DAY-92
#100DAYSRTL

"UVM: UVM Base Class"

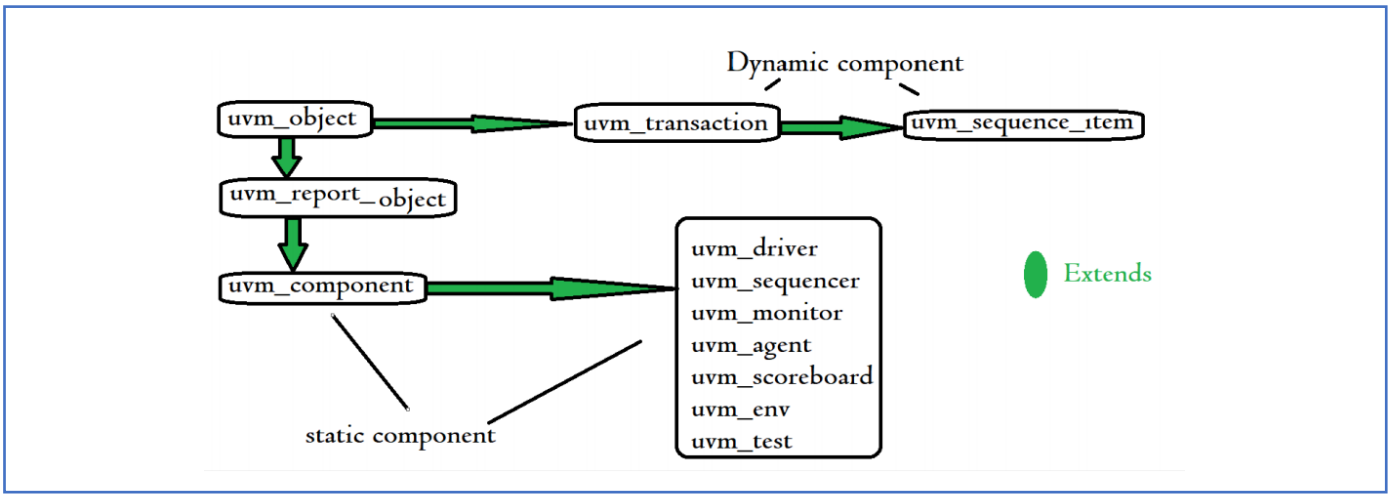
- Let's discuss the importance of base class with the code

```
class add;

  bit [3:0] a;
  bit [3:0] b;
  bit [4:0] c;
  function new(bit [3:0] a, bit [3:0] b);
    this.a = a;
    this.b = b;
    this.c = a + b;
  endfunction
endclass
class mul extends add;
  bit [3:0] p;
  bit [3:0] q;
  bit [7:0] r;
  function new(bit [3:0] p, bit [3:0] q);
    super.new(p,q);
    this.r = p * q;
  endfunction
endclass
module tb;
  mul t;
  int a, m;
  initial begin
    t = new(4'b0001, 4'b0010);
    a = t.c;
    m = t.r;
    $display("add : %0d and mul : %0d",a,m);
  end
endmodule
```

- You can access the parent class properties using the child class.
- In UVM we represent the Generator as a sequence and Sequencer sends data from the sequence to the driver
- Base Classes: UVM Components & UVM Object:-
  - ➢ Static components:- Which stay in the verification environment till the process done is called static components
    - ✓ Eg:-Scoreboard, Driver, monitor, Sequencer
  - ➢ Dynamic components:-These varies in the verification environment
    - ✓ Eg:- Transaction
- Static components are built using UVM Components

- Dynamic components are built using UVM object
- Building the constructor is different for uvm_object and uvm_component
- Let's check the default constructor for the base classes
- For uvm_object:-
  - ✓ function new(input string name=”chinnu”);
    super.new(name)
- For uvm_component:-
  - ✓ function new(string name = “Chinnu”,uvm_component parent=null);
    super(name,parent)//This parent helps us in the uvm_tress



- **Building a class using uvm_object:-**

```
`include "uvm_macros.svh"
import uvm_pkg::*;
class obj extends uvm_object;
  `uvm_object_utils(obj)
  function new(string inst="obj");
    super.new(inst);
  endfunction
endclass
module tb;
  obj o;
  initial begin
  o=new("O");
  end
endmodule
```

- **Building a class using uvm_component:-**

```
`include "uvm_macros.svh"
import uvm_pkg::*;
class comp extends uvm_component;
  `uvm_component_utils(comp) // To register our class to factory
  function new(string inst="comp",uvm_component parent=null);
    super.new(inst,parent);
  endfunction
endclass
module tb;
  comp c;
  initial begin
    c=new("c",null);
  end
endmodule
```