Nang Thiri Wutyi

20113

9/23/2024

**Verilog Lab Week #1**

1. **“Hello World” in Verilog**

//-----------------------------------------------------

// This is my first Verilog Program

// Design Name : hello\_world

// File Name : hello\_world.v

// Function : This program will print 'hello world'

//-----------------------------------------------------

module helloWorld;

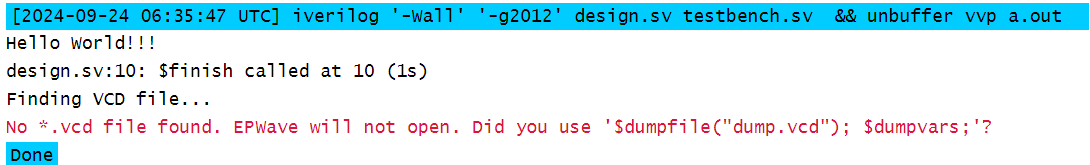
initial begin

$display ("Hello World!!!");

#10 $finish;

end

endmodule // End of Module helloWorld

**Results** 

1. **The digital circuit schematics based on the module gate**

**Main module “gate”**

module gates(

output wire out0, // Declare outputs

output wire out1,

output wire out2,

input wire in1, // Declare inputs

input wire in2,

input wire in3,

input wire in4

);

// Logic operations

not U1(out0, in1);

and U2(out1, in1, in2, in3, in4);

xor U3(out2, in1, in2, in3);

endmodule

**Testbench for gate**

// Testbench for the "gates" module

module testbench();

// Inputs to the gates module (reg type for testbench)

reg in1, in2, in3, in4;

// Outputs from the gates module (wire type)

wire out0, out1, out2;

// Instantiate the "gates" module

gates uut (

.out0(out0),

.out1(out1),

.out2(out2),

.in1(in1),

.in2(in2),

.in3(in3),

.in4(in4)

);

// Test logic

initial begin

// Display the header

$display("in1 in2 in3 in4 | out0 out1 out2");

$display("-----------------------------");

// Apply different input combinations and display results

in1 = 0; in2 = 0; in3 = 0; in4 = 0; #10;

$display("%b %b %b %b | %b %b %b", in1, in2, in3, in4, out0, out1, out2);

in1 = 0; in2 = 1; in3 = 1; in4 = 0; #10;

$display("%b %b %b %b | %b %b %b", in1, in2, in3, in4, out0, out1, out2);

in1 = 1; in2 = 1; in3 = 0; in4 = 1; #10;

$display("%b %b %b %b | %b %b %b", in1, in2, in3, in4, out0, out1, out2);

in1 = 1; in2 = 1; in3 = 1; in4 = 1; #10;

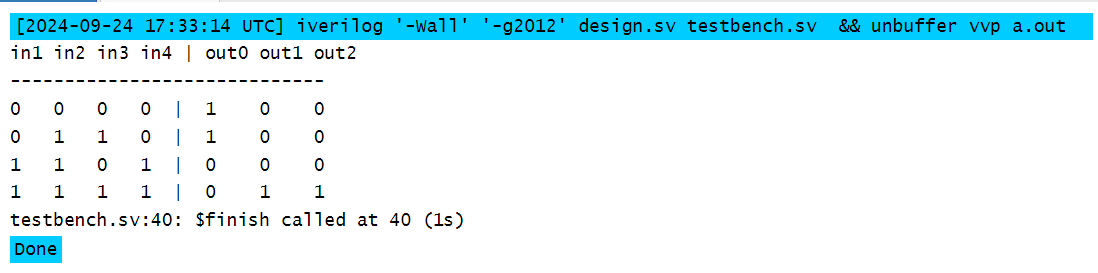
$display("%b %b %b %b | %b %b %b", in1, in2, in3, in4, out0, out1, out2);

$finish;

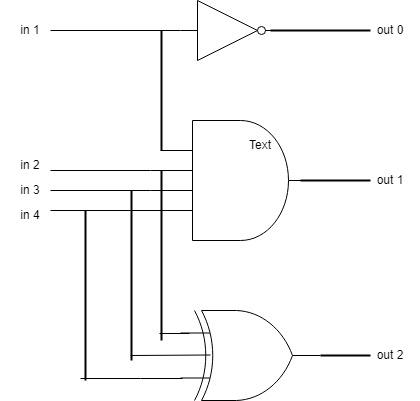
end

endmodule

**Results**



**Digital circuit schematics**

****

1. **Main Module for oneBitFA1 (Continuous Assignment)**

module oneBitFA1(

input wire a, // Input bit a

input wire b, // Input bit b

input wire ci, // Input carry-in

output wire co, // Output carry-out

output wire sum // Output sum

);

// Continuous assignment to calculate sum and carry-out

assign {co, sum} = a + b + ci;

endmodule

**Testbench for oneBitFA1**

module tb\_oneBitFA1;

// Inputs

reg a, b, ci;

// Outputs

wire co, sum;

// Instantiate the full adder module

oneBitFA1 uut (

.a(a),

.b(b),

.ci(ci),

.co(co),

.sum(sum)

);

// Initial block to apply inputs

initial begin

// Apply different combinations of inputs

a = 0; b = 0; ci = 0;

#10 a = 0; b = 0; ci = 1;

#10 a = 0; b = 1; ci = 0;

#10 a = 0; b = 1; ci = 1;

#10 a = 1; b = 0; ci = 0;

#10 a = 1; b = 0; ci = 1;

#10 a = 1; b = 1; ci = 0;

#10 a = 1; b = 1; ci = 1;

#10 $finish; // End the simulation

end

// Monitor to display the results

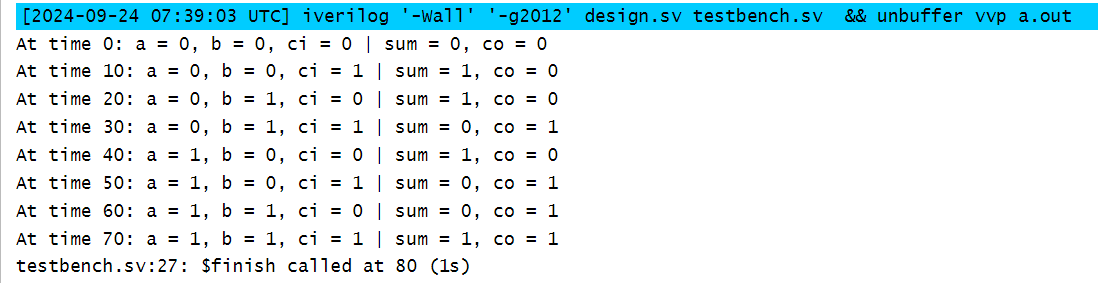
initial begin

$monitor("At time %0d: a = %b, b = %b, ci = %b | sum = %b, co = %b", $time, a, b, ci, sum, co);

end

endmodule

**Results**

****

**Main Module for oneBitFA2 (1-bit Full Adder using Always Block)**

module oneBitFA2 (

input wire a\_i,

input wire b\_i,

input wire ci\_i,

output reg sum\_o,

output reg co\_o

);

// Always block for sum and carry-out

always @(\*) begin

{co\_o, sum\_o} = a\_i + b\_i + ci\_i;

end

endmodule

**Testbench for oneBitFA2**

module tb\_oneBitFA2;

// Declare testbench inputs and outputs

reg a\_i, b\_i, ci\_i;

wire sum\_o, co\_o;

// Instantiate the oneBitFA2 module

oneBitFA2 uut (

.a\_i(a\_i),

.b\_i(b\_i),

.ci\_i(ci\_i),

.sum\_o(sum\_o),

.co\_o(co\_o)

);

// Testbench procedure

initial begin

$display("Testing 1-bit full adder using always block (oneBitFA2)...");

$monitor("a\_i=%b, b\_i=%b, ci\_i=%b -> sum\_o=%b, co\_o=%b", a\_i, b\_i, ci\_i, sum\_o, co\_o);

// Test case 1

a\_i = 0; b\_i = 0; ci\_i = 0;

#10;

// Test case 2

a\_i = 0; b\_i = 0; ci\_i = 1;

#10;

// Test case 3

a\_i = 0; b\_i = 1; ci\_i = 0;

#10;

// Test case 4

a\_i = 0; b\_i = 1; ci\_i = 1;

#10;

// Test case 5

a\_i = 1; b\_i = 0; ci\_i = 0;

#10;

// Test case 6

a\_i = 1; b\_i = 0; ci\_i = 1;

#10;

// Test case 7

a\_i = 1; b\_i = 1; ci\_i = 0;

#10;

// Test case 8

a\_i = 1; b\_i = 1; ci\_i = 1;

#10;

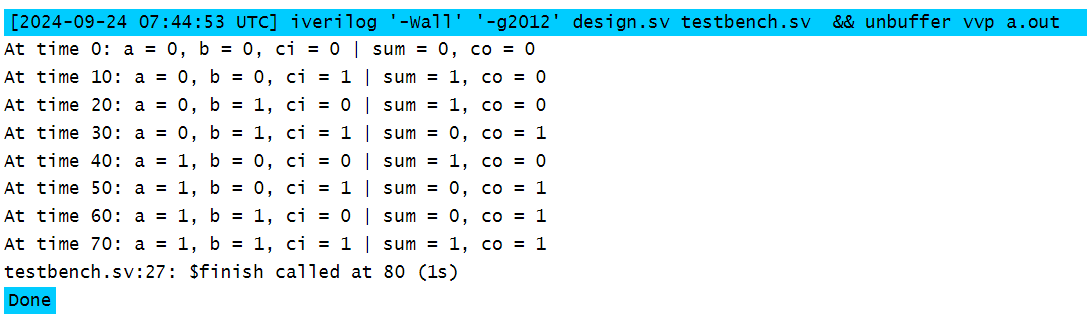
// End simulation

$finish;

end

endmodule

**Results**

****