**Homework 1**

Ashmitha Murali

Prof. Alex Yang

Digital Design and HDL EE461

16th September 2025

**Q1.**

* **Code for White space(tab):**

module hello();

initial begin

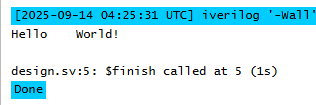
$display(&quot;Hello\t World!\n&quot;);

#5 $finish();

end

endmodule

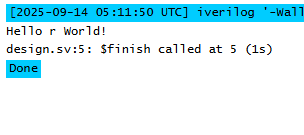
**Result:**



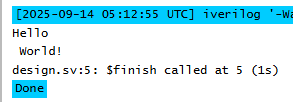
* **Result for Carriage return:**

I tried both \r and \x0d for Carriage return:

With \r it printed the character ‘r’

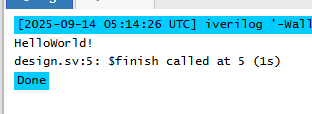


With \x0d it added a new line instead of returning to the current line.



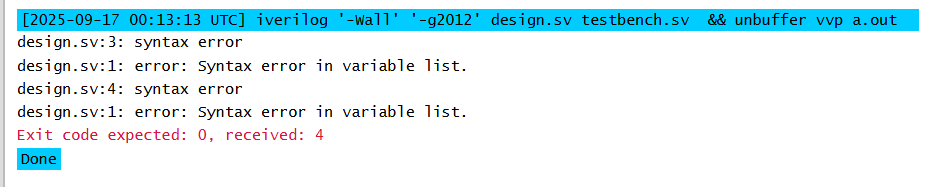
* **Result for Formfeed**:

I tried both \f and \x0c. Neither of them added any space in the terminal.



**Q2.**

Below is the result while using a keyword as a variable. I got a syntax error:



**Code used:**

module Q2;

integer input;

reg inout;

endmodule

**Q3.**

To make a module with the module name “&$$abc\_123” to compile, I added an **escape character(\)** at the start of the name and **a space** at the end of the module name. Below is an example of the module that compiles successfully:

**Code used:**  
module \&$$abc\_123 ;

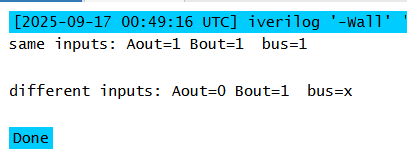
integer int1;

reg reg1;

endmodule

**Q4.**

I wrote a module using the tri variable and tested it with different values. Below is the result and Verilog code.



When driving with the same input (1,1), the tristate bus was assigned the value of these inputs.

When driving with different inputs(1,0), the tristate bus was assigned to **X**(unknown).

**Code used:**module triState;

tri bus;

reg Aout, Bout;

assign bus = Aout;

assign bus = Bout;

initial begin

Aout = 1'b1;

Bout = 1'b1;

$display("same inputs: Aout=%b Bout=%b bus=%b\n", Aout, Bout, bus);

Aout = 1'b0;

Bout = 1'b1;

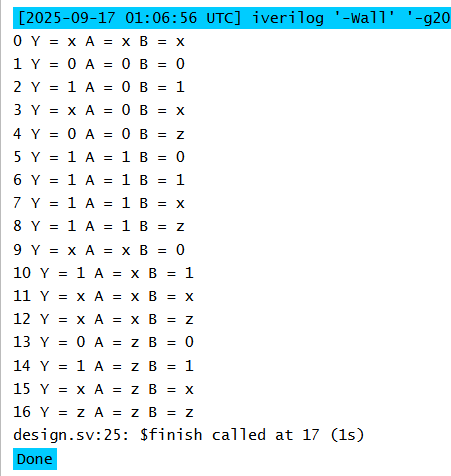
$display("different inputs: Aout=%b Bout=%b bus=%b\n", Aout, Bout, bus);

end

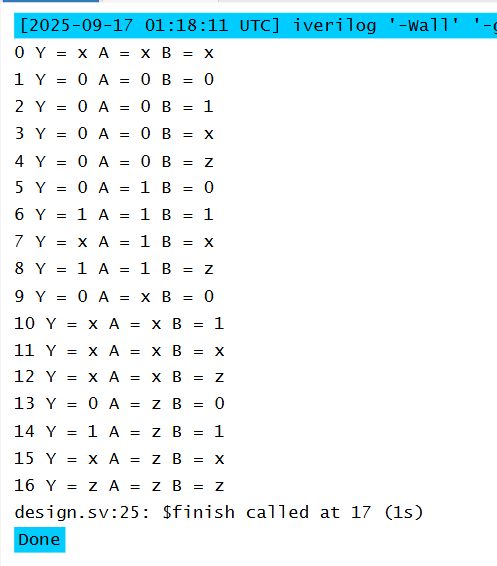
endmodule

**Q5.**

Result for trior/wor:



Result for triand/wand:



**Code used:**

module TestWor();

wor Y;

reg A, B;

assign Y = A;

assign Y = B;

initial begin

$monitor("%g Y = %b A = %b B = %b", $time, Y, A, B);

#1 A = 1'b0; B = 1'b0;

#1 A = 1'b0; B = 1'b1;

#1 A = 1'b0; B = 1'bx;

#1 A = 1'b0; B = 1'bz;

#1 A = 1'b1; B = 1'b0;

#1 A = 1'b1; B = 1'b1;

#1 A = 1'b1; B = 1'bx;

#1 A = 1'b1; B = 1'bz;

#1 A = 1'bx; B = 1'b0;

#1 A = 1'bx; B = 1'b1;

#1 A = 1'bx; B = 1'bx;

#1 A = 1'bx; B = 1'bz;

#1 A = 1'bz; B = 1'b0;

#1 A = 1'bz; B = 1'b1;

#1 A = 1'bz; B = 1'bx;

#1 A = 1'bz; B = 1'bz;

#1 $finish;

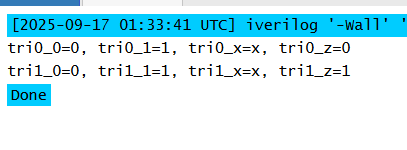
end

endmodule

**Q6.**

With tri0 and tri1, I am getting the assigned value for inputs(0, 1, X), but for **Z and unassigned**

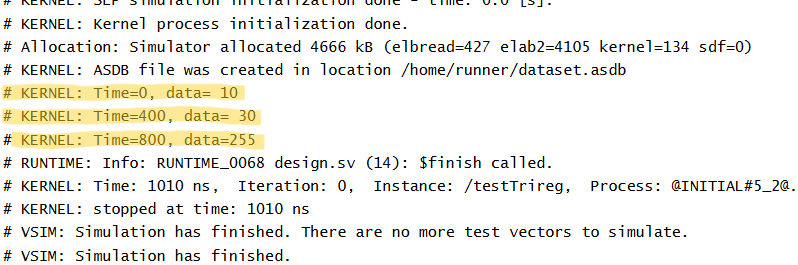
**values**, I am getting **0 for tri0** and **1 for tri1**.



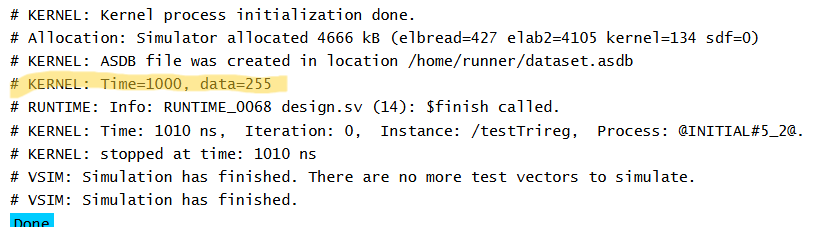
**Q7.**

Trireg was **not supported on Icarus** Verilog 12. I used **Aldec Riviera Pro** for this code:

I tested by adding ‘monitor’ at the start of the initial block to see how it works:

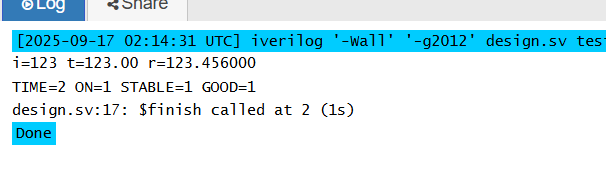


I tested the code with the ‘monitor’ at the end of the initial block to see the final state of the data:



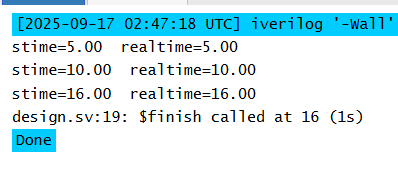
**Q8.**

Result for testInteger.



**Q9.**

By assigning $stime and $realtime to ‘time’ type the values get converted into integer as shown below:



**Code used:**

module time\_demo;

time t1, t2;

initial begin

#5.25;

t1 = $stime;

t2 = $realtime;

$display("stime=%2.2f realtime=%2.2f", t1, t2);

#5.25;

t1 = $stime;

t2 = $realtime;

$display("stime=%2.2f realtime=%2.2f", t1, t2);

#5.5;

t1 = $stime;

t2 = $realtime;

$display("stime=%2.2f realtime=%2.2f", t1, t2);

$finish;

end

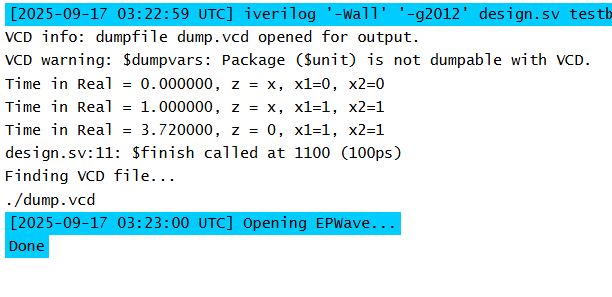
endmodule

**Q10.**

The real delay calculated is **27.1828 ns** since the time unit is 10ns (i.e., 2.71828\*10ns).

The displayed delay is **27.2ns** [(3.72-1)\*10] since we are using a rounding precision of 100ps.

The difference is **0.0172ns or 17.2ps**



**Code used:**

`timescale 10ns/100ps

module sampleDesign();

reg x1,x2;

wire z;

nor #2.71828(z,x1,x2);

initial begin

$monitor("Time in Real = %0f, z = %0b, x1=%0b, x2=%0b", $realtime,z,x1,x2);

x1=1'b0; x2=1'b0;

#1 x1=1'b1;x2=1'b1;

#10 $finish();

end

initial begin

$dumpfile("dump.vcd");

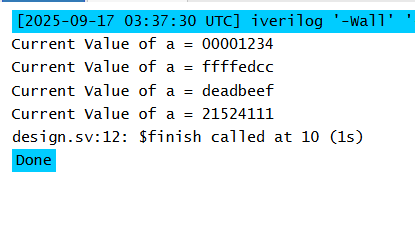
$dumpvars();

end

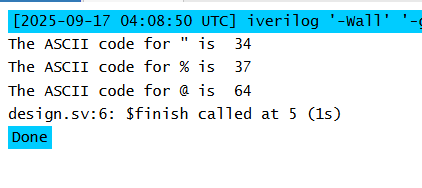
endmodule

**Q11.**

Result of signedNumber



**Q12.**



**Code used:**

module Ascii\_num();

initial begin

$display("The ASCII code for %c is %d", 8'd34, 8'd34);

$display("The ASCII code for %c is %d", 8'd37, 8'd37);

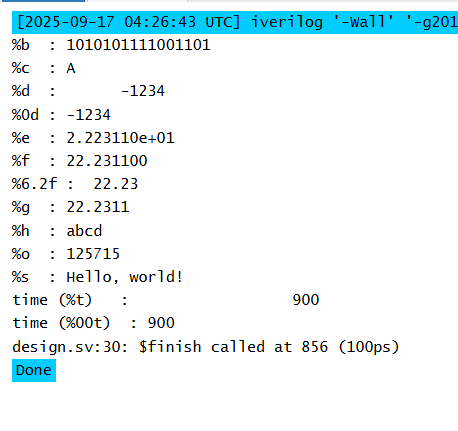
$display("The ASCII code for %c is %d", 8'd64, 8'd64);

#5 $finish();

end

endmodule

**Q13**



**Code used:**

`timescale 10ns/100ps

module display\_formats();

reg [15:0] hex\_val = 16'hABCD;

integer i = -1234;

integer ch = "A";

time t;

real r = 22.2311;

reg [103:0] s = "Hello, world!";

initial begin

#8.56;

t = $time;

$display("%%b : %b", hex\_val);

$display("%%c : %c", ch);

$display("%%d : %d", i);

$display("%%0d : %0d", i);

$display("%%e : %e", r);

$display("%%f : %f", r);

$display("%%6.2f : %6.2f", r);

$display("%%g : %g", r);

$display("%%h : %h", hex\_val);

$display("%%o : %o", hex\_val);

$display("%%s : %s", s);

$display("time (%%t) : %t", t);

$display("time (%%00t) : %00t", t);

$finish;

end

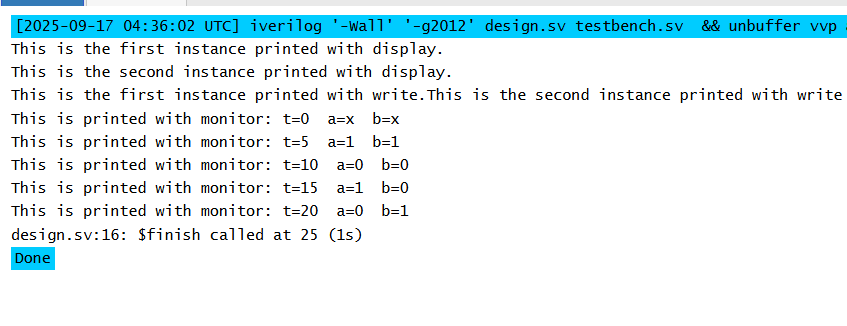
endmodule

**Q14**

As seen below, $display automatically adds a new line.

$write does not add a new line automatically and needs to be explicitly added.

$monitor automatically prints when one of the variables in the monitor block changes.



**Code used:**

module Ascii\_num();

reg a,b;

initial begin

$display("This is the first instance printed with display.");

$display("This is the second instance printed with display.");

$write("This is the first instance printed with write.");

$write("This is the second instance printed with write\n");

$monitor("This is printed with monitor: t=%0t a=%0d b=%0d", $time, a, b);

#5 a = 1; b=1;

#5 a = 0; b=0;

#5 a = 1; b=0;

#5 a = 0; b=1;

#5 $finish();

end

endmodule