/ Verilog HDL for " lab3\_sims " , " verilog\_3in\_stim " " functional "

‘timescale 1ns/1ps

module verilog\_3in\_stim (

inout wire VDD,

inout wire VSS,

output reg[ 7 : 0 ] OUT ) ;

assign VDD = 1 ’b1 ;

assign VSS = 1 ’b0 ;

reg clock ;

initial begin

clock = 1 ’b0 ;

OUT = 8 ’h00 ;

end

always begin

#25.0 clock = ~clock ;

end

always @ (posedge clock )

begin

OUT = #1.0 OUT + 8 ’h01 ;

end

always @ (negedge clock )

begin

if ( OUT == 8 ’hFF )

#50

$finish;

end

endmodule

//Half Adder

initial

begin

A = 1 ’b0 ; B = 1 ’b0 ;

#10 A = 1 ’b0 ; B = 1 ’b1 ;

#10 A = 1 ’b1 ; B = 1 ’b0 ;

#10 A = 1 ’b1 ; B = 1 ’b1 ;

#10

$finish;

end

//Decoder

module booth\_decode\_9bit\_v1\_tb1 (

output wire Xm1,

output wire X0,

output wire Xp1,

output reg [ 7 : 0 ] Y,

input wire [ 8 : 0 ] PP,

inout wire VDD,

inout wire VSS ) ;

parameter NUM0 = 8 ’ b00000000 ;

parameter NUM1 = 8 ’ b00000001 ;

parameter NUM2 = 8 ’ b11111110 ;

parameter NUM3 = 8 ’ b11111111 ;

parameter NUM4 = 8 ’ b10101010 ;

parameter NUM5 = 8 ’ b01010101 ;

parameter NUM6 = 8 ’ b10000000 ;

parameter NUM7 = 8 ’ b11001100 ;

reg clk ;

reg [ 2 : 0 ] X;

reg [ 3 : 0 ] sel ;

assign VDD = 1 ’b1 ;

assign VSS = 1 ’b0 ;

assign Xm1 = X[ 0 ] ;

assign X0 = X[ 1 ] ;

assign Xp1 = X[ 2 ] ;

// I n i t i a l i z e variables .

initial begin

clk = 1 ’b0 ;

X = 3 ’b000 ;

sel = 4 ’h0 ;

end

// Clock for advancing through the test cases .

always begin

#50.0 clk = ~clk ;

end

// Counter for X to generate a l l p o s s i b i l i t i e s .

always @ ( posedge clk )

begin

X = X + 3 ’b001 ;

end

// ’ sel ’ variable to iterate through the case statement below .

always @ ( posedge clk )

begin

if ( X == 3 ’b000 )

begin

sel = sel + 4 ’h1 ;

end

end

// Different cases for Y.

Always @ ( sel or Y)

begin

case ( sel )

4 ’h1 : begin

Y = NUM0;

end

4 ’h2 : begin

Y = NUM1;

end

4 ’h3 : begin

Y = NUM2;

end

4 ’h4 : begin

Y = NUM3;

end

4 ’h5 :begin

Y = NUM4;

end

4 ’h6 : begin

Y = NUM5;

end

4 ’h7 : begin

Y = NUM6;

end

4 ’h8 : begin

Y = NUM7;

end

default: begin

Y = 8 ’h00 ;

end

endcase

end

// End simulation when we finish looking at a l l our cases .

always @ ( posedge clk )

begin

if ( sel == 4 ’h8 )

begin

if ( X == 3 ’b111 )

begin

#100;

$finish;

end

end

end

endmodule

initial

begin

A = 1 ’b0 ; B = 1 ’b0 ; Cin = 1 ’b0 ;

#10 A = 1 ’b0 ; B = 1 ’b0 ; Cin = 1 ’b1 ;

#10 A = 1 ’b0 ; B = 1 ’b1 ; Cin = 1 ’b0 ;

#10 A = 1 ’b0 ; B = 1 ’b1 ; Cin = 1 ’b1 ;

#10 A = 1 ’b1 ; B = 1 ’b0 ; Cin = 1 ’b0 ;

#10 A = 1 ’b1 ; B = 1 ’b0 ; Cin = 1 ’b1 ;

#10 A = 1 ’b1 ; B = 1 ’b1 ; Cin = 1 ’b0 ;

#10 A = 1 ’b1 ; B = 1 ’b1 ; Cin = 1 ’b1 ;

#10

$finish;

end

//12 Bit Adder

module adder\_12bit\_vstim (

output reg [ 1 1 : 0 ] A,

output reg [ 1 1 : 0 ] B,

output reg Cin ,

inout wire VDD,

inout wire VSS ) ;

reg clk ;

reg [ 3 : 0 ] sel ;

initial begin

sel = 4 ’ b0000 ;

clk = 1 ’b0 ;

A = 12 ’h000 ;

B = 12 ’h000 ;

Cin = 1 ’b0 ;

end

always begin

#25.0 clk = ~clk ;

end

always @ ( posedge clk )

begin

sel = sel + 4 ’ b0001 ;

end

always @ ( sel )

begin

case ( sel )

4 ’ b0001 : begin

A = 12 ’h7FF;

B = 12 ’h001 ;

Cin = 1 ’b0 ; // S = 800

end

4 ’ b0010 : begin

A = 12 ’h08F ;

B = 12 ’h07F ;

Cin = 1 ’b0 ; // S = 10E

end

4 ’ b0011 : begin

A = 12 ’hF00 ;

B = 12 ’h100 ;

Cin = 1 ’b0 ; // S = (1) 000

end

4 ’ b0100 : begin

A = 12 ’h800 ;

B = 12 ’hFFF;

Cin = 1 ’b0 ; // S = (1)7FF

end

4 ’ b0101 : begin

A = 12 ’h07F ;

B = 12 ’hF71 ;

Cin = 1 ’b0 ; // S = FF0

end

4 ’ b0110 : begin

A = 12 ’h7FF;

B = 12 ’h001;

Cin = 1 ’b1; // S = 801

end

4 ’ b0111 : begin

A = 12 ’h08F;

B = 12 ’h07F;

Cin = 1 ’b1; // S = 10F

end

4 ’ b1000 : begin

A = 12 ’hF00;

B = 12 ’h100;

Cin = 1 ’b1; // S = (1) 001

end

4 ’ b1001 : begin

A = 12 ’h800;

B = 12 ’hFFF;

Cin = 1 ’b1; // S = (1) 800

end

4 ’ b1010 : begin

A = 12 ’h07F ;

B = 12 ’hF71 ;

Cin = 1 ’b1 ; // S = FF1

end

default : begin

#50

$finish;

end

endcase

end

endmodule