

计算机系统结构实验

Lab01:flowing light

常烁晨 521021910369

2023.3.15

摘要

在lab01中，我借助Vivado开发环境，学习了Verilog硬件编程语言，控制硬件并行电路实现flowing_light功能，学习了如何新建module，与simulation文件，初步理解了Vivado的语法、项目流程、仿真方法与调试手段，收获颇丰。此外，我们在完成Vivado环境的开发以及虚拟仿真实验后，还通过管脚约束，在并行硬件FPGA上完成了上板验证，加深了我们对flowing_light项目的理解与认知。

目录

摘要	1
1.实验目的	3
2、实验原理	3
2.1 Vivado工程文件的组织架构	3
2.2 flowing_light 原理	3
3、仿真模拟	4
3.1 创建激励文件	4
3.2仿真实验验证	5
4、工程实现	6
4.1 修改代码:	6
4.2 管脚约束:	7
5、致谢	9

1.实验目的

- (1) 通过基础实验，依据实验说明PDF的步骤指示，了解并学习使用Xilinx研发的集成开发环境Vivado,掌握仿真以及调试手段。
- (2) 了解硬件描述语言 Verilog HDL的语法规则，相关寄存器的使用，时钟的设置以及各项描述功能行为的代码逻辑。
- (3) 完成源代码以及仿真激励文件的编写，进而实现开发环境的仿真模拟功能，观察高低电平的结果，检验电路设计是否符合预期循环目标。
- (4) 学习使用 I/O Planing 或者手动输入xdc约束文件，添加管脚约束。
- (5) 将代码加载到FPGA板上，并在实物上实现 flowing light 的功能。
- (6) 总结整理实验结果，完成实验报告，熟悉FPGA系统硬件开发的实验流程。

2、实验原理

2.1 Vivado工程文件的组织架构

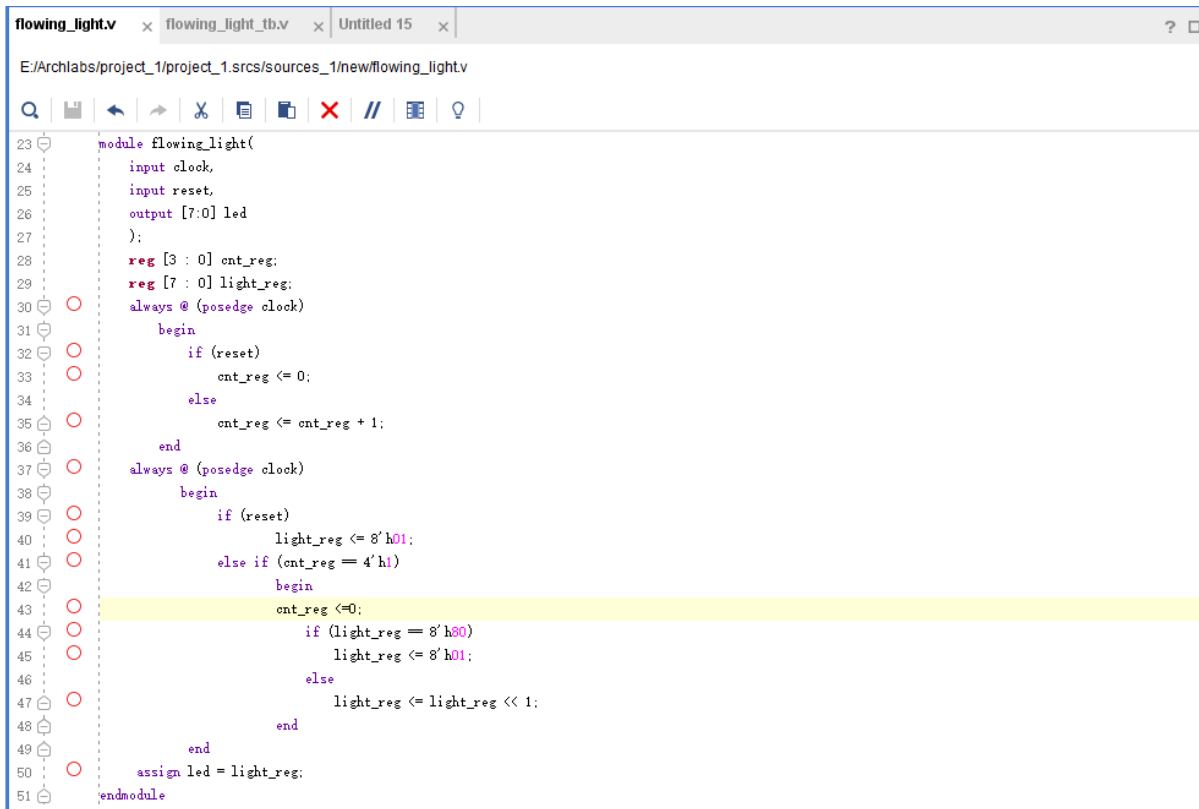
- (1) design source.v 文件（源代码文件）
- (2) simulation source.v 文件（仿真激励文件）
- (3) constraints.xdc 文件（管脚约束文件）

2.2 flowing_light 原理

本次实验要求实现8个LED小灯间隔一定时间循环点亮并熄灭的过程，该过程被称为flowing_light。要求每一时刻只有同一小灯点亮，其余小灯处于熄灭状态。当最后一个LED小灯熄灭后，第一个LED小灯重新点亮，并从此不断循环。源代码保存在flowing_light.v文件。

按照实验说明书的指导，只需通过时钟控制，达到一定的时间间隔之后，通过Verilog语言的位运算，左移逻辑使得下一个LED灯对应的状态位置1，而前一个小灯的状态位被0填充，实现高低电平的转化。

实验中我的代码如下：



```
23 module flowing_light(  
24     input clock,  
25     input reset,  
26     output [7:0] led  
27 );  
28     reg [3 : 0] cnt_reg;  
29     reg [7 : 0] light_reg;  
30     always @ (posedge clock)  
31     begin  
32         if (reset)  
33             cnt_reg <= 0;  
34         else  
35             cnt_reg <= cnt_reg + 1;  
36     end  
37     always @ (posedge clock)  
38     begin  
39         if (reset)  
40             light_reg <= 8'h01;  
41         else if (cnt_reg == 4'h1)  
42             begin  
43                 cnt_reg <= 0;  
44                 if (light_reg == 8'h80)  
45                     light_reg <= 8'h01;  
46                 else  
47                     light_reg <= light_reg << 1;  
48             end  
49         end  
50     assign led = light_reg;  
51 endmodule
```

实验指导说明书要求将原本的代码做出修改以支持flowing_light功能。观察到原本代码中间隔时间过长（采用24位寄存器计数，因此每次计数结束所需的时间远远超过仿真模拟的时间上限），因此此处将代码修改为使用4位寄存器保存计数信息，每个时钟周期进行一次自增。且当8位LED状态寄存器最高位置1后，再次经过相同时间间隔，将需要恢复默认状态（即最后一个LED灯亮起后经过对应时间间隔，恢复到只有第一个LED点亮的状态）。

3、仿真模拟

3.1 创建激励文件

完成上述源代码编程后，需要进行仿真激励文件的编写，以便使开发环境辅助进行虚拟仿真。首先创建flowing_light_tb.v激励文件，根据实验指导说明书的要求完成激励文件的编写。

实验指导书给出的代码如下：

```

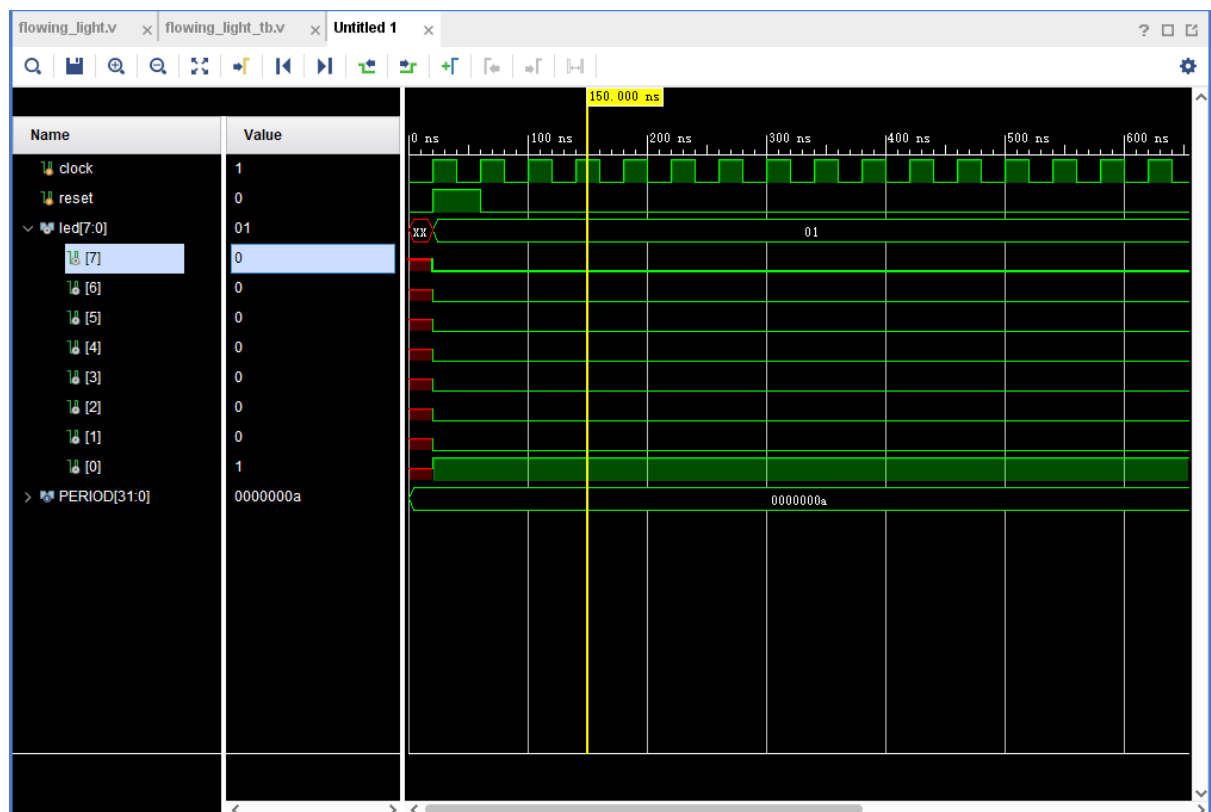
1
2  reg  clock;
3  reg  reset ;
4  wire [7:0] led;
5
6  flowing_light u0(
7      .clock(clock),
8      .reset(reset),
9      .led(led));
10
11  parameter PERIOD= 10;
12
13  always #(PERIOD*2) clock =  !clock;
14
15  initial begin
16      clock = 1'b0;
17      reset = 1'b0;
18      #(PERIOD*2) reset = 1'b1;
19      #(PERIOD*4) reset = 1'b0;
20
21      // #580;  reset = 1'b1;
22  end

```

可以看到激励文件中时钟周期是40个时间单位。

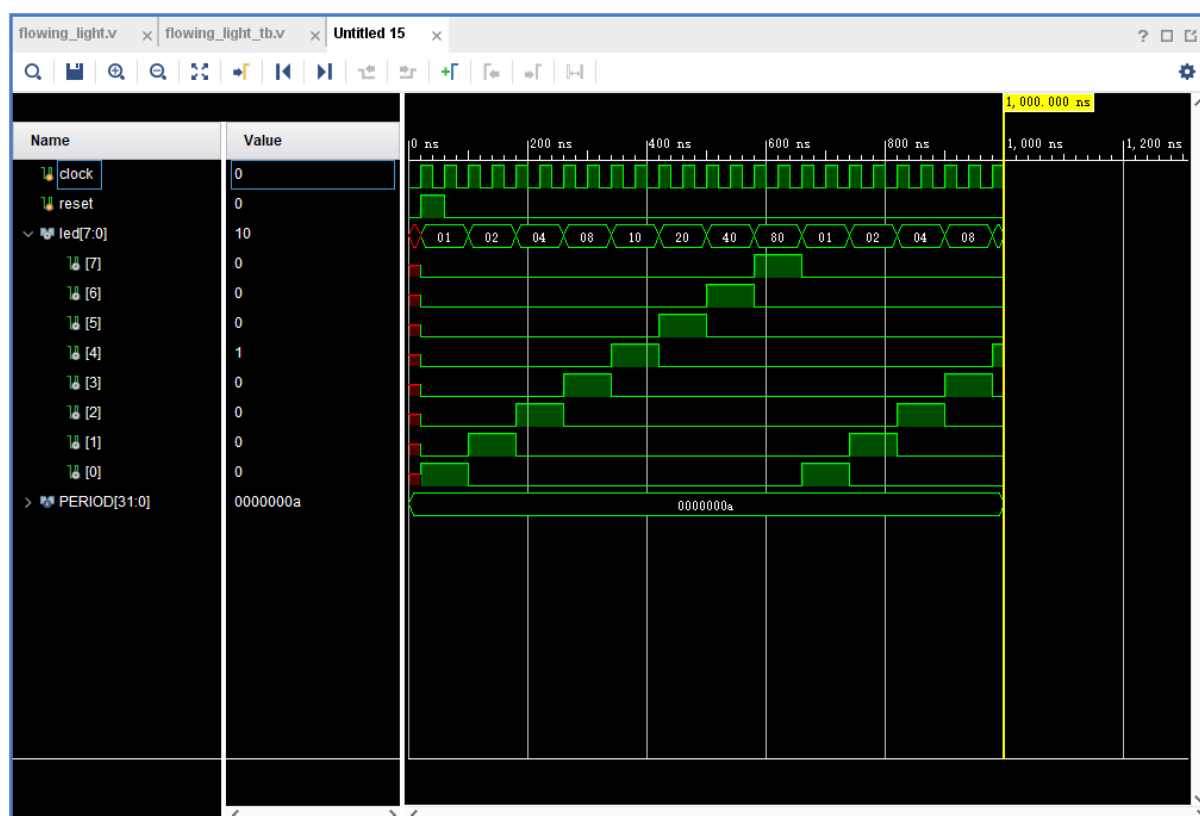
3.2仿真实验验证

在修改代码之前，运行仿真的结果如图所示：



可以看到由于仿真运行的时间有限，在24位寄存器达到最大值（即切换LED灯对应的时间间隔结束）之前，仿真运行就已经结束，因此整个仿真过程虽然各项信号正常，但是无法体现flowing_light的效果。

修改代码后，再次进行模拟实验。这次观察到的仿真结果如图所示：



实验结果中可以看到，LED小灯的高电平段相继出现，时间间隔相等，同时在完成一轮flowing之后，第一个LED再次点亮，开启下一轮循环。

4、工程实现

4.1 修改代码：

由于实验板载高频时钟，下载验证需要查分时钟，因此对flowing_light模块做出时钟方面的修改如下：

```

module flowing_light (
    input clock_p,
    input clock_n,
    .....
);

IBUFGDS IBUFGDS_inst (
    .O ( CLK_i )
    .I (clock_p )
    .IB (clock_n )
);

Always @ ( posedge CLK_i )
    begin
        if ( !reset )
            .....
    end

.....

```

4.2 管脚约束：

- (1) 利用 IO planing 功能，首先对工程进行综合（Flow Navigator —> Synthesis —> Run Synthesis）
- (2) 新建约束文件，手动输入 FPGA 管脚约束信息。内容如下所示。

```
set_property PACKAGE_PIN W23 [get_ports {led[7]}]
set_property PACKAGE_PIN AB26 [get_ports {led[6]}]
set_property PACKAGE_PIN Y25 [get_ports {led[5]}]
set_property PACKAGE_PIN AA23 [get_ports {led[4]}]
set_property PACKAGE_PIN Y23 [get_ports {led[3]}]
set_property PACKAGE_PIN Y22 [get_ports {led[2]}]
set_property PACKAGE_PIN AE21 [get_ports {led[1]}]
set_property PACKAGE_PIN AF24 [get_ports {led[0]}]
set_property PACKAGE_PIN AC18 [get_ports clock_p]
set_property PACKAGE_PIN W13 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property IOSTANDARD LVDS [get_ports clock_p]
set_property IOSTANDARD LVDS [get_ports clock_n]
set_property IOSTANDARD LVCMOS18 [get_ports reset]
```

(3) 下载验证，生成 bit 流文件（Flow Navigator —> Program and Debug —>Generate Bitstream）供 FPGA 板运行工程。生成 Bitstream 后直接进行烧写。

(4) 连接及演示，首先将实验板连接到电源线上，并通过数据线连接电脑，打开电脑的防火墙相关权限。

之后点击 Flow Navigator —> Open HardWare Manager 进入硬件编程管理界面，点击 Program Device将实验板与电脑连接。点击 Program，将比特流文件传输到 FPGA 芯片，观察实验板上的 8 位流水灯循环点亮的过程。

5、致谢

感谢刘老师以及三位实验助教学长学姐的悉心指导和帮助。在完成第一次实验的过程中，由于对实验环境和实验语言不够熟悉，因此我遇到了许多问题，再次感谢几位不厌其烦指导我解决问题和 debug 的助教学长学姐。

感谢上海交通大学的计算机系统结构实验室，为我们准备了相关的实验环境以及 FPGA 开发板，学校和学院提供的优秀设备让我们受益匪浅。