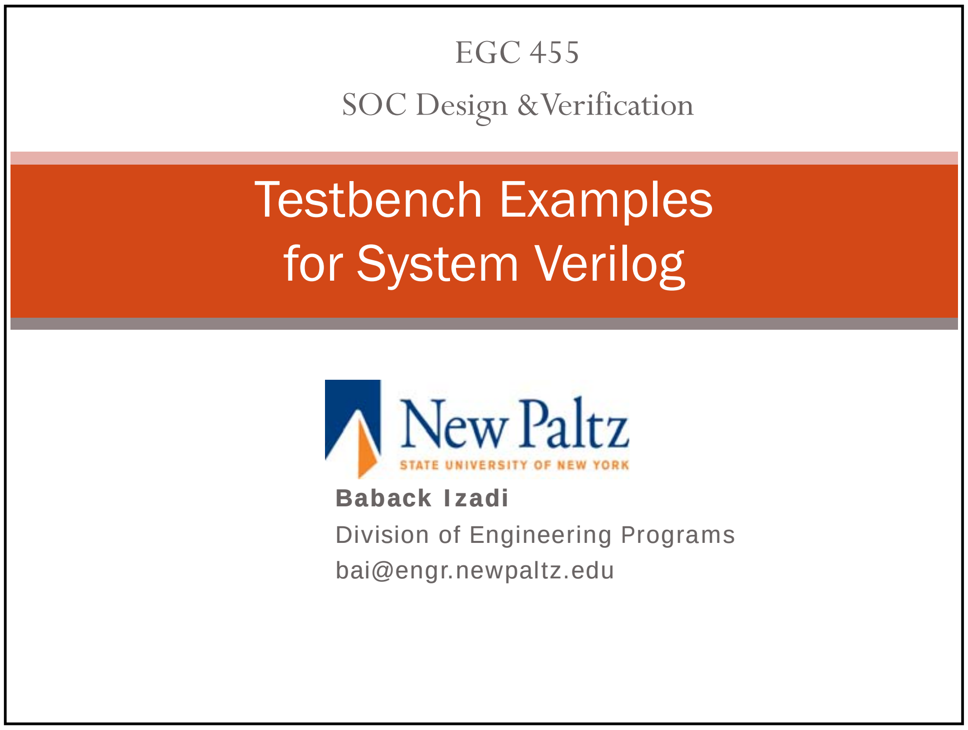
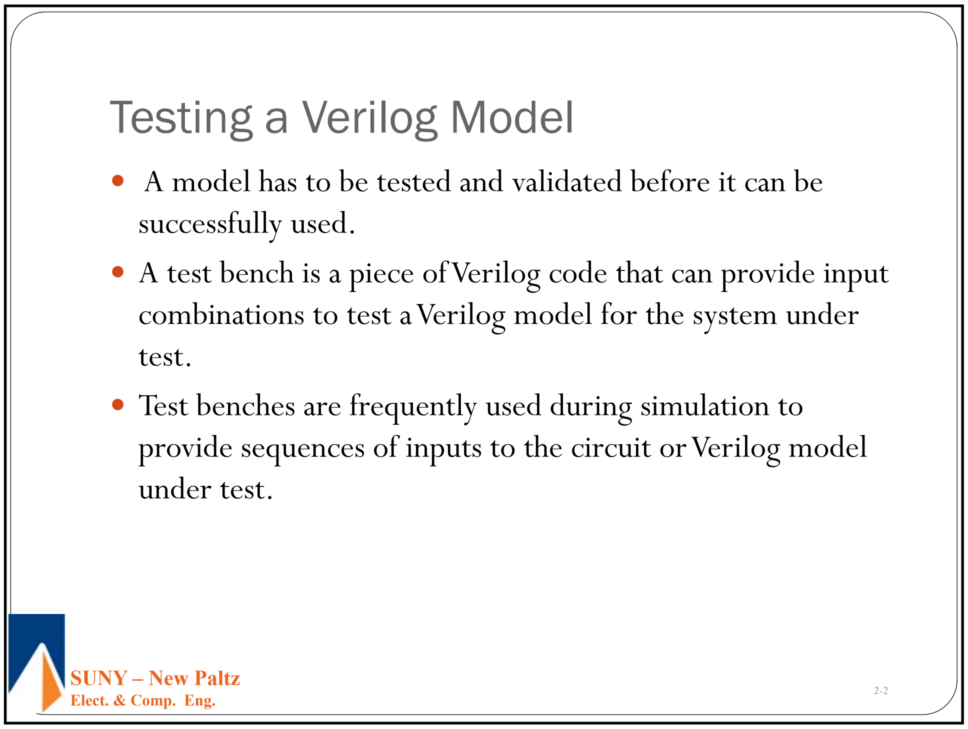


EGC 455
SOC Design &Verification

Testbench Examples for System Verilog

New Paltz
STATE UNIVERSITY OF NEW YORK
Baback Izadi
Division of Engineering Programs
bai@engr.newpaltz.edu



Testing a Verilog Model

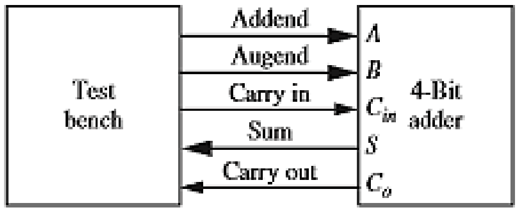
- A model has to be tested and validated before it can be successfully used.
- A test bench is a piece of Verilog code that can provide input combinations to test a Verilog model for the system under test.
- Test benches are frequently used during simulation to provide sequences of inputs to the circuit or Verilog model under test.

SUNY – New Paltz
Elect. & Comp. Eng.

2-2

Testing a Verilog Model

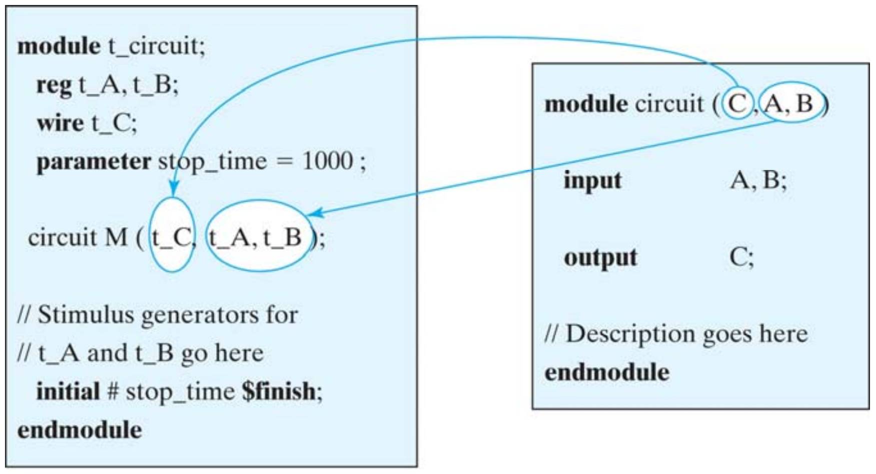
- Test bench for testing a 4-bit binary adder:



SUNY – New Paltz
Elect. & Comp. Eng.

2-3

Interaction between stimulus and design modules



```
module t_circuit;
  reg t_A, t_B;
  wire t_C;
  parameter stop_time = 1000 ;

 circuit M ( t_C, t_A, t_B );

// Stimulus generators for
// t_A and t_B go here
  initial # stop_time $finish;
endmodule
```

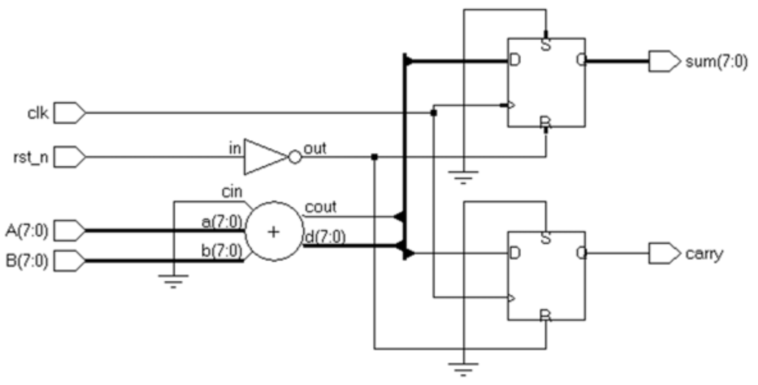
```
module circuit ( C, A, B )

  input          A, B;

  output         C;

// Description goes here
endmodule
```

SUNY – New Paltz
Elect. & Comp. Eng.

```
module top;
byte a, b, vhdl_out, sv_out;
bit sel;
mux_sv SV_MUX (.*, .dat_out(sv_out));
mux_vhdl VHDL_MUX(.*, .dat_out(vhdl_out));
initial
  begin
    $monitor("a: %h  b: %h  sel: %b   vhdl_out: %h   sv_out:
%h", a, b, sel, vhdl_out, sv_out);
    a = 8'h55;
    b = 8'hAA;
    sel = 0;
    #1;
    sel = 1;
    #1;
    sel = 0;
    #1;
    sel = 1;
  end
Endmodule
```

SUNY – New Paltz
Elect. & Comp. Eng.

A simple adder



SUNY – New Paltz
Elect. & Comp. Eng.

```
// Design file
module adder
       ( input [7:0] A,
       input [7:0] B,
       input clk,
       input rst_n,
       output reg [7:0] sum,
       output reg carry);

       always @(posedge clk or negedge rst_n)
              if (!rst_n)
                     {carry,sum} <= 0;
              else
                     {carry,sum} <= A + B;
endmodule
```

SUNY – New Paltz
Elect. & Comp. Eng.

```
//TestBench file
Module adder_tester (
output reg[7:0]   A,
output reg[7:0]   B,
output reg        clk,
output reg        rst_n,
input  wire       carry,
input  wire [7:0] sum);

adder ADDER (.*);

initial begin
  clk = 0;
  rst_n = 0;
  A = 0;
  B = 0;
  @(posedge clk);
  @(posedge clk);
  rst_n = 1;
end

always #10 clk = ~clk;
```

8

SUNY – New Paltz
Elect. & Comp. Eng.

```
//Continue TestBench
initial begin
   @(posedge rst_n);
   for (A = 0; A <= 25; A ++) begin
      for (B = 0; B <= 10; B++) begin
         @(posedge clk);
         @(negedge clk);
         assert({carry,sum} == A + B)
          else begin
          $display("%0d + %0d = %0d not %0d",A, B, A+B,{carry,sum}
          $stop;
              end
      end
   end
   $stop;
end
endmodule // adder_tester
```

9 SUNY – New Paltz
Elect. & Comp. Eng.

```
module multadd_sv
       (input [7:0] a, b, c, d,
        input clk, rst_n,
       output logic [16:0] mulsum
       );
       logic[16:0] ab, cd, ab_cd;

always @(a or b or c or d)
       begin
              ab = a * b;
              cd = c * d;
              ab_cd = ab + cd;
       end

always @(posedge clk)
       if (!rst_n)
              mulsum <= 0;
       else
              mulsum <= ab_cd;
endmodule
```

SUNY – New Paltz
Elect. & Comp. Eng.

```
//TestBench file
module top;
logic[7:0] a, b, c, d;
logic [16:0] vhdl_out, sv_out, predicted;
bit clk, rst_n;

multadd_vhdl VHDL_MUX(.*, .mulsum(vhdl_out));
multadd_sv SV_MUX (.*, .mulsum(sv_out));

initial begin
 $monitor("%t: a: %h  b: %h  c: %h  d: %h   predicted: %h
vhdl_out: %h   sv_out:  %h", $time,a, b, c, d, predicted,
vhdl_out, sv_out);
 clk = 0;
 rst_n = 0;
 @(posedge clk);
 @(negedge clk);
 rst_n = 1;
end

always #10ns clk = ~clk;
```

SUNY – New Paltz
Elect. & Comp. Eng.

```
initial begin
   @(negedge clk);
   @(negedge clk);
   repeat (10) begin
     @(negedge clk);
     a = $random;
     b = $random;
     c = $random;
     d = $random;
     predicted = a*b + c*d;
     @(posedge clk);
     @(posedge clk);
     #1ns;
  assert((vhdl_out == predicted) && (sv_out == predicted));
    end
    $stop;
  end

endmodule
```

SUNY – New Paltz
Elect. & Comp. Eng.

```
module top;
   reg [7:0]  reg8;
   reg [63:0] reg64_ext, reg64_zero;
   integer    my_random, i, seed;

   initial
      begin
         seed = 5;
         for (i = 1; i<= 4; i=i+1) begin
            my_random = $random (seed);
            reg8       =  my_random;
            reg64_ext  =  my_random;
            reg64_zero = {my_random};
            $display;
            $display ("my_random(decimal):  %d", my_random);
            $display ("my_random(hex     ):  %h",my_random);
            $display ("reg8:                 %h", reg8);
            $display ("reg64_ext:            %h", reg64_ext);
            $display ("reg64_zero:           %h",reg64_zero);
         end
      end
endmodule
```

SUNY – New Paltz
Elect. & Comp. Eng.

“do.run” Script

```
//You can store the file in your project
directory and use load->macro from QuestaSim
to compile and simulate
// if files in “work” exists, delete them.
if [file exists "work"] {vdel -all}
//create a work library
vlib work
//Compile Verilog “file”
vlog file.sv
//Run the simulation on QuestaSim
vsim -voptargs="+acc" top
//Run all cases
run -all
```

SUNY – New Paltz
Elect. & Comp. Eng.