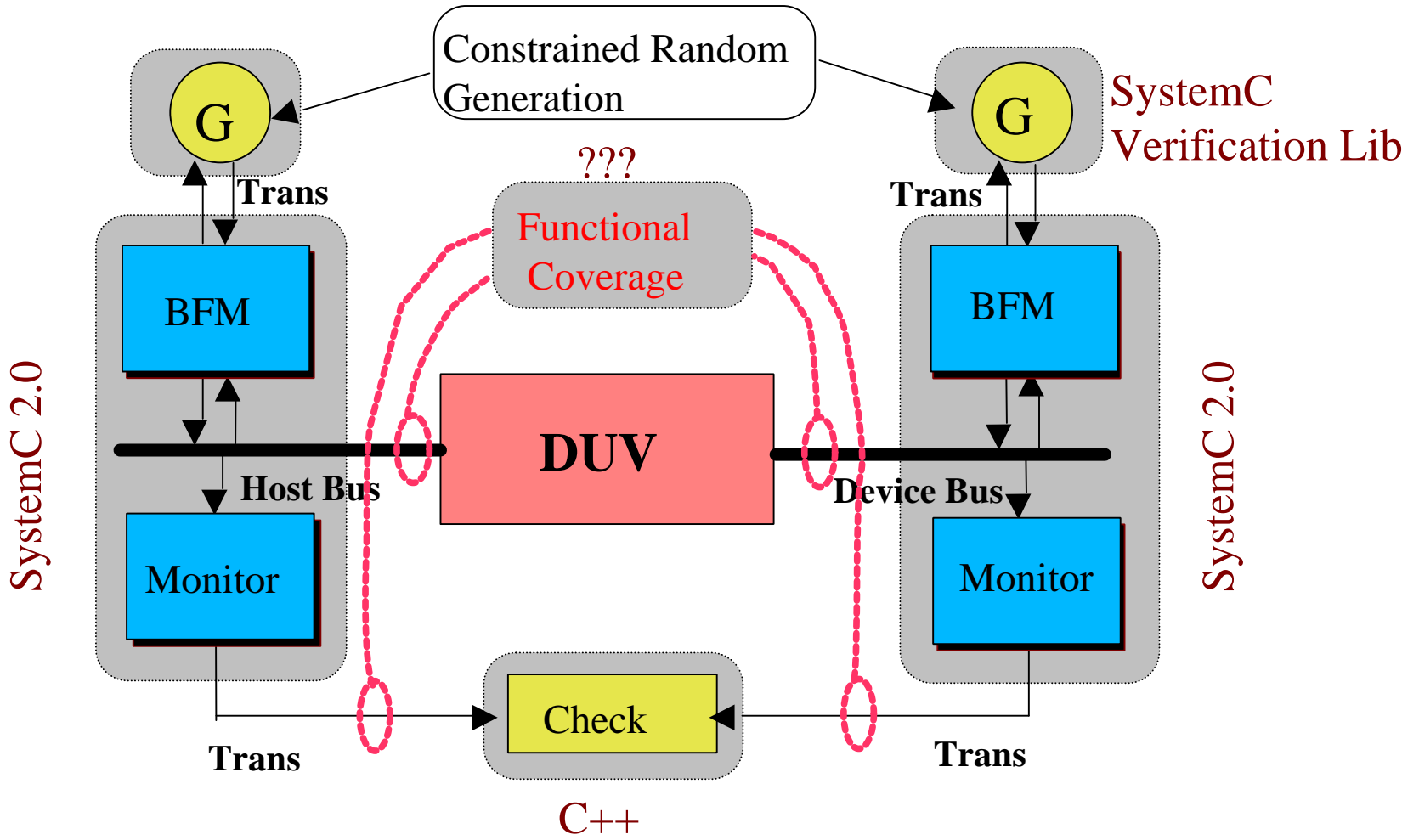




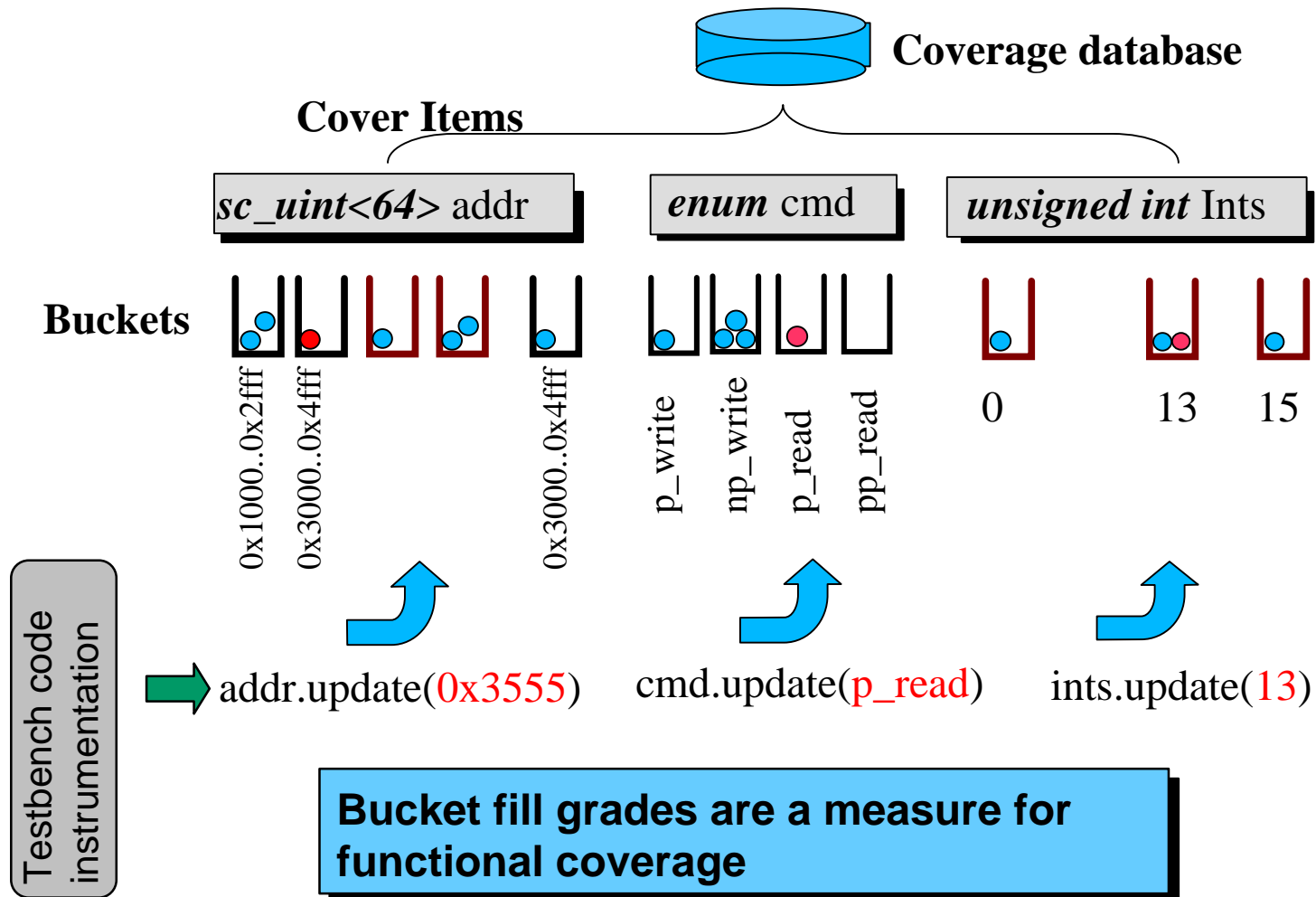
A Functional Coverage Prototype for SystemC-based Verification of Chipset Designs

Robert Siegmund, Ulrich Hensel,
Andreas Herrholz, Isa Volt
AMD Dresden Design Center

- Chipset Verification using SystemC
- SystemC Coverage API Prototype
 - Cover items and buckets
 - Toggle coverage
 - Cross coverage
 - Coverage reports
- Coverage Hole Analysis
- Conclusions and Summary



Concept of Functional Coverage



Definition of coverage items and buckets

```
// create/switch to active coverage database
sr_cover::set_database("bus_trans_coverage");

// define coverage items
sr_cover::create_item(trans.get_addr(), "addr");

// specify buckets for each coverage item
// create bucket [0x1000,0x2fff] with hit goal 5
sr_cover::get_item("addr").bucketize(sc_uint<64>(0x1000),
                                     sc_uint<64>(0x2fff),
                                     "ac97_space", 5);

// OPTIONAL:special OTHERS bucket
sr_cover::get_item("addr").bucketize(SR_OTHERS);
```

Item type deduced here

Goal: 5 hits required

Coverage API supports definition of 'illegal' and 'ignore' ranges

```
// specify ignorable ranges for addr
sr_cover::get_item("addr").ignore(
    sr_cover::get_item("addr")() > sc_uint<64>(0x20) &&
    sr_cover::get_item("addr")() < sc_uint<64>(0x3F));

// specify illegal ranges for addr
sr_cover::get_item("addr").illegal(
    sr_cover::get_item("addr")() == sc_uint<64>(0x80));
```

Instrumentation of testbench code for coverage analysis

```
void systemc_process() {  
  
    trans* t; // Bus transaction  
    ...  
  
    if(cover_condition) {  
  
        // cover value of fields in transaction  
        sr_cover::get_item("addr").update(t->get_addr());  
        sr_cover::get_item("cmd").update(t->get_cmd());  
        ...  
    }  
}
```

Goal: Have all pins/selected signals toggled at least once?

```
sc_in<bool> clk;  
sc_out<sc_logic> dout;  
sc_signal<bool> data;
```

Ports and signals of type
bool and sc_logic supported

```
// include pins 'clk','dout' in toggle coverage  
sr_toggle_coverage::add_pin(clk,"clk");  
sr_toggle_coverage::add_pin(dout,"dout");  
  
// specify transitions to 'X' illegal for data  
sr_toggle_coverage::illegal_trans(  
    "data",sc_logic('0'),sc_logic('X'));  
  
// ignore 1->0 transitions on clk  
sr_toggle_coverage::ignore_trans("clk",1,0);
```

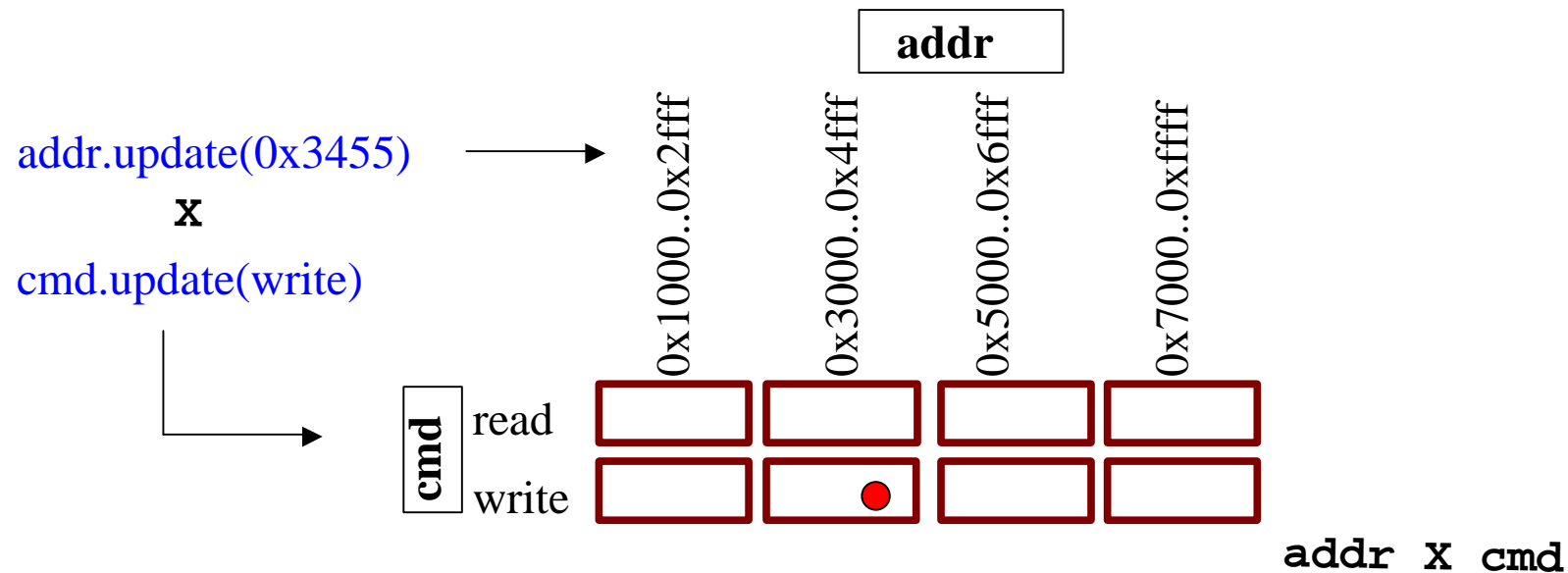
Cross Coverage API



Cross coverage = cross product of cover items at times t

Application: Covering verification corner cases

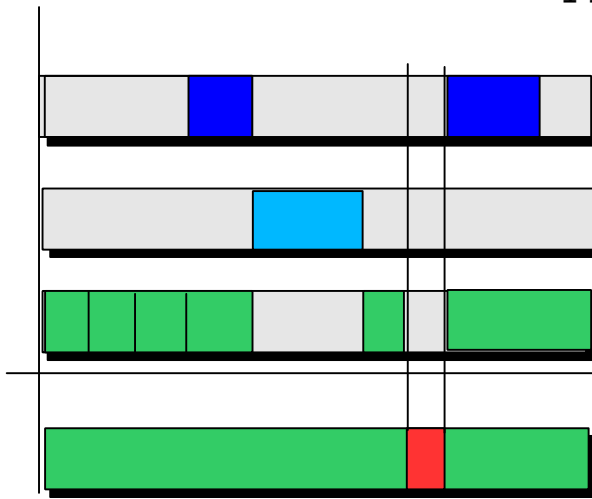
```
// cross items "addr", "cmd"  
// the cross product item is named "addr_X_cmd"  
sr_cover::cross("addr_X_cmd", "addr", "cmd");
```



Post Run Analysis of Coverage Holes



Domain of covered type



User defined item buckets

minus Ignore expression

minus Buckets with hits after simulation

= Coverage hole!

Coverage holes are computed automatically using a BDD-based analysis algorithm

**** rqst_type ****

Samples: 81 ignored: 0 Grade: 1

grade	goal	samples	%total	rqst_type
1	1	6	7.41	rqst_trans::msr_write
1	1	44	54.3	rqst_trans::msr_read
1	1	4	4.94	rqst_trans::io_read
1	1	4	4.94	rqst_trans::io_write
1	1	4	4.94	rqst_trans::non_coh_read

**Coverage holes: rqst_trans::coh_read_be
rqst_trans::coh_write
rqst_trans::non_coh_write**

Result of coverage hole analysis

- We presented a prototype to measure and analyze functional coverage in SystemC-based verification.
- Functional coverage prototype has been used in SystemC-based verification of an AMD chipset design
- Using the proposed coverage API, coverage analysis was easy to integrate in in-house SystemC verification environment
- Coverage API is fully SystemC 2.0 compliant (implemented as add-on library to SystemC)
- Automatic coverage hole analysis provided us a tool to
 - measure the quality of stimulus
 - check if corner cases have been thoroughly exercised

Trademark Attribution



AMD, the AMD Arrow Logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this presentation are for identification purposes only and may be trademarks of their respective companies.