**Course Code: EE461**

**Assignment**

**PREPARED BY**

**Khandoker Samiul Hoque**

**Student Id: 19837**

**(a)**

**Question No Answer: 01**

module helloWorld;

initial begin

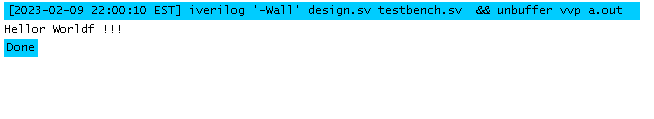
$display ("Hello\r World\f !!!");

#10 $finish;

end

endmodule

Here is the following code but in the observation in eda player I find it that it remain work as character and do not function, as it should be and it does not work here. It is ignored here. Result is below



**Question No Answer: 02**

**module keywordVariable;**

**integer begin = 1;**

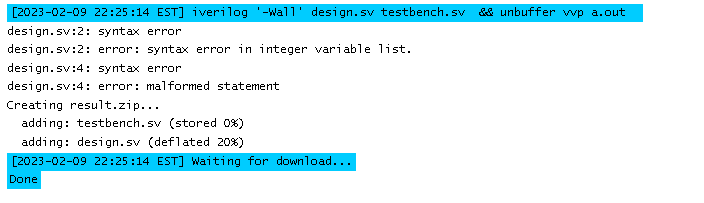
**initial begin**

**$display("Value of begin: %d", begin);**

**end**

**endmodule**

**Error result:**

****

Here in Verilog “begin” use as a dedicated keyword that is why we cannot use it as variable although for that reason I am having error in my code.

**Question No Answer: 03**

&$$abc\_123, in Verilog it does not work like that it must start with The first character must start with \_ or a-z/A-Z, like \_a-zA-Zxxxx. And then the rest of them must be \_ or a-z/A-Z/0-9 or $

So correction will be :

module abc\_123 ( );

This module will pass the compilation without having an error

**Question No Answer: 04**

module triduladevicevariablee;

tri tri\_var;

assign tri\_var = 1'b1;

wire device2 = 1'bz;

assign tri\_var= device2;

initial begin

$monitor("At time %0t, tri\_variable = %b", $time, tri\_var);

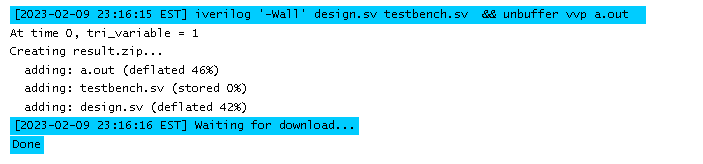
#10;

$finish;

end

endmodule

**output result :**

****

**Question No Answer: 05**

module triOR (input x, y, u, output z);

assign z = x | y | u;

endmodule

module triAND (input x, y, u, output z);

assign z = x & y & u;

endmodule

module testbench;

reg x, y, u;

wire z1, z2;

triOR dut1 (x, y, u, z1);

triAND dut2 (x, y, u, z2);

initial begin

for (x=0; x<2; x=x+1)

for (y=0; y<2; y=y+1)

for (u=0; u<2; u=u+1)

begin

#10 $display("x=%b y=%b u=%b z1=%b z2=%b", x, y, u, z1, z2);

end

endmodule

here I execute triOR and triAND in the code and it defines all 16 combination of a, b and c and result has been displayed although with the truth table it define the whole result accordingly.

**Question No Answer: 06**

module tri\_4\_val();

tri0 x;

tri1 y;

reg [3:0] input\_data;

initial begin

input\_data = 4'b0000;

#1;

input\_data = 4'b0001;

#1;

input\_data = 4'bxzx;

#1;

input\_data = 4'bzzz;

#1;

$finish;

end

assign x = input\_data[0];

assign y = input\_data[1];

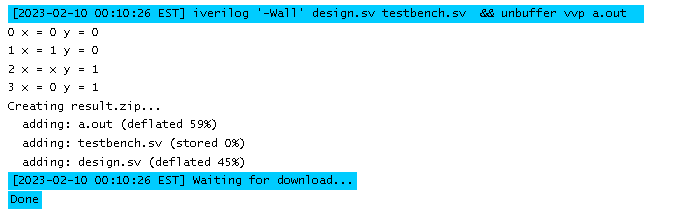
initial begin

$monitor("%g x = %b y = %b", $time, x, y);

end

endmodule

result :



**Question No Answer: 07**

module testTrireg ();

trireg[7:0] data;

reg[1:0] flag;

assign data = (flag==1)? 10 :

(flag==0)? 8’bzz:

(flag==3)? 30 : 255;

initial begin

flag = 1;

#200 flag=0;

#200 flag=3;

#200 flag=0;

#200 flag=2;

#200 flag=0;

$monitor(“Time=%g, data=%d”, $time, data);

#10 $finish;

end

endmodule

in the output it shows trireg nets not supported.

**Question No Answer: 08**

module testInteger();

wire pwrGood, pwrOn, pwrStable;

integer i;

time t;

real r;

assign pwrStable = 1'b1;

assign pwrOn = 1; // 1 or 1'b1

assign pwrGood = pwrOn & pwrStable;

initial begin

i=123.456;

r=123456e-3;

t=123456e-3;

$display("i=%0g",i," t=%6.2f",t," r=%f",r);

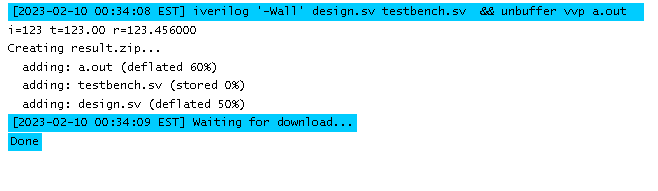
// #2 $display("TIME=%0d",$time," ON=",pwrOn," STABLE=",pwrStable,“ GOOD=",pwrGood);

#10 $finish();

end

endmodule

Result:



**Question No Answer: 09**

module stimerealtime();

time t;

reg clk;

initial begin

#10 clk = 0;

#10 clk = 1;

end

always @(posedge clk)

t = $realtime;

initial

begin

#100

$display("S time: %0d, Real time: %0d", $time, t);

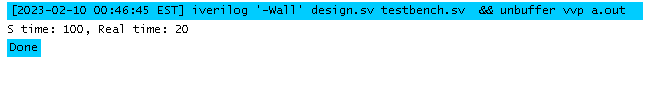
#100

$finish;

end

endmodule

**Result:**



Here simulation time is 100 and real time is 20 .

**Question No Answer: 10**

module comparison\_Real\_Display();

time real\_time, display\_time;

initial begin

real\_time = $realtime;

#2.71828 display\_time = $time;

$display("Real time: %t", real\_time);

$display("Display time: %t", display\_time);

$finish;

end

endmodule

**Result :**

Real time: 0  
Display time: 3

one simulation unit is equivalent to 100 picoseconds in real time. Real delay is 2.71828 \* 10ns = 27.1828ns . it should be 3+ 27.1828ns .

**Question No Answer: 11**

module signedNumber;

reg [31:0]

a;

initial begin

a = 14'h1234;

$display ("Current Value of a = %h", a);

a =

14'h1234;

$display ("Current Value of a = %h", a);

a = 32'hDEAD\_BEEF;

$display ("Current Value of a = %h", a);

a =

32'hDEAD\_BEEF;

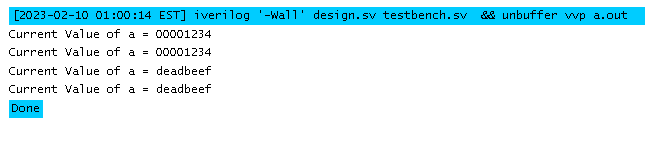
$display ("Current Value of a = %h", a);

#10 $finish();

end

endmodule

**Result:**

****

**Question No Answer: 12**

module helloWorld;

initial begin

$display("ASCII code for \" (double quote) character: %d", 34);

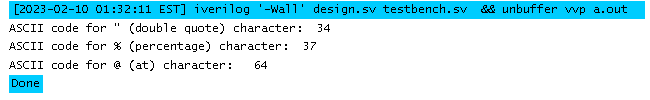
$display("ASCII code for %% (percentage) character: %d", 37);

$display("ASCII code for @ (at) character: %d", 64);

end

endmodule

**Result:**

****

**Question No Answer: 13**

module display\_formats;

reg [7:0] a = 8'hF0;

reg [3:0] b = 4'h3;

integer c = 65;

real d = 3.14159265358979323846;

initial begin

$display("a in binary: %b", a);

$display("a in character: %c", a);

$display("b in decimal: %d", b);

$display("b in decimal with zero padding: %0d", b);

$display("c in ASCII: %c", c);

$display("d in scientific notation: %e", d);

$display("d in floating-point: %f", d);

$display("d with 6 total digits, 2 after the decimal point: %6.2f", d);

$display("d in shortest representation: %g", d);

$display("a in hexadecimal: %h", a);

$display("a in octal: %o", a);

$display("b in binary: %b", b);

$display("b in decimal: %d", b);

$display("b in hexadecimal: %h", b);

$display("b in binary, with a width of 8 bits: %08b", b);

$display("b in binary, with a width of 4 bits: %04b", b);

end

endmodule

**Result:**

a in binary: 11110000

a in character:

b in decimal: 3

b in decimal with zero padding: 00000003

c in ASCII: A

d in scientific notation: 3.141593e+00

d in floating-point: 3.141593

d with 6 total digits, 2 after the decimal point: 3.14

d in shortest representation: 3.14159

a in hexadecimal: f0

a in octal: 360

b in binary: 0011

b in decimal: 3

b in hexadecimal: 3

b in binary, with a width of 8 bits: 00000011

b in binary, with a width of 4 bits: 0011

**Question No Answer: 14**

module compare;

reg [7:0] m = 8'hF0;

reg [3:0] n = 4'h3;

initial begin

$display("$display system task: m = %h, n = %d", m, n);

$write("$write system task: m = %h, n = %d", m, n);

$monitor("$monitor system task: m = %h, n = %d", m, n);

end

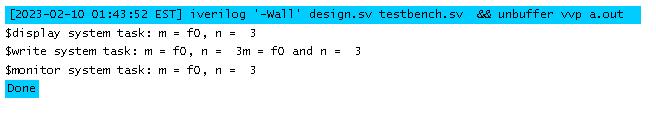
always @(m or n) begin

$display("m = %h and n = %d", m, n);

end

endmodule

**Result:**

****

Github : <https://github.com/KhandokerSamiulHoque/ee461-assignment-01-19837.git>