# Module #11 : Verilog HDL Tasks

**11.1: Verilog HDL Tasks (Not Synthesizable):**

A task is similar to a function, but:

- A function runs in “0” time, a task can have time control statements , @, #, wait.
- A task does not return values in-line like a function, but it can have output as well as input ports. Thus it can send back multiple values. These ports are declared using input and output.
- A function cannot invoke a task, but a task can invoke functions or other tasks.
- A function must have at least one input argument. A task can have zero.

***Syntax:***

***task task_name;***
*input [msb:lsb] input_port_list;*
*output [msb:lsb] output_port_list;*
*reg [msb:lsb] reg_variable_list;*
*parameter [msb:lsb] parameter_list;*
*integer [msb:lsb] integer_list;*
*... statements ...*
***endtask***

# Module #11 : Verilog HDL Tasks

***Example 11.1:***

```
module alu (func, a, b, c);
input [1:0] func;
input [3:0] a, b;
output [3:0] c;
reg [3:0] c; // so it can be assigned in always
block
task my_and;
input[3:0] a, b;
output [3:0] andout;
integer i;
begin
for (i = 3; i >= 0; i = i - 1)
andout[i] = a[i] & b[i];
end
endtask

always @(func or a or b) begin
case (func)
2'b00: my_and (a, b, c);
2'b01: c = a | b;
2'b10: c = a - b;
default: c = a + b;
endcase
end
endmodule
```

# Module #11 : Verilog HDL Tasks

***Example 11.2:***

```
module alu (func, a, b, c);
input [1:0] func;
input [3:0] a, b;
output [3:0] c;
reg [3:0] c; // so it can be assigned in always
block
task my_and;
input[3:0] a, b;
output [3:0] andout;
integer i;
begin
for (i = 3; i >= 0; i = i - 1)
#10;
andout[i] = a[i] & b[i];
end
endtask
always @(func or a or b) begin
case (func)
2'b00: my_and (a, b, c);
2'b01: c = a | b;
2'b10: c = a - b;
default: c = a + b;
endcase
end
endmodule
```