2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

UVM – Stop Hitting Your Brother Coding Guidelines

Chris Spear & Rich Edelman

Mentor, A Siemens Business

Mentor® A Siemens Business

accellera SYSTEMS INITIATIVE

Are We There Yet?

- What is keeping your project from completion?
  - Hint: It's not your simulator speed
  - Brain speed for verification engineers

- Write clear code for maximum reuse
  - Follow common coding styles

- First, tell people what to do
  - Then what NOT to do

Configuration Tips

- What is the most confusing part of UVM?
  - Configuration!
- Group a compont's configuration values together: "config object"
  - Instead of 1000's of uvm_config_db::set()/get() for individual values
  - Two flavors: environment and agent config objects
- When uvm_config_db::get() fails, end simulation
- Minimize wildcards in uvm_config_db::set()

```
uvm_config_db #(int)::set(null, "*", "addr", 42); // BAD - Global scope!
```

```
% find / -name emacs -print
```

Configuration Tips (Cont.)

- Better yet: pass config objects between components with OOP-style

```
class test_base extends …;
  env_config env_cfg;

  function void build_phase(…);
    env = …::create("env", this);
    env_cfg = …create("env_cfg);
    env.set_config(env_cfg);
  endfunction
```

```
class env …;
  env_config env_cfg;

  virtual function void set_config(env_config env_cfg);
    this.env_cfg = env_cfg;
  endfunction
```

  – One argument vs. five

```
uvm_config_db #(env_config)::set(this, "env", "env_cfg", env_cfg);
```

```
uvm_config_db #(env_config)::get(this, "", "env_cfg", env_cfg);
```

Transaction Tips

- How to write sequence item classes?
- Quickly with field macros!

- Avoid the field macros
  - What about conditional fields?
    - if (cmd==NOP) don't compare src, dst
  - Write do_copy(), do_compare(), …

- In any case, never mix the two

```
class tx_item extends uvm_sequence_item;
  rand logic [31:0] src, dst;
  rand command_t    cmd;
  rand tx_payload   pay_h;
       logic [31:0] result;
       logic [99:0] temp;

  `uvm_object_utils_begin(tx_item)
    `uvm_field_int(src, UVM_ALL_ON)
    `uvm_field_int(dst, UVM_ALL_ON)
    `uvm_field_enum(command_t, cmd, UVM_ALL_ON)
    `uvm_field_int(result, UVM_ALL_ON)
    `uvm_field_object(pay_h, UVM_ALL_ON)
  `uvm_object_utils_end

  function new(string name="tx_item");
    super.new(name);
  endfunction
endclass
```

Sequence Tip

- Simplify starting your sequence
  - Pass sequencer, config object, status, …

```
class tx_seq extends uvm_sequence #(tx_item);
  int num_xacts;

  virtual task init_start(output uvm_status_e aok,
                          input uvm_sequencer #(tx_item) sqr,
                          input int num_xacts);
    this.num_xacts = num_xacts;  // Parameter to seq
    this.start(sqr);             // Start this sequence
    aok = UVM_IS_OK;             // All is cool
  endtask

  …
endclass
```

```
class tx_test extends test_base;
  tx_seq seq;

  task run_phase(…);
    seq = …::create("seq");
    seq.init_start(aok, sqr, 42);
  endtask
```

Sequence Tips

- Learn the uvm_sequence_item create(), start_item(), finish_item
  - Avoid the "training wheels" `uvm_do() macros

```
virtual task tx_sequence::body();
  tx_item tx;

  `uvm_do(tx);
endtask
```

```
virtual task tx_sequence::body();
  tx_item tx;

  tx = tx_item::type_id::create("tx");
  start_item(tx);
  if(!tx.randomize()) `uvm_fatal(...)
  finish_item(tx);
endtask
```

Misc. Tips

- Minimize objections to just test level, and maybe scoreboard
  - Avoid objections in sequences, drivers, …
- Construct objects with factory create() method, not new()
  - Except uvm_sequencer and TLM connections

```
`define my_create_o(_type, _name) _name = _type::type_id::create(`"_name`")
`define my_create_c(_type, _name, _p=this) _name = _type::type_id::create(`"_name`", _p)

tx_item t;
tx_agent agt;
`my_create_o(tx_item, t);    // t = tx_item::type_id::create("t")
`my_create_c(tx_agent, agt); // agt = tx_agent::type_id::create("agt", this)
```

- If uvm_object::create() had a second argument, only need a single macro

Acknowledgements

- Mike Baird
- Cliff Cummings
- Bob Oden
- Stuart Sutherland
- James Chang
- And many more!