2211290 姚知言 Verilog 作业 4

1.请自行设计一个模块，完成统计 32 位 2 进制数 0 和 1 出现的次数。

count.v 设计：

```
`timescale 1ns / 1ps
module count(
    input [31:0] innum,
    output [5:0] cnt0,
    output [5:0] cnt1
    );
    reg [5:0] c1,i;
    always @(*)
    begin
        c1 = 0;
        for (i = 0; i < 32; i = i + 1)
        begin
            if (innum[i] == 1'b1)
                c1 = c1 + 1;
            end
    end
    assign cnt0 = 6'd32 - c1;
    assign cnt1 = c1;
endmodule
```

设计思路：

模块有一个输入和两个输出，输入为 32 位 2 进制数 innum，也是我们要分析的数串。

输出是两个 6 位 2 进制数 cnt0 和 cnt1，分别为 0 出现的次数和 1 出现的次数。

使用 6 位 2 进制数以保证能够保证 0-32 的所有数。

设计一个 always 块，每当任意数值发生变化时执行一次，根据当时的 innum 遍历每一位，统计 1 的个数，更新寄存器 c1，并以此更新 cnt0 和 cnt1。

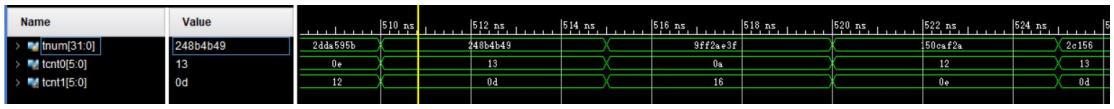
testbench.v 设计：

```
`timescale 1ns / 1ps
module testbench();
reg [31:0]tnum;
wire [5:0] tcnt0,tcnt1;
count mycnt(tnum,tcnt0,tcnt1);
initial begin
    tnum = 32'b0;
end
always #5 tnum=$random;
endmodule
```

设计思路：

寄存器 tnum 初始化为 0，每 5 秒用 random 生成 32 位随机二进制数将其刷新，调用 count 模块，计算 cnt0，cnt1，存入 tcnt0 和 tcnt1 中，进行验证。

测试结果：



任举两例分析：

510ns：248b4b49 => 0010 0100 1000 1011 0100 1011 0100 1001

共有 19（0x13）个 0，13（0x0d）个 1。

515ns：9ff2ae3f => 1001 1111 1111 0010 1010 1110 0011 1111

共有 10（0x0a）个 0，22（0x16）个 1。通过验证。

2.请使用 always 块语句，实现一个十分频器，divider10(input clk_in,input reset, output count, output clk_out)。其功能可以理解为将时钟降频为原来的 10 分之一（思路：对 clk_in 进行 count 计数，count 取值 0~9，count

数到 5 时，clk_out 由 1 变 0，count 数到 10 时自动归零同时 clk_out 由 0 变 1）。

divider10.v 设计：

```
`timescale 1ns / 1ps
module divider10(
    input clk_in,
    input reset,
    output reg [3:0] count,
    output reg clk_out
   );
    always @(posedge clk_in) begin
        if (reset) begin
            count <= 4'd0;
            clk_out <= 1'b0;
        end else begin
            if (count == 4'd4 || count == 4'd9) begin
                clk_out <= ~clk_out;
            end
            count <= (count + 1'd1) % 4'd10 ;
        end
    end
endmodule
```

设计思路：

每个 clk_in 上升沿执行 always 语句块，若 reset 为 1，count 和 clk_out 清零。否则，count+1%10，同时若 count 为 4 或 9 翻转 clkout(clk_out 会和 count 的 5/0 一起到来)。

testbench.v 设计：

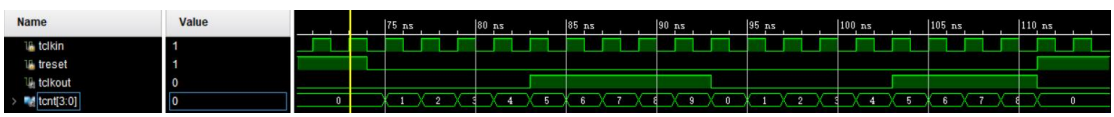
```
`timescale 1ns / 1ps
module testbench();
reg tclkin,treset;
wire tclkout;
wire [3:0] tcnt;
divider10 mydiv(tclkin,treset,tcnt,tclkout);
initial begin
    tclkin = 1'b0;
    treset = 1'b1;
    #2;
    treset = 1'b0;
end
always #1 tclkin = ~tclkin;
always #37 treset = ~treset;
endmodule
```

设计思路：

设计为 treset 先设为 1，2 周期后设为 0，tclkin 设为 0，这样保证更好的冷启动。

随后每 1 周期翻转 tclkin，每 37 周期翻转 treset，进行验证。

测试结果：



reset 验证：reset 为 1 的时候，tclkout 为 0，tcnt 置 0，不受 tclkin 影响。reset 为 0 的时候，当 tcnt 数到 5 的时候 tclkout 置为 1，数到 10(0)的时候 tclk 置为 0。图中 74ns treset 置 0，随每 ns 翻转可以看到 tcnt 正常计数，到 5 和 0 翻转，直到 111ns treset 置 1 清零，通过验证。