梦幻呼吸灯实验

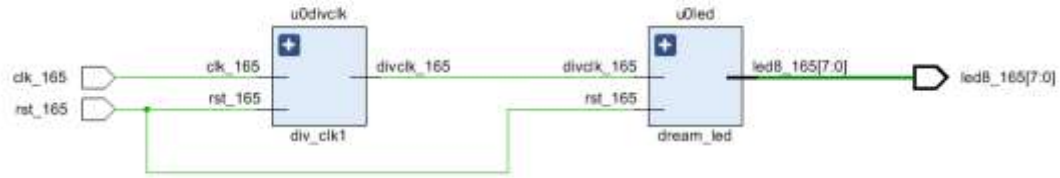
本实验包括基本实验部分和改进实验部分(梦幻呼吸灯)

一、基本实验

1、顶层模块

top.v

```
module top(
input rst_165,
input clk_165,
output[7:0] led8_165
      );
      wire divclk_165;
      div_clk1
u0divclk(.rst_165(rst_165),.clk_165(clk_165),.divclk_165(divclk_165));
      dream_led
u0led(.rst_165(rst_165),.divclk_165(divclk_165),.led8_165(led8_165));
endmodule
```



图一、详细描述后的结果

顶层模块中包括 div_clk1 分频时钟模块、dream_led 梦幻 LED 灯设计部分。两部分的连接关系如图一所示，100MHz 的时钟信号 clk_165 分频成 divclk_165 作为 dream_led 的模块的时钟输入信号，驱动 led8_165 实现不同的输出效果。

2、分频时钟模块

div_clk.v

```
`timescale 1ns / 1ps
module div_clk1(input clk_165,input rst_165,output reg divclk_165);
reg[31:0] counter_165;
always @(posedge clk_165 or posedge rst_165)
begin
      if(rst_165)
            begin
              counter_165<=32'h00000000;
            divclk_165<='h0;
              end
        else
            begin
            if(counter_165==32'h02faf07f)
```

```
                begin
                  counter_165<=32'h00000000;
                  divclk_165<=~divclk_165;
                end
            else
               counter_165<=counter_165+1;
          end
end
endmodule
```

由于 FPGA 中提供的时钟引脚是 100MHz,要实现梦幻 LED 灯在每 1s 发生一次变化，要对时钟进行分频。

T=1s $f_{输入} = 100MHz$

$f_{输出}=\frac{1}{T}$=1Hz

分频因子计算入下：

$$\frac{\frac{f_{输入}}{f_{输出}}}{2}-1=N$$

计算求得 N=49999999，其十六进制表示为 N=02FAF07F

3、梦幻 LED 控制文件 dream_led.v

```
`timescale 1ns / 1ps
module dream_led(
input divclk_165,
input rst_165,
output reg [7:0] led8_165
      );
reg[5:0] counter_165;
always@(posedge rst_165 or posedge divclk_165)
begin
      if(rst_165)
         counter_165<=0;
      else if(counter_165==18)
         counter_165<=0;
      else
          counter_165=counter_165+1;
end
always @*
      begin
            case(counter_165)
              5'b00000:   led8_165=8'b11111111;
              5'b00001:   led8_165=8'b00000000;
              5'b00010,5'b10000:   led8_165=8'b00000001;
              5'b00011,5'b01111:   led8_165=8'b00000010;
```

```
            5'b00100,5'b01110:    led8_165=8'b00000100;
            5'b00101,5'b01101:    led8_165=8'b00001000;
            5'b00110,5'b01100:    led8_165=8'b00010000;
            5'b00111,5'b01011:    led8_165=8'b00100000;
            5'b01000,5'b01010:    led8_165=8'b01000000;
            5'b01001: led8_165=8'b10000000;
            5'b10001:led8_165=8'b01010101;
            5'b10010:led8_165=8'b10101010;
              default:led8_165='b00000000;
          endcase
    end
endmodule
```

设计思路：

- 用 reg 型的 led8_165 定义 8 个 LED 的亮和灭。
- 运用模为 19 的计数器计数，根据计数器每一秒计的不同的数字，case 语句进行判断，执行不同情况下的 LED 的不同变化。Case 语句并行触发，执行速度会比较快。
- 计数器从 0~18 循环计数，使 8 个 LED 周而复始地在 19 中状态下不断地切换。

4、引脚约束文件

top.xdc

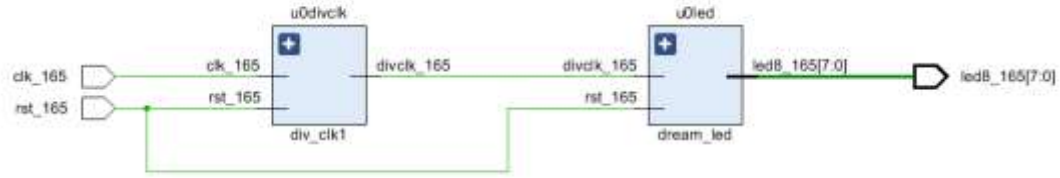
```
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk_165]
set_property IOSTANDARD LVCMOS33 [get_ports rst_165]
set_property PACKAGE_PIN J19 [get_ports clk_165]
set_property PACKAGE_PIN T5 [get_ports rst_165]
set_property PACKAGE_PIN Y17 [get_ports {led8_165[0]}]
set_property PACKAGE_PIN Y16 [get_ports {led8_165[1]}]
set_property PACKAGE_PIN AA16 [get_ports {led8_165[2]}]
set_property PACKAGE_PIN AB16 [get_ports {led8_165[3]}]
set_property PACKAGE_PIN AB17 [get_ports {led8_165[4]}]
set_property PACKAGE_PIN AA13 [get_ports {led8_165[5]}]
set_property PACKAGE_PIN AB13 [get_ports {led8_165[6]}]
set_property PACKAGE_PIN AA15 [get_ports {led8_165[7]}]
```

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
All ports (10)										
led8_165 (8)	OUT			✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[7]	OUT		AA15	✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[6]	OUT		AB13	✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[5]	OUT		AA13	✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[4]	OUT		AB17	✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[3]	OUT		AB16	✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[2]	OUT		AA16	✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[1]	OUT		Y16	✓	13	LVCMOS33*	3.300		12	SLOW
led8_165[0]	OUT		Y17	✓	13	LVCMOS33*	3.300		12	SLOW
Scalar ports (2)										
clk_165	IN		J19	✓	15	LVCMOS33*	3.300			
rst_165	IN		T5	✓	34	LVCMOS33*	3.300			

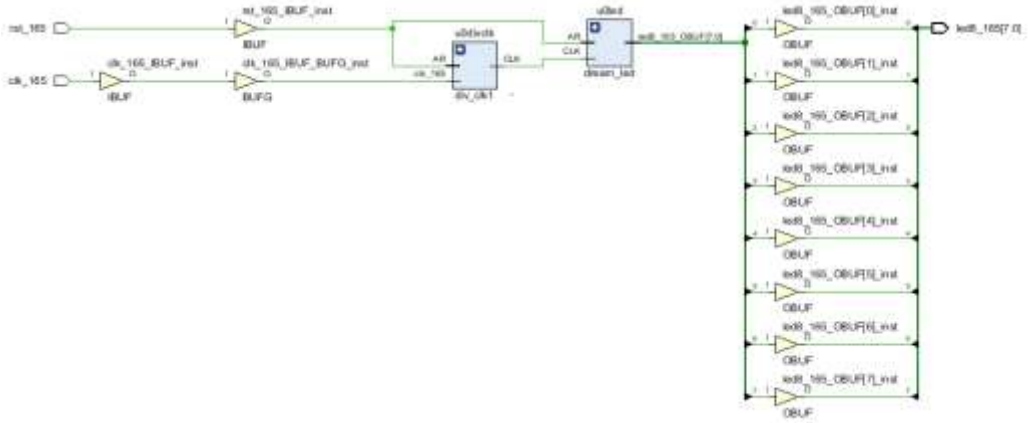
说明：

- 将 led8_165[0]、led8_165[1]、led8_165[2]、led8_165[3]、led8_165[4]、led8_165[5]、led8_165[6]、led8_165[7]输出引脚约束到 A7-EDP-1 开发平台上的 XC7A75TFGG484-1 器件的八个引脚上，以达到驱动 LED 的目的。
- 将 clk_165 约束到 100MHz 的时钟引脚上，将 rst_165 约束到开关 T5 引脚上。

详细描述后的结果



综合结果

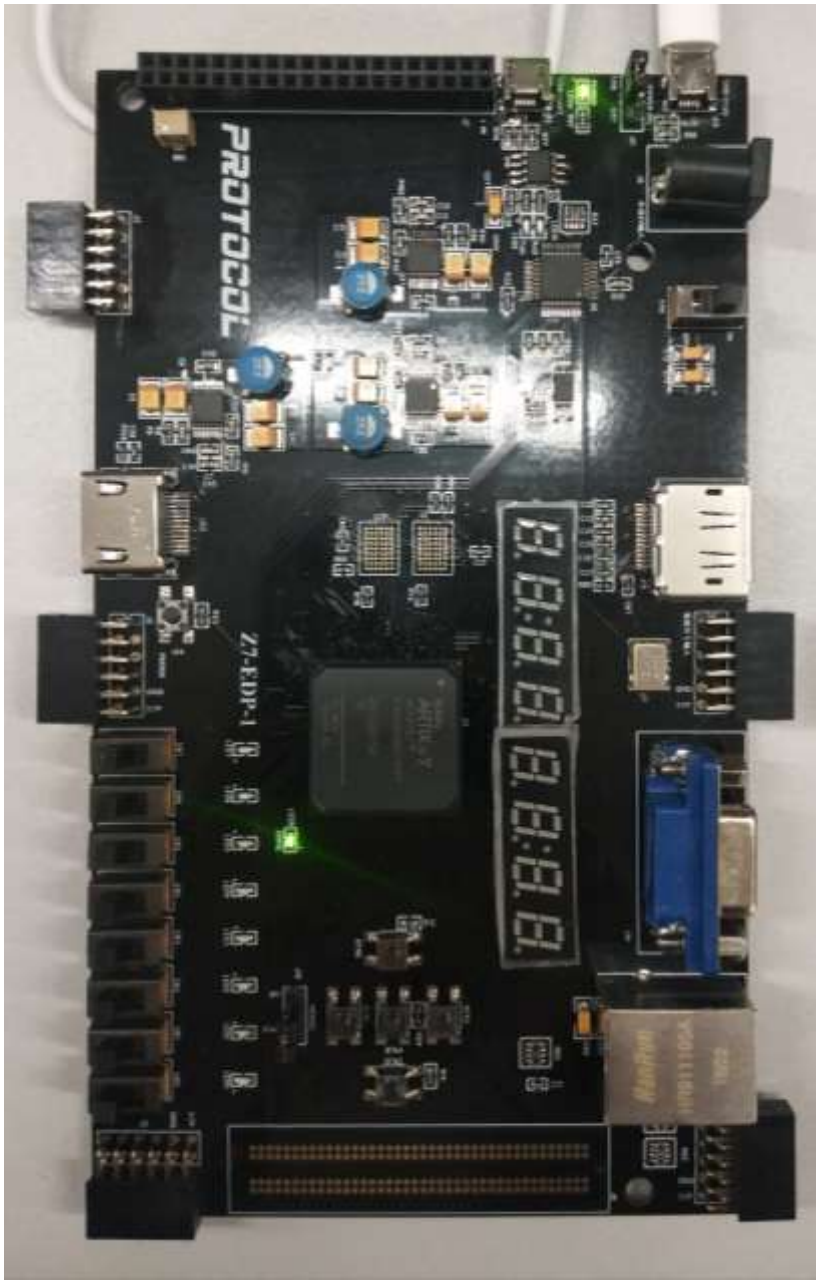


5、实物效果图

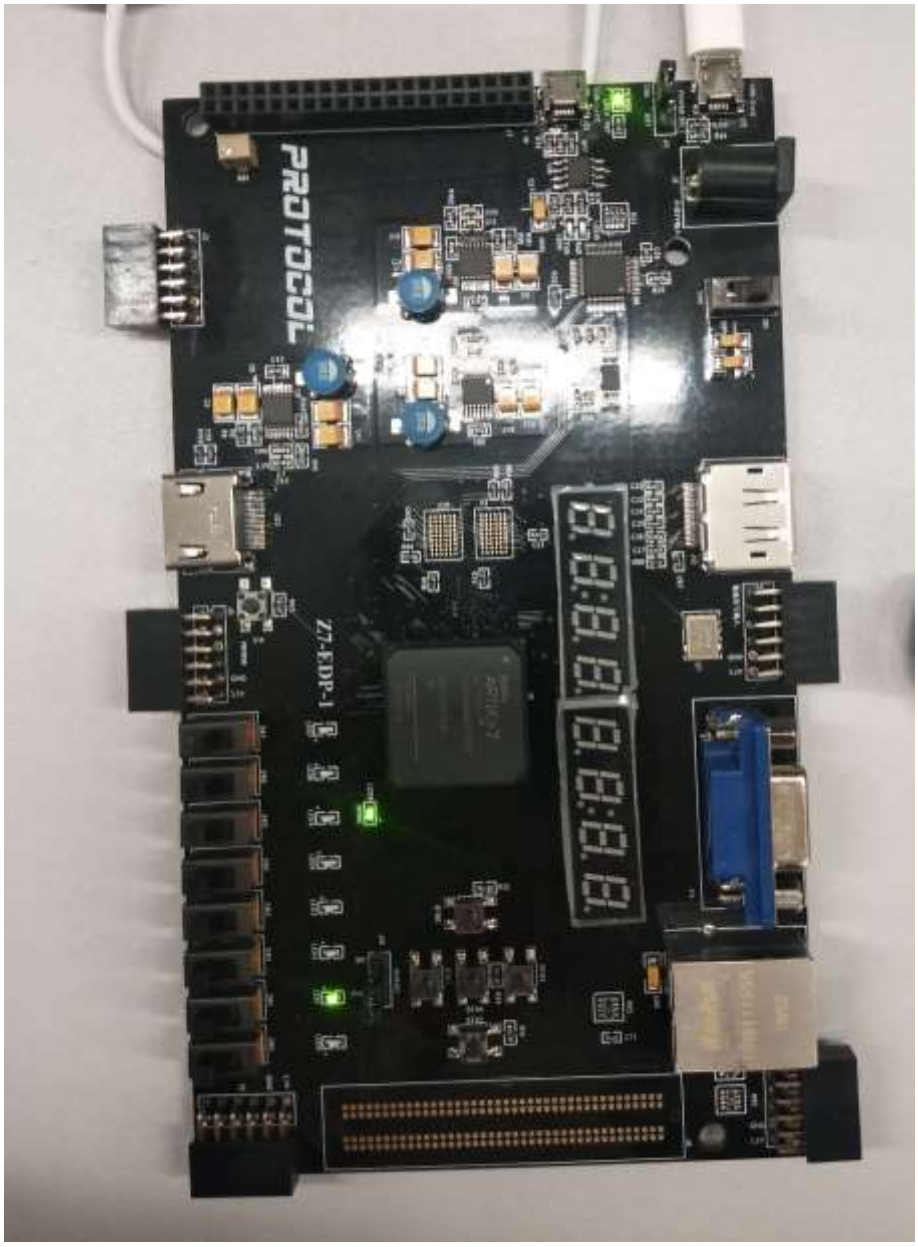
实现效果梦幻 LED 灯变化规律如下：

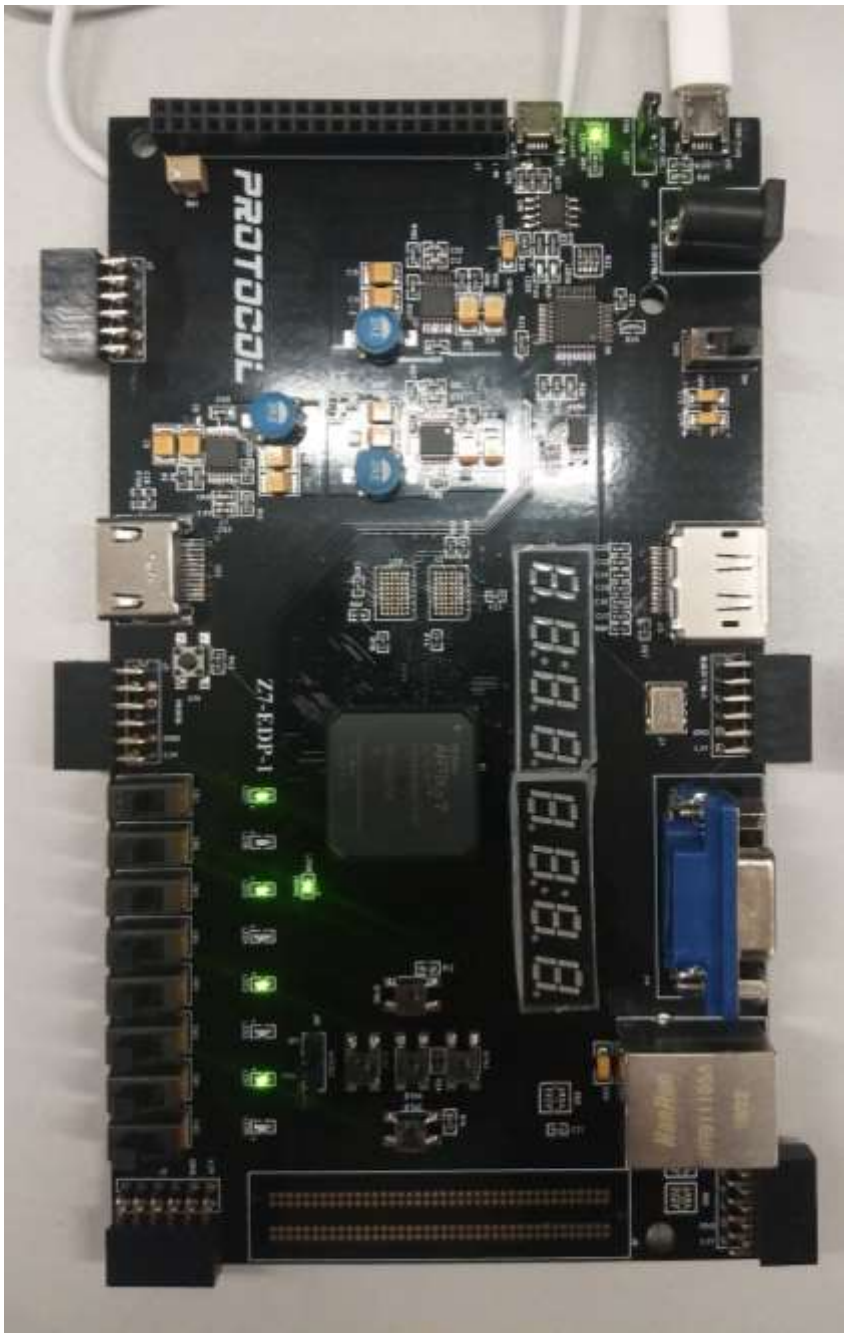
1) 第 1s,8 个 LED 全亮。
2) 第 2s,8 个 LED 全灭。
3) 第 3s 开始，LED 开始从右向左移动，当移动到最左边时。
4) LED 开始从最左向最右移动，当移动到最右边时。
5) 8 个 LED 其中 4 个 LED 间隔亮，剩下 4 个 LED 间隔灭。
6) 返回 1)，周而复始的变化。


PROTOCOL
Z7-EDP-1


PROTOCOL
Z7-EDP-1


PROTOCOL
Z7-EDP-1


PROTOCOL
Z7-EDP-1


PROTOCOL
Z7-EDP-1

二、发挥改进实验（将呼吸灯与梦幻 LED 融合—梦幻呼吸灯）

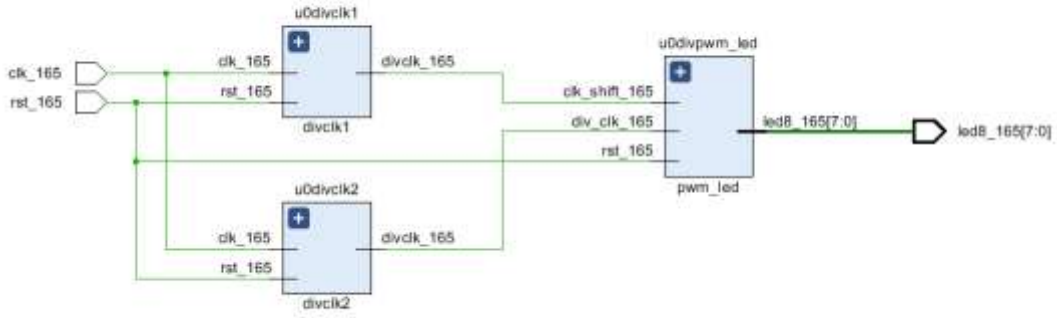
1、顶层设计模块 top. v

```
`timescale 1ns / 1ps
module top(
        input clk_165,
      input rst_165,
      output [7:0] led8_165
      );
```

```
    wire divclk1_165,divclk2_165;
    divclk1
u0divclk1(.clk_165(clk_165),.rst_165(rst_165),.divclk_165(divclk1_165));
    divclk2
u0divclk2(.clk_165(clk_165),.rst_165(rst_165),.divclk_165(divclk2_165));
    pwm_led
u0divpwm_led(.div_clk_165(divclk2_165),.clk_shift_165(divclk1_165),.rst_165(rs
t_165),.led8_165(led8_165));
endmodule
```

顶层模块中包括 div_clk1 分频时钟模块、divclk2 分频时钟模块、pwm_led 梦幻呼吸灯设计部分。将 clk_165 经过 divclk1 和 divclk2 时钟分频模块，进行分频后得到输出 div_165 分别作为控制 pwm 波占空比的时钟信号 div_clk_165 和控制 LED 灯亮灭不断变化的 clk_shift_165，驱动 pwm_led 模块实现梦幻呼吸灯的效果。

两部分的连接如图一所示。



图一、详细描述结果

2、分频文件

divclk1.v

```
`timescale 1ns / 1ps
module divclk1(input clk_165,input rst_165,output reg divclk_165);
reg[31:0] counter_165;
always @(posedge clk_165 or posedge rst_165)
begin
    if(rst_165)
        begin
          counter_165<=32'h00000000;
        divclk_165<='h0;
          end
     else
        begin
        if(counter_165==32'h02faf07f)    //02faf07f
            begin
              counter_165<=32'h00000000;
```

```
                divclk_165<=~divclk_165;
               end
            else
              counter_165<=counter_165+1;
          end
end
endmodule
```

由于 FPGA 中提供的时钟引脚是 100MHz,在每 1 秒中设置不同的 LED 灯亮和灭，对时钟进行分频。

T=1s $f_{输入} = 100MHz$

$f_{输出}=\frac{1}{T}$=1Hz

分频因子计算入下：

$$\frac{\frac{f_{输入}}{f_{输出}}}{2}-1=N$$

计算求得 N=49999999，其十六进制表示为 N=02FAF07F

divclk2.v

```
`timescale 1ns / 1ps
module divclk2(input clk_165,input rst_165,output reg divclk_165);
reg[31:0] counter_165;
always @(posedge clk_165 or posedge rst_165)
begin
      if(rst_165)
           begin
             counter_165<=32'h00000000;
           divclk_165<='h0;
            end
       else
           begin
           if(counter_165==32'h00001387)    //00001388
               begin
                 counter_165<=32'h00000000;
                 divclk_165<=~divclk_165;
               end
           else
              counter_165<=counter_165+1;
         end
end
endmodule
```

此分频文件设置呼吸灯的分频频率：

T=1s $N_1 = N_2 = 99$

$$f_{输出} = 1Hz$$

分频因子计算如下：

$$\frac{\frac{f_2}{f_{输出}}}{2}-1=N_2 \quad \frac{\frac{f_1}{f_2}}{2}-1=N_1$$

求得$f_1 = 10000Hz$

f=100MHz

$N=\frac{\frac{f}{f_1}}{2}-1=4999$，其十六进制是 1387。

3、呼吸梦幻灯设计文件

pwm_led.v

```
`timescale 1ns / 1ps
module pwm_led(
        input div_clk_165,
        input clk_shift_165,
        input rst_165,
        output reg [7:0] led8_165
        );
parameter period=99,period1=99;
parameter step=1;
parameter s0=1'b0,s1=1'b1;
reg [31:0] count_165;
reg signed [31:0] duty_165=0;
reg dir_165=1;
reg state_165;
reg pwm_165;

reg [5:0] i_165; //修改了位宽

initial i_165=0;
always @(posedge div_clk_165 or posedge rst_165)
begin
if(rst_165)
        begin
            count_165<=0;
            duty_165<=0;
            state_165<=s0;
        end

else if(count_165==period-1)   //计数器到达某一个值之后，呼吸灯开始变化

          begin
               count_165<=0;
                case (state_165)
```

```
            s0:   //由暗到亮
                  begin
                     duty_165<=duty_165+step;   //灯开始亮
                    if(duty_165>=period1) //到达最亮之后，要变暗
                      begin
                        state_165<=s1;
                        duty_165<=period1;
                      end
                  end
            s1:   //由亮到暗
                  begin
                    duty_165<=duty_165-step;
                     if(duty_165<=0) //到达最暗之后，要变亮
                       begin
                         duty_165<=0;
                         state_165<=s0;
                       end
                  end
            endcase
      end
else
          count_165<=count_165+1;   //否则，呼吸灯接着计数
end

always @ (*)
begin
   if(count_165<duty_165) //控制脉冲宽度
       pwm_165<=1;
   else
       pwm_165=0;
end

always @(posedge rst_165 or posedge clk_shift_165) //i 控制哪个灯亮
begin
 if(rst_165)
     begin
```

```
        i_165<=0;
        end
    else if(i_165==18)
        i_165<=0;
    else
         i_165=i_165+1;
end

always @ *
begin
   case (i_165)   //通过 i 控制点亮不同的灯
         5'b00000:   led8_165={8{pwm_165}};
         5'b00001:   led8_165={8{pwm_165}};
         5'b00010,5'b10000:   led8_165={7'b0000000,pwm_165};
         5'b00011,5'b01111:   led8_165={6'b000000,pwm_165,1'b0};
         5'b00100,5'b01110:   led8_165={5'b00000,pwm_165,2'b00};
         5'b00101,5'b01101:   led8_165={4'b0000,pwm_165,3'b000};
         5'b00110,5'b01100:   led8_165={3'b000,pwm_165,4'b0000};
         5'b00111,5'b01011:   led8_165={2'b00,pwm_165,5'b00000};
         5'b01000,5'b01010:   led8_165={1'b0,pwm_165,6'b000000};
         5'b01001: led8_165={pwm_165,7'b0000000};
         5'b10001:led8_165={4{1'b0,pwm_165}};
         5'b10010:led8_165={4{pwm_165,1'b0}};
         default:led8_165='b00000000;
    endcase
 end
endmodule
```

设计思路：

- count_165 是模为 99 的计数器，count_165 每循环完一次，duty_165 加或减 step，duty_165 也相当于一个模为 99 的计数器。通过 duty_165<count_165，则 pwm_165 为 1，否则 pwm_165 为 0。据此改变占空比，使占空比先依次不断增加，后依次不断减小。实现呼吸灯的效果。
- state_165=0 时，呼吸灯慢慢变亮，duty_165 慢慢变大，当 duty_165 达到最大（period1）的时候，此时 LED 灯最亮，将 state_165 置 1，使接下来的 LED 灯慢慢变暗，duty_165 慢慢变小，当 duty_165 减到 0 时，LED 灯最暗，之后将 state_165 置 0，如此循环，使 LED 灯不断由亮变暗再由暗变亮，再由亮变暗等等。
- 一开始的时候，duty_165=0 且 state_165=0，count_165 从 0 记到计数到 period-1 时（在此区间 pwm_165 一直为 0），duty_165 加 step，之后 count_165 再从 0 记到 period-1 期间，存在 count_165< duty_165 的时候，在这个时候 pwm_165 输出 1，驱动 LED 灯亮。随着 count_165 计数循环次数的不断增加，duty_165 不断变大，致使 count_165 每次循环期间，count_165< duty_165 的时间越来越长，从而改变了固定时间段内的占空比。同理，当 state_165=1

时的情况和此类似，故不再赘述。

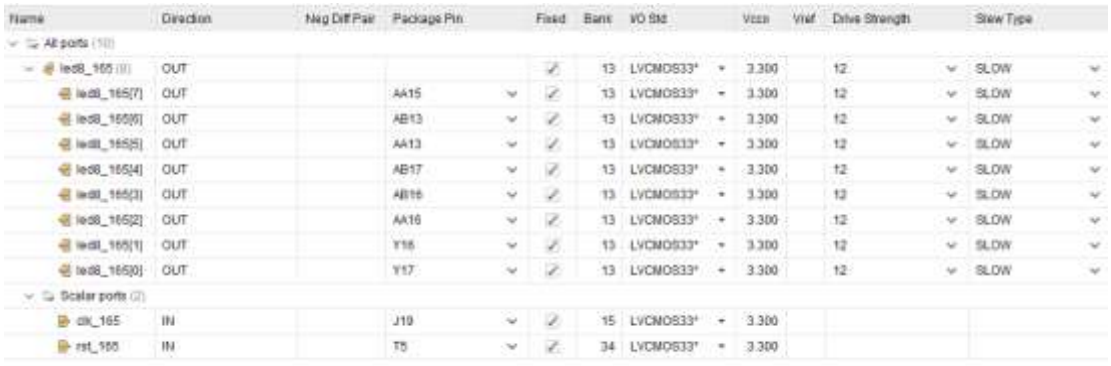
- 通过计数器 i_165 从 0 到 18 不断循环计数，控制 case 语句的那一句执行，实现 8 个 LED 灯亮和不亮状态的切换，实现梦幻灯的效果。
  第 1s 和第 2s 通过复制操作符，实现 8 个 LED 同时呼吸变亮和变暗。
  之后通过拼接操作符，实现 8 的 LED 流水呼吸效果。
  最后，通过复制操作符，实现 8 个 LED 间隔呼吸变亮和间隔呼吸变暗。

注：要实现梦幻呼吸灯的效果，count_165 和 i_165 计数的频率选择至关重要，在计算的时候比较困难。

4、约束文件

top.xdc

```
set_property PACKAGE_PIN Y17 [get_ports {led8_165[0]}]
set_property PACKAGE_PIN Y16 [get_ports {led8_165[1]}]
set_property PACKAGE_PIN AA16 [get_ports {led8_165[2]}]
set_property PACKAGE_PIN AB16 [get_ports {led8_165[3]}]
set_property PACKAGE_PIN AB17 [get_ports {led8_165[4]}]
set_property PACKAGE_PIN AA13 [get_ports {led8_165[5]}]
set_property PACKAGE_PIN AB13 [get_ports {led8_165[6]}]
set_property PACKAGE_PIN AA15 [get_ports {led8_165[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led8_165[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk_165]
set_property IOSTANDARD LVCMOS33 [get_ports rst_165]
set_property PACKAGE_PIN J19 [get_ports clk_165]
set_property PACKAGE_PIN T5 [get_ports rst_165]
```

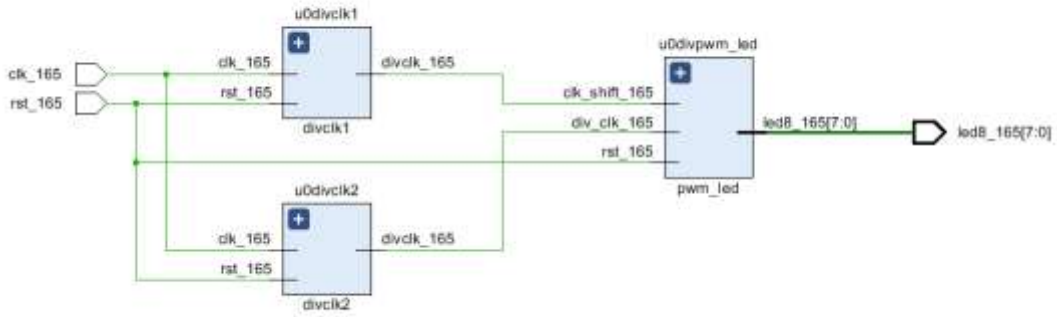


说明：

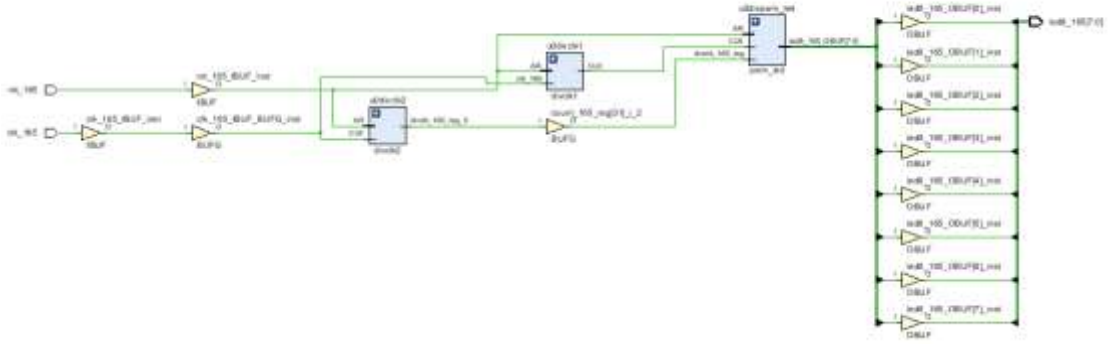
- 将 led8_165[0]、led8_165[1]、led8_165[2]、led8_165[3]、led8_165[4]、led8_165[5]、led8_165[6]、led8_165[7]输出引脚约束到 A7-EDP-1 开发平台上的 XC7A75TFGG484-1 器件的八个引脚上，以达到驱动 LED 的目的。

- 将 clk_165 约束到 100MHz 的时钟引脚上，将 rst_165 约束到开关 T5 引脚上。

详细描述后的图像



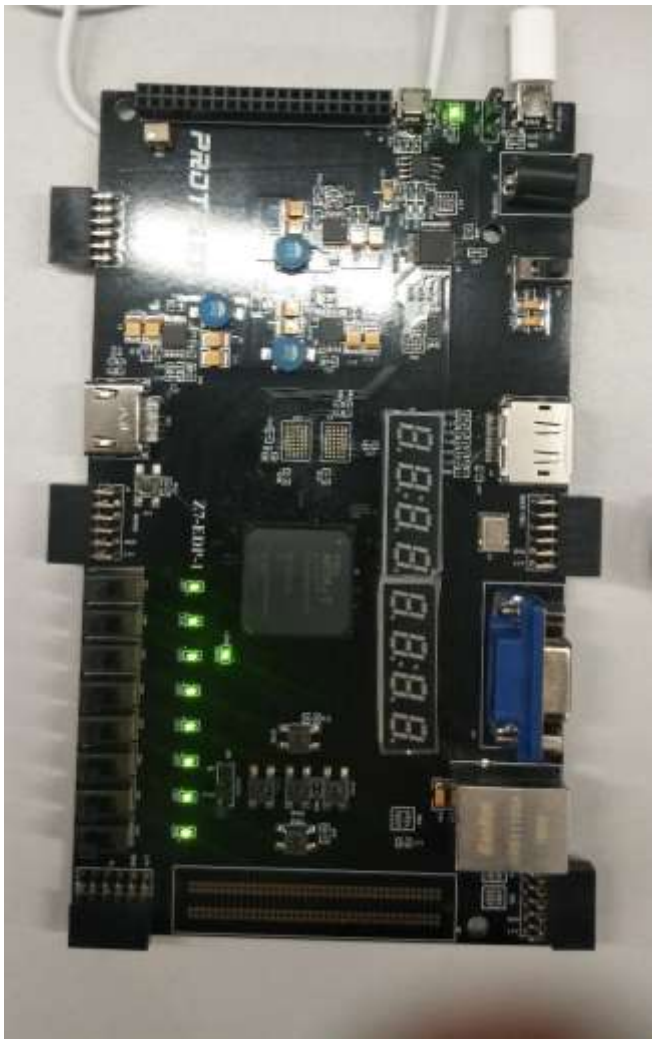
综合后的结果

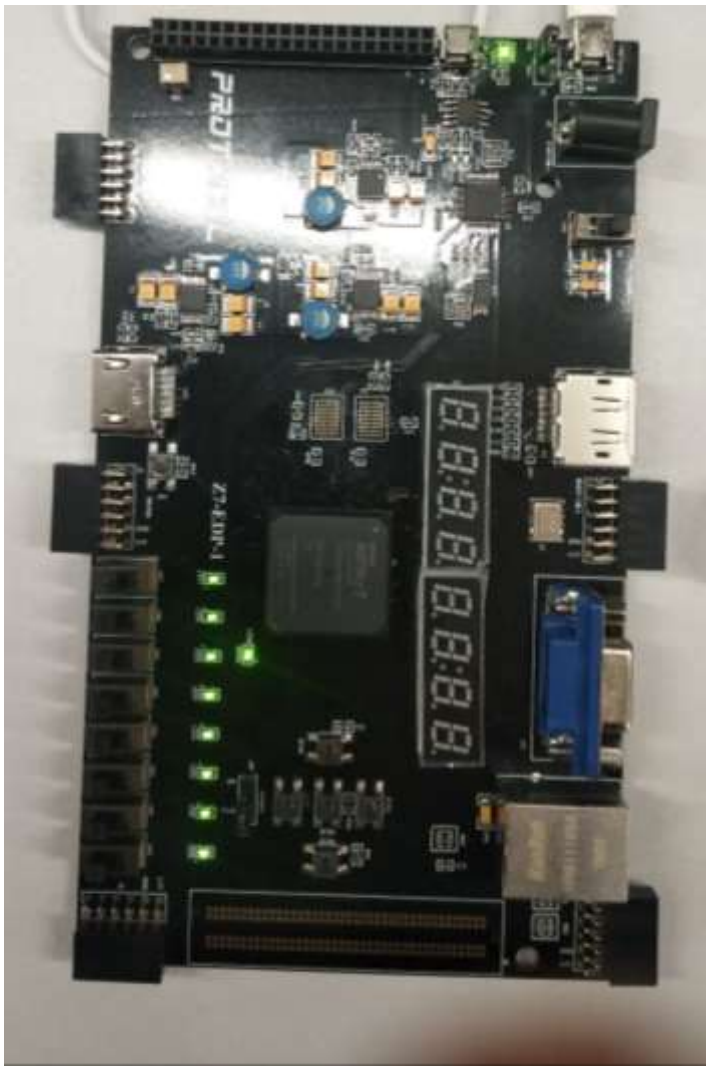


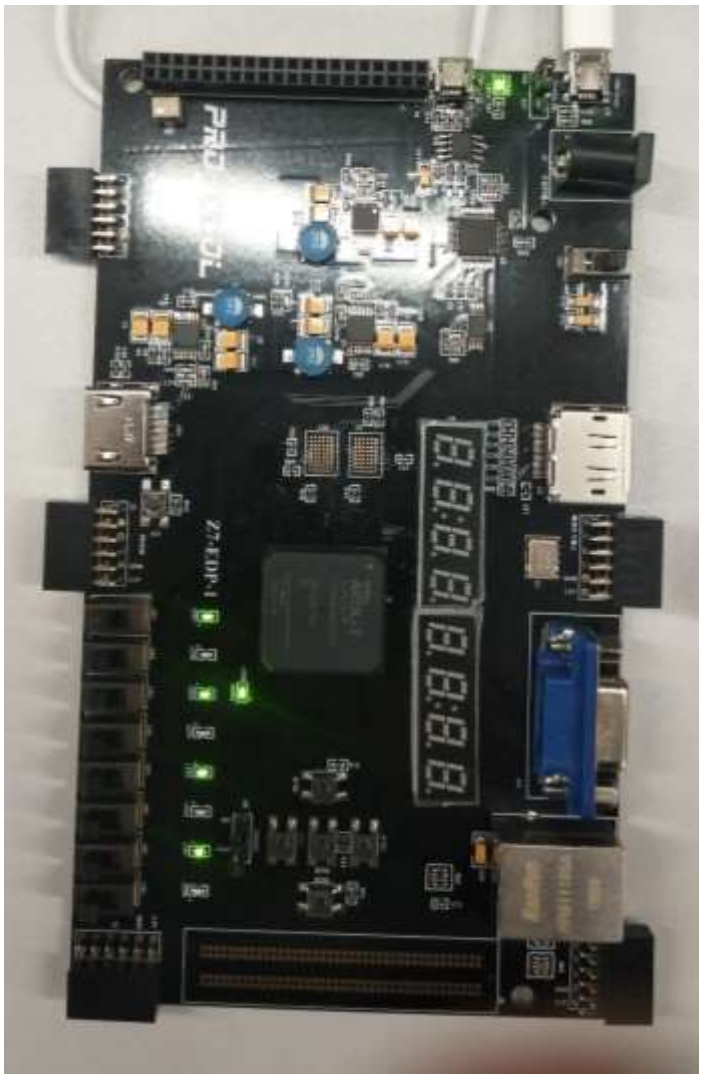
5、实物效果图（将梦幻 LED 和呼吸灯融合后，已经实现，但是放图不太容易看出，已发给老师视频）

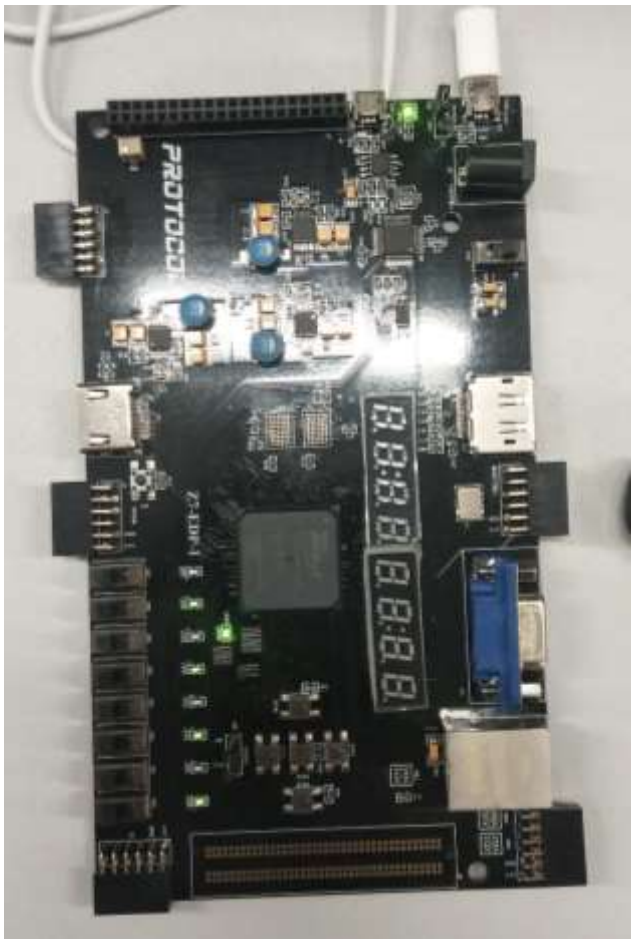
实现效果梦幻呼吸灯变化规律如下：

7) 第 1s,8 个 LED 慢慢变亮。
8) 第 2s,8 个 LED 慢慢熄灭。
9) 第 3s 开始,LED 开始从右向左移动,当移动到最左边时。每移动到一个位置，LED 灯一个慢慢变亮，下一个 LED 灯慢慢熄灭。
10) LED 开始从最左向最右移动，当移动到最右边时。每移动到一个位置，LED 灯一个慢慢变亮，下一个 LED 灯慢慢熄灭。
11) 8 个 LED 其中 4 个 LED 间隔慢慢亮，剩下 4 个 LED 间隔慢慢熄灭。
12) 返回 1)，周而复始的变化。








PROTOCO