

UVM → System Verilog - Base class library

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

OVM - Mentor & Cadence

→ Constrained random, coverage driven verification

Generation of test from test bench

- Transaction level communication, layered sequential stimulus, Standardized messaging, Register Layer (adv<sup>xx</sup>)

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

Functional coverage → to know (when to end)

☞ Header, payload, checksum.

Configurable & reuse of IP

Constraints modify to increase coverage.

sequence (set of trans)



transaction

pack

→ driver

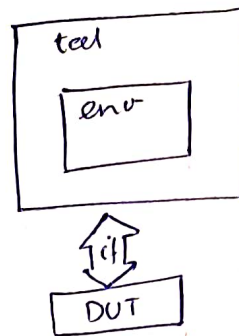
→ min level signal.

Fixed part - env

Variable part - test

Class based - test, env

Structural - if, dut



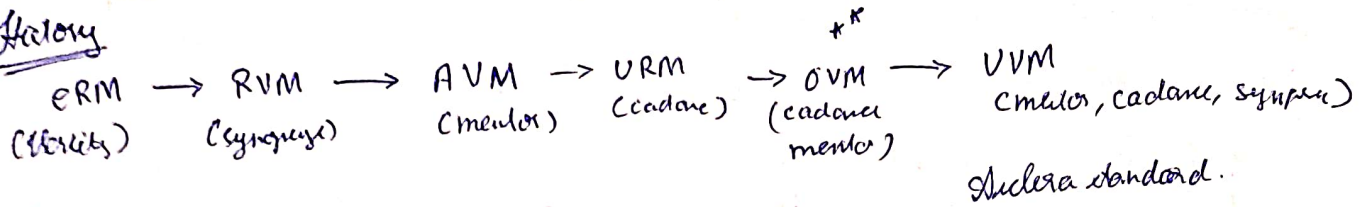
25-12-2024

## UVM

- Umesh Saw

- UVM automatically no build, compile & run test
- UVM architecture (sequence ~ gen)

### History



### UVM 1.2

UVM macro, UVM package

```
`include "uvm-macro.svh" import uvm_pkg::*;
```

```
`UVM_info(LID, <MSG>, <verbosity>)
```

→ Macro

```
`include " .svh"
```

```
$sformatf("Var1 is %0d", Var)
```

```
`define DATA 1
```

### UVM component

- type → queue 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

= null to handle

run\_test ("class") → starts test in UVM.

```
`include "uvm_macros.svh"
```

```
import uvm_pkg::*;
```

```
class my_component extends uvm_component;
```

```
  `uvm_component_util(Cmy_compnd); // 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);
```

```
  endfunction
```

```
endclass
```

```
module my_test
```

```
  initial begin
```

```
    run_test C"my_compnd"; // runtest
```

```
  end
```

```
endmodule
```

---

`uvm\_component\_util(Ces) → factory registration.

↳ cannot create component of type "my\_compnd" because  
it is not registered with the factory

---

```
run_test C"";
```

```
querilog <filename> -R +UVM-TESTNAME=my_component  
                                component_util name
```

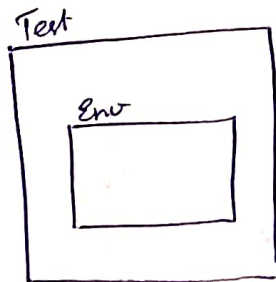
class variable cannot be added to wave → done through interface.

### build\_phase

```
function void build_phase (uvm_phase phase)
  super.build_phase (phase);
  'uvm.info C "": < >;
endfunction
```

### Nested component

env env\_h;



```
function build_phase (....);
```

```
...:
```

```
env_h = env::typeid::create("env_h", this);
```

```
endfunction
```

### order of phases

build, Topdown Tasks, are parallel  
final,

### UVM - reporting

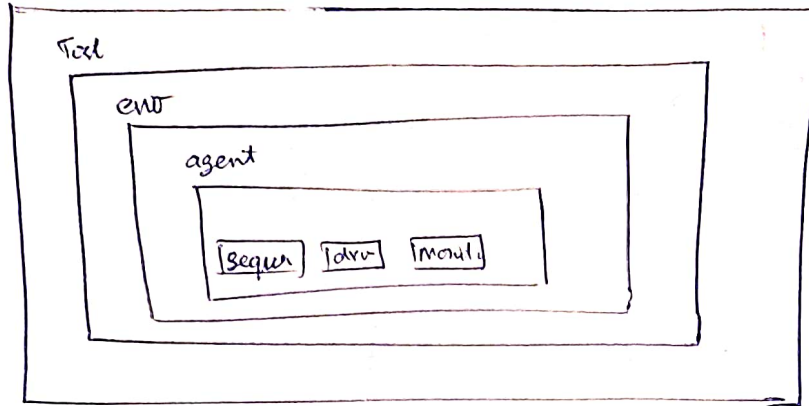
UVM - info, UVM - error, UVM - fatal, UVM - warning,  
(<string>), (<string>), (<enum>) → stops simulation & exits.

default ← {  
UVM - NONE - 0  
UVM - LOW - 100  
UVM - HIGH - 300  
UVM - MEDIUM - 200  
UVM - FULL - 400  
UVM - DEBUG - 500

+ UVM-VERBOSITY = <verbosity>

↳ UVM-NONE, UVM-DEBUG, ...

Top



uvm\_top.print\_topology();

uvm-test-top

env-h

agent-h

drv-h

mon-h

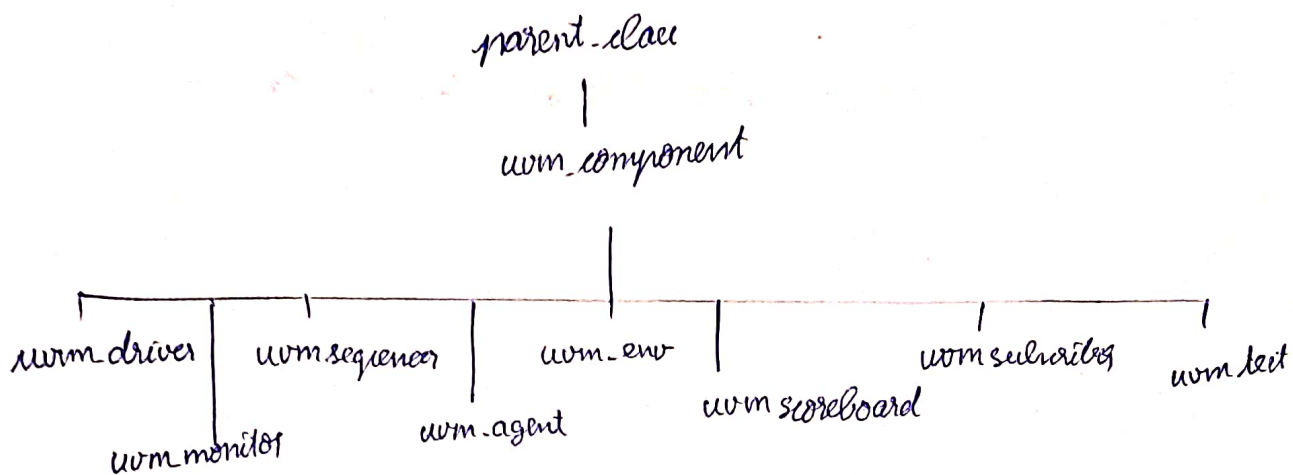
seq-h

get\_name() → obj name.

get\_full\_name() → obj name full with hierarchy

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 & retrieve data.

uvm\_config\_db #(type T) :: set(context, → null / this

"inst\_name", → 'key'

"field\_name", → i/o

value);

from → test to → top

raise exception,

#10

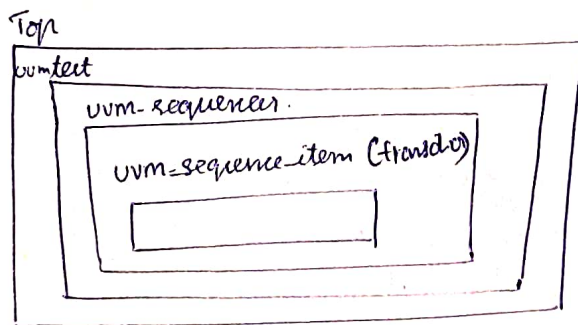
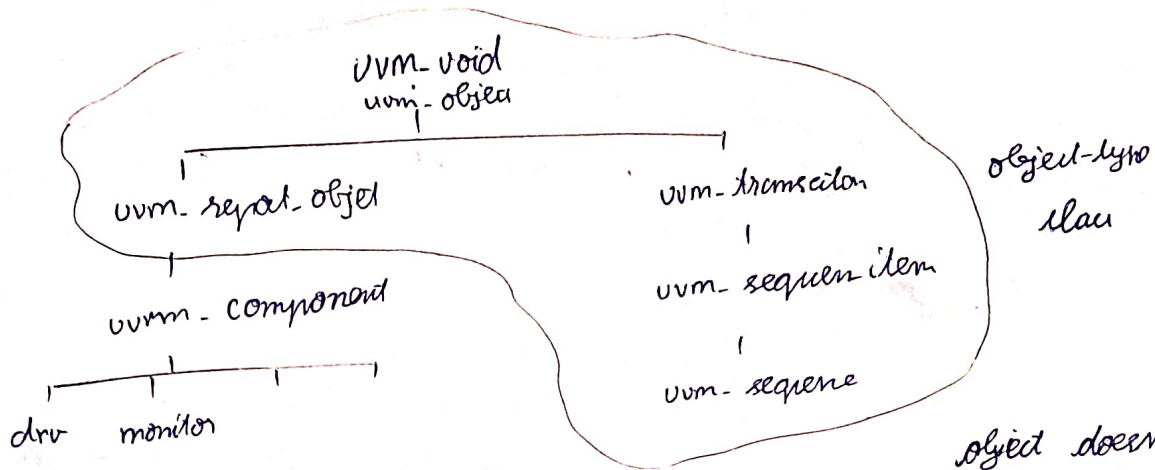
drop exception

#2

uvm\_config\_db #(type T) :: get (context, inst\_name, field\_name, value);

26.12.2024

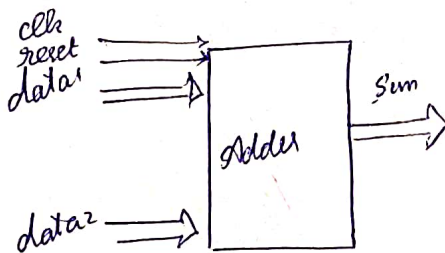
Using config db send "virtual interface" from top.sv to env.sv  
 ←



→ sequence item class.

copy,  
compare,  
print,  
get,  
create

record.  
make



→ ports to be declared in my-sequence-class

as bit

Factory registry → 'vvm-object-util (user-sequence-item) [to register class]

'vvm-object-util.begin(<object name>)  
 'uvm-field-int-<object name>  
 'vvm-object-util.end.

[to register object]



class user-seq-item extends uvm-sequence-item;

bit[30] data1, data2;  
 bit[4:0] sum;  
 'uvm-object util - begin()

'uvm-field-int (data1); → factory registry.

'uvm-field-int (data2);

'uvm-object util - end()

If not registered output won't be there  
 for .print()  
 & cannot be used inside any method

| UVM-DEC

- OCT

- HEX

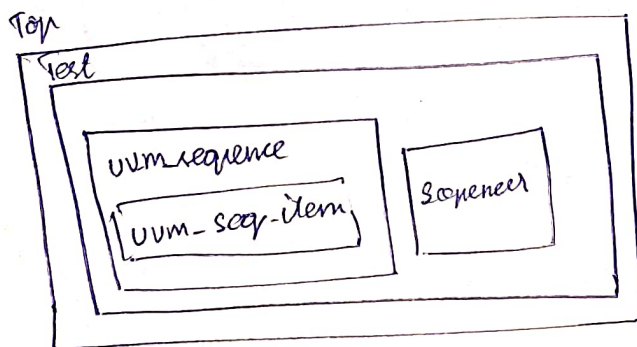
copy, compare, (clone)

uvm-sequence (<sup>generator</sup> ~~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-sequences, m-sequences.

typedef  $\Rightarrow$  forward declaration.

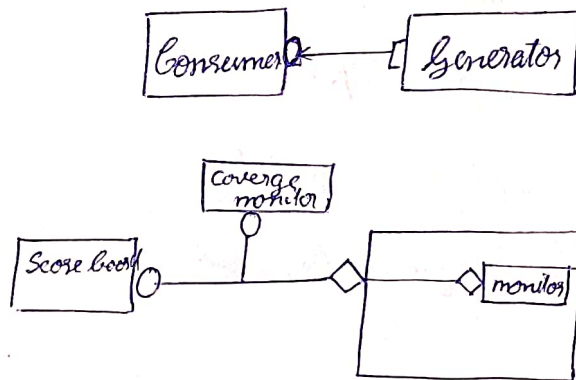
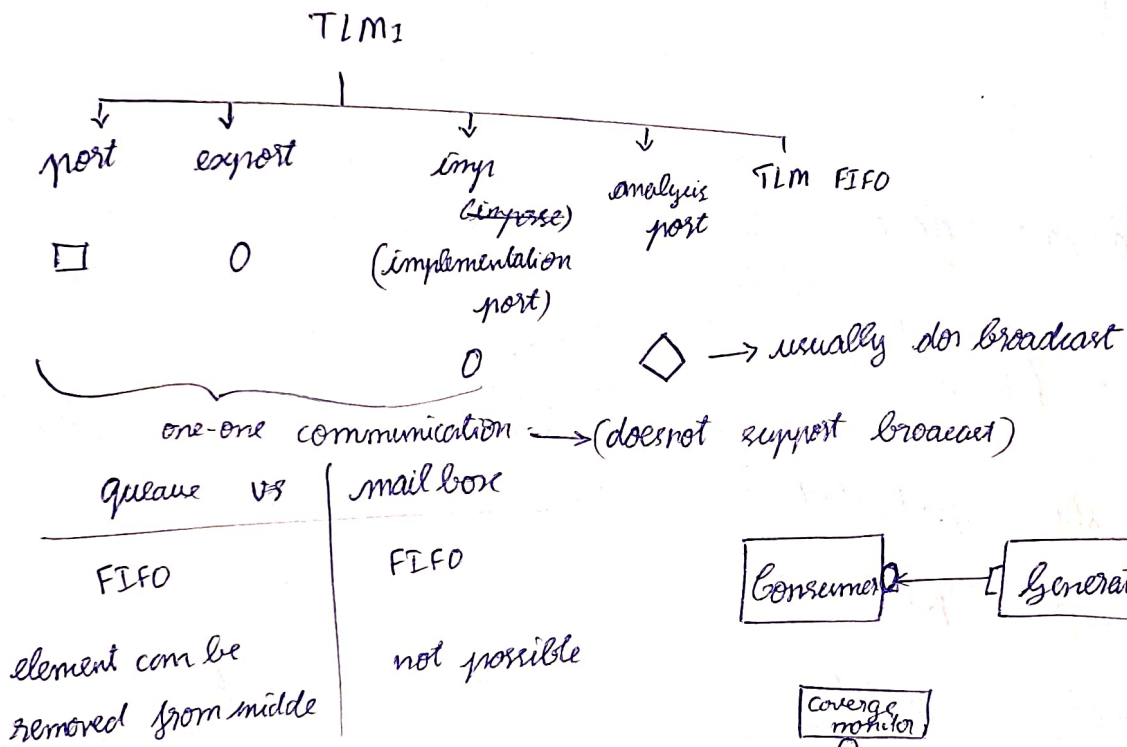
'vom declare p-sequences (use-sequences)

p-sequences variable defined.

27-12-2024

## Transaction Level Modeling

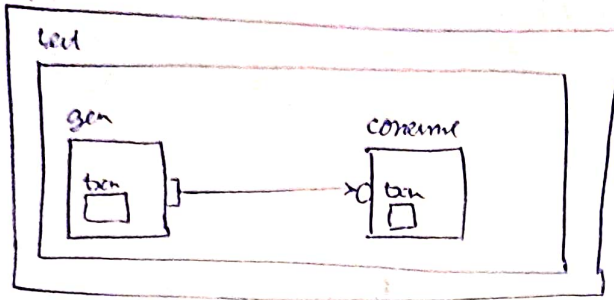
$\rightarrow$  Similar to tnx (mailbox)



payload  $\rightarrow$  ethernet payload.

$\downarrow$   
TLM<sub>2</sub> can be used.

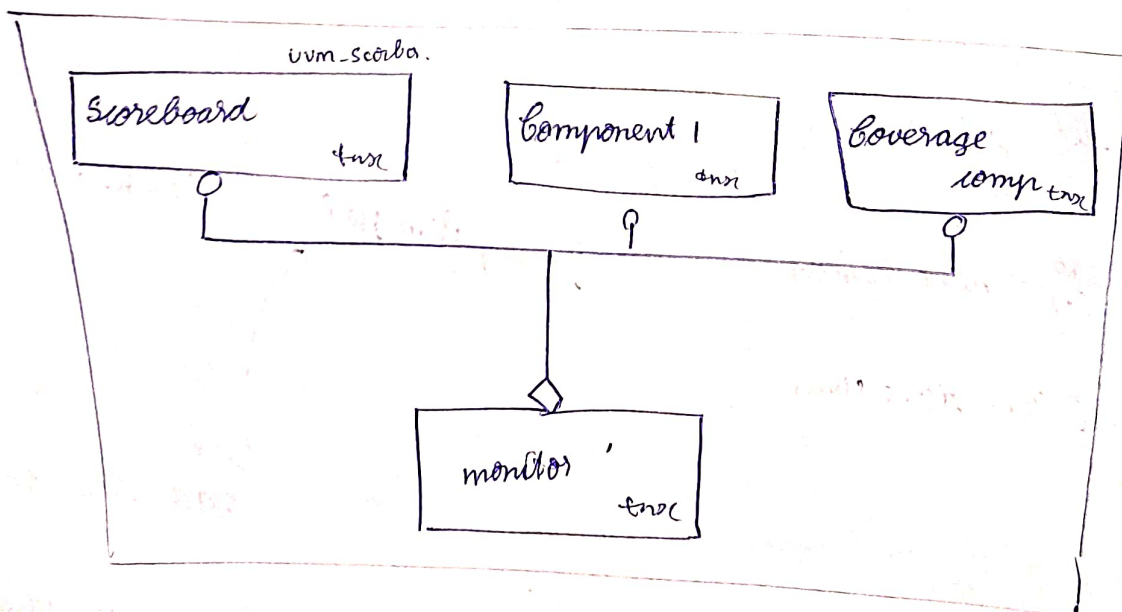
test.sv



Configure dp vs TLM

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

non-blocking null → from consumer request to generator to generate seq.  
⇒ get method in generator gets triggered.



Line folding VS  
Ctrl+K, ctrl

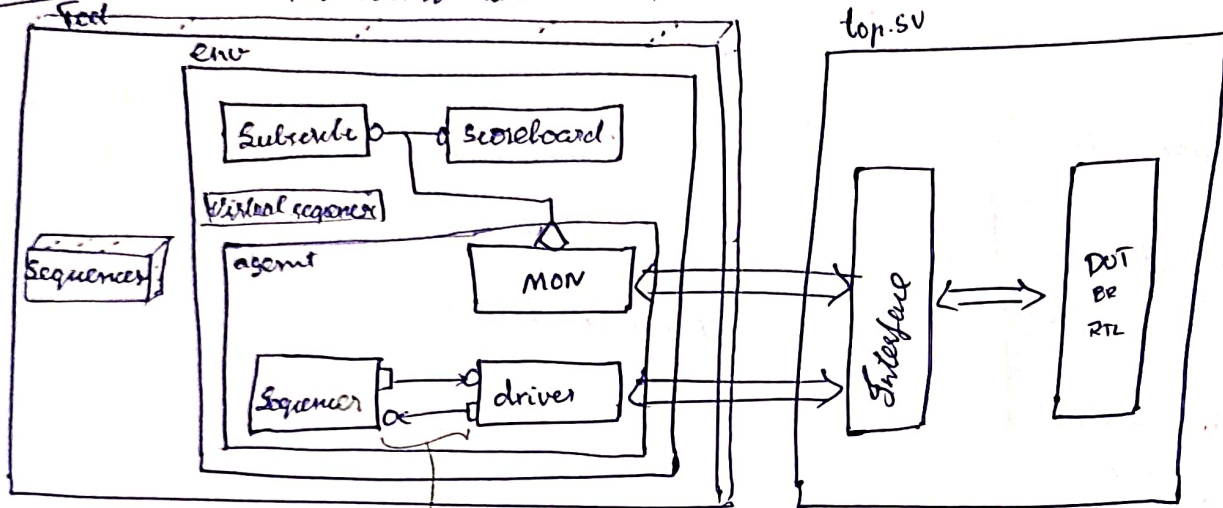
.print (vmm default tree-printer);

line

change print format.

25.12.2024

Test bench architecture



[resp. port  
seq item port]

Implicit  
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 not  
get. next item (b)

seq-item not item done (b)

Seq-item port

Seq-h. start (env-h agent  
.seqrl)

test  
start-item (b)

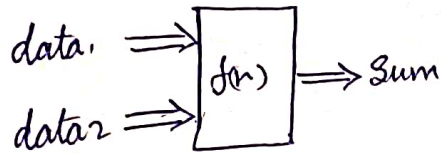
finish-item (b)

30-12-2024

## Reference Model,

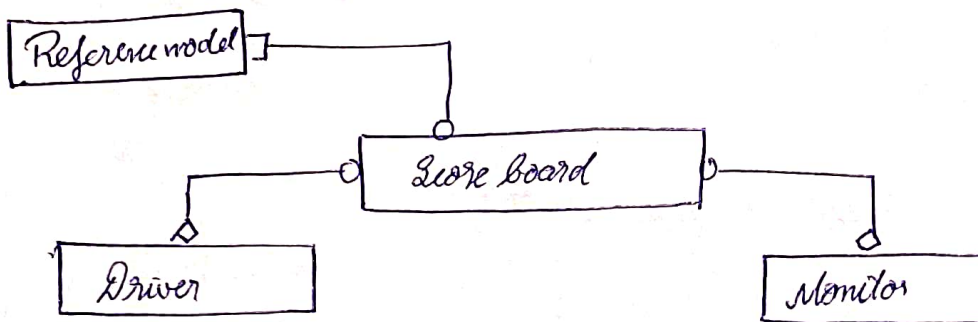
? of DUT for verification

(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\_crsps

set\_response()

start\_item(item)

pre\_do()

item.randomize()

mid\_do()

finish\_item(item)

post\_do()

get\_response()

→ (Optional)



## TLM FIFO

Producer → [FIFO] → Consumer.

Soft IP, Hard IP

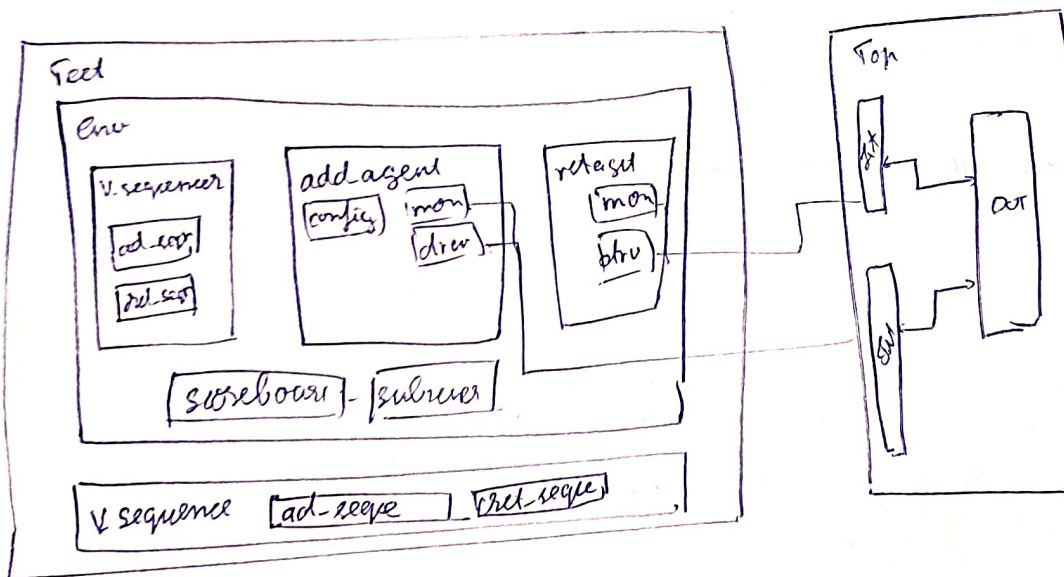
## Factory Overriding

global override, → set type override by type } used to change behaviour  
instance override → set inst override by type } of class by substitution  
('path', . without editing or recompiling

UVM-resource-db → unstructured config db

05-01-2025

Virtual sequence, Virtual sequencer.





'uvm-do (seq-item)



create

start-item (seq-it)

randomize

finish-item (seq-it)

'uvm-do-with ( )

'uvm-do-pri ( )

'uvm-do-pri-with ( )

'uvm-create ( )

'uvm-send ( )

11/01/2025

vvm-top. set error-limit (noc)

→ Directory Structure

Design,

Sim, UVM - (layer hierarchy)

Run,

→ Run

g qverilog +acc=ncr -R +UVM-INFO=

<simargs>

change logfile.

+UVM-TEST =

coverage,

+incdir+../design/

config tb

+incdir+../Sim/UVM/.

→ handle h

P-sequences, m-sequences

config tb

Basic UVM structure

uvm-driver

mon

seq

seq\_r

seq\_item

port connection → analysis port, export

(inbuilt port for subscriber)

Draw UVM structure ppt