2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

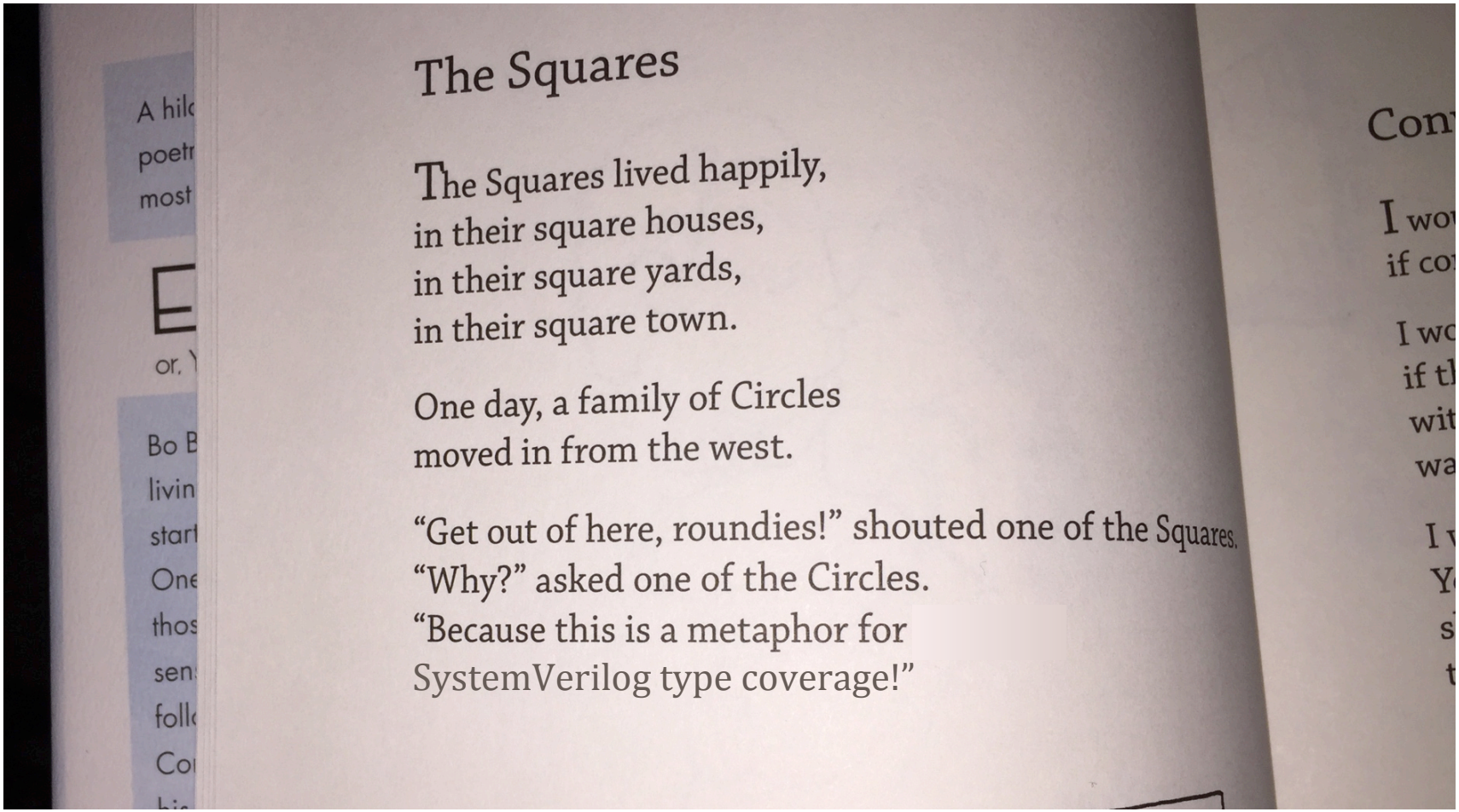
Method for Generating Unique Coverage Classes to Enable Meaningful Covergroup Merges Across Testbenches

Eldon Nelson, M.S., P.E.

Verification Engineer, Micron Technology, Inc.

eldon_nelson@ieee.org

A Poem by Bo Burnham



The Squares

The Squares lived happily,
in their square houses,
in their square yards,
in their square town.

One day, a family of Circles
moved in from the west.

"Get out of here, roundies!" shouted one of the Squares.
"Why?" asked one of the Circles.
"Because this is a metaphor for
SystemVerilog type coverage!"

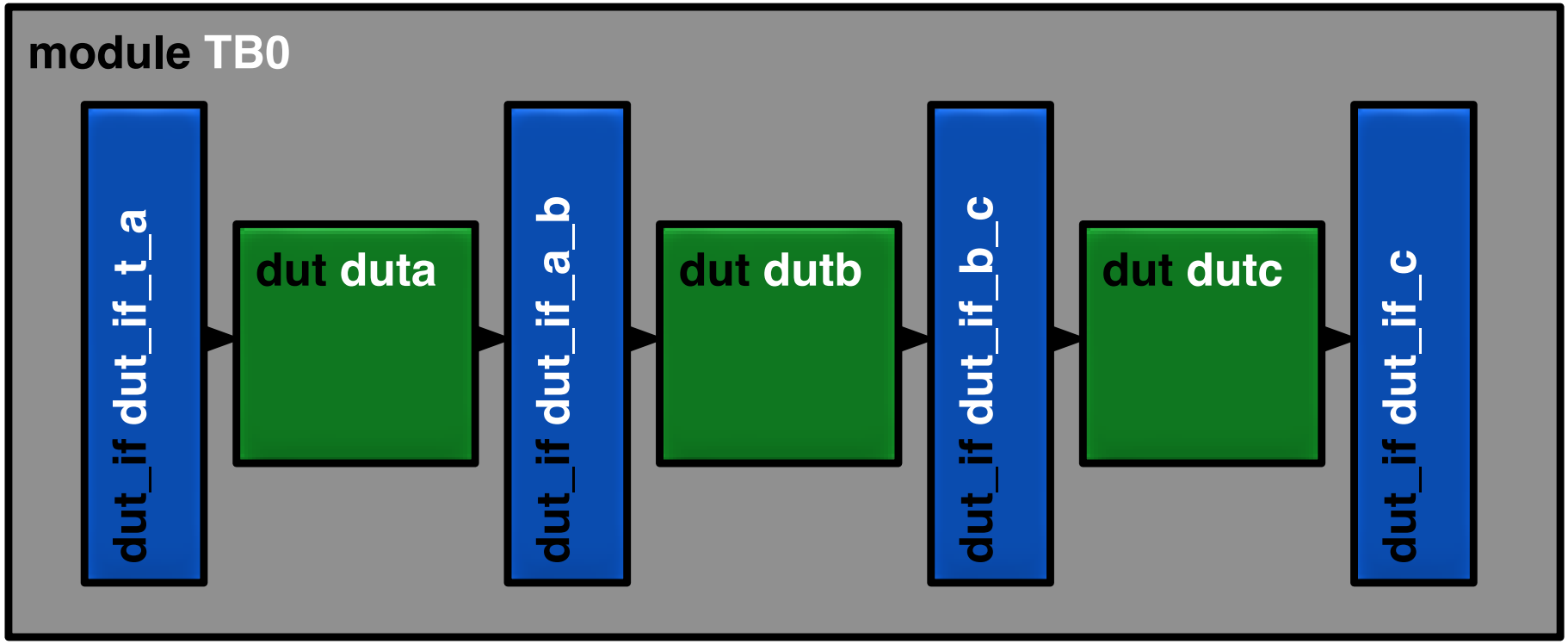
Examples Available under the
General Public License Version 2

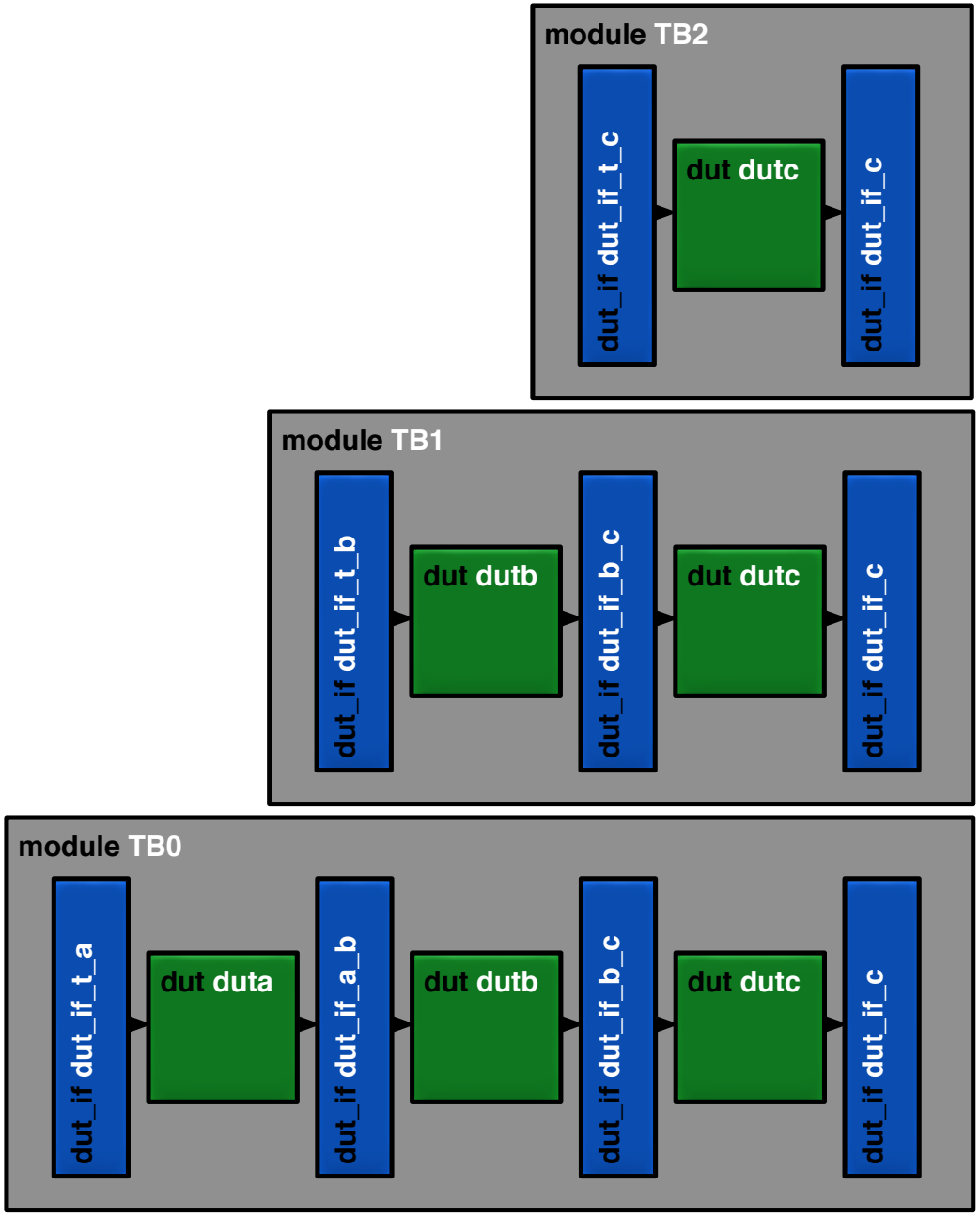
https://github.com/tenthousandfailures/uniquecoverage

Overview

- The Simple and Embedded Covergroup
- What Coverage-Driven Testplans Crave
- A Better Embedded Covergroup
- SystemVerilog and UVM Implementation

Testbench


module TB0
dut_if dut_if_t_a
dut duta
dut_if dut_if_a_b
dut dutb
dut_if dut_if_b_c
dut dutc
dut_if dut_if_c


module TB2
dut_if dut_if_t_c
dut dutc
dut_if dut_if_c
module TB1
dut_if dut_if_t_b
dut dutb
dut_if dut_if_b_c
dut dutc
dut_if dut_if_c
module TB0
dut_if dut_if_t_a
dut duta
dut_if dut_if_a_b
dut dutb
dut_if dut_if_b_c
dut dutc
dut_if dut_if_c

Covergroup Definition

dut_if_cg.svh

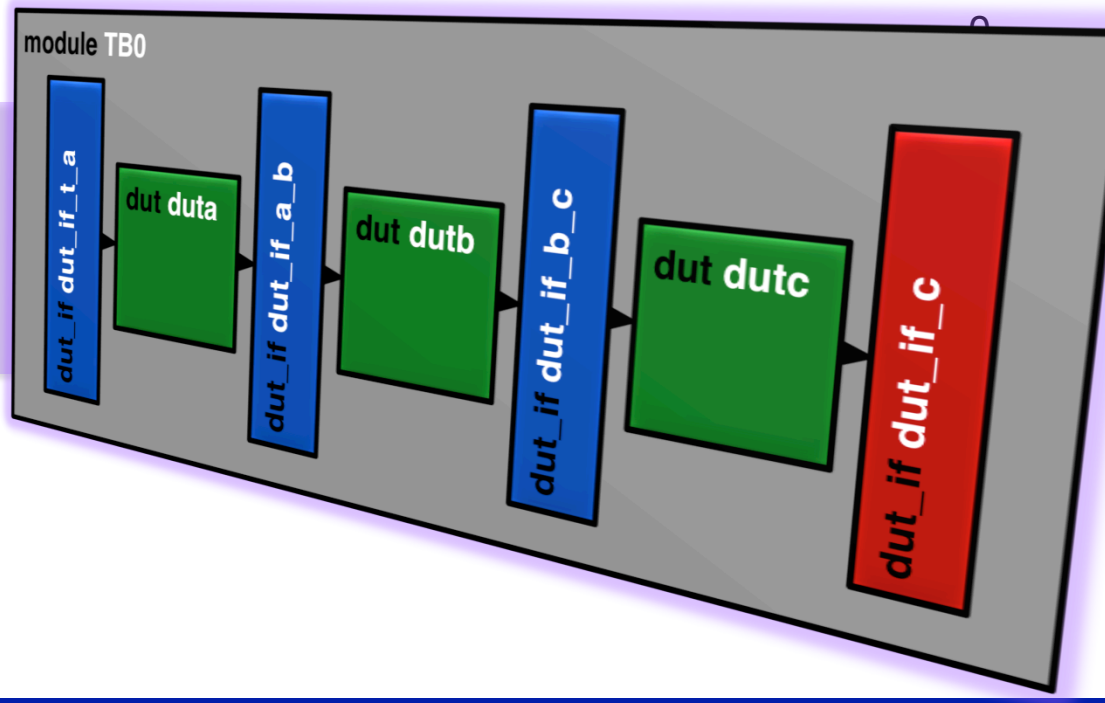
```
covergroup dut_if_cg (ref logic [3:0] cmd,
                      input string    inst_name,
                      input string    comment);

    option.name = {name, ".", inst_name};

    coverpoint cmd {
        bins _null  = {'h0};
        bins _read  = {'h1};
        bins _write = {'h2};
        bins _cfg   = {'h3};
    }

endgroup
```

Simple Covergroup

```
interface dut_if (input logic clk);
    logic [3:0] adr, cmd, data;

    `include "dut_if_cg.svh" // included covergroup definition
    dut_if_cg simple_inst;   // simple covergroup

    always @(posedge clk) begin
        simple_inst.sample();
    end

    initial begin
        simple_inst = new(.cmd(cmd),
                          .inst_name(inst_name),
                          .comment(simple_comment)
                          );
    end

endinterface
```

Simple Covergroup

Name	Class Type	Coverage
/TB0/dut_if_c		
TYPE dut_if_cg		50.0%
CVP dut_if_cg::cmd		50.0%
bin _null		1
bin _read		0
bin _write		
bin _cfg		
INST simple.TB0_dut_if_c		
bin _null		
bin _read		
bin _write		
bin _cfg		

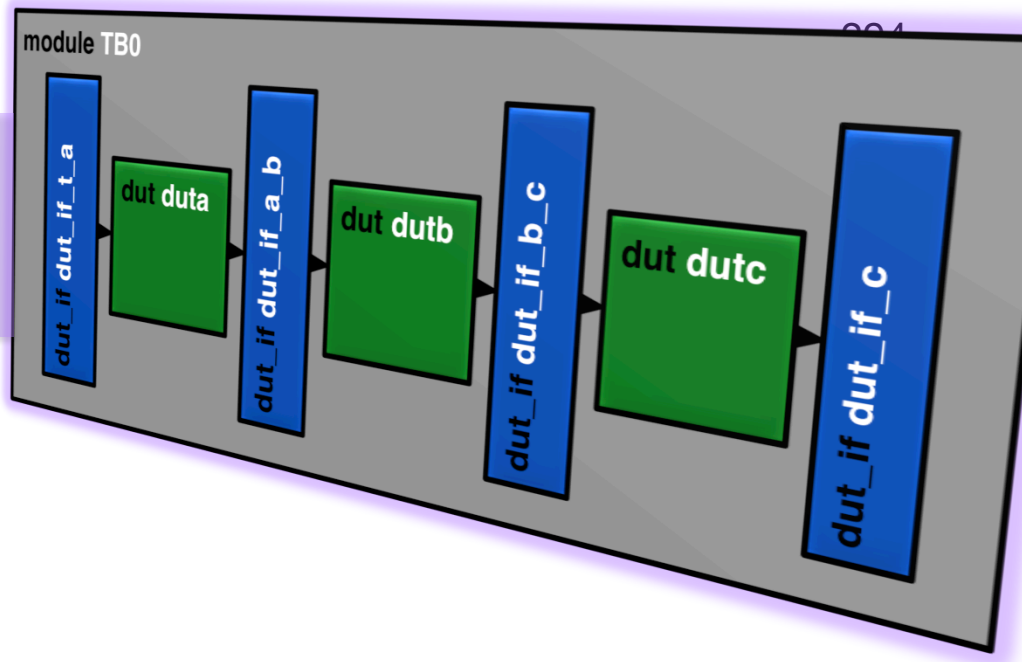


Embedded Covergroup

```
interface dut_if (input logic clk);
    logic [3:0] cmd, adr, data;

    emb_pkg::cov emb_inst; // embedded covergroup

    always @(posedge clk) begin
        emb_inst.sample();
    end

    initial begin
        emb_inst = new(.cmd(cmd),
                       .inst_name(inst_name)
                       );
    end

endinterface
```

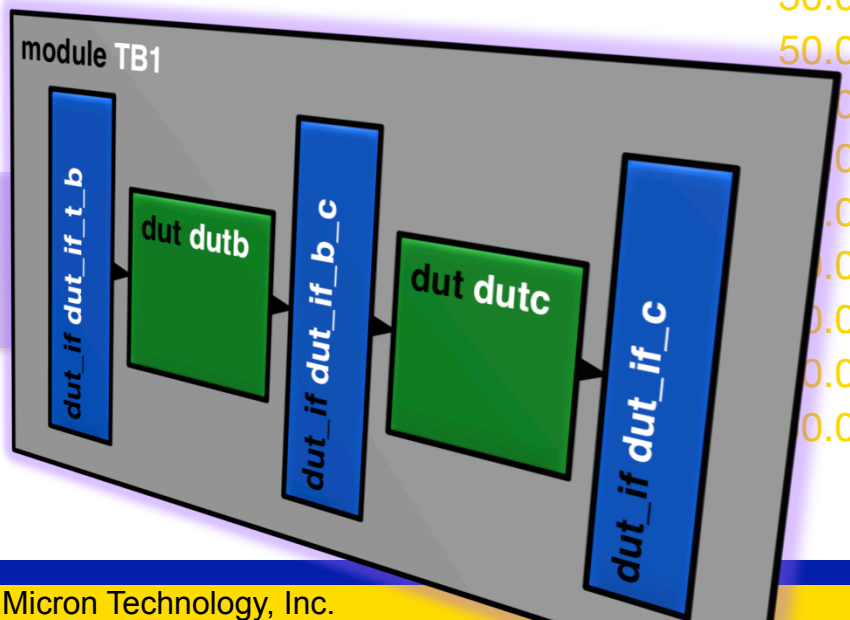
Embedded Covergroup

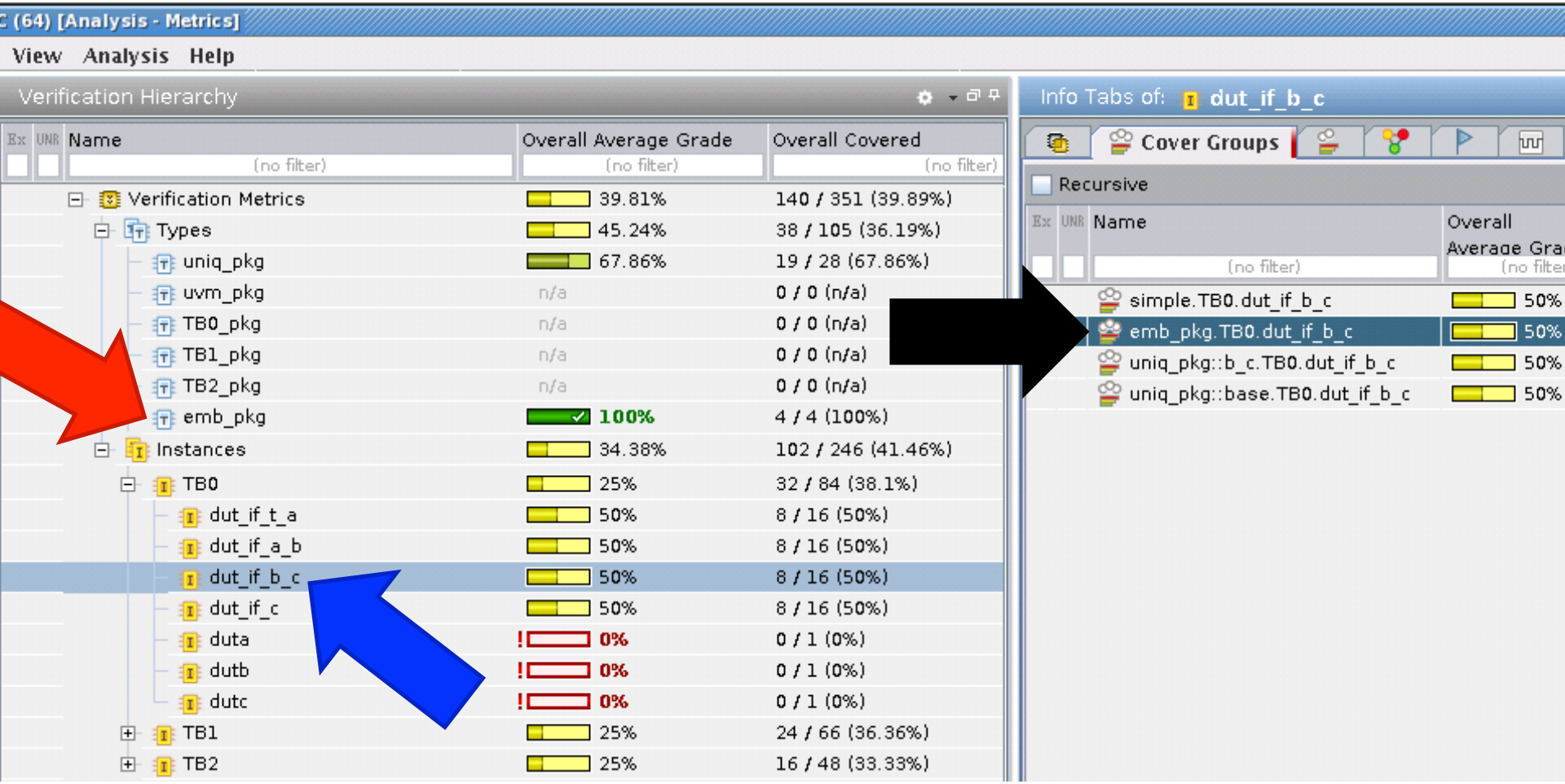
Name	Class Type	Coverage
/emb_pkg/cov		
TYPE dut_if_cg	cov	100.0%
CVP dut_if_cg::cmd		100.0%
bin _null		9
bin _read		390
bin _write		
bin _cfg		
INST emb_pkg.TB0.dut_if_t_a		
INST emb_pkg.TB0.dut_if_a_b		
INST emb_pkg.TB0.dut_if_b_c		
INST emb_pkg.TB0.dut_if_c		
INST emb_pkg.TB1.dut_if_t_b		
INST emb_pkg.TB1.dut_if_b_c		
INST emb_pkg.TB1.dut_if_c		
INST emb_pkg.TB2.dut_if_t_c		
INST emb_pkg.TB2.dut_if_c		



Embedded Covergroup

Name	Class Type	Coverage
/emb_pkg/cov		
TYPE dut_if_cg	cov	100.0%
CVP dut_if_cg::cmd		100.0%
bin _null		9
bin _read		390
bin _write		294
bin _cfg		197
INST emb_pkg.TB0.dut_if_t_a		50.0%
INST emb_pkg.TB0.dut_if_a_b		50.0%
INST emb_pkg.TB0.dut_if_b_c		0%
INST emb_pkg.TB0.dut_if_c		0%
INST emb_pkg.TB1.dut_if_t_b		.0%
INST emb_pkg.TB1.dut_if_b_c		.0%
INST emb_pkg.TB1.dut_if_c		.0%
INST emb_pkg.TB2.dut_if_t_c		0.0%
INST emb_pkg.TB2.dut_if_c		0.0%




(64) [Analysis - Metrics]
View Analysis Help
Verification Hierarchy
Ex UNR Name Overall Average Grade Overall Covered
(no filter) (no filter) (no filter)
Verification Metrics 39.81% 140 / 351 (39.89%)
Types 45.24% 38 / 105 (36.19%)
uniq_pkg 67.86% 19 / 28 (67.86%)
uvm_pkg n/a 0 / 0 (n/a)
TB0_pkg n/a 0 / 0 (n/a)
TB1_pkg n/a 0 / 0 (n/a)
TB2_pkg n/a 0 / 0 (n/a)
emb_pkg 100% 4 / 4 (100%)
Instances 34.38% 102 / 246 (41.46%)
TB0 25% 32 / 84 (38.1%)
dut_if_t_a 50% 8 / 16 (50%)
dut_if_a_b 50% 8 / 16 (50%)
dut_if_b_c 50% 8 / 16 (50%)
dut_if_c 50% 8 / 16 (50%)
duta 0% 0 / 1 (0%)
dutb 0% 0 / 1 (0%)
dutc 0% 0 / 1 (0%)
TB1 25% 24 / 66 (36.36%)
TB2 25% 16 / 48 (33.33%)
Info Tabs of: dut_if_b_c
Cover Groups
Recursive
Ex UNR Name Overall Average Gra
(no filter) (no filter)
simple.TB0.dut_if_b_c 50%
emb_pkg.TB0.dut_if_b_c 50%
uniq_pkg::b_c.TB0.dut_if_b_c 50%
uniq_pkg::base.TB0.dut_if_b_c 50%

Covergroup Type

- Simple covergroups cannot be merged
- Embedded covergroups can lose meaning when there are many contributors

Overview

- The Simple and Embedded Covergroup ✔
- What Coverage-Driven Testplans Crave
- A Better Embedded Covergroup
- SystemVerilog and UVM Implementation

Testplan Instance Coverage

Section	Title	Link	Type
1	Simple Covergroup		
1.1	cmd coverpoint for interface _c on TB0	/TB0/dut_if_c/dut_if_cg,simple.TB0.dut_if_c:cmd	CoverPoint
1.2	cmd coverpoint for interface _c on TB1	/TB1/dut_if_c/dut_if_cg,simple.TB1.dut_if_c:cmd	CoverPoint
1.3	cmd coverpoint for all interfaces of _c with wildcard	/TB*/dut_if_c/dut_if_cg,simple.TB*.dut_if_c:cmd	CoverPoint

What Testplans Crave

- Testplans Crave **Type** Coverage

Instance Problems 1 of 3

- **Instance** names change
- **Instance** names don't make sense across testbenches

Instance Problems 2 of 3

- Repeating **Instances** names is not DRY

Instance Problems 3 of 3

- Glob patterns in **Instance** names is a dirty fragile hack

Instance Vs. Embedded in a Testplan

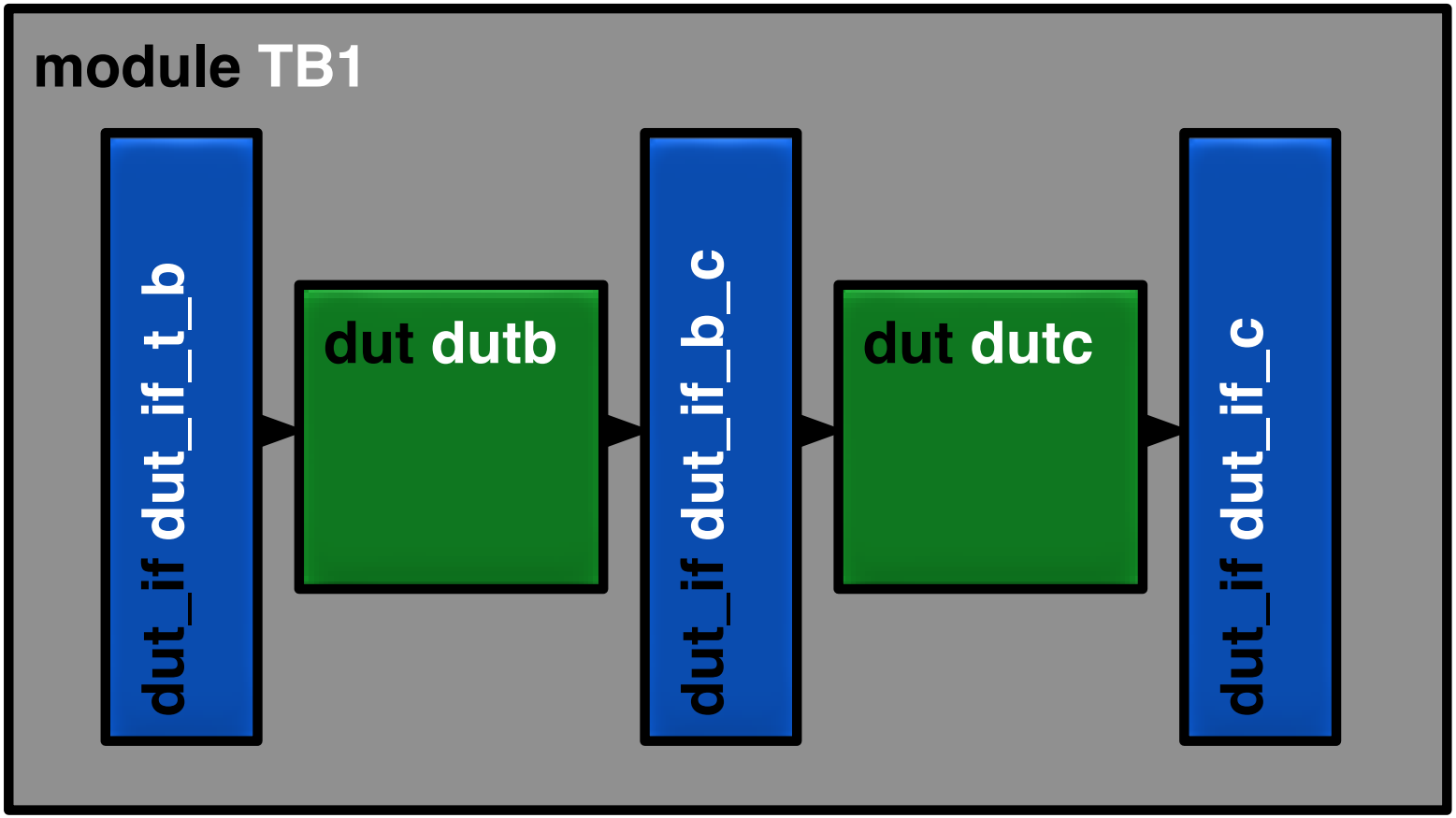
Section	Title	Link	Type
1	Simple Covergroup		
1.1	cmd coverpoint for interface _c on TB0	/TB0/dut_if_c/dut_if_cg,simple.TB0.dut_if_c:cmd	CoverPoint
1.2	cmd coverpoint for interface _c on TB1	/TB1/dut_if_c/dut_if_cg,simple.TB1.dut_if_c:cmd	CoverPoint
1.3	cmd coverpoint for all interfaces of _c with wildcard	/TB*/dut_if_c/dut_if_cg,simple.TB*.dut_if_c:cmd	CoverPoint

Section	Title	Link	Type
2	Embedded Covergroup Method		
2.1	cmd coverpoint for covergroup type /emb_pkg/cov	/emb_pkg/cov/dut_if_cg:cmd	CoverPoint

A Better Embedded Covergroup

- Instance coverage done with Simple Covergroups in a coverage driven testplan is non-optimal
- Type coverage done with Embedded Covergroups is not specific enough

Testbench


module TB1
dut_if dut_if_t_b
dut dutb
dut_if dut_if_b_c
dut dutc
dut_if dut_if_c

The Solution

Name	Class Type	Coverage
/uniq_pkg/b_c		
TYPE dut_if_cg	b_c	75.0%
CVP dut_if_cg::cmd		75.0%
bin _null		2
bin _read		97
bin _write		98
bin _cfg		0
INST uniq_pkg::b_c.TB0.dut_if_b_c		50.0%
INST uniq_pkg::b_c.TB1.dut_if_b_c		50.0%

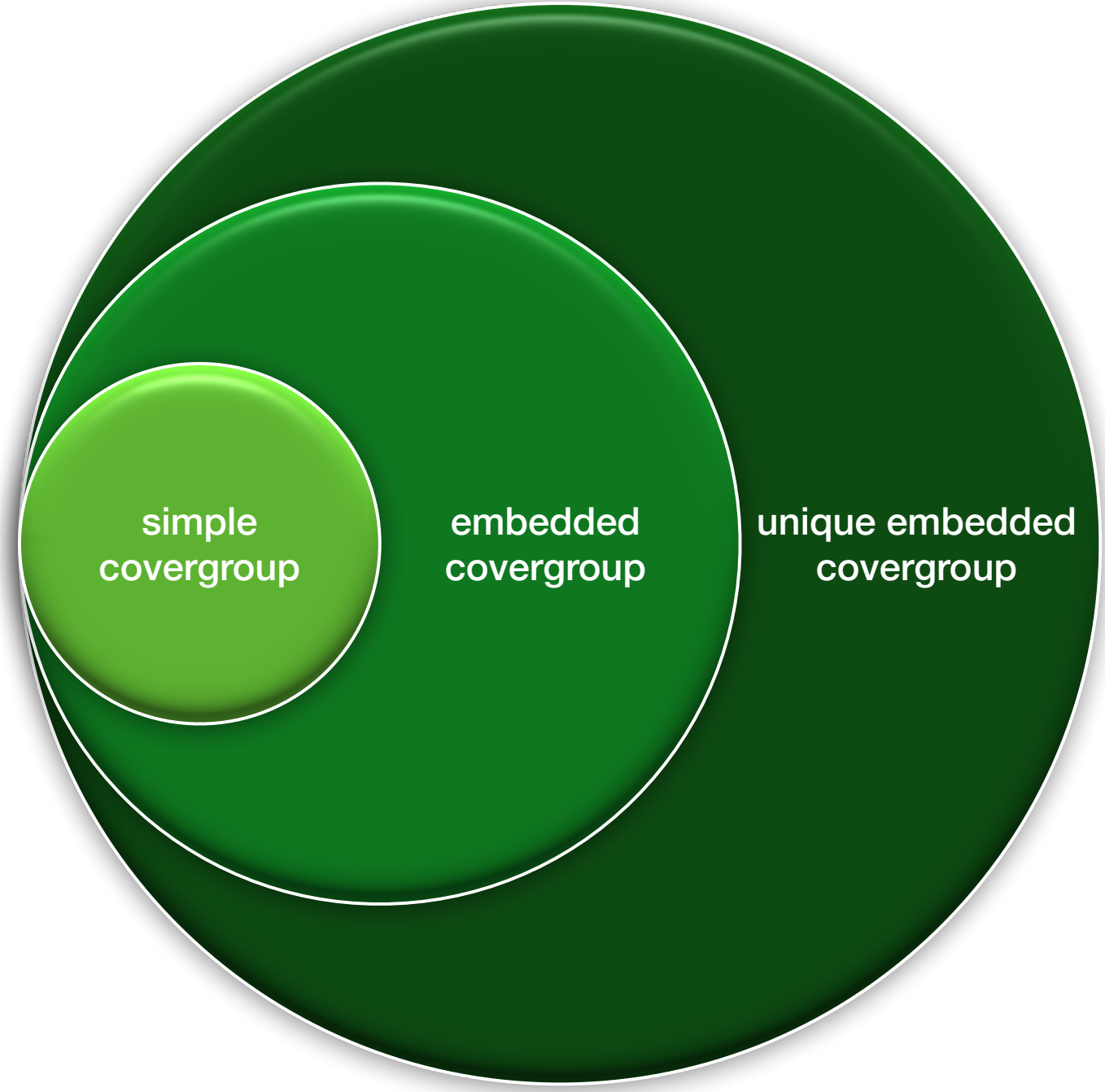
Embedded Vs. Unique Embedded

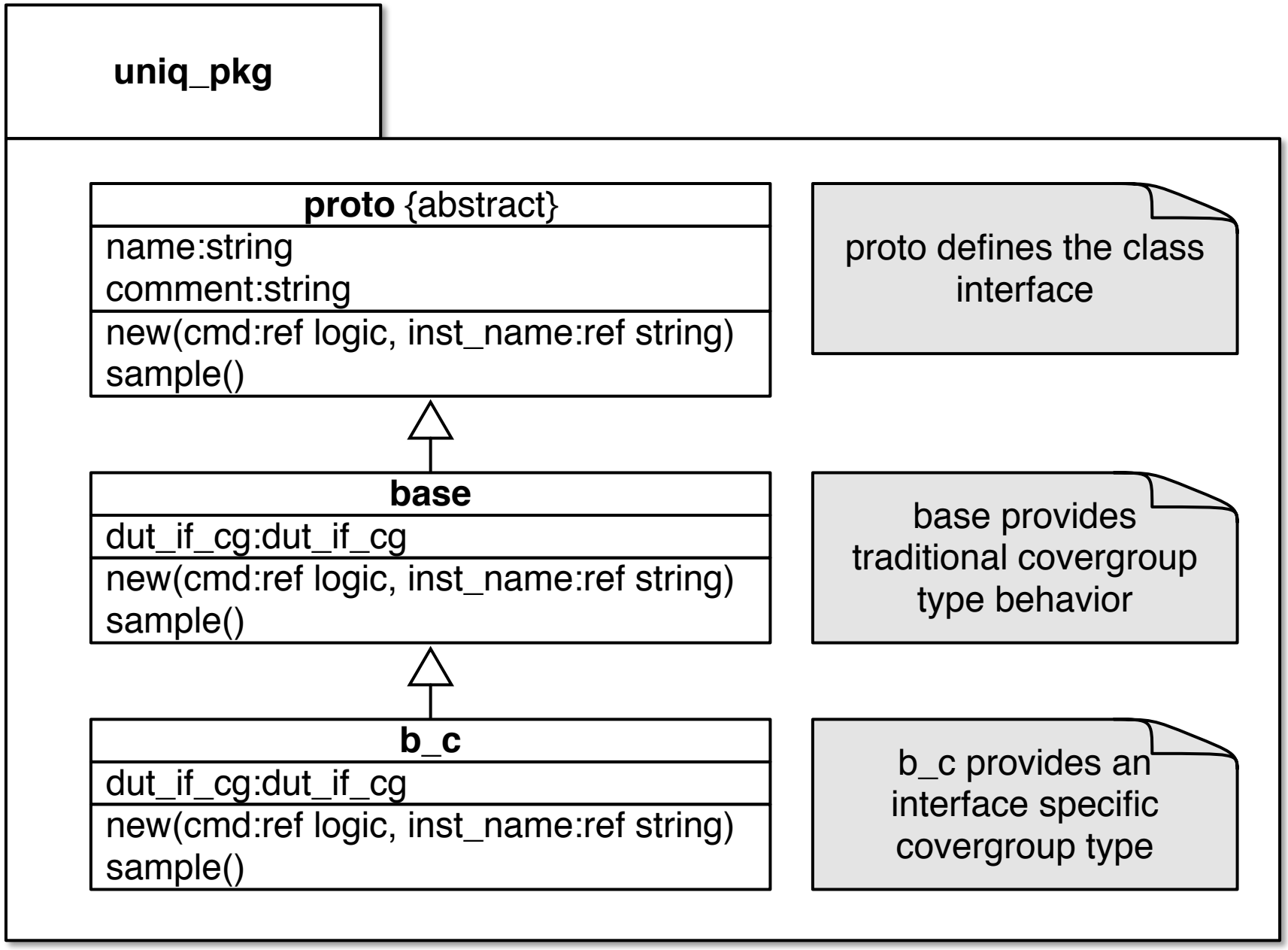
Section	Title	Link	Type
2	Embedded Covergroup Method		
2.1	cmd coverpoint for covergroup type /emb_pkg/cov	/emb_pkg/cov/dut_if_cg:cmd	CoverPoint

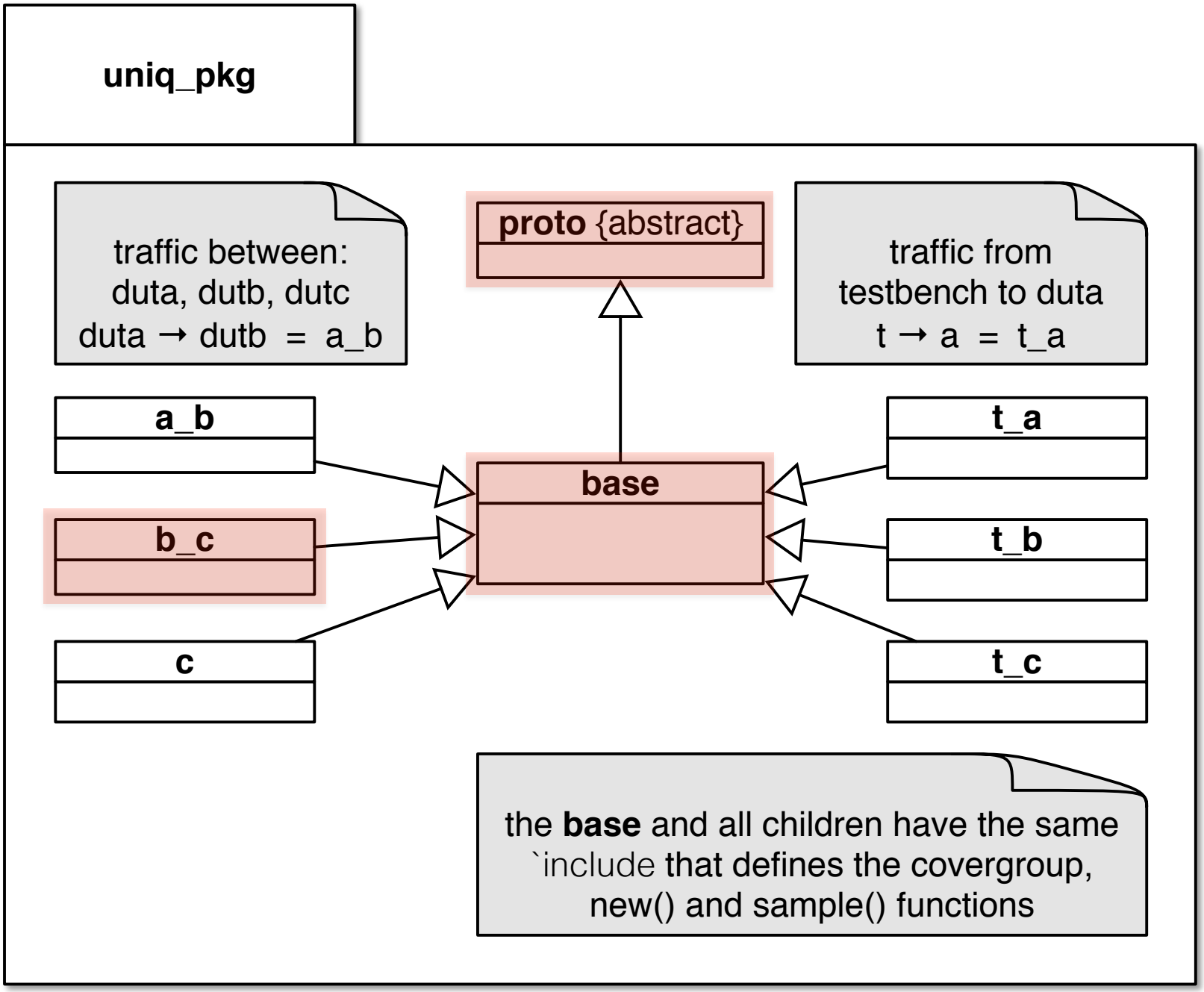
Section	Title	Link	Type
3	Proposed Unique Embedded Covergroup		
3.1	Coverage between `dutb` and `dutc`	/uniq_pkg/b_c/dut_if_cg:cmd	CoverPoint

Agenda

- The Simple and Embedded Covergroup ✔
- What Coverage-Driven Testplans Crave ✔
- A Better Embedded Covergroup
- SystemVerilog and UVM Implementation


simple
covergroup
embedded
covergroup
unique embedded
covergroup


uniq_pkg
proto {abstract}
name:string
comment:string
new(cmd:ref logic, inst_name:ref string)
sample()
proto defines the class interface
base
dut_if_cg:dut_if_cg
new(cmd:ref logic, inst_name:ref string)
sample()
base provides traditional covergroup type behavior
b_c
dut_if_cg:dut_if_cg
new(cmd:ref logic, inst_name:ref string)
sample()
b_c provides an interface specific covergroup type


uniq_pkg
traffic between:
duta, dutb, dutc
duta → dutb = a_b
proto {abstract}
traffic from
testbench to duta
t → a = t_a
a_b
b_c
c
base
t_a
t_b
t_c
the base and all children have the same
`include that defines the covergroup,
new() and sample() functions

Benefits 1 of 3

- Merging across testbenches **with** specificity
  - Reused covergroup definition
  - Testplan line identical for any testbench

Benefits 2 of 3

- Hierarchical structure
  - Multiple covergroup samples
  - Groupings of covergroups that make sense

Benefits 3 of 3

- Consistent type coverage that makes coverage-driven testplans happy
  - Names available early before the code
  - Write your covergroup include file before your verif environment and link up your coverage!

Agenda

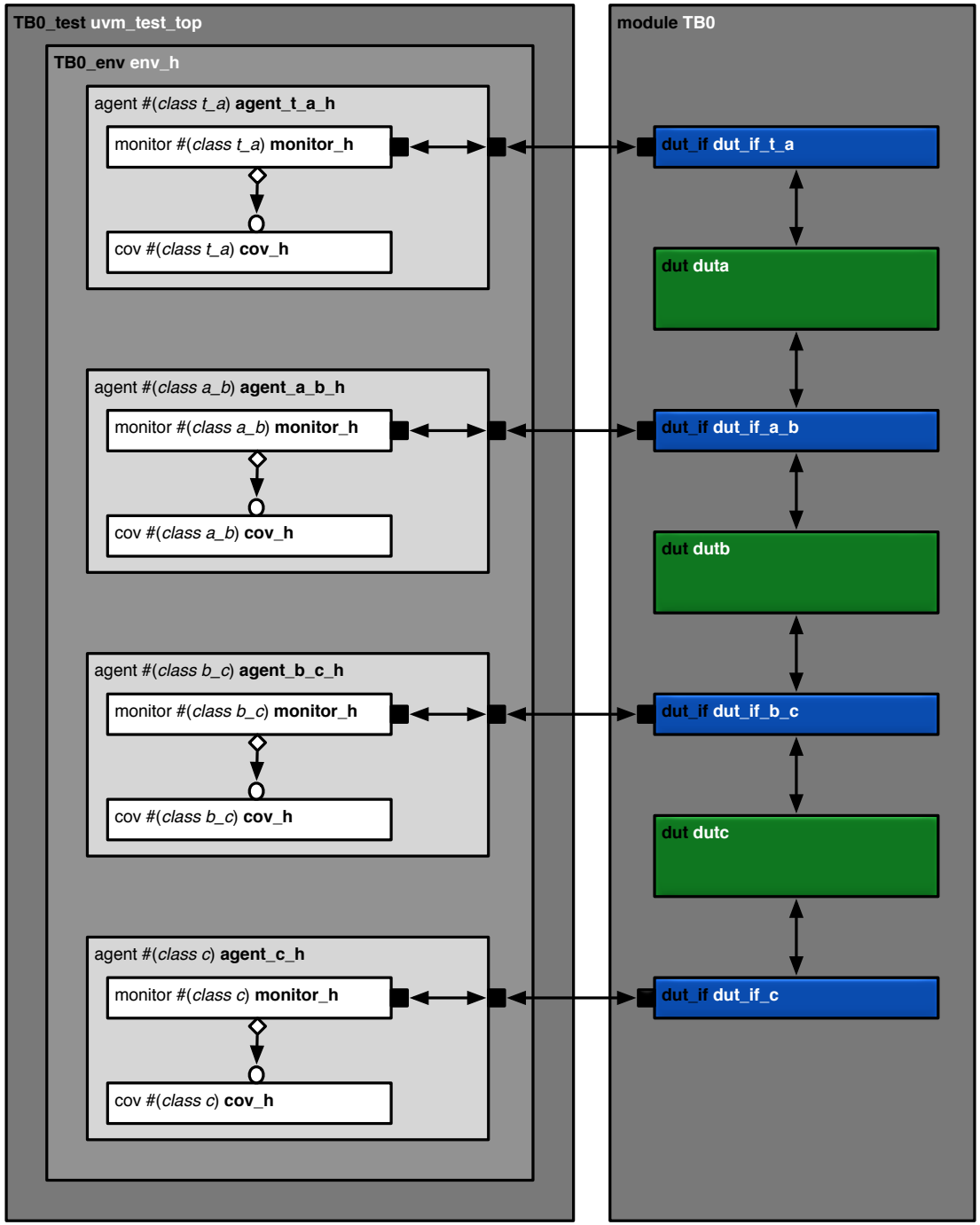
- The Simple and Embedded Covergroup ✔
- What Coverage-Driven Testplans Crave ✔
- A Better Embedded Covergroup ✔
- SystemVerilog and UVM Implementation

```
package uniq_pkg;

  virtual class proto;
    ...
  endclass

  class base extends proto;
    ...
  endclass

  class b_c extends base;

    string name = {pkg_prefix, "b_c"};
    string comment = "b_c comment";

    `include "dut_if_cg.svh"
    `include "uniq_pkg_fcn.svh"

  endclass

endpackage
```

```
interface dut_if #(type T = uniq_pkg::base) (input logic clk);

    string inst_name = "";
    logic [3:0] cmd, adr, data;

    T cov_inst; // parameterized class containing covergroup

    always @(posedge clk) begin
        cov_inst.sample();
    end

    initial begin
        $sformat(inst_name, "%m");
        cov_inst = new(.cmd(cmd),
                       .inst_name(inst_name)
                       );
    end

endinterface
```

```
module TB0 ();

    dut_if #(uniq_pkg::t_a) dut_if_t_a(clk);
    dut_if #(uniq_pkg::a_b) dut_if_a_b(clk);
    dut_if #(uniq_pkg::b_c) dut_if_b_c(clk);
    dut_if #(uniq_pkg::c)   dut_if_c(clk);

    dut duta(
             .slave(dut_if_t_a), // input
             .master(dut_if_a_b) // output
             );
    dut dutb(
             .slave(dut_if_a_b),
             .master(dut_if_b_c)
             );
    dut dutc(
             .slave(dut_if_b_c),
             .master(dut_if_c)
             );

endmodule
```

UVM Solution

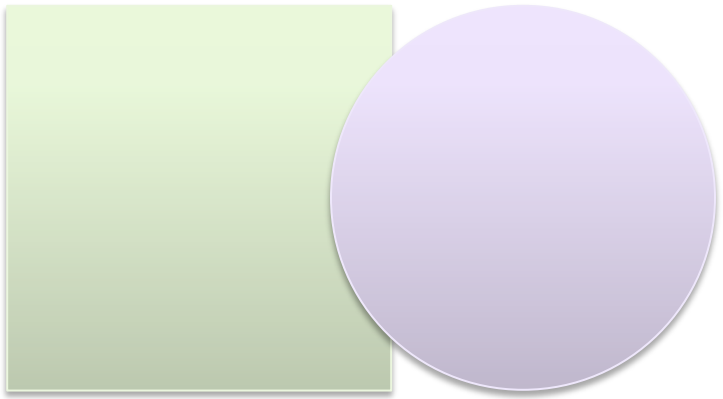

TB0_test uvm_test_top
TB0_env env_h
agent #(class t_a) agent_t_a_h
monitor #(class t_a) monitor_h
cov #(class t_a) cov_h
agent #(class a_b) agent_a_b_h
monitor #(class a_b) monitor_h
cov #(class a_b) cov_h
agent #(class b_c) agent_b_c_h
monitor #(class b_c) monitor_h
cov #(class b_c) cov_h
agent #(class c) agent_c_h
monitor #(class c) monitor_h
cov #(class c) cov_h
module TB0
dut_if dut_if_t_a
dut duta
dut_if dut_if_a_b
dut dutb
dut_if dut_if_b_c
dut dutc
dut_if dut_if_c

Summary

- The Simple and Embedded Covergroup ✔
- What Coverage-Driven Testplans Crave ✔
- A Better Embedded Covergroup ✔
- SystemVerilog and UVM Implementation ✔

Conclusion

- Testplans Crave Type Coverage
- Unique Embedded Coverage Superset of Embedded Coverage
  - Third Heat!
- Unique Embedded Coverage Added Features



Backup Slides

Parameterized Embedded

```
typedef virtual dut_if #(uniq_pkg::base#("t_a")) dut_if_t_a_t;


...
# ** while parsing file included at TB_common.svh(9)
# ** at agent.svh(21): (vlog-2181) Use of a parameterized class base
as a type creates a default specialization.
```

Parameterized Covergroup

Covergroups

Name	Class Type	Coverage
/emb_pkg/cov		
/TB0/dut_if_a_b		
/TB0/dut_if_b_c		
/TB0/dut_if_c		
/TB0/dut_if_t_a		
/uniq_pkg/base/base__2		
TYPE dut_if_cg	base #(<non-int val>)	50.0%
/uniq_pkg/base/base__3		
TYPE dut_if_cg	base #(<non-int val>)	50.0%
/uniq_pkg/base/base__4		
TYPE dut_if_cg	base #(<non-int val>)	50.0%
/uniq_pkg/base/base__5		
TYPE dut_if_cg	base #(<non-int val>)	50.0%