DAY-70
#100DAYSRTL

“System Verilog:-Class & Objects”

“Introduction”:-

System Verilog introduces class and object-oriented programming paradigms to hardware description languages, facilitating modular design, encapsulation, and reusability in hardware development. Classes define structures, behaviors, and hierarchies, while objects instantiate these classes, enabling robust and efficient design methodologies in System Verilog.

“Class”:-

- Class is a blueprint that defines the properties and behavior of an object.
- The class encapsulates properties and methods in a single entity. In System Verilog, all classes are dynamic in nature by default

```
class VLSI;
  //Body of class
endclass
```

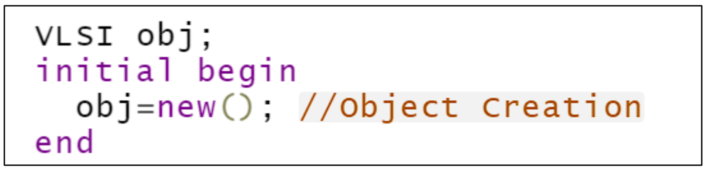
- A class will define all the properties and methods that will be present in the object

  Eg:-

```
class VLSI;
  int Size_Of_Transistor;//5nm,7nm
  string Techonology_Used;//CMOS,NMOS,PMOS
  function synthesis();
    //Performing synthesis
  endfunction
endclass
```

“**Object**”:-

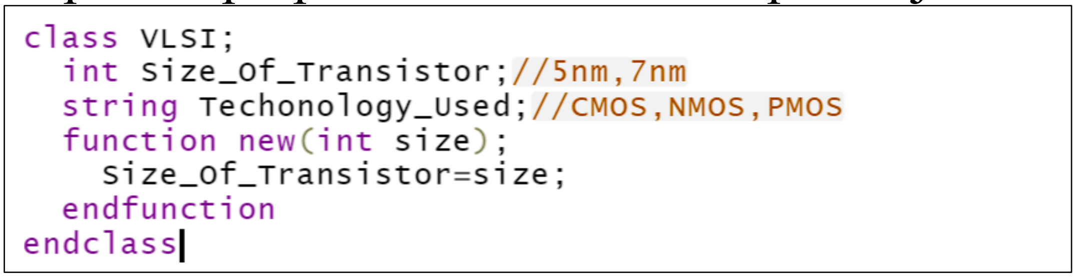
- Objects are a unique entity of the class.
- They are created dynamically whenever needed and destroyed when their need is over



```
VLSI obj;
initial begin
  obj=new(); //Object Creation
end
```

“**Constructors**”:-

- Constructors are in-built methods that are called to create a new object
- Basically, when we use new() to create a new class this acts as a constructor.
- In contrast to a default constructor, a parameterized constructor allows specific properties to be defined upon object creation.



```
class VLSI;
  int Size_Of_Transistor;//5nm,7nm
  string Techonology_Used;//CMOS,NMOS,PMOS
  function new(int size);
    Size_Of_Transistor=size;
  endfunction
endclass
```

- **Note:-**Constructors are special methods used to create objects so unlike a normal method, constructors don’t have a return type. In the example mentioned above please note there is no return type for function new**.**

“**Destructors**”:-

- These are just the opposite of constructors and are called when we want to destroy an object.
- In SV, we can’t manually destroy objects and thus destructors are not present in System Verilog

“**this keyword**”:-

this keyword represents the current object. This can be used to specifically point to properties that belong to the current object.

```
class VLSI;
  int Size_Of_Transistor;//5nm,7nm
  string Techonology_Used;//CMOS,NMOS,PMOS
  function new(int size);
    this.Size_Of_Transistor=size;
  endfunction
endclass
```

Code Practising:-

```
class packet;
    bit [4:0] length;
    bit [15:0] addr;
    function new(bit [15:0] addr, bit [4:0] l=1);
        this.length = l;
        this.addr = addr;
    endfunction: new
    function void print_packet();
        $display("addr = 0x%0h", addr);
        $display("lenght = %0d", length);
    endfunction: print_packet
endclass: packet
module class_ex_1;
    initial begin
        packet pkt1, pkt2;
        pkt1 = new('ha4a4,10);
        pkt1.print_packet();
        pkt2 = new('hb623);
        pkt2.length = 22;
        pkt2.print_packet();
    end
endmodule
```

Result:-

```
addr = 0xa4a4
lenght = 10
addr = 0xb623
lenght = 22
Simulation has finished. There are no more test vectors to simulate.
```

“**Automatic**”:-

When a method is automatic in nature these variables won’t be allocated memory when we create an object but rather when the methods are called. Once the method call is over the memory is released.

Code Practising:-

```
module test_1;
    initial begin
       $display("Without Automatic ");
        for (int i = 0; i<3; i++) begin
            fork
                $display("i = %0d", i);
            join_none
        end
    end
endmodule
module test_2;
    initial begin
       $display("With Automatic ");
        for (int i = 0; i<3; i++) begin
            fork
                automatic int k = i;
                $display("k = %0d", k);
            join_none
        end
    end
endmodule
```

Result:-

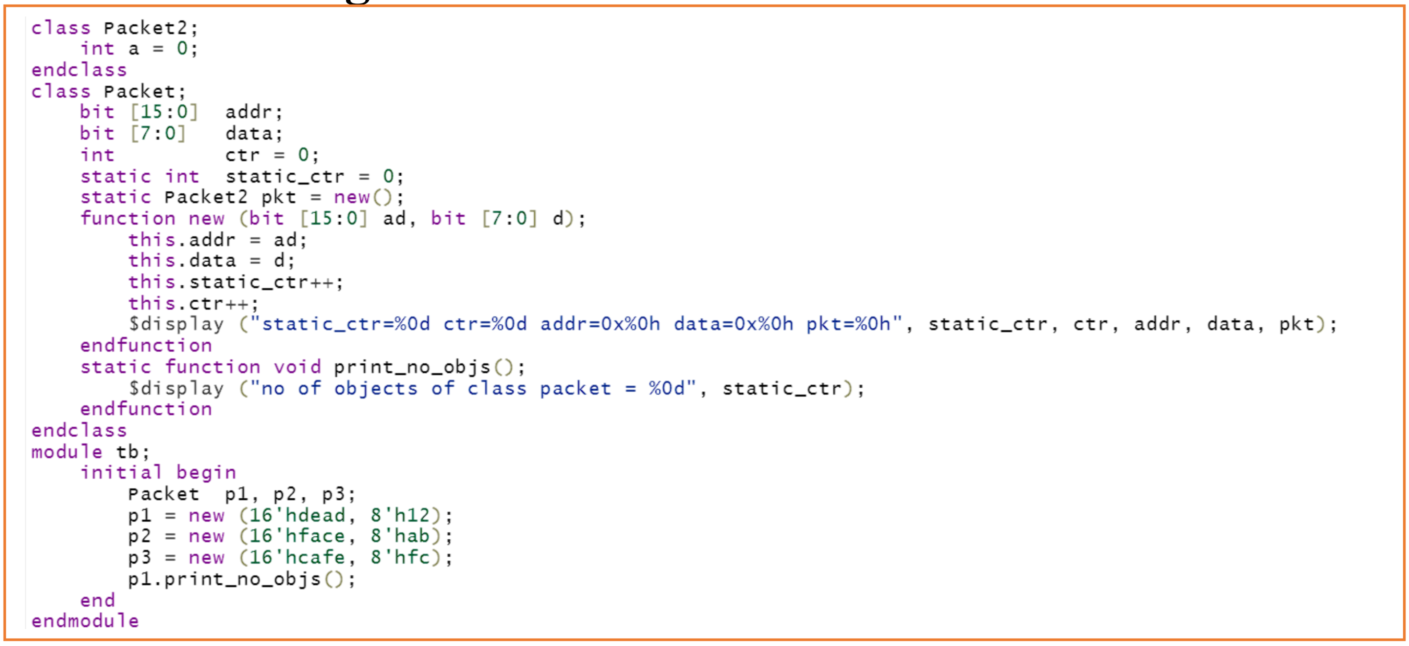
```
Without Automatic
i = 3
i = 3
i = 3
With Automatic
k = 2
k = 1
k = 0
Simulation has finished. There are no more test vectors to simulate.
```

If we have a fork_join inside a for loop and we are using the loop variable inside fork_join then changes made by any one of the processes to that loop variable will be reflected on all other processes. But if we declare that variable as automatic, then all processes will have individual memory space for that variable, and thus, changes in the variable won't be reflected in other processes.

“<u>Static</u>”:-

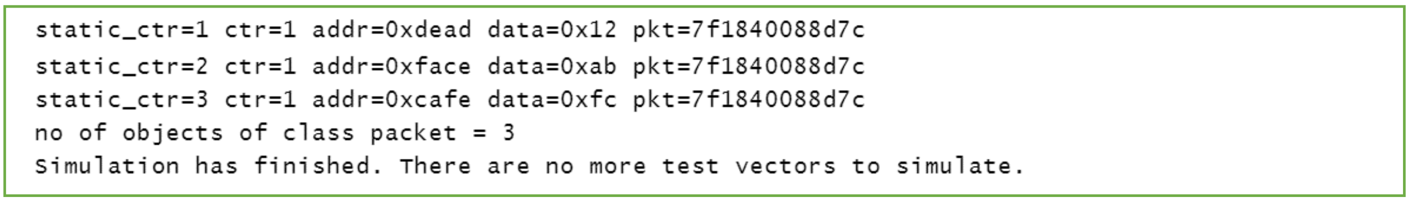
- Static keyword is used to make a method or class static in nature.
- Static methods would share internal variables between different method calls. Thus, if we set some internal value of the method to some value, it will be reflected on all other method calls.
- Similarly, we can use static keyword for class as well. This makes class of static nature and thus can be used without making objects

Code Practising:-



```
class Packet2;
    int a = 0;
endclass
class Packet;
    bit [15:0]  addr;
    bit [7:0]   data;
    int         ctr = 0;
    static int  static_ctr = 0;
    static Packet2 pkt = new();
    function new (bit [15:0] ad, bit [7:0] d);
        this.addr = ad;
        this.data = d;
        this.static_ctr++;
        this.ctr++;
        $display ("static_ctr=%0d ctr=%0d addr=0x%0h data=0x%0h pkt=%0h", static_ctr, ctr, addr, data, pkt);
    endfunction
    static function void print_no_objs();
        $display ("no of objects of class packet = %0d", static_ctr);
    endfunction
endclass
module tb;
    initial begin
        Packet  p1, p2, p3;
        p1 = new (16'hdead, 8'h12);
        p2 = new (16'hface, 8'hab);
        p3 = new (16'hcafe, 8'hfc);
        p1.print_no_objs();
    end
endmodule
```

Result:-



```
static_ctr=1 ctr=1 addr=0xdead data=0x12 pkt=7f1840088d7c
static_ctr=2 ctr=1 addr=0xface data=0xab pkt=7f1840088d7c
static_ctr=3 ctr=1 addr=0xcafe data=0xfc pkt=7f1840088d7c
no of objects of class packet = 3
Simulation has finished. There are no more test vectors to simulate.
```

An object of another class Packet2 inside class Packet. The object of class Packet2 is static in nature, thus it can be observed that the object handle pkt in Packet class points to the same address and no new object is created for each new object of Packet.