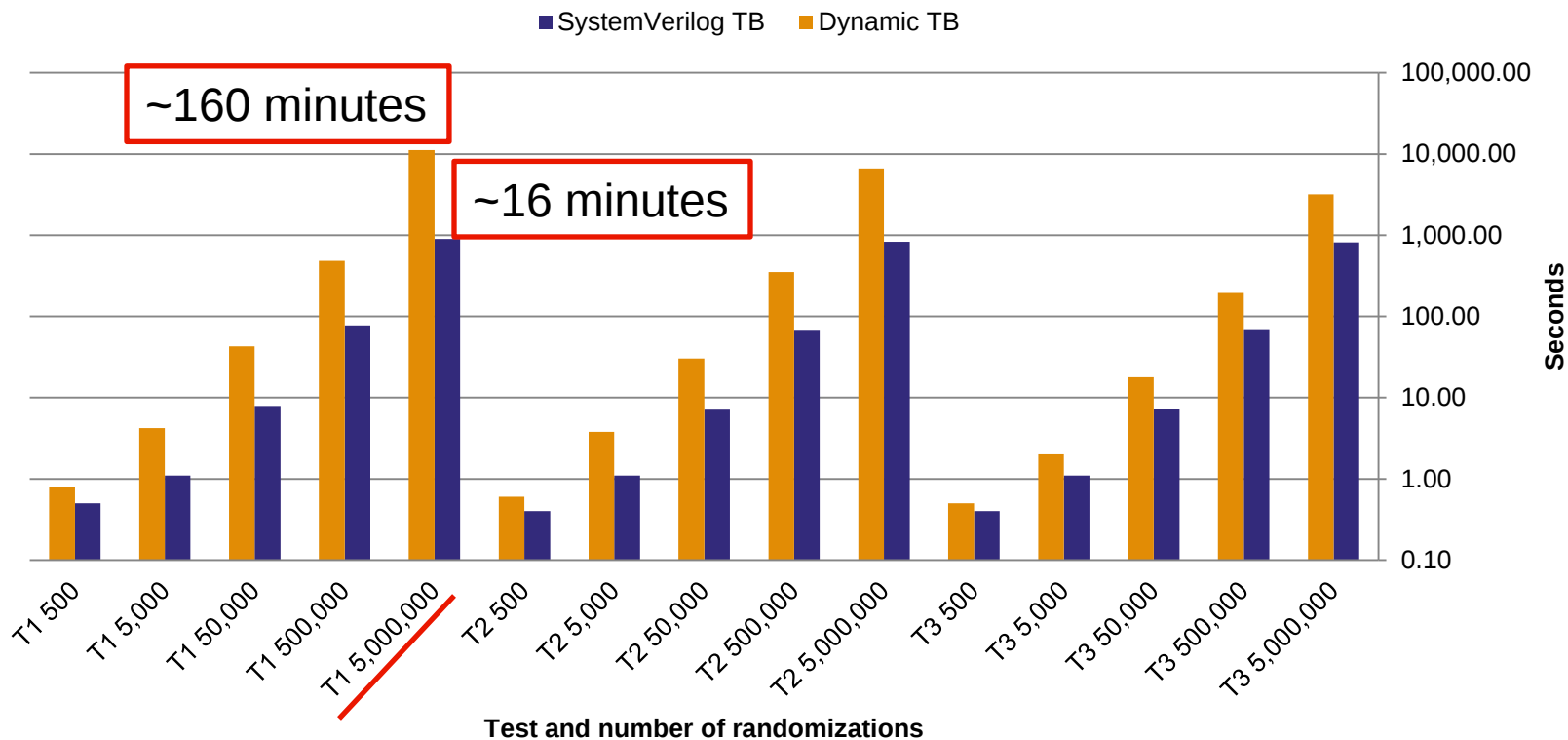


See paper for more detail

Performance

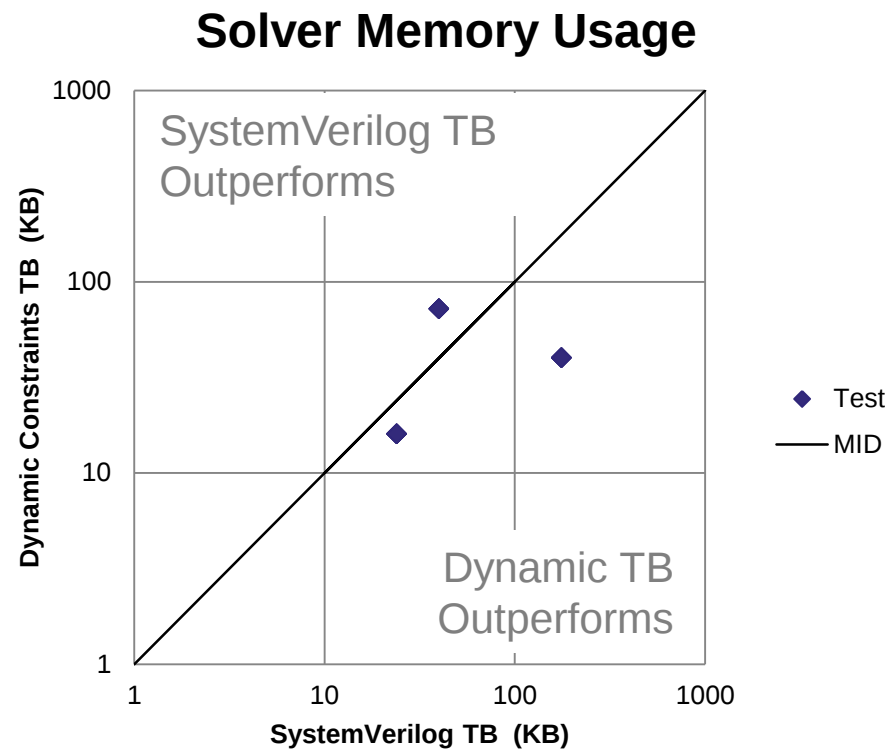
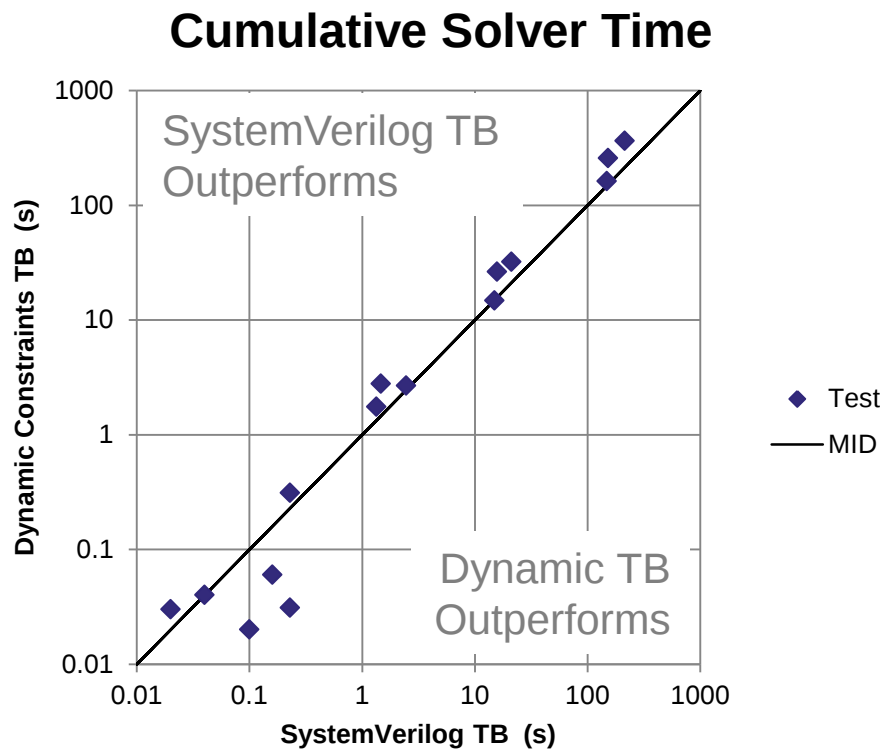
- Cost is significant for *fully* interchangeable constraints
 - UVM database access!

Cumulative Simulation Time



Performance

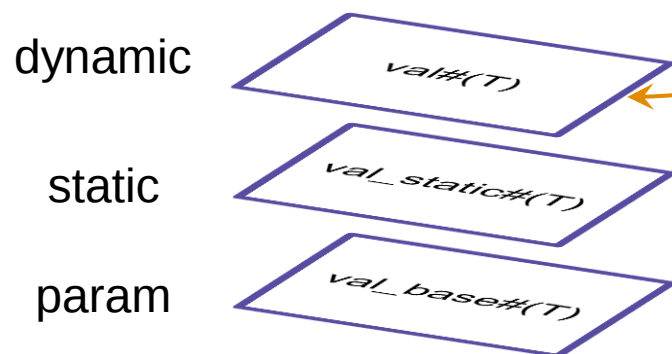
- We found solver performance comparable



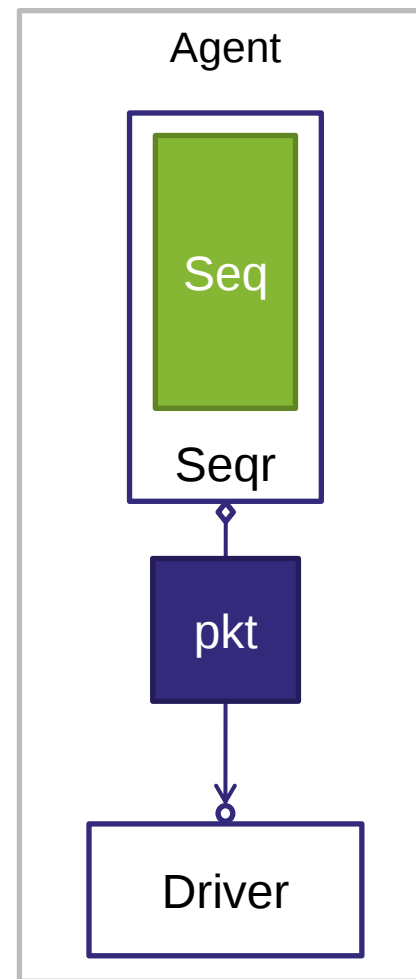
Performance Guidance

- Instantiate dynamic rand once
- Push default constraint
- Interchange once

```
pkt.next()  
pkt.clear_dynamic();
```



Randomized Value



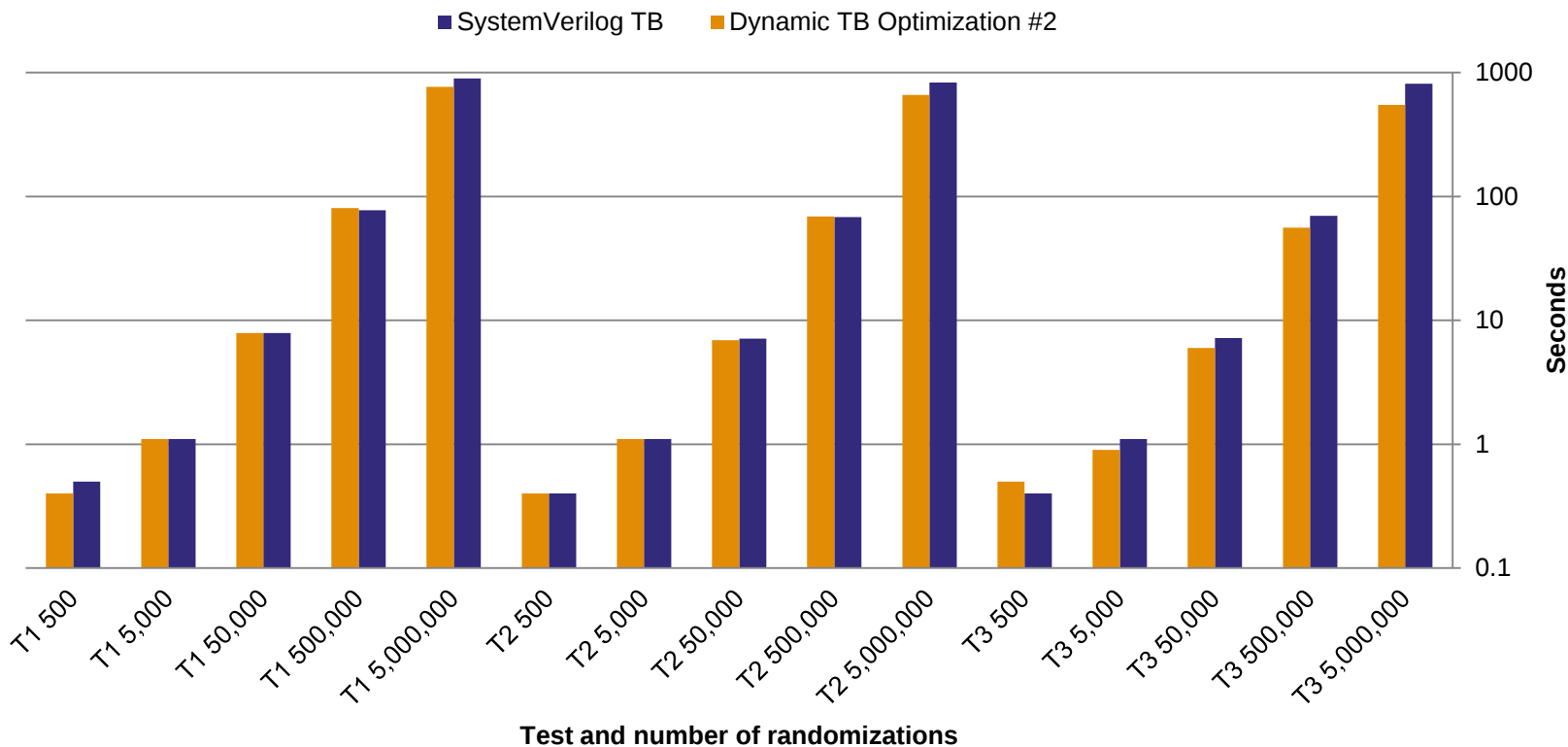
Compile once, simulate many times

Performance Guidance

Compile once, simulate many times

- Allow 1 interchange = same as SystemVerilog constraints

Cumulative Simulation Time



Summary

Interchangeable Constraints

- SystemVerilog constraints
 - Set in code with some flexibility
- Dynamic constraints
 - Set in a string providing lots of flexibility
 - Pass via global resource database or command line
- Implementation
 - Template library of “standard” constraints
 - Template random value class
 - Front-end to instantiate constraint on-the-fly
- Performance
 - Set a default, override on command-line
 - Compile once, run many times

Agenda

SystemVerilog Constraints

Dynamic Constraints

Overview

Implementation

Performance Review



References

- [1] Lorenzo Perrone, *Constraint*, Abecedarian Gallery, online:
<http://www.flickr.com/photos/abecedariangallery/5449083311/in/photostream/>
- [2] Mark Strickland, et.al. *Soft constraints in SystemVerilog, semantics and challenges*. Design and Verification Conference, 2012.
- [3] Kristian Bjornard, *Constraints / Sacrifices*, online:
<http://www.flickr.com/photos/bjornmeansbear/4727475559/>
- [4] Woolworth Bldg. (LOC), Bain News Services publisher, Library of Congress, online:
http://www.flickr.com/photos/library_of_congress/2163937408/
- [5] Nicola, *Woolworth Building*, online:
<http://www.flickr.com/photos/15216811@N06/4631107091/in/photostream/>
- [6] Lorenzo Perrone, *The Big Wave*, Abecedarian Gallery, online:
<http://www.flickr.com/photos/abecedariangallery/5449082679/in/photostream/>



Thank You