DAY-74
#100DAYSRTL

"System Verilog:-Encapsulation"

"Introduction":-

Encapsulation is a programming practice that involves the **concealment of data** within a class, making it exclusively accessible through the class's methods. This technique effectively safeguards the data (along with internal methods) within the "**capsule**" of the class, thereby **limiting access** solely to authorized users, typically the methods of the class.

"Encapsulation":-

Frequently, we utilize Base Classes or Base Class libraries offered by third-party sources. Typically, the Class Members in these libraries are inherently public, allowing direct access from outside the Class. However, there are instances where the providers of these Base Classes might **impose restrictions** on how external entities can access the Class members. This is implemented as a **safety and security measure** to prevent unauthorized alterations to internal states and logic. These restrictions are put in place to fortify the integrity of the Class and uphold its intended functionality, promoting a more controlled and secure environment for the users and maintainers of the code.

Class Members can be declared with the following two Qualifiers:

- **Local:** It is available only to methods inside the class. Further, these local members are not visible within subclasses.
- **Protected:** It is a property or method that has all of the characteristics of a local member, except that it can be inherited; it is visible to subclasses.

“Local”:-

- External access to the class members can be avoided by declaring members as local, Within the class is allowed
- Any violation could result in a compilation error.

“Code Practising”:-

```
class parent_class;
  local bit [31:0] tmp_addr;
  function new(bit [31:0] r_addr);
    tmp_addr = r_addr + 10;
  endfunction
  function display();
    $display("tmp_addr = %0d",tmp_addr);
  endfunction
endclass
module encapsulation;
  initial begin
    parent_class p_c = new(5);
    p_c.tmp_addr = 20;  //Accessing local variable outside the class
    p_c.display();
  end
endmodule
```

“Result”:-

```
ERROR VCP5248 "Cannot access local/protected member ""p_c.tmp_addr"" from this scope." "testbench.sv" 13  18
WARNING VCP2803 "Function p_c.display result is ignored." "testbench.sv" 14  18
FAILURE "Compile failure 1 Errors 3 Warnings  Analysis time: 0[s]."
```

“Code Practising”:-

```
class parent_class;
  local bit [31:0] tmp_addr;
  function new(bit [31:0] r_addr);
    tmp_addr = r_addr + 10; //Accessing local variable inside the class
  endfunction
  function display();
    $display("tmp_addr = %0d",tmp_addr);
  endfunction
endclass
module encapsulation;
  initial begin
    parent_class p_c = new(5);
    p_c.display();
  end
endmodule
```

“Result”:-

```
tmp_addr = 15
Simulation has finished. There are no more test vectors to simulate.
```

“Protected”:-

- In some use cases, it is required to access the class members only by the derived class’s, this can be done by prefixing the class members with the protected keyword.
- Any violation could result in a compilation error.

“Code Practising”:-

```
class parent_class;
  protected bit [31:0] tmp_addr;
  function new(bit [31:0] r_addr);
    tmp_addr = r_addr + 10;
  endfunction
  function display();
    $display("tmp_addr = %0d",tmp_addr);
  endfunction
endclass
class child_class extends parent_class;
  function new(bit [31:0] r_addr);
    super.new(r_addr);
  endfunction
  function void incr_addr();
    tmp_addr++;
  endfunction
endclass
module encapsulation;
  initial begin
    parent_class p_c = new(5);
    child_class  c_c = new(10);
    // variable declared as protected cannot be accessed outside the class
    p_c.tmp_addr = 10;
    p_c.display();s
    c_c.incr_addr();  //Accessing protected variable in extended class
    c_c.display();
  end
endmodule
```

“Result”:-

```
ERROR VCP5248 "Cannot access local/protected member ""p_c.tmp_addr"" from this scope." "testbench.sv" 23  18
WARNING VCP2803 "Function p_c.display result is ignored." "testbench.sv" 24  18
ERROR VCP2000 "Syntax error. Unexpected token: c_c[_IDENTIFIER]." "testbench.sv" 25  5
WARNING VCP2803 "Function c_c.display result is ignored." "testbench.sv" 26  18
FAILURE "Compile failure 2 Errors 5 Warnings  Analysis time: 0[s]."
```

“Code Practising”:-

```
class parent_class;
  protected bit [31:0] tmp_addr;
  function new(bit [31:0] r_addr);
    tmp_addr = r_addr + 10;
  endfunction
  function display();
    $display("tmp_addr = %0d",tmp_addr);
  endfunction
endclass
class child_class extends parent_class;
  function new(bit [31:0] r_addr);
    super.new(r_addr);
  endfunction
  function void incr_addr();
    tmp_addr++;
  endfunction
endclass
module encapsulation;
  initial begin
    child_class  c_c = new(10);
    c_c.incr_addr();  //Accesing protected variable in extended class
    c_c.display();
  end
endmodule
```

“Result”:-

```
tmp_addr = 21
Simulation has finished. There are no more test vectors to simulate.
```