Test Benches (Test Fixtures)

Verilog for Testing
Overview

• We have concentrated on Verilog for synthesis
• Can also use Verilog as a test language
• Very important to conduct comprehensive verification on your design
• To simulate your design you need to produce an additional module that includes your *synthesizable* Verilog design.
  - Usually referred to as a TEST BENCH or TEST FIXTURE
    • Not hardware, just additional Verilog!
Test Bench

- A virtual platform containing the design to be tested (UUT) and virtual wires connected to the UUT inputs and outputs.
- To exercise and verify the correctness of a design to be implemented in hardware
- Has three main purposes
  - to generate stimulus for simulation
  - to apply this stimulus to the module under test and to collect output responses
  - to compare output responses with expected values
- Test Bench should be created by a different engineer than the one who created the synthesizable Verilog
Test Bench – Virtual Platform

Apply Stimulus  \rightarrow  Module to be Tested (UUT)  \rightarrow  Verify Responses
Test Bench Overview

- A Test Bench module consists of
  - Port list has NO ports
  - Instantiate module to be tested (UUT)
  - Declare internal signals to wire to UUT inputs and outputs
  - Verilog statements to provide stimulus and verify UUT responses
  - Designing a test bench that has good coverage can be a very involved project!
module decoder_tf;

    internal signal declarations

    UUT: test_component instantiation

    signals to generate stimulus

    statements to verify responses

endmodule
Example Decoder – is the design correct?

```verilog
module decoder(
    input [2:0] sw,
    output reg [7:0] led
);

always @ (sw)
    case (sw)
        0: led = 8'b00000001;
        1: led = 8'b00000010;
        2: led = 8'b00000100;
        3: led = 8'b00011000; // mistake
        4: led = 8'b00010000;
        5: led = 8'b00100000;
        6: led = 8'b01000000;
        7: led = 8'b10000000;
    endcase

decoder
```
Create Test Fixture

- Select Simulation View
- Select Project => New Source

Select decoder source to associate with test fixture
Outline of Test Fixture produced

```verilog
`timescale 1ns / 1ps

// Comments deleted
module decoder_tf;

    // Inputs
    reg [2:0] sw;

    // Outputs
    wire [7:0] led;

    // Instantiate the Unit Under Test (UUT)
    decoder uut (.
        .sw(sw),
        .led(led)
    );

    initial begin
        // Initialize Inputs
        sw = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        end

    endmodule
```
Apply input stimulus

```verilog
// Wait 100 ns for global reset to finish
#100;

sw = 3'b001; // check all input combinations
#100;
sw = 3'b010;
#100;
sw = 3'b011;
#100;
sw = 3'b100;
#100;
sw = 3'b101;
#100;
sw = 3'b110;
#100;
sw = 3'b111;
#100;

end

endmodule
```
Simulate Behavioral Model
Testing sequential logic

- Creating clock signal

```
wire [7:0] led;

// Instantiate the Unit Under Test (UUT)
decoder uut (  
  .sw(sw),  
  .led(led)
);

reg clk_50M;

always // create 50MHz clock
begin
  clk_50M = 0;
  #10;    // 10 times time units = 10ns
  clk_50M = 1;
  #10;
end

initial begin
  // Initialisation
end
```

- Can also ‘restart’, ‘run’, and add internal signals to wave