2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

METAX 沐曦集成电路

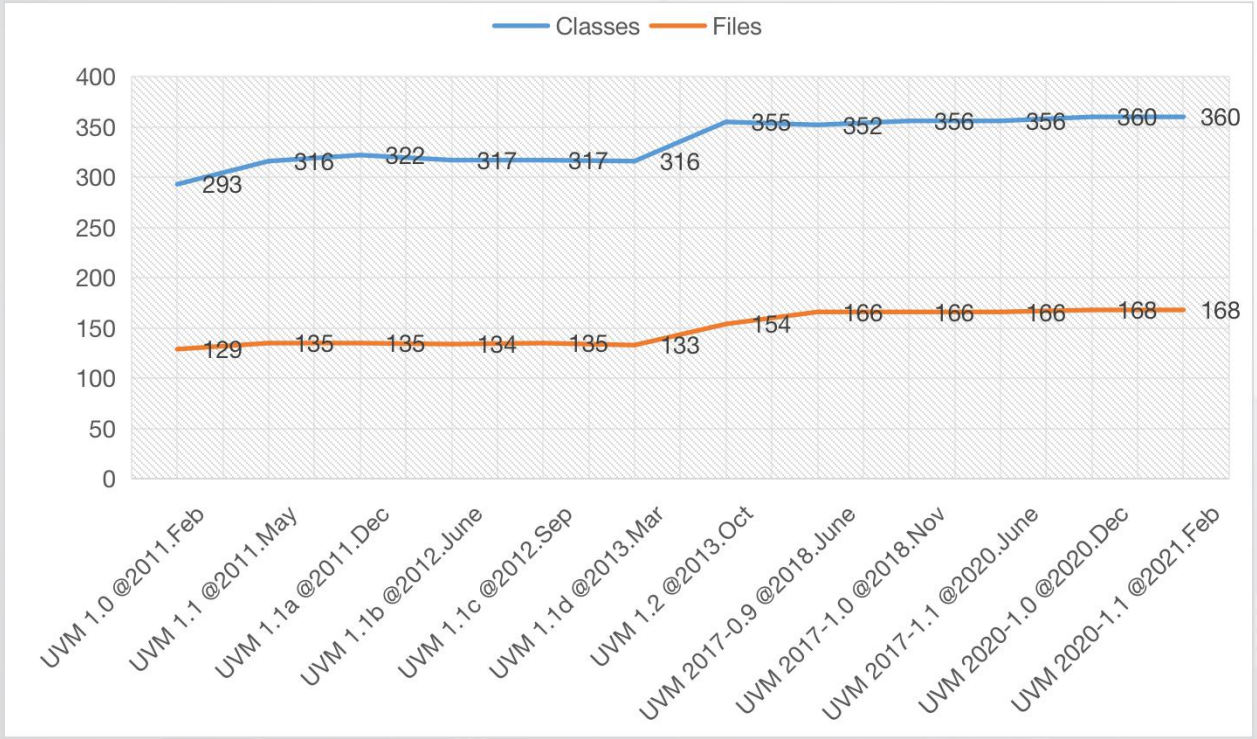
UVM is Now IEEE1800.2-2020 Standard: A 2020 Adoption Primer

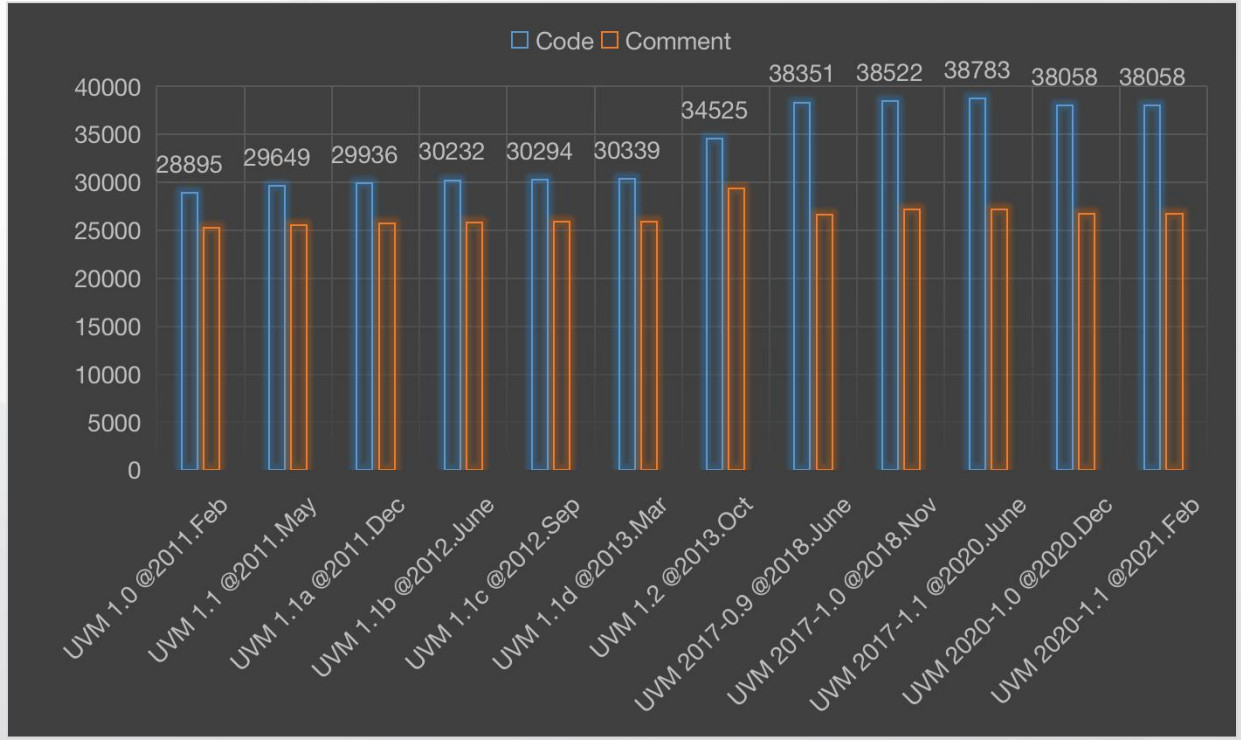
Roman Wang

MetaX Integrated Circuits (Shanghai) Co., Ltd. 沐曦集成电路

Happy Birthday UVM

- Before UVM: eRM, VMM, AVM, OVM, etc.
- 10th Years Anniversary
  - First UVM-1.0 release: 2011. Feb.
  - UVM-2020.1.1 release: 2021. Feb.





Changes to UVM RAL and Memory

- Rand mode property is not changed correctly
  - In the uvm_reg_field, invoking set_access() will change the register access mode, but the rand mode property is not change accordingly.
  - NOT one of "RW", "WRC", "WRS", "WO", "W1", or "WO1", the value of ~is_rand~ is ignored and the rand_mode() for the field instance is turned off since it cannot be written.

Scenario	Previous Access	New Access	Rand Mode
1	Non-writable	Non-writable	0
2	Non-writable	Writable	?
3	Writable	Non-writable	0
4	Writable	Writable	x

Two A

```
// modifies the ~rand_mode~ for the field instance to the specified one
set_rand_mode(bit rand_mode);
//  returns the rand_mode of the field instance.
bit get_rand_mode();
```

Changes to UVM RAL and Memory

- Order issue post_read() invoking.

In the section 18.6.9.4 of 1800.2 LRM, it says "the registered callback methods are invoked before the invocation of this method" where the method is post_read().

```
// POST-READ CBS
post_read(rw);
for (uvm_reg_cbs cb=cbs.first(); cb!=null;
cb=cbs.next())
        cb.post_read(rw);
```

```
// POST-READ CBS
for (uvm_reg_cbs cb=cbs.first(); cb!=null;
cb=cbs.next())
      cb.post_read(rw);
post_read(rw);
```

- uvm_mem::get_addresses() function doesn't check if offset argument is smaller than array size-1.

For example,

if the ABC memory has 10 addresses, but you call model.ABC.get_addresses(11, model.map, addr), it will issue a warning like "RegModel, Offset 11 Lies outside of memory ABC, which has a size of 10" .

Changes to UVM RAL and Memory

- The uvm_reg::do_read has malfunction in post_read() function description.

As post_read() function description, it says "if the specified readback data or ~status~ is modified, the updated readback data or status will be returned by the register operation." .

The do_read function in which the post_read is previously called restores the rw.value so that any change made in post_read is restored to its original value.

Adding get/set_* API in uvm_reg_item.

```
value = rw.value[0]; // preserve
 // POST-READ CBS - REG
……
// POST-READ CBS – FIELDS
……
rw.value[0] = value; // restore
rw.element = this;
rw.element_kind = UVM_REG;
```

```
// POST-READ CBS - REG
…….
value = rw.get_value(0);
   // POST-READ CBS - FIELDS
……
rw.set_value(value, 0);
rw.set_element(this);
rw.set_element_kind(UVM_REG);
```

Changes to UVM RAL and Memory

- The uvm_reg::do_read has malfunction in set_check_on_read() function description.

The LRM for set_check_on_read (18.2.5.3 in 1800.2-2017) refers to checking against the mirrored value, but in do_read(), the previous code for both FRONTDOOR and BACKDOOR compares against get() not get_mirrored_value(). If you are already on 1800.2 2017 or 2020, please call **map.set_check_on_read(1)** to enable this feature.

```
UVM_BACKDOOR: begin
 ……..
   if (map.get_check_on_read()) exp = get();
   ……..
UVM_FRONTDOOR: begin
  ……..

  if (local_map.get_check_on_read()) exp = get();
```

```
UVM_BACKDOOR: begin
 ……..
   if(map.get_check_on_read())
      exp = get_mirrored_value();
   ……..
UVM_FRONTDOOR: begin
  ……..

   if (local_map.get_check_on_read())
      exp = get_mirrored_value();
```

What's set_check_on_read?

The register model will automatically check any value read back fro
m
 a register or field against the current value in its mirror and report any discrepancy.

This effectively combines the functionality of the <uvm_reg::read()> and ~uvm_reg::mirror(UVM_CHECK)~ method.

This mode is useful when the register model is used passively.

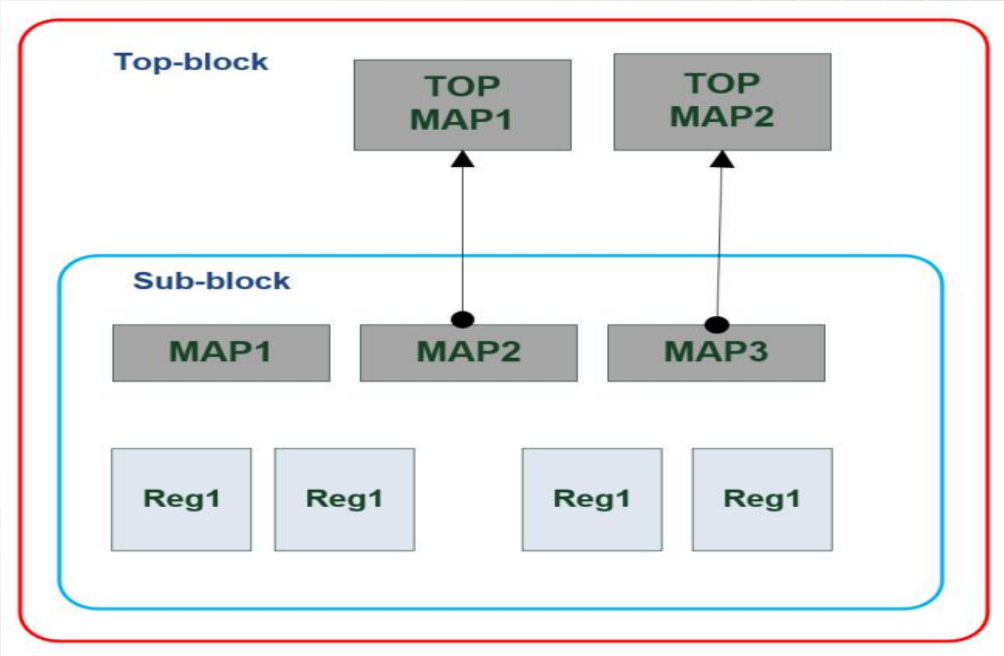
This mode setting doesn't impact on register prediction at the end of the read

Changes to UVM RAL and Memory

- UVM map issue in uvm_re_block

uvm_reg_block, when sub-block has more than one map defined then only default (used) map is marked initialized when locking the model.

UVM_WARNING ..../src/reg/uvm_reg_map.svh(1280) @ 0: reporter [RegModel] map '**map1**' does not seem to be initialized correctly, check that the top register model is locked().



- Random issue in register test sequences

The register test sequences rely on the non-deterministic **$random** system call.

(specially uvm_mem_access_seq.svh & uvm_reg_mem_shared_access_seq.svh)

Now it replaces it with **::randomize()** call

Changes to UVM Reporting

- **uvm_default_report_server::report_summarize()**

The result of report_summarize() should be sent to the descriptor in the argument, but the implementation always sends to STDOUT regardless of argument value.

```
function void report_summarize(UVM_FILE file = 0);
```

```
function void report_summarize(UVM_FILE file = UVM_STDOUT);
```

```
virtual task run_phase (uvm_phase phase);
    uvm_report_server rs = uvm_report_server::get_server();
    fh= $fopen("uvm_report_summary", "w");
    ……..
    rs.report_summarize(fh);
    …….
    $fclose(fh);
```

Changes to UVM Command Line

- **Command line control message order issue**

Any attempt to control the verbosity/action/severity of messages traversing through the uvm_root instance is currently ignored by the library.

The code which is responsible for dealing with the command line arguments of verbosity happens later in the uvm_component constructor.

If uvm_root is the *only* target of these command line arguments, you'll get a warning

```
UVM_WARNING uvm_root.svh(1069) @ 0.000ns: reporter [INVLCMDARGS]
"+uvm_set_action=/^$/,FOO,UVM_WARNING,UVM_NO_ACTION" never took effect due to a
mismatching component pattern
```

However, if uvm_root is one of many targets, then no warning is emitted, and the arguments are silently ignored.

UVM 1800.2 2020 now allows command line control of messages before the uvm_root::build_phase().

Changes to UVM Sequence

- Enhancement for stacking sequencers

When stacking sequencers, this will cause a problem as the single call in a low-level sequencer is absorbed by the next call in the higher-level sequencer(s).

UVM 1800.2 2020 adds **wait_for_sequences_count** (higher than 1) to improve sequencer layering implementation. Increasing this value will decrease the efficiency

```
forever begin
    start_item(LOW_req); // will return when LOW_driver calls try_next_item
    if (HIGH_req == null) begin
      HIGH_driver.seq_item_port.try_next_item(HIGH_req); // PROBLEM
HERE
    end
    translate(HIGH_req,LOW_req);
    finish_item(LOW_req);
    HIGH_driver.seq_item_port.item_done(HIGH_req);
```

Try_next_item is a tas

Low-level sequencer

```
uvm_config_db#(int)::set(this, "path.to.sequencer", "wait_for_sequences_count", 4);
```

Changes to UVM Callback and comparer

- UVM Callback

Inside of the the uvm_callbacks#() class, a warning is sent if the user has two callbacks with the same name.

The name only ever occurs in debug information (uvm_callbacks#().display()), no functionality is based of the callback name.

- UVM Comparer

uvm_compare::compare_field_int doesn't have check the size to be less than 64, which means users can supply a size value greater than 64 causing "mask" field to overflow.

Mask controls the comparison and if it overflows then the whole comparison is masked. Thereby hiding failures.

Changes to UVM Packer and component

- uvm_packer::big_endian

1.2	The default value is 1
1800.2 2017	The default value is 0
1800.2 2020	The big_endian is removed and implementation is little endian by default.

- UVM Component

1. Add accessors set_print_config_matches() and get_print_config_matches() in uvm_component
2. Deprecate directed access to print_config_matches field

Migration

- UVM-1.2 -> IEEE1800.2

1. **Ensure that your code runs with UVM 1800.2-2017 which was a baseline for the UVM 1800.2-2020 library development**

Compile/Run using a UVM 1800.2-2017/2020 library with `UVM_NO_DEPRECATED` defined

2. **Identify the areas where your code may need modifications to comply with the standard**

Compile/Run using this library with `UVM_ENABLE_DEPRECATED_API` defined

3. **Ensures that only the 1800.2 API documented in the standard, along with any non-deprecation Accellera supplied API, is used**

Compile/Run using this library without `UVM_ENABLE_DEPRECATED_API` defined

[Note] All code deprecated in UVM 1800.2-2017 has been removed from UVM 1800.2-2020 library.

- **The 3rd part UVM VIP and UVM environment**

Contact with your vendor and get support.

Conclusion

2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

- IEEE 1800.2 2020 is coming, more features with bugs fixes, why not to adopt with your code.
- Report issues to the Accellera Forum (http://forums.accellera.org/forum/) or ask your simulator vendor to submit an issue

roman.wang@metax-tech.com

Disclaimer & Attribution

DISCLAIMER

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurately for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between different manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. MetaX assumes no obligation to update or otherwise correct or revise this information. However, MetaX reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of MetaX to notify any person of such revisions or changes.

ATTRIBUTION

http://www.metax-tech.com/