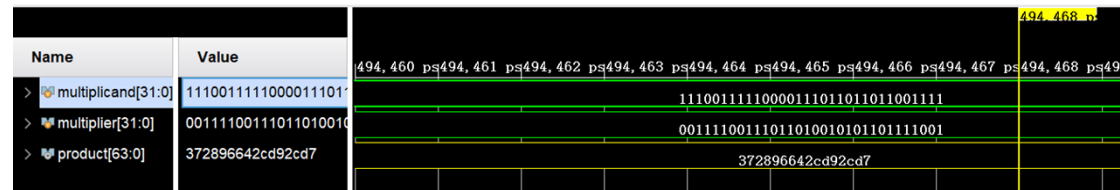


参照循环语句中的例子，完成一个 32 位二进制数的乘法

```
22 module multiply32(  
23     input [31:0]multiplicand,  
24     input [31:0]multiplier,  
25     output reg [63:0]product  
26 );  
27     integer i;  
28     always @(multiplicand or multiplier)  
29     begin  
30         product=64'b0;  
31         for(i=0;i<32;i=i+1)  
32         begin  
33             if (multiplier[i]==1'b1)  
34             begin  
35                 product=product +(multiplicand <<i);  
36             end  
37         end  
38     end  
39 endmodule  
40  
23 module multiplytb();  
24     reg [31:0]multiplicand,multiplier;  
25     wire [63:0]product;  
26     multiply32 op(multiplicand,multiplier,product);  
27     initial begin  
28         multiplicand=32'b0000_0000_0000_0000_0000_0000_0000;  
29         multiplier =32'b0000_0000_0000_0000_0000_0000_0000;  
30     end  
31     always #5 multiplicand=$random%33'b1_0000_0000_0000_0000_0000_0000_0000;  
32     always #5 multiplier=$random%33'b1_0000_0000_0000_0000_0000_0000_0000;  
33 endmodule
```

multiply32.v	multiplytb.v	Untitled 10*
		
Name	Value	
> multiplicand[31:0]	11100111110000111011011011001111	494,460 ps
> multiplier[31:0]	00111100111011010010101101111001	494,461 ps
> product[63:0]	372896642cd92cd7	494,462 ps

multiplicand[31:0] = 11100111110000111011011011001111 (二进制)

multiplier[31:0] = 00111100111011010010101101111001 (二进制)

转为十进制: multiplicand = 3967854847 multiplier = 999999999

product_decimal = 3967854847 * 999999999 = 3967854843032145153

product_hex = 0x372896642cd92cd7

仿真给出的结果是: product[63:0] = 0x372896642cd92cd7

可见结果正确

总结:

在本次实验中，我成功实现了 32 位二进制数的乘法运算。通过参照循环语句中的例子，我们构建了一个基于 Verilog 的乘法器模块 multiply32，该模块利用移位和累加的方式模拟了多位乘法的过程。测试平台 multiplytb 引入了随机生成的测试数据以确保模块的鲁棒性和通用性。这次实验让我深刻体会到细节的重要性，以及如何通过系统性的方法来验证复杂电路的功能。