SV TASK AND FUNCTION

Presented By:
Neeli Anu

CONTENTS

1. WHAT IS TASK AND FUNCTION
2. TYPES OF TASKS WITH EXAMPLES
   - i . Static Tasks
   - ii . Automatic Tasks
3. TYPES OF FUNCTIONS AND EXAMPLES
   - i . Static Functions
   - ii . Automatic Functions
   - iii .Pass by position and name-function
   - iv . Pass by reference function
   - v . Pass by value-function
4. DIFFERENCES BETWEEN TASK AND FUNCTION

TASK:

- In System Verilog, A Task is a block of code that can do multiple things and can be called whenever needed.
- Task is used when we want to perform a set of actions that may take time (like delay or @event).

FEATURES OF TASK:

- Timing Control
- Accept Inputs
- Give Outputs
- Reusable Code Block
- Can Call Other Task and Functions

Code Example:

```
task display_values(input int a, input int b);
  #5; // wait  for 5 time units
  $display("Values are: %0d and %0d", a, b);
endtask
```

Output:

time 5: Values are a and b

Types of Tasks:

1 . Static Task:

- These are the **default** type of tasks.
- Declared without **Automatic** keyword

Example:

```
module static_task_demo;
task say_hello; // Task definition (static by default)
    $display("Hello from static task at time %0t", $time);
  endtask
initial begin  // Initial block to call the task
    say_hello();  // First call
    #5;
    say_hello();  // Second call after 5 time units
  end
endmodule
```

Output : Hello from static task at time 0
Hello from static task at time 5

2. Automatic Tasks:

- Every time the task is called, **a new copy** of its variables is created.
- Useful in **testbenches** and concurrent environments.
- Declared with the **Automatic** keyword.

Example:

```
module auto_task_demo;
 task automatic double_value(input int num); // Automatic task with a local variable
   int result;
   result = num * 2;
   $display("Time: %0t , Input: %0d , Result: %0d", $time, num, result);
 endtask
 initial begin // Call the task
   double_value(3);
   double_value(7);
 end
endmodule
```

Output: Time: 0, Input: 3, Result: 6

Time: 0 ,Input: 7, Result: 14

FUNCTION:

- Function is a block of reusable code that takes input and performs computation and returns a single value
- For example function is like a calculator, when you give the numbers it instantly give you the answer.

FEATURES OF FUNCTION:

- Returns a value
- Executes instantly(No time delay)
- Reusable Code
- It can only have inputs(no outputs and inouts are allowed)
- It can be declared inside/outside modules/classes
- Function can be written in modules , interfaces, packages, classes.

Code Example:

```
module function_example;
int result; // Declare a variable to hold the result
// Function definition
  function int square(input int num);
    square = num * num;
  endfunction
 // Initial block to call the function
  initial begin
    result = square(4); // Call the function with input 4
    $display("Square of 4 is: %0d", result);
   $finish;
  end
endmodule
```

Output: Square of 4 is: 16

TYPES OF FUNCTIONS:

1 . Static Function:

- **A static function belongs to the class itself, not to any object.**
  You can call it without creating an object.
- **It can only access static variables or other static functions of the class.**
  It cannot use non-static (object-specific) data.

Example:

```
class Calc;
  static function int add(int a, int b);
    return a + b;
  endfunction
endclass

module test;
  initial $display("Sum = %0d", Calc::add(5, 3)); // Output: Sum = 8
endmodule
```

2 . Automatic Function:

- **Each call gets its own copy of variables**
  This means values don’t get mixed up when the function is called multiple times or recursively.
- **Supports recursion and parallel execution**
  Automatic functions can safely call themselves (recursive) or run in parallel blocks.

Example:

```
module test;
 // Automatic function to double a number
  function automatic int double(int x);
    return x * 2;
  endfunction

  initial begin
    $display("Result = %0d", double(5));  // Output: Result = 10
  end
endmodule
```

3. Pass by position and name –function

Example:

```
module tb;
  int x, y;
  int z;

  function void fn(string name, int value);
    $display("%s, %0d", name, value);
  endfunction

  initial begin
    fn(.value(10), .name("sv"));
  end
Endmodule
```

Output: sv, 10

4. pass by reference-function

```
module tb;
  int a, b;
  int out;
  function automatic int mul( ref int a, b);
    // function automatic int mul(const ref int a, b);//to make sure value not updated
    a = a*b;
    return a;
     $display("out = %0d, a = %0d, b = %0d", out, a, b);
  endfunction
  initial begin
    a = 5;
    b = 5;
    out = mul(a,b);
    $display("out = %0d, a = %0d, b = %0d", out, a, b);
  end
endmodule
```

Output: out = 25, a = 25, b = 5

5. pass by value-function:

```
module tb;
  int a, b;
  int out;
  function int mul(int a = 5, b = 5);
    a = a*b; //a became updated here but global "a" wont update
    return a;
  endfunction
  initial begin
    out = mul(); //No arguments are passed
    $display("out = %0d, a = %0d,b = %0d", out, a, b); //a and b values will be 0 since their values are not
updated
     a = 2;
    b = 2;
    out = mul(a, b);
    $display("out = %0d,a = %0d,b = %0d", out, a, b);
  endmodule
```

output :out = 4, a = 2, b = 2

DIFFERENCES BETWEEN TASK AND FUNCTION:

FUNCTION	TASK
A Function can enable another function but not another task	A Task can enable another task and function
Function always executes in zero Simulation time	Task always executes in non-zero simulation time
Function must not contain any delay, event and timing control statements	Task may contain delay, event and timing control statements
Functions must have at least one input argument. They can have more than one input argument.	Task may have zero or more arguments of type input, output or inout.

THANK YOU!