Testbench Infrastructure

- Testbench
- Testbench Architecture
- Simulation Control
- Data and Transactions
- Transactors
- Transaction-Level Interfaces
- Timing Interface
- Callback Methods
- Ad-Hoc Testbenches
- Legacy Bus-Functional Model

Reference: Verification Methodology Manual by Janick Bergen – Chapter 4

Testbench

- Testbench mimic the environment in which the design will reside.
- It checks whether the RTL Implementation meets the design spec or not.
- This environment creates invalid and unexpected as well as valid and expected conditions to test the design.

Linear(Direct) TestBench

```
module top();          //TestBench code start
      reg [15:0] a,b;
      wire [16:0] c;

      adder DUT(a,b,c);
   initial
   begin
            a = 16'h45;
            b = 16'h12;
          #10 $display(" a=%0d,b=%0d,c=%0d",a,b,c);
   end
  endmodule            //TestBench code end
```

Random TestBench

```
module top();        //TestBench code start
      reg [15:0] a,b;
      wire [16:0] c;

   adder DUT(a,b,c);
   initial  begin
   repeat  (100) begin
            a = $random;
            b = $random;
            #10 $display(" a=%0d,b=%0d,c=%0d",a,b,c);
       end
   end
  endmodule            //TestBench code end
```

Self Checking TestBench

- Visually inspecting simulation results to determine functional correctness is not an acceptable long-term strategy
- Whatever intellectual process you would go through to identify an error visually in the simulation result must be coded in your testbench.
- This technique will let the testbench detect errors and declare success or failure on its own

Self Checking TestBench

```
module top();        //TestBench code start
    ….  //declarations

  adder DUT(a,b,c);
  initial  begin
  repeat  (100) begin
          a = $random;
          b = $random;
          #10 $display(" a=%0d,b=%0d,c=%0d",a,b,c);
          if(a+b !=c) $display(“ERROR”);
      end
  end
  endmodule          //TestBench code end
```

Self Checking TestBench (Example -1)

```
//Adder & Adder test with all the combinations, compare with expected results and stop
//when adder output fails

`timescale 1ns/1ps
module adder_tb_cmp;

  // signals for connecting the adder
  reg  [3:0] a, b;
  reg        cin;
  wire [3:0] sum;
  wire       carry;
  reg  [3:0] expected;
  integer    i,j,k;
  // adder instance
  fadd4 DUT(.a(a),.b(b),.cin(cin),.sum(sum),.carry(carry));
/*
  initial begin
    #0 a = 2, b = 2, cin = 0, expected = 4;
    #10 a = 3, b = 5, cin = 2, expected = 10;
    #20 a = 8, b = 6, cin = 3, expected = 17;
  end
*/
```

Self Checking TestBench (Example -2)

```
//Adder testbench with all the possible values
initial begin
 for(i = 0; i <16; i++) begin
   a <= i;
   for(j = 0; j <16; j++) begin
     b <= j;
       for(k = 0; k <2; k++) begin
         cin <= k;
         #10
         expected <= a + b + cin;
         if(sum !== a + b + cin) begin
           $display("Error - Sum is wrong");
           $stop;
         end
      end
   end
  end
end


initial
  $monitor($time, "ns a=%b, b=%b, cin=%b, carry=%b, sum=%b, expected=%b",a,b,cin,sum,carry,expected);
endmodule
```

Self Checking TestBench (Example -3)

```
//Adder & Adder test with all the combinations and stop when adder output fails using assert
`timescale 1ns/1ps
module adder_tb;

  // signals for connecting the adder
  reg  [3:0] a, b;
  reg        cin;
  wire [3:0] sum;
  wire       carry;
  integer    i,j,k;
  // adder instance
  fadd4 DUT(.a(a),.b(b),.cin(cin),.sum(sum),.carry(carry));
```

Self Checking TestBench (Example -3)

```
//Adder testbench with all the possible values
initial begin
 for(i = 0; i <16; i++) begin
   a <= i;
   for(j = 0; j <16; j++) begin
     b <= j;
       for(k = 0; k <2; k++) begin
         cin <= k;
         #10
         assert (sum == a + b + cin) begin
           $display("Test Passed");
         end
         else begin
           $error("Error  - Sum is wrong");
           $stop;
         end
       end
     end
    end
  end

initial
  $monitor($time, "ns a=%b, b=%b, cin=%b, carry=%b, sum=%b,",a,b,cin,sum,carry);
endmodule
```