DAY-72
#100DAYSRTL

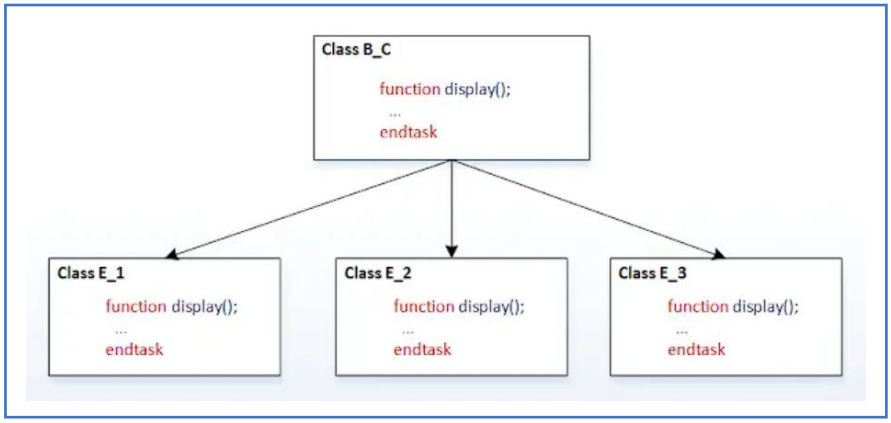
“System Verilog:-Polymorphism”

“Introduction”:-

In System Verilog, polymorphism is primarily achieved through **function overriding**, enabling varied function definitions in child classes, known as **static polymorphism**. Parameterized classes also offer versatility by allowing different class behaviors based on parameters. However, **function overloading**, a means of polymorphism, **isn't supported** in the latest System Verilog version (IEEE 1800-2008), restricting **dynamic polymorphism**. This language leans towards static polymorphism through overriding and parameterization, steering away from runtime-defined function behaviors, and emphasizing compile-time decision-making for function execution.

“Polymorphism”:-

- Polymorphism means **many forms**. Polymorphism in System Verilog provides the ability for an object to take on many forms.
- Method handle of super-class can be made to refer to the subclass method, this allows polymorphism or different forms of the same method.



“Virtual”:-

- In SV, to override a function present in parent class, we need to use virtual keyword in the function declaration present in parent class.
- **Note:-**It is not necessary for the child class to have the virtual keyword, but if we intend to create a child class from this class then we must use virtual keyword to enable function overriding.

“Code Practising”:-

```
class DevicePacket;
    bit [31:0] data;
    bit [9:0] addr;
    int device_version;
    function new();
        device_version = 1;
    endfunction //new()
    function void printInfo();
        $display("Device version 1 packet");
        $display("device_version: %0d", device_version);
        $display("addr: 0x%0h", addr);
        $display("data: %0d", data);
    endfunction
    virtual function int sendData(int data);
        this.data = data;
        $display("Sending device packet");
        return data;
    endfunction
endclass: DevicePacket
class Device2Packet extends DevicePacket;
    function new();
        data = 0;
        device_version = 2;
    endfunction //new()
    function void printInfo();
        $display("Device version 2 packet");
        $display("addr: 0x%0h", addr);
        $display("data: %0d", data);
    endfunction

    function int sendData(int data);
        this.data = data;
        $display("Sending device version 2 packet");
        return this.data;
    endfunction
endclass: Device2Packet
module test;
    DevicePacket devicePacket;
    Device2Packet device2Packet;
    initial begin
        device2Packet = new();
        devicePacket = device2Packet;
        devicePacket.printInfo();
        devicePacket.sendData(45);
    end
endmodule
```

“Result”:-

```
Device version 1 packet
device_version: 2
addr: 0x0
data: 0
Sending device version 2 packet
Simulation has finished. There are no more test vectors to simulate.
```

“Code practising”:-

```
// base class
class base_class;
  virtual function void display();
    $display("Inside base class");
  endfunction
endclass
// extended class 1
class ext_class_1 extends base_class;
  function void display();
    $display("Inside extended class 1");
  endfunction
endclass
// extended class 2
class ext_class_2 extends base_class;
  function void display();
    $display("Inside extended class 2");
  endfunction
endclass
// extended class 3
class ext_class_3 extends base_class;
  function void display();
    $display("Inside extended class 3");
  endfunction
endclass
// module
module class_polymorphism;
  initial begin
    //declare and create extended class
    ext_class_1 ec_1 = new();
    ext_class_2 ec_2 = new();
    ext_class_3 ec_3 = new();
    //base class handle
    base_class b_c[3];
    //assigning extended class to base class
    b_c[0] = ec_1;
    b_c[1] = ec_2;
    b_c[2] = ec_3;
    //accessing extended class methods using base class handle
    b_c[0].display();
    b_c[1].display();
    b_c[2].display();
  end
endmodule
```

“Result”:-

```
Inside extended class 1
Inside extended class 2
Inside extended class 3
Simulation has finished. There are no more test vectors to simulate.
```