



SyoSCB VIP Introduction

kasper@syosil.com, jacob@syosil.com

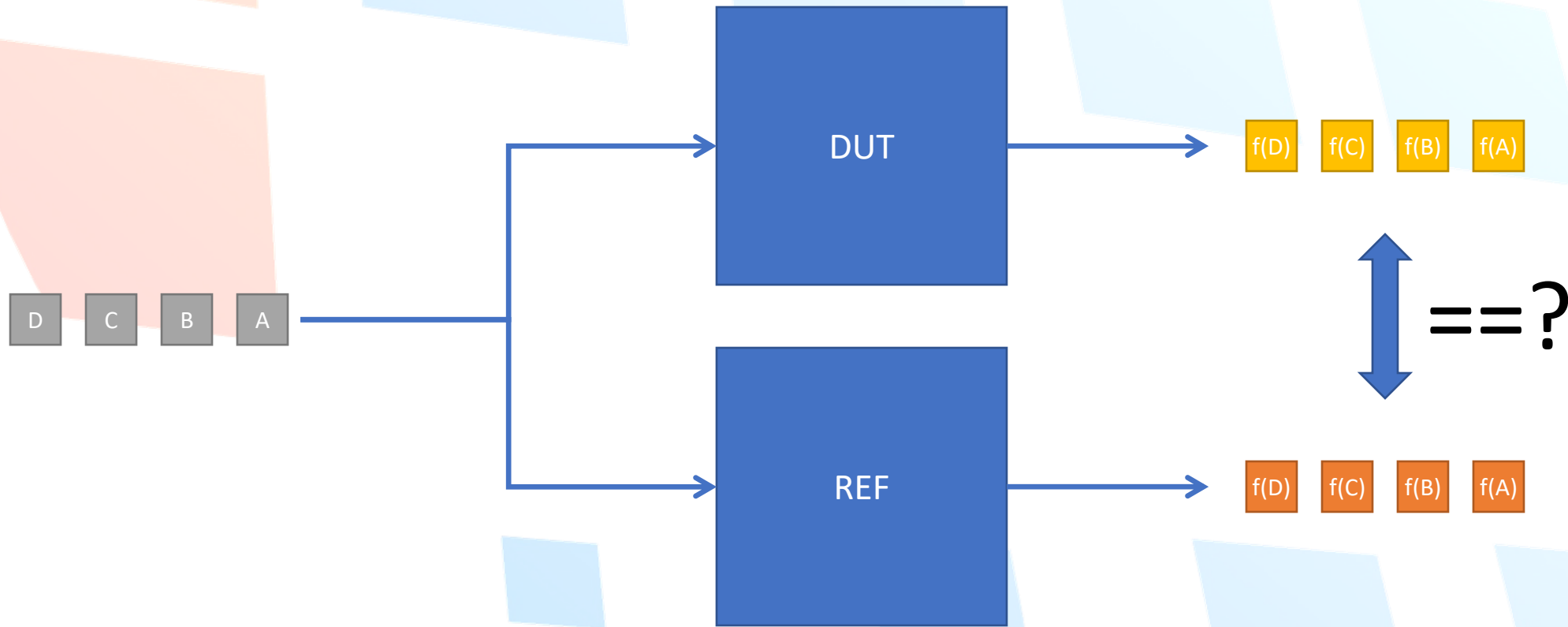
Methodologies, design & verification
www.syosil.com

Agenda

- Brief intro to score boarding in a UVM TB in general
- Overview of the SyoSCB VIP
- In depth details of selected features

Scoreboarding Introduction

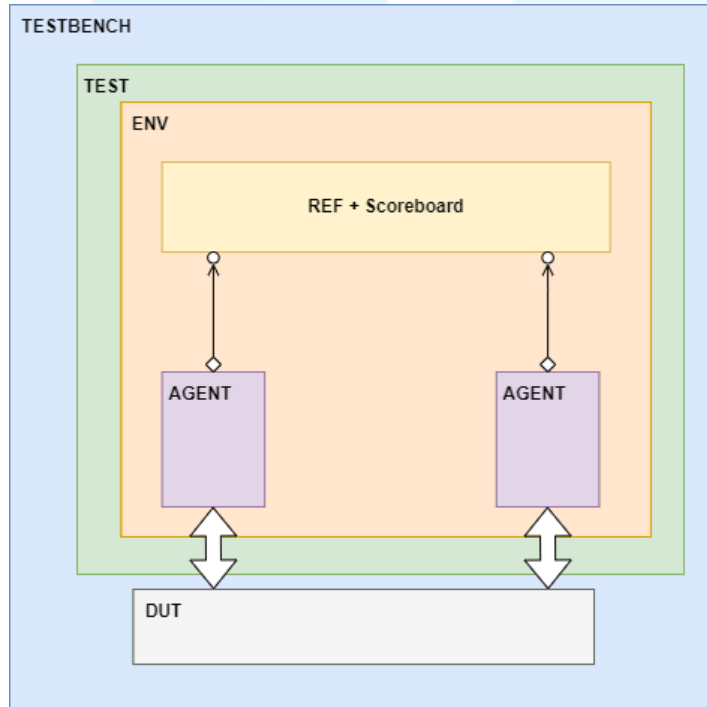
HDL Scoreboards (UVM, VMM, OVM, AVM, xVM...)



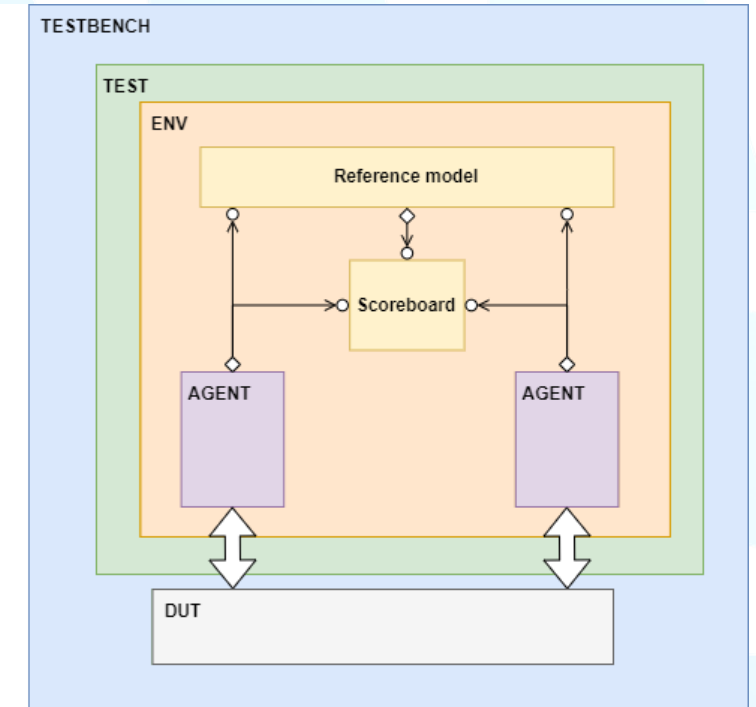
UVM Scoreboards

- General approach
 - Scoreboard + Ref in one
 - Makes it difficult to separate scoreboard and reference model development
- SyoSil Approach
 - Scoreboard and ref are separate
 - Can instantiate SCB and REF as necessary, increases modularity and flexibility

General Approach



SyoSil Approach



Motivation – UVM Scoreboard Landscape

- UVM native scoreboard is empty
- Existing user donations are limited in versatility
 - Employ blocking "expect" function as REF
 - Inhibits use of time-consuming REFs (e.g. SystemC)
 - Inhibits use of multiple concurrent models
- A reusable SCB is key for productivity and easy debug
 - We identify same principal structure across designs

SCB User Needs – What are those?

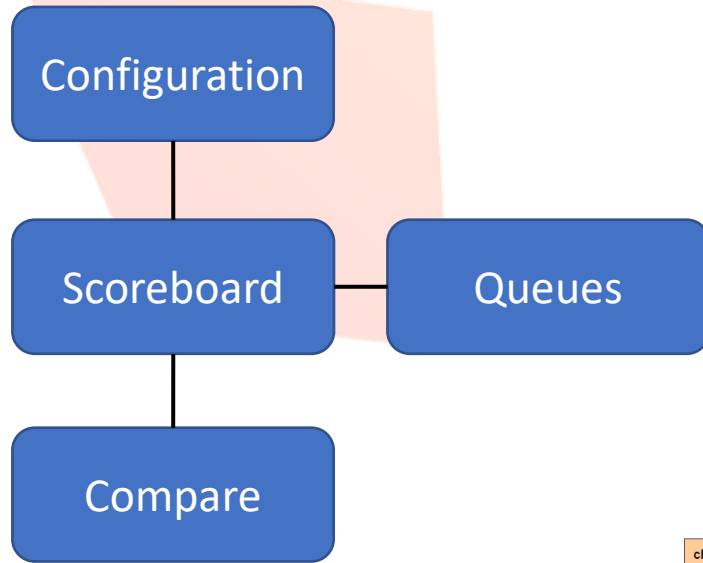
- Fast out of the box, easy to configure
- Consistent re-use
- Scalability (any number of models, queues, producers, compare methods)
- Clean interfaces to selfcontained models, e.g. SC
- Accelerated debug
- Inherently best performance
 - Search has linear and not polynomial complexity

Success Stories

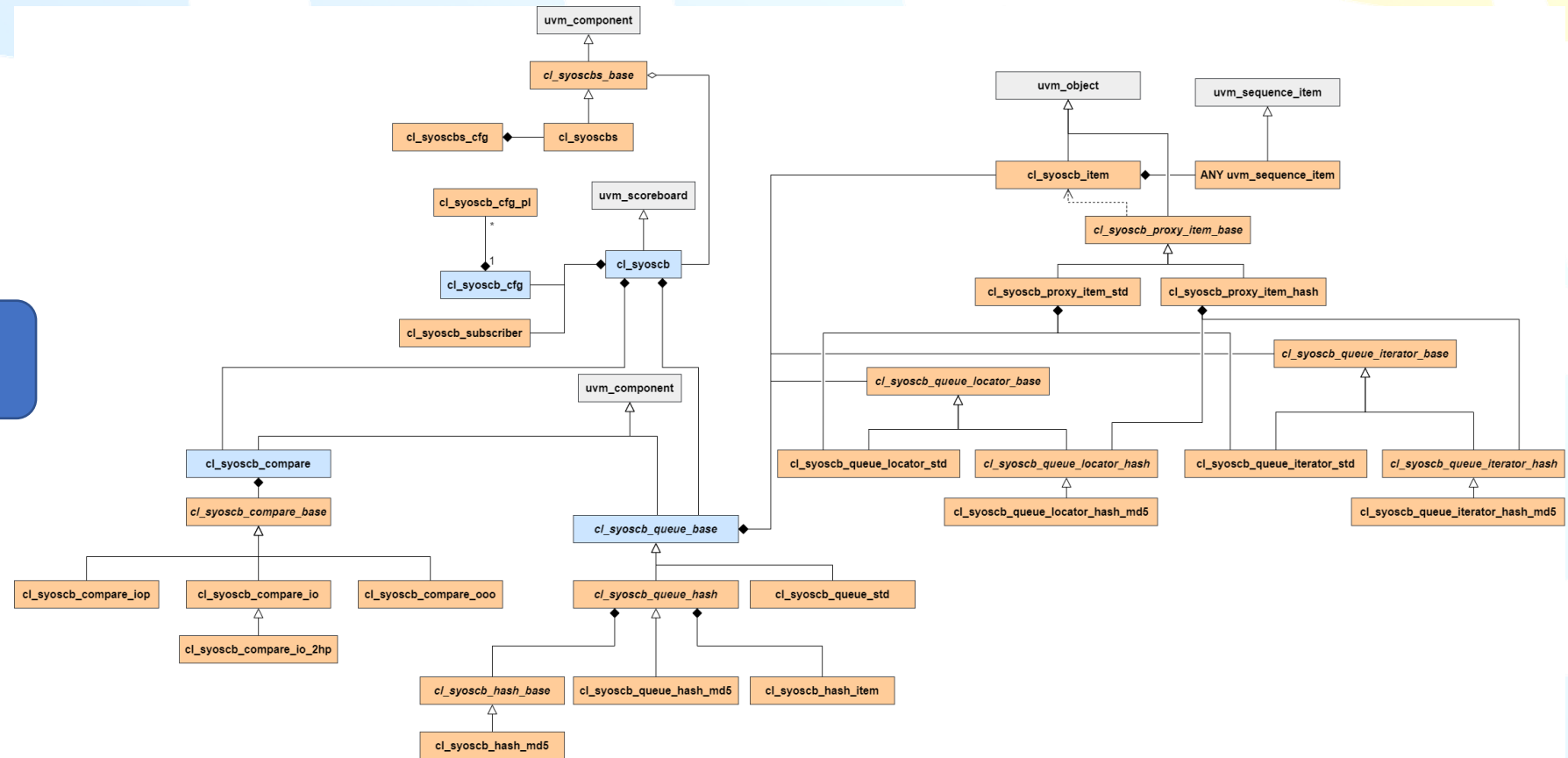
- Used across many UVM projects over the years
- Some test bench stats from 8 SV test benches
 - 10 – 75K lines
 - Scoreboards were 3-25% of the test bench size, in avg 15%
 - 15% code saved = 15% time saved
- We do SCB setup/config plus validation in less than a day for even complex designs
- Easy for newbies
- Same look&feel across all SCBs
- Out of the box
 - Top performance
 - Very good debug capabilities
 - Documentation
 - Numerous examples and self-tests

Overview of the SyoSCB VIP

Key Features - Architecture



Basic Architecture



UML Diagram

Key Features – Configuration

- In `build_phase` of your test, create and forward `cfg`

```
function void my_test::build_phase(uvm_phase phase);  
    cl_syoscb_cfg cfg = cl_syoscb_cfg::create("cfg");    //Create config  
    cfg.set_compare_type(pk_syoscb::SYOSCB_COMPARE_IO); //Use in-order comparisons  
    cfg.set_queue_type(pk_syoscb::SYOSCB_QUEUE_STD);    //Use standard SV-queues  
    cfg.init("SCB 1", `{"REF", "DUT"}, `{"P1", P2});    //Initialize 1 SCB named "SCB 1", two queues  
                                                         //(REF and DUT), and two producers for each (P1 and P2)  
    uvm_config_db #(cl_syoscb_cfg)::set(this, "scb_env", "cfg", cfg); //Set config in DB for environment  
    this.scb_env = my_env::type_id::create("scb_env"); //Create env which will get cfg and create scb  
endfunction: build_phase
```

- In `connect_phase` of your environment, get scoreboard subscribers and connect to correct producers

```
function void my_env::connect_phase(uvm_phase phase);  
    uvm_subscriber dut_p1_sub = this.scb.get_subscriber("DUT", "P1"); //Get subscriber for DUT, producer P1  
    this.dut_mon.p1_port.connect(dut_p1_sub.analysis_export);          //Connect to DUTs P1 analysis port  
    //do the same for remaining subscribers and producers  
endfunction: connect_phase
```

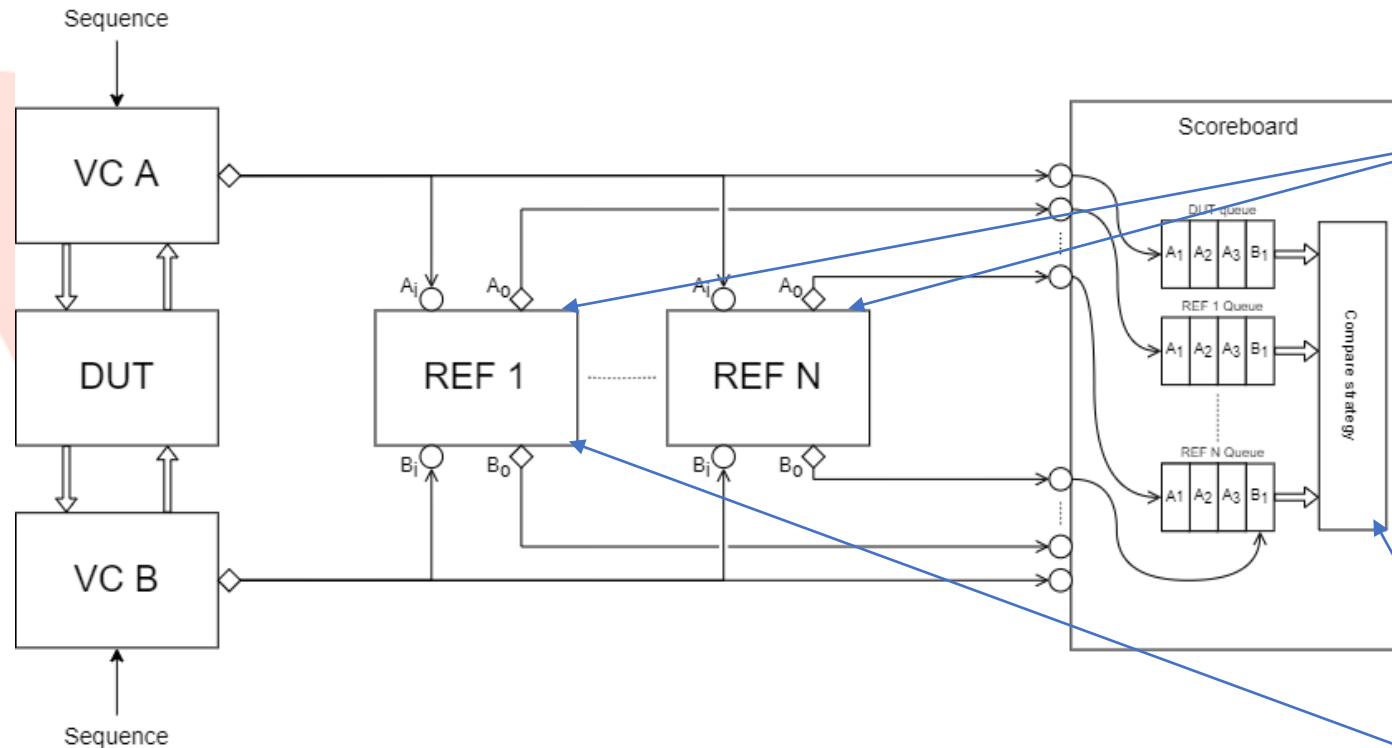
Key Features – UVM Versions&Configurability

- Supports different UVM versions
 - UVM1.1d, UVM1.2 and UVM IEEE
 - Directly without any configuration
- Configurability
 - Simple initial configuration and connection
 - 30+ configuration knobs
 - Two different queue implementations for optimal performance
 - Four different compare mechanisms for different scenarios
 - Supports multiple instances in the same test bench
 - Supports user defined compares and queue types if needed
 - Works with reals as well (analog simulation)

Key Features – Scalability and Architectural Separation

- Any number of models (multiple reference models...)
 - One primary model, multiple trailing models
 - Design models: RTL, gate
 - Timed/untimed reference models: SV, SC, Python, C/C++, ...
- Any number of queues
 - One per model, per transaction type, ...
 - Items in queues are tagged with metadata

Key Features - Scalability and Architectural Separation

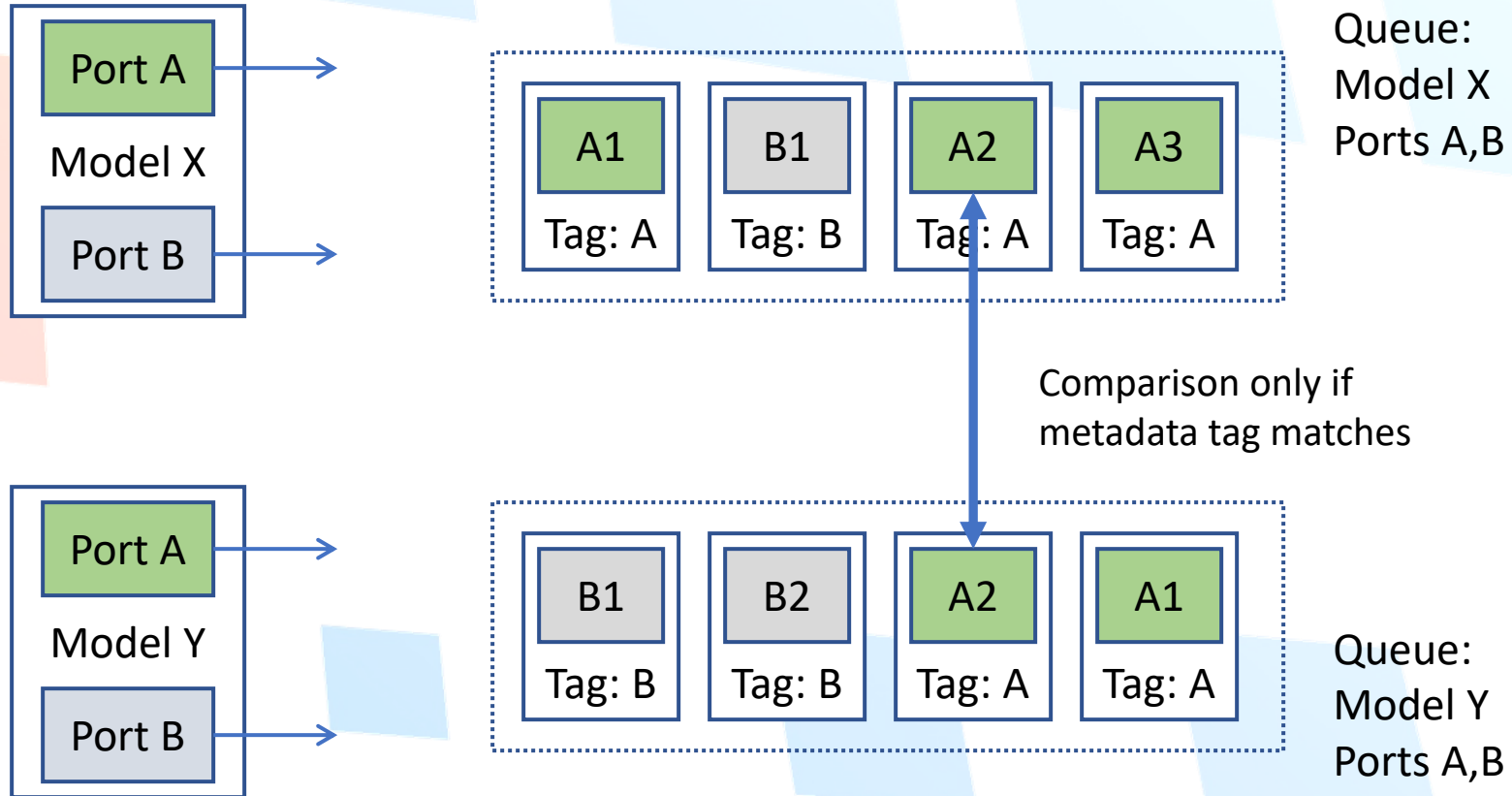


Scalable
Multiple Reference
Models

**Architectural
Separation**
SCB and Reference
models are
separated

Bi-dir Ports in DUT and reference models (e.g. read/write SoC protocol)

Key Features – Producer Tagging

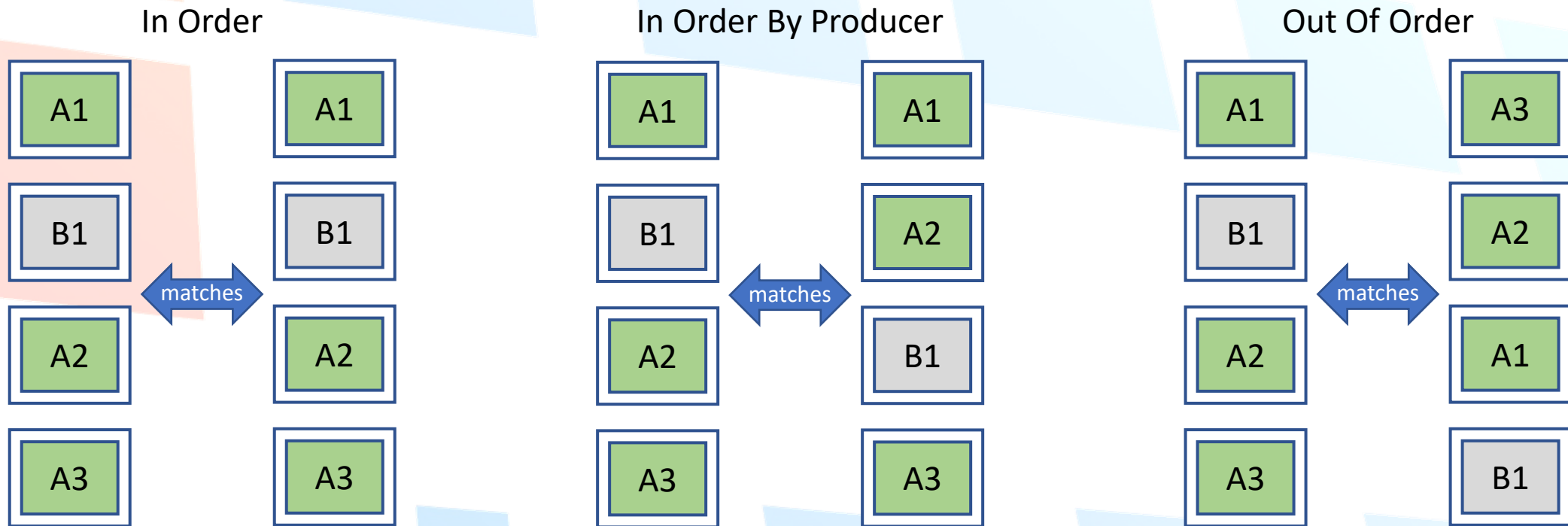


Metadata tagging of Sequence Items: One Queue per Model

Key Features - Debugging

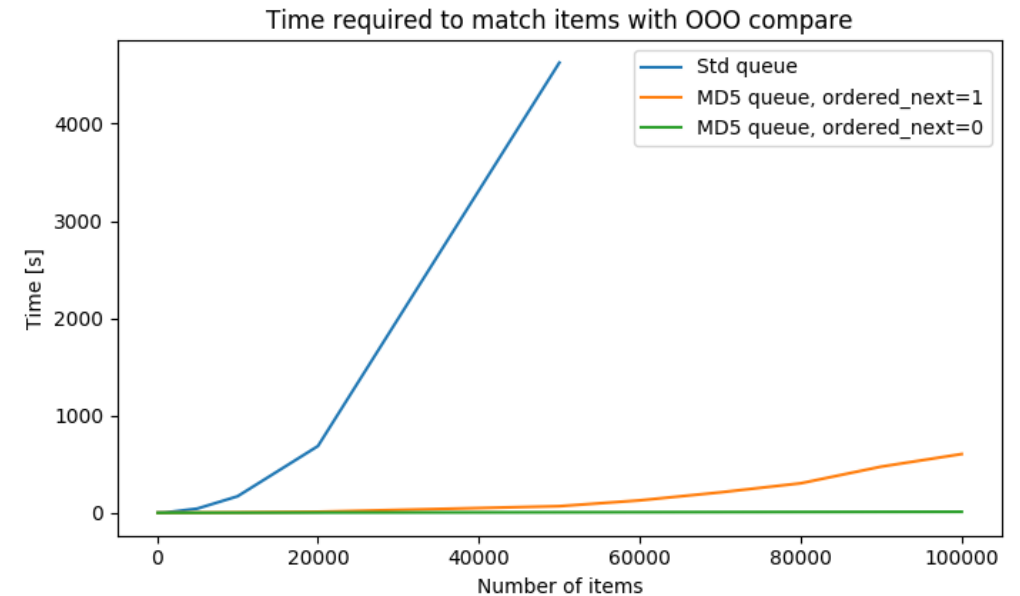
- No items inserted detection
 - Fails if scoreboard is empty
- Item isolation
 - Transactions are cloned per default when inserted, disallowing future modification
 - May be disabled by config knob `cl_syoscb_cfg::set_disable_clone(bit sdc)`
- Detection of orphans in queue(s)
- Content dump to file
 - All transactions, orphans
 - TXT or XML
- Side-by-side transaction view when errors occur

Key Features -Built-in Comparison Methods



Key Features – Queue Types

- Standard Queue
 - Using SV queues
 - Good for almost all applications
 - Poor performance for OOO compare with many elements in the queues
- MD5 Queue
 - Calculates hash key by MD5ing `<item>.pack()`
 - Inserts `<item>` into associative array with MD5 hash value as key
 - On comparison
 - Calculates hash key by MD5ing the other item
 - Checks whether the key exists in the associative array



In depth details of selected features

Configuration Features - Examples

Config knob	Purpose
<code>void set_max_queue_size(string queue_name, int unsigned mqs)</code>	Per-queue limit for the maximum number of elements that can be in a given queue. Issue a UVM_ERROR if exceeded.
<code>void set_max_search_window(string queue_name, int unsigned msw)</code>	Per-queue value determining the max number of elements that should be tried in each queue when performing OOO compare with std queues. If 0, all elements in the queue are tried. (Hash queues bypass this by using hash in AA)
<code>void set_comparer(uvm_comparer comparer, string queue_names[], string producer_names[])</code>	For all combinations of given queue names and producer names, use that comparer when comparing items. Allows for fine-tuned comparison behaviour.
<code>void set_scb_stat_interval(int unsigned ssi)</code>	Set an interval such that insertions, matches, flushes and remaining items in the SCB are printer after every N insertions into the scoreboard
<code>void set_disable_compare_after_error(bit dcae)</code>	Toggle whether the SCB should be able to proceed without comparing items once a UVM_ERROR has been detected

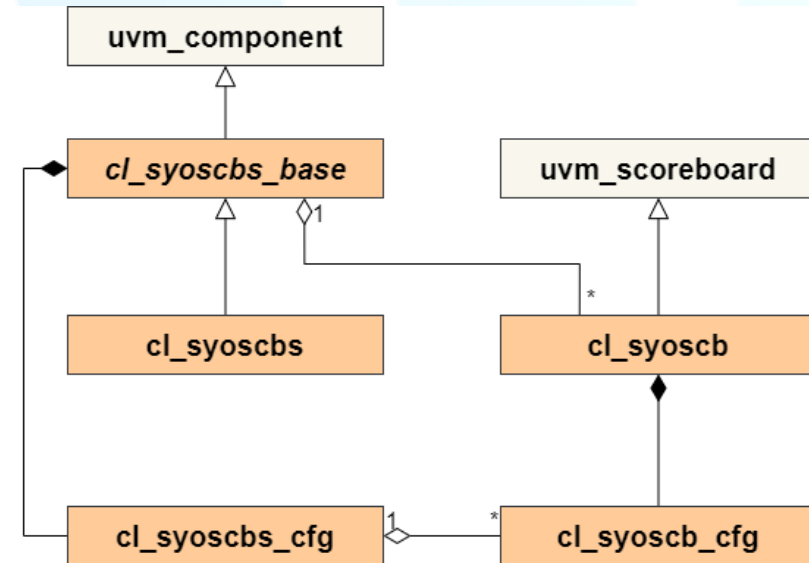
Runtime Features - Examples

- Some scoreboard features can be controlled at runtime from e.g. a `uvm_sequence`

Function	Purpose
<code>void cl_syoscb::compare_trigger(...)</code>	Manually trigger a comparison at any time
<code>void cl_syoscb::flush_queues_all()</code>	Flush all queues in the scoreboard,
<code>Void cl_syoscb::flush_queues(...)</code>	Flush a specific queue
<code>Void cl_syoscb::compare_control(...)</code>	Enable or disable comparisons

Wrapper for Multiple Instances

- Implemented by the `cl_syoscbs` class
 - Boosts implementation speed if many similar scoreboards are needed
 - Holds an array of `cl_syoscb` objects
 - Provides a method for accessing the subscriber for each combination of scoreboard, queue and producer for easy connection.
- Configuration done in similar fashion to `cl_syoscb_cfg` by the `cl_syoscbs_cfg` class.
 - Convenience function `cl_syoscbs_cfg::init(...)`;
 - Creates N scoreboards
 - Supports automatic naming or manual naming
 - Very simple setup and configuration
- Similar runtime features
 - Flush all queues in all score boards:
`flush_queues_all()`;
 - Toggle comparison in all scoreboards:
`cl_syoscbs::compare_control_all(...)`;
 - And more ...



Side-by-side Compare Error View

The scoreboard prints a side-by-side comparison of failing sequence items if they do not match

- Visual inspection of failing items
- Includes miscompare information to quickly identify which fields are failing in large sequence items

```
UVM_ERROR ./src/cl_syoscb_compare_io.svh(102) @ 0: reporter [COMPARE_ERROR]
#####
# [syoscb0]: cmp-io: Item from primary queue (Q1) not found in secondary queue (Q2) #
#####
-----
Name                Type                Size  Value                Name                Type                Size  Value
-----
P1-item-3881        cl_syoscb_item    -      @3881                P1-item-3982        cl_syoscb_item    -      @3982
  producer          string            2      P1                    producer          string            2      P1
  insertion_index    integral          64     'd3                    insertion_index    integral          64     'd3
  queue_index        integral          64     'd0                    queue_index        integral          64     'd0
  item              large_seq_item    -      @3922                item              large_seq_item    -      @3959
    int_a            integral          32     'hb9cc997a            int_a            integral          32     'hb0cca0e2
    int_b            integral          32     'h0                    int_b            integral          32     'h0
    int_arr          da(integral)      9      -                      int_arr          da(integral)      9      -
      [0]            integral          32     'h325ea203            [0]            integral          32     'hfd0e2295
      [1]            integral          32     'h6bfce43a            [1]            integral          32     'h706ab852
      [2]            integral          32     'h87cde414            [2]            integral          32     'h9cce61cb
      [3]            integral          32     'h2f81a427            [3]            integral          32     'hf3c23513
      [4]            integral          32     'hc6f99f4c            [4]            integral          32     'h69bb5ea7
      [5]            integral          32     'h4cb314e2            [5]            integral          32     'h5acc6b0b
      [6]            integral          32     'h84d81aab            [6]            integral          32     'h4227f3a3
      [7]            integral          32     'h12e31653            [7]            integral          32     'h5c79005
      [8]            integral          32     'h97bfa2c4            [8]            integral          32     'h1665fb34
-----
#####
Results from uvm_comparer::get_miscompares() [show_max=5]
-----
P1-item-3982.item.int_a: lhs = 'hb0cca0e2 : rhs = 'hb9cc997a
P1-item-3982.item.int_arr[0]: lhs = 'hfd0e2295 : rhs = 'h325ea203
P1-item-3982.item.int_arr[1]: lhs = 'h706ab852 : rhs = 'h6bfce43a
P1-item-3982.item.int_arr[2]: lhs = 'h9cce61cb : rhs = 'h87cde414
P1-item-3982.item.int_arr[3]: lhs = 'hf3c23513 : rhs = 'h2f81a427
#####
```


Dump to file

- Scoreboarded items may be dumped to a TXT or XML file.
- Full scoreboard dump
 - Dumps all scoreboarded transactions to file for visual inspection or automated postprocessing
 - May either dump all transactions to the same file, or one file per queue (config knob `full_scb_dump_split`)
- Orphans dump
 - Dumps items remaining in queues after simulation finishes. Always dumps orphans to one file per queue.

Text dump

- Simple visual inspection
- Example from `cl_scb_test_io_std_dump`, using very simple sequence items:

```
////////////////////////////////////  
// Queue: Q1                                     //  
////////////////////////////////////
```

```
//--item:      0 -----//  
int_a: 0, data.size: 0, use_data: 0
```

```
//--item:      1 -----//  
int_a: 1, data.size: 0, use_data: 0
```

```
//--item:      2 -----//  
int_a: 2, data.size: 0, use_data: 0
```

```
...
```

XML Dump

- Allows for transformation to other formats via XSLT
- Scoreboard comes with XSLT files for transforming to HTML and GraphML
- HTML transform supports nested items to any depth by hiding them until clicked

Queue Q1				Queue Q2			
P1-item-3841		Producer: P1		P1-item-3951		Producer: P1	
Queue index: -1		Insertion index: 0		Queue index: -1		Insertion index: 0	
Name	Type	Size	Value	Name	Type	Size	Value
a	integral	32	'h66cf6bd4	arr	sa(integral)	15	► arr
b	integral	8	'h92	children	sa(object)	3	► children
t	t_on_off	32	ON				
r	real	64	3.140000				
str	string	12	Hello, world				
q	da(integral)	3	► q				
aa	aa(int,string)	3	▼ aa				
			[hundred]				'h64
			[one]				'h1
			[two]				'h2
child	uxp_child_seq_item	-	▼ child				
			Name	Type	Size	Value	
			arr	sa(integral)	15	▼ arr	
						[0]	'h0
						[1]	'h0
						[2]	'h0
						[3]	'h0
						[4]	'h0
					
						[10]	'h0
						[11]	'h0
						[12]	'h0
						[13]	'h0
						[14]	'h0
			children	sa(object)	3	▼ children	
						[0]	▼ [0] (uxp_small_seq_item)
						i	integral 32 'h0
						by	integral 8 'h0
						[1]	<null>
						[2]	► [2] (uxp_small_seq_item)
empt	da(integral)	0	[]				
null_object	object	-	<null>				
P1-item-3830		Producer: P1					
Queue index: -1		Insertion index: 1					
Name	Type	Size	Value				
a	integral	32	'heb0bf046				
b	integral	8	'hd8				
t	t_on_off	32	ON				
r	real	64	6.280000				
str	string	12	Hello, world				
q	da(integral)	6	► q				
aa	aa(int,string)	3	► aa				
child	uxp_child_seq_item	-	► child				
empt	da(integral)	0	[]				
null_object	object	-	<null>				

Release Contents

- License: APACHE - Like UVM ☺
- Current official released version is: 1.0.2.5
- New version out in beta: 1.0.3.beta
- Shipped as a tar.gz file
- To get it:
 - [Accellera](#)
 - Old version 1.0.2.5, downloaded 800+ times
 - Write email to: scoreboard@syosil.com
 - Will also be available on our website soon

Release Contents

- Source code
 - Fully commented, all methods and classes are described
 - Easy to navigate, clear separation of user-facing API and backend
- Testbench
 - Examples and installation validation
 - Runs on all major commercial simulators
 - Siemens EDA – Questa, Synopsys – VCS, Cadence – Xcelium
 - Contains Makefile support for all of them
 - Contains functional coverage model for features
 - Delivered AS-IS
 - But comprehensive, 70+ tests showcasing features and ensuring functionality
- Documentation
 - User manual in HTML and PDF format + API descriptions
 - The original Papers from DVCon EU 2014 and US 2015
 - This presentation

Thank you for listening 😊

- Questions?