UVM → System Verilog - Base class library

- Connecting Environment to DUT
- Transactions
- Sequence & Tests
- Monitors & Subscribers
- Reporting

OVM - Mentor & Cadence

→ Constrained random, coverage driven verification
Seperation of test from test bench
- Transaction level communication, layered sequential stimulus, Standardized messaging, Register layer. (adv)**

Checkers → to check expected op reached. (Does it work)

Functional coverage → to know (when to end)
↻ Header, payload, checksum.

Configurable & reuse of IP

Constrains modify to increase coverage.

sequence (set of tranx)

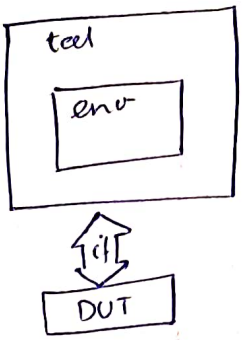
transaction → driver → pin level signal.
packet

Fixed part - env

Variable part - test

Class based - test, env

Structural - if, dut



25-12-2024

# UVM

-Umesh Saw

- UVM automatically re build, compile & run test
- uvm architecture (sequence → gen)

History

eRM (Verisity) → RVM (Synopsys) → AVM (Mentor) → URM (Cadence) → OVM (Cadence Mentor) ** → UVM (Mentor, Cadence, Synopsys)

Accellera standard.

UVM 1.2

UVM macro, UVM package

`include "uvm_macro.svh"  import uvm_pkg::*;

`UVM_info(<ID>, <MSG>, <verbosity>

Macros

→ `include " .svh"

`define DATA 1

$sformatf("Var1 is %0d", var)

---

UVM component

:null ` to handel

- type_quasi static
- phases (function, task)

build_phase, connect_phase, end_of_elaboration_phase, start_of_simulation, extract phase, check_phase, report_phase, final phase

run_phase

↓

task

run_test("<class_name>") ⟶ starts test in UVM.

```
`include "uvm_macros.svh"
import uvm_pkg::*;

class my_component extends uvm_component;
  `uvm_component_utils(my_compond); // semicolon not require as macro expansion

  function new(string name="", uvm_component parent);
    super.new(name, parent);
    `uvm_info("", "This is my_component", UVM_NONE);
  end function
endclass

module my_test;
  initial begin
    run_test("my_compond"); // runstest
  end
endmodule
```

---

`uvm_component_utils ⟶ factory registration.

↳ Cannot create compoent of type "my_compond" because it is not registered with the factory

---

```
run_test("");
```

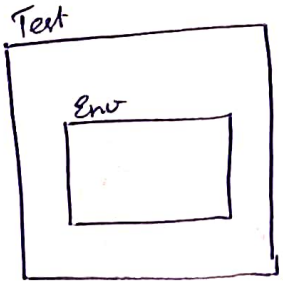
```
qverilog <filename> -R +UVM_TESTNAME=my_component
                                     component_utils name
```

class variable cannot be added to wave → done through interface.

## build_phase

```
function void build_phase (uvm_phase phase)
    super.build_phase (phase);
    `uvm_info(" ", < >, ;)
endfunction
```

## Nested component




```
env env_h;

function build_phase ( .... );
  ...
  ...
  env_h = env::type_id::create("env_h",this);
endfunction
```

## order of phases

build : Top down  Tasks, are parallel
final :

## UVM_reporting

UVM_info , UVM_error → sim - doesn't stop, error - displayed
, UVM_fatal → stops simulation & exits, UVM_warning,

(<string>,<string>)

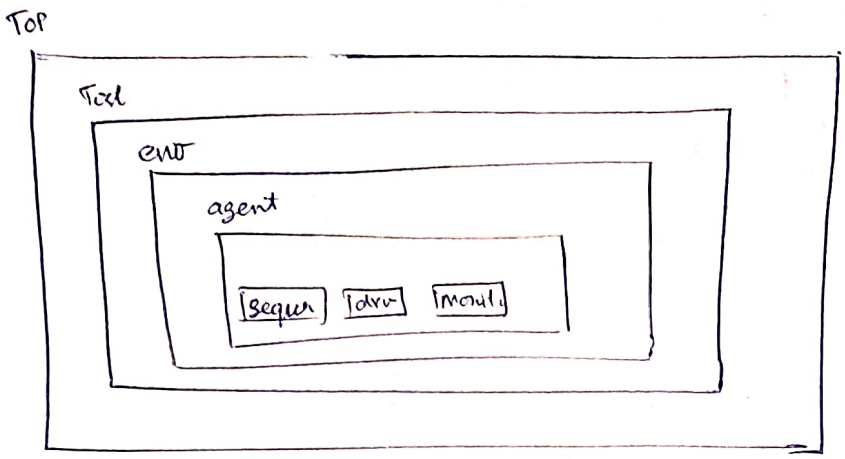
("<string>", <string>, <enum>)

default ← UVM_MEDIUM - 200

UVM_NONE - 0
UVM_LOW - 100
UVM_HIGH - 300
UVM_FULL - 400
UVM_DEBUG - 500

+UVM_VERBOSITY = <verbosity>
↳ UVM_NONE, UVM_DEBUG, ...

---




---

uvm_top.print_topology();

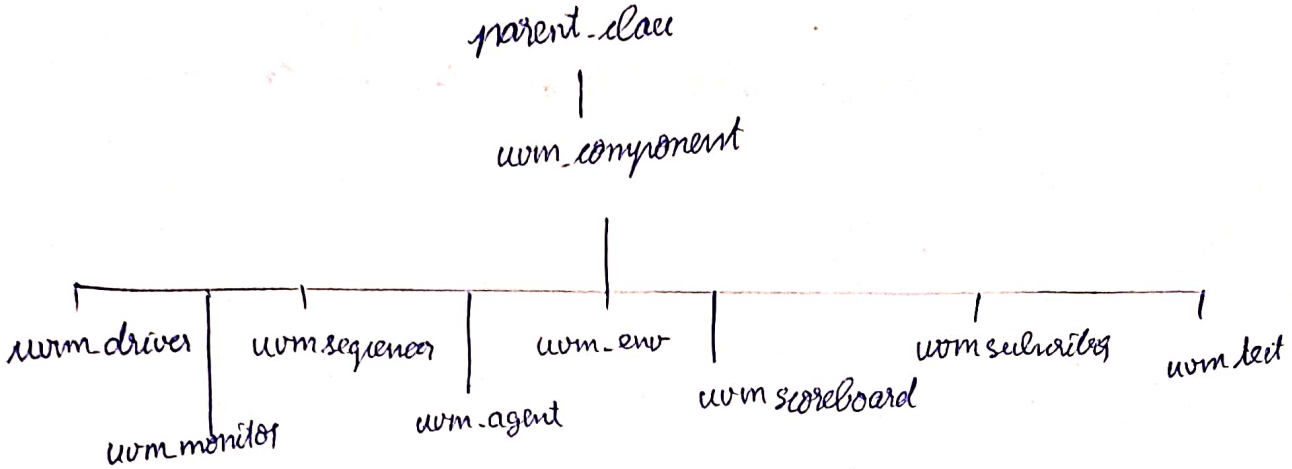
uvm_test_top
  env_h
    agent_h
      drv_h
      mon_h
      seq_h

---

get_name() ⟶ obj name.

get_ful_name() ⟶ obj name full with heirarchy

get_type_name() ⟶ class name




---

initial begin
  $display("before test");
  run_test("test");
  $display("after test");
end

user_defined_phase → don't use - debug problem.

uvm_config_db → single_ton class .uvm_top

↳ static database → set(), get() to put & retrive data.

uvm_config_db #(typeT) :: set(cntxt, → null / this
"inst_name", → 'key'
"field_name", → i/10
value);

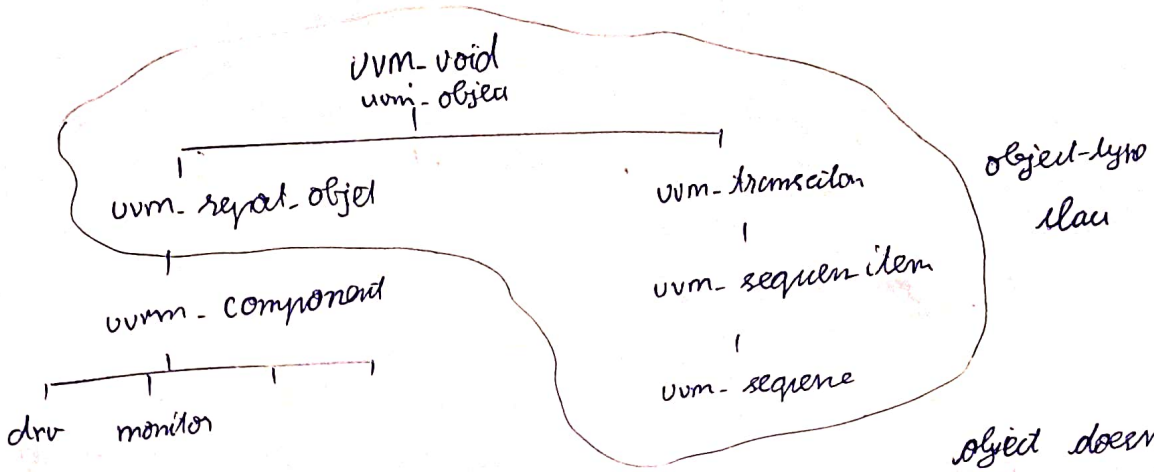
from → test to → top

raise exception,
#10
drop exception

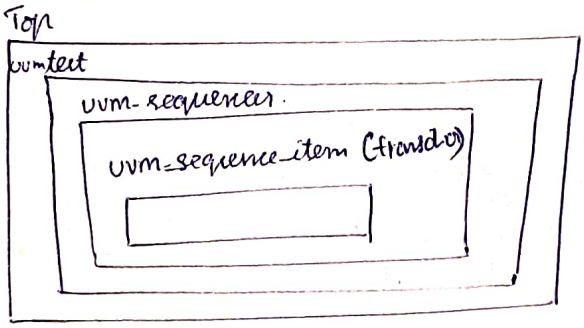
#2
uvm_config_db#(typ)::get(...);

26-12-2024

Using config db send "virtual interface" from top.sv to env.sv




object doesnot have phases

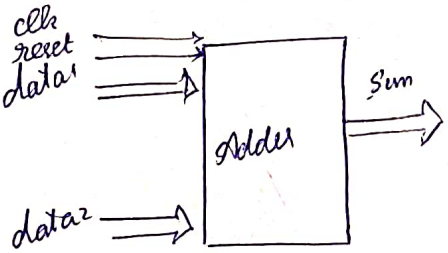



→ sequence item class.

copy,
compare,
print,
get,
create

record.
pack

↓ ports to be declared in my_sequence class




as bits

Factory registry → `uvm_object_utils (user_sequence_item) [to register class]

`uvm_object_utils_begin(<component> / ~~<object name>~~)
`uvm_field_int-(<objname>)
`uvm_object_utils_end.

[to register object]

class user_seq_item extends uvm_sequence_item;

bit[30] data1, data2;
bit[4:0] sum;
`uvm_object_utils_begin()
`uvm_field_int (data1);
`uvm_field_int (data2);
`uvm_object_utils_end()

→ factory registry.

If not registered output won't be there

for_print()
x cannot be used inside any method

| UVM_DEC
- OCT
- HEX

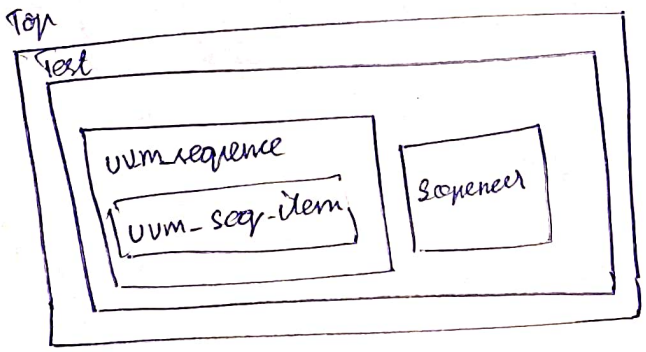
copy, compare, (clone)

uvm_sequence (generator ~~transaction~~)

1. pre_start, start, post start,
2. pre_body, body, post body,
3. pre_do, mid_do, post_do
4. start_item, finish item, get response.

seq.start (seqr)

to call body() of seq




pre_start
pre_body
pre_do
mid_do
post_do
post_body
post_start

P-sequencer , m-sequencer.

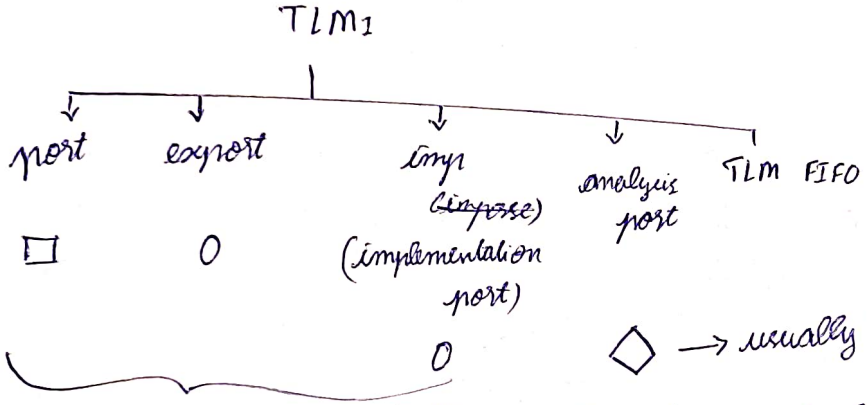
typedef ⇒ forward declaration.

`uvm_declare_p_sequencer(user_sequencer))`

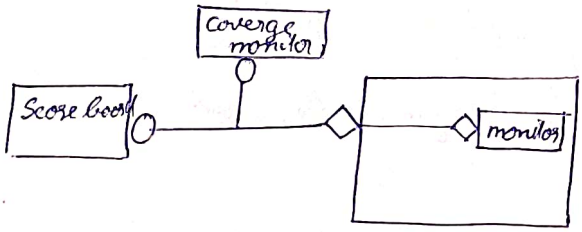
p_sequencer variable defined.

27-12-2024

## Transaction Level Modeling → Similar to fifo (mailbox)




| queue vs | mailbox |
|---|---|
| FIFO | FIFO |
| element can be removed from midde | not possible |

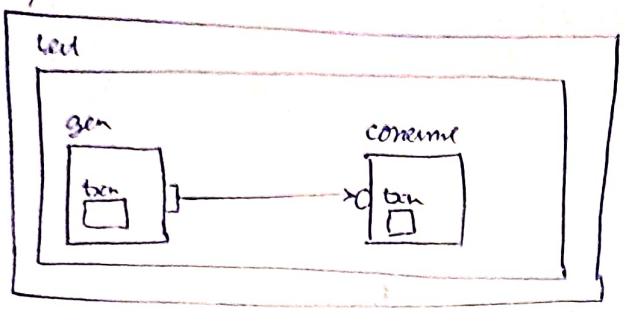




payload → ethernet payload.

↓

TLM2 can be used.

top.sv

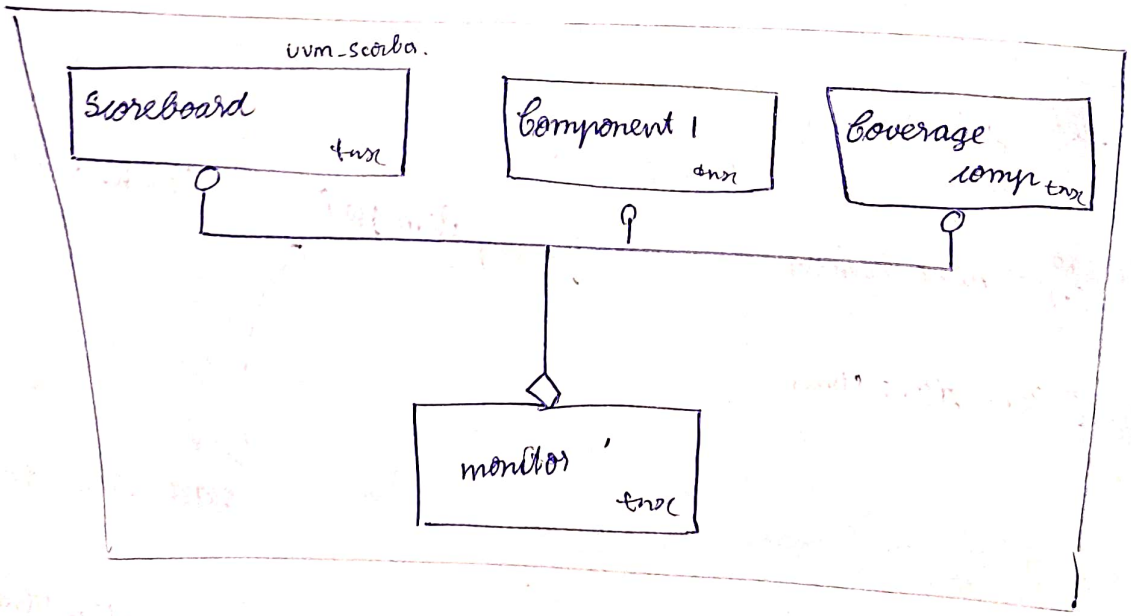



## Configure db vs TLM

| | |
|---|---|
| Static data type passed. ↴ memory persist till end of program | Dynamic data type passed. ↳ created & deleted default |

non-blocking [pull] → from consumer request to generator to generate seq.

⇒ get method in generator gets triggered.



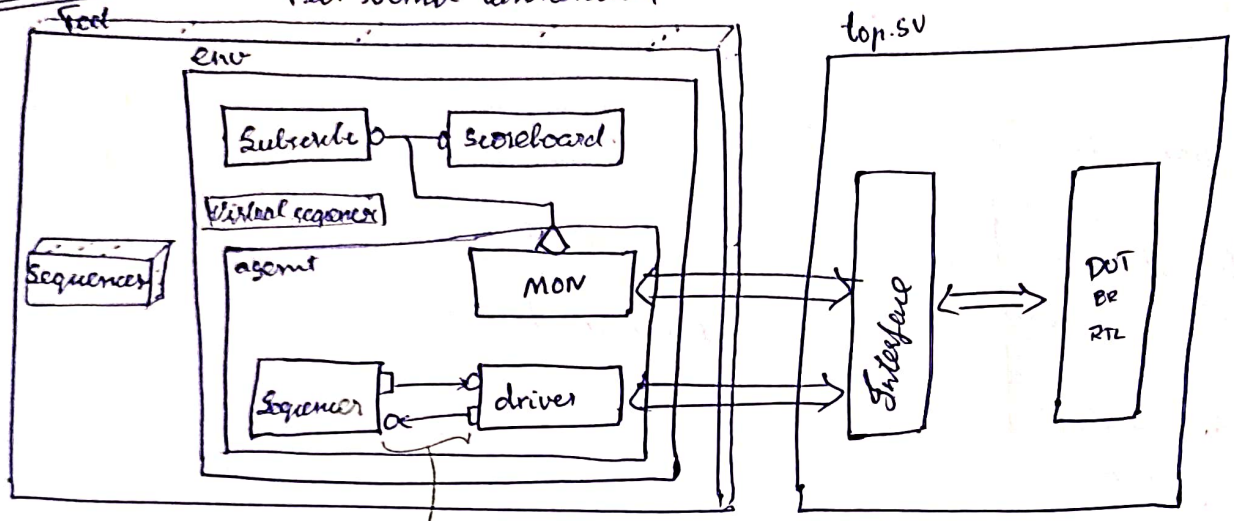
Line folding VS
ctr+K, ctr

.print (uvm_default_tree_printer);

-line- ⤳ change print format.

25-12-2024

Test bench architecture




[rsp - post
seq_item_post]

Implisit
exist

Sequence priority can be set to start first or later on parallel

fifo to be used to make it run in parallel

Sequencer & driver

→ refer git commit

Drivers

seq_item_port.get_next_item(t);

seq_item_port.item_done(t);

seq_item_post

seq_h.start(env_h.agent_h.seqrL)

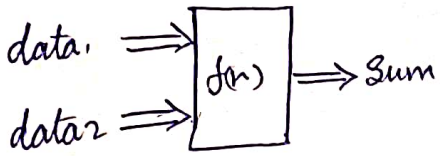
test
start_item(t);

finish_item(t);

30-12-2024

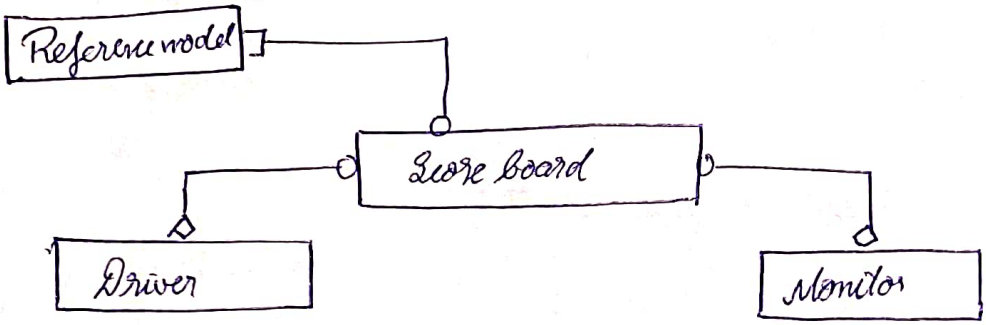
## Reference Model

→ of DUT for verification (extends from uvm_component)




fn() : data1 + data2 ⟶ Need not be synthesizable

↗ can be in SystemVerilog, C/C++, Python




if SystemVerilog used TLM ports not available ; need to use mailbox

## Transaction Flow

Driver ⟶ Sequencer ⟶ Sequence

```
get_next_item()
...
...
item_done()
write(rsp)
set_response()
```

```
start_item(item)
pre_do()
item.randomize()
mid_do()
finish_item(item)
post_do()
get_response()
  ↳ (optional)
```

## TLM FIFO

Producer → [ FIFO ] → Consumer.

SoftIP, HardIP

## Factory Overriding

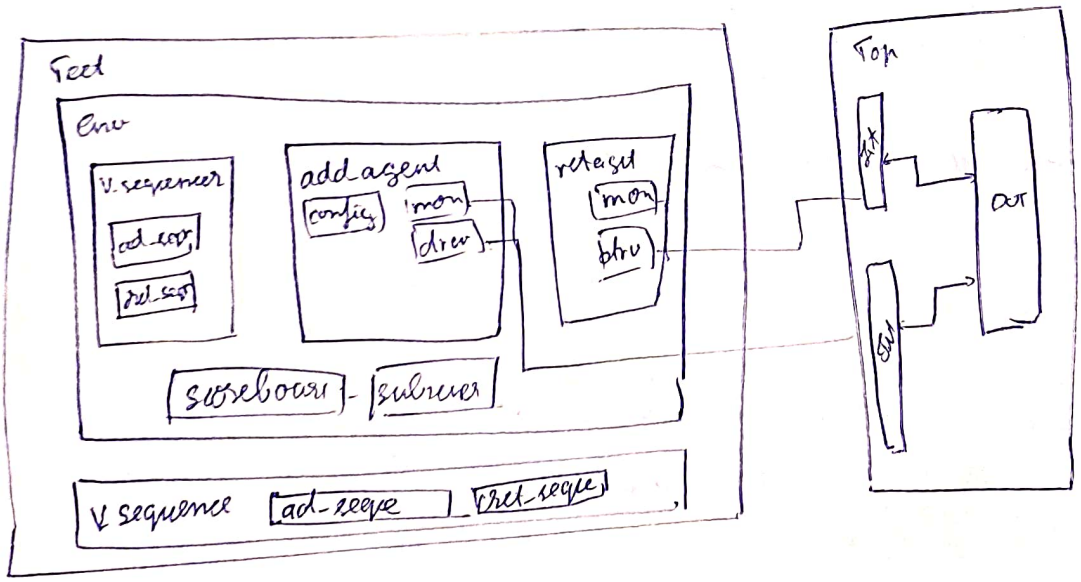
global override, → set_type_override_by_type
instanance override → set_inst_override_by_type } used to change behaviour of class by substitution without editing or recompli

("path", .

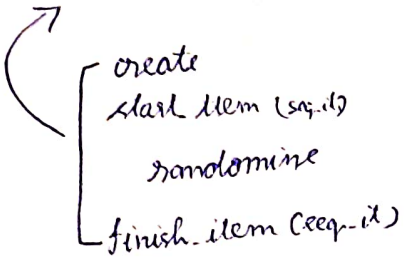
uvm_resource_db → unstructured configuc_db

## 05-01-2025

Virtual sequence, Virtual sequencer.



`uvm_do (seq_item)




`uvm_do_with ( )

`uvm_do_pri ( )

`uvm_do_pri_with ( )

`uvm_create ( )

`uvm_send ( )