• **Tasks** – act like subroutines in Verilog

```verilog
task check_output;
    input [1:0] num, device_out, good_out;
    if (device_out != good_out)
        $display("Test results mismatch on %b : expected %b, got %b", num, good_out, device_out);
endtask
```

• **Initial blocks** – execute once at the start of simulation (used in simulation)

```verilog
// example Verilog snippet
reg [1:0] test_in, correct_out;
wire [1:0] test_out;
test_device my_chip (test_in,test_out);

initial begin
    for(i = 0; i < num_tests; i = i + 1) begin
        test_in = i;
        correct_out = test_in + 1;
        check(i,test_out,correct_out);
    end
    $finish();
end
```

**Problem 1: FSM Design – Verilog Edition**

Below is a template corresponding with the recommended CS150 Verilog FSM design pattern. Implement the FSM (from last weeks discussion) that detects the pattern ‘0101’ on a 1-bit input.

```verilog
module fsm (input data,clk,reset,
            output success);

    reg [__:_] current_state, next_state;

    always @(posedge clk) begin
        if (_____ ) ______________;
        else ________________________;
    end

    always @(*) begin

    end

    assign success = ____________;
endmodule
```