What will be the output

Assign y = 4’b1010 & 4’b0110;

It is a bitwise and operator. So multiply corresponding bits. (starts from Lsb to Msb) 0.0=0;1.1=1; 0.1=0; 1.0=0.

RESULT=0010.

Assign y= 4’b0010 && 4’b0110;

This is logical and operator. Here take total bit considered as a num don’t take individual bits.

Here two bits are more than one so the output is true.

RESULT=4’b0001

What will be the output in VHDL

Y<=4’b0010 & 4’b0110;

In vhdl the &operator acts as a concatenation operator.

RESULT= 8’b00100110;

What will be the output in verilog

Assign y=&(8’b11110111);

Here the & operator acts as a unary and operator. The operation is multiply all bits but the output of 1st two bits multiply with the third variable. The shifting is moved up to end of bits.

RESULT =8’b00000000;

what is equivalent operator in vhdl for unary AND operator

there is no equivalent operator for unary and, nor or operator.

What will be the output in verilog

reg X1=32’b1111;

display(“~X1=%b”,~X1);

it is a bitwise not operator. It contains total 32 bits. All bits are should be reversed. The hidden bits are 28 bits all are 0’s. now convert it becomes 1’s. the visual bits are 4 one’s now convert becomes 4 zero’s.

RESULT=11111111111111111111111111110000

reg X1=32’b1111;

display(“!X1=%b”,!X1);

It is a logical not operator. Here the output is only one bit. The visual bits only inversion. Remaining all bits are 0’s only no need to change.

RESULT !X1=0;

what is equivalent operator in vhdl for unary Not (~)operator

NOT Operator

Y<=NOT A;

All input bits are inverted.

what is equivalent operator in vhdl for Logical Not (!)operator

NOT Operator.

Y<=not A; it will acts as bitwise inverter(~)

Differences b/w $finish , $display, $monitor

**$display** executes when the statement is executed at the start of simulation.

**$monitor** is used to monitor a signal when it’s value changes (Values keep updating during simulation).

**$strobe** prints the values at the end of simulation.

module tp;  
reg i1;  
initial

$monitor("\$monitor: i1 = %b", i1);  
initial

begin  
$strobe ("\$strobe : i1 = %b", i1);  
i1 = 0;  
i1 <= 1;  
$display ("\$display: i1 = %b", i1);  
#1 $finish;

end  
endmodule

**Answer)** $display: i1 = 0  
$monitor: i1 = 1  
$strobe : i1 = 1

Differences b/w $stop and $finish

finish exits the simulation and gives control back to the operating system.

$stop suspends the simulation and puts a simulator in an interactive mode

module tb ;

int x =5 ;

initial begin

$display ("Hi",$time);

#10ns;

$stop ;

$display("Hi ",$time);

#20ns ;

$display("Hi",$time);

if(x ==5)begin

$error("this is Error");

$finish(1);

end

$display("After the $finish");

end

endmodule

Points to look : 1)So at 0ns, a "hi" message will be displayed after 10ns $stop will be called & it suspends the simulation.

2)now if we simply comment out $stop, then re-run our simulation we will observe that at 30 ns again hi message will be printed and if the condition will be stratified and $finish will be called so it will terminate the simulation, so after $finsh the Display message "After the $finish" will not be displayed.