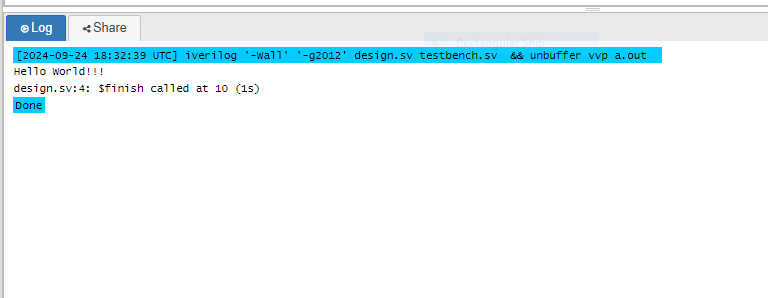
**Verilog HDL**  **DATE : 9/24/2024**

**S A SABBIRUL MOHOSIN NAIM**

**ID; 20176**

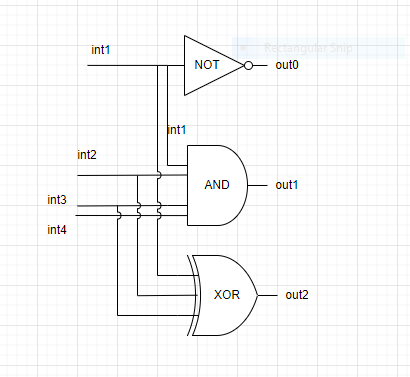
**HW: 01**

# 1. Hello World Module

module helloWorld;  
 initial begin  
 $display ("Hello World!!!");  
 #10 $finish;  
 end  
endmodule  
  


# 2. Digital Circuit Schematic for gates Module

module gates();  
 wire out0;  
 wire out1;  
 wire out2;  
 reg in1, in2, in3, in4;  
 not U1(out0, in1);  
 and U2(out1, in1, in2, in3, in4);  
 xor U3(out2, in1, in2, in3);  
endmodule

Based on the module gates,the digital circuit schematic.  


# 3. Testbenches and Simulation for Design Modules

## \*oneBitFA1 Module

module oneBitFA1(

input a,

input b,

input ci,

output co,

output sum

);

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

endmodule

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

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

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;

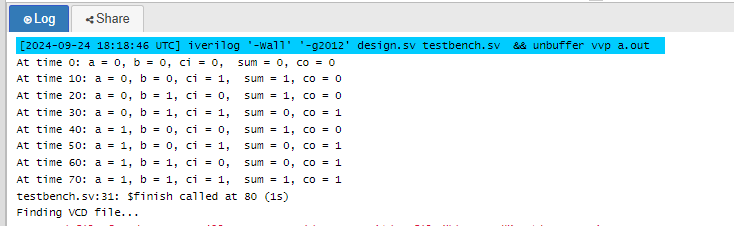
#10 a = 1; b = 1; ci = 0;

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

#10 $finish(); // End the simulation

end

endmodule



## \*oneBitFA2 Module

module oneBitFA2(

input wire a\_i,

input wire b\_i,

input wire ci\_i,

output reg sum\_o,

output reg co\_o

);

always @(a\_i, b\_i, ci\_i)begin

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

end

endmodule

//testbr

module tb\_oneBitFA2;

// Inputs

reg a\_i;

reg b\_i;

reg ci\_i;

// Outputs

wire sum\_o;

wire co\_o;

// Instantiate the design under test (DUT)

oneBitFA2 dut (

.a\_i(a\_i),

.b\_i(b\_i),

.ci\_i(ci\_i),

.sum\_o(sum\_o),

.co\_o(co\_o)

);

// Test procedure

initial begin

// Monitor the changes in inputs and outputs

$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);

// Apply test cases

a\_i = 0; b\_i = 0; ci\_i = 0; #10; // Test Case 1

a\_i = 0; b\_i = 0; ci\_i = 1; #10; // Test Case 2

a\_i = 0; b\_i = 1; ci\_i = 0; #10; // Test Case 3

a\_i = 0; b\_i = 1; ci\_i = 1; #10; // Test Case 4

a\_i = 1; b\_i = 0; ci\_i = 0; #10; // Test Case 5

a\_i = 1; b\_i = 0; ci\_i = 1; #10; // Test Case 6

a\_i = 1; b\_i = 1; ci\_i = 0; #10; // Test Case 7

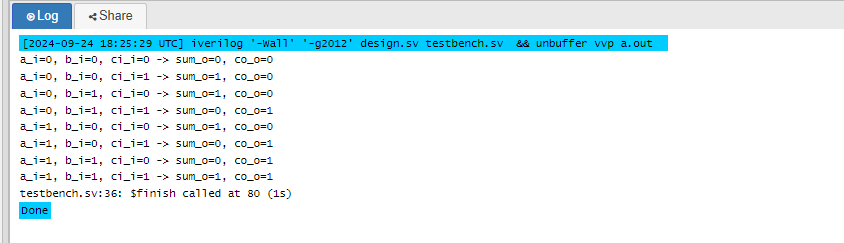
a\_i = 1; b\_i = 1; ci\_i = 1; #10; // Test Case 8

// End the simulation

$finish;

end

endmodule



## \* DFFSynch Module

module DFFSynch(

d\_i,

rst\_i,

clk\_i,

q\_o

);

input d\_i,rst\_i,clk\_i;

output q\_o;

reg q\_o;

always @(posedge clk\_i)begin

if(rst\_i) q\_o <= 0;

else q\_o <= d\_i;

end

endmodule

//testbr

module tb\_DFFSynch;

// Inputs

reg d\_i;

reg rst\_i;

reg clk\_i;

// Outputs

wire q\_o;

// Instantiate the design under test (DUT)

DFFSynch dut (

.d\_i(d\_i),

.rst\_i(rst\_i),

.clk\_i(clk\_i),

.q\_o(q\_o)

);

// Clock generation

initial begin

clk\_i = 0;

forever #5 clk\_i = ~clk\_i; // 10-unit time period clock

end

// Test procedure

initial begin

// Monitor the changes in inputs and outputs

$monitor("clk\_i=%b, rst\_i=%b, d\_i=%b -> q\_o=%b", clk\_i, rst\_i, d\_i, q\_o);

// Initialize inputs

rst\_i = 1; d\_i = 0; #10; // Test Case 1: Reset is active, output should be 0

rst\_i = 0; d\_i = 1; #10; // Test Case 2: Remove reset, d\_i=1, output should follow d\_i

d\_i = 0; #10; // Test Case 3: d\_i=0, output should follow d\_i

d\_i = 1; #10; // Test Case 4: d\_i=1, output should follow d\_i

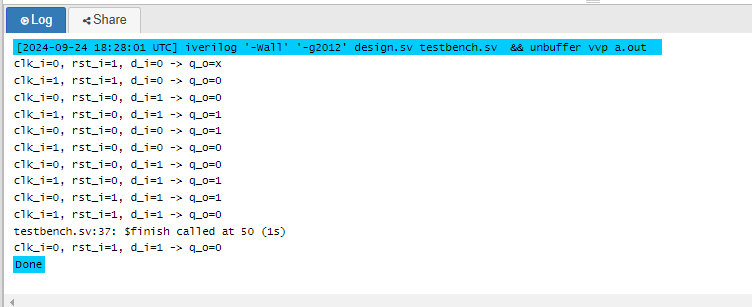
rst\_i = 1; #10; // Test Case 5: Activate reset, output should go to 0

// End the simulation

$finish;

end

endmodule



## \* DFFAsynch Module

module DFFAsynch(

d\_i,

rst\_i,

clk\_i,

q\_o

);

input d\_i,rst\_i,clk\_i;

output q\_o;

reg q\_o;

always @(posedge clk\_i or posedge rst\_i)begin

if(rst\_i) q\_o <= 0;

else q\_o <= d\_i;

end

endmodule

//tr

module tb\_DFFAsynch;

// Inputs

reg d\_i;

reg rst\_i;

reg clk\_i;

// Outputs

wire q\_o;

// Instantiate the design under test (DUT)

DFFAsynch dut (

.d\_i(d\_i),

.rst\_i(rst\_i),

.clk\_i(clk\_i),

.q\_o(q\_o)

);

// Clock generation

initial begin

clk\_i = 0;

forever #5 clk\_i = ~clk\_i; // 10-unit time period clock

end

// Test procedure

initial begin

// Monitor the changes in inputs and outputs

$monitor("clk\_i=%b, rst\_i=%b, d\_i=%b -> q\_o=%b", clk\_i, rst\_i, d\_i, q\_o);

// Initialize inputs

rst\_i = 1; d\_i = 0; #10; // Test Case 1: Reset is active, output should be 0

rst\_i = 0; d\_i = 1; #10; // Test Case 2: Remove reset, d\_i=1, output should follow d\_i

d\_i = 0; #10; // Test Case 3: d\_i=0, output should follow d\_i

d\_i = 1; #10; // Test Case 4: d\_i=1, output should follow d\_i

rst\_i = 1; #10; // Test Case 5: Activate reset, output should go to 0

rst\_i = 0; d\_i = 1; #10; // Test Case 6: After reset, d\_i=1, output should follow d\_i

// Add a delay to see the effect of the clock edge

d\_i = 0; #10; // Test Case 7: d\_i=0, output should follow d\_i on next clock edge

#10; // Wait for the clock edge

d\_i = 1; #10; // Test Case 8: d\_i=1, output should follow d\_i on next clock edge

// End the simulation

$finish;

end

endmodule

