计算机系统结构实验 1

FPGA 基础实验：LED Flow Water Light

胡晨志 521021910107

摘要

在 lab01 中，我学习了 Verilog 语言并实现了 flowing light 功能，对 Verilog 的基本语法有了大致的了解，并对 Vivado 的编程环境、项目流程、调试方合仿真手段有了基本的了解。

关键字：Vivado, Verilog

目录

1 实验目的

- 掌握 Xilinx 逻辑设计工具 Vivado 的基本操作
- 掌握使用 Verilog HDL 进行简单的逻辑设计
- 掌握仿真功能
- 使用 1/0 Planing 添加管脚约束
- 生成 Bitstream 文件
- 上板验证

2 原理分析

2.1 Vivado 工程的基本组成

Vivado 工程的基本组成如下：

- design source .v 文件
- simulation source .v 文件
- constrain .xdc 文件（上板验证所需的管脚约束文件）

在本次实验中，对应的文件依次为 flowing_light.v 文件、flowing_light_tb.v 文件和 lab01_xdc.xdc 文件。

2.2 flowing light 原理

Flowing light 要求在一定时间内 8 个 LED 等依次亮灭，最后一个 LED 灯熄灭，第一个 LED 灯循环亮起。在实现代码中，用时钟周期的上升沿来同步灯的亮灭，代码如 </>CODE 1所示。由代码克制控制 LED 的 `light_reg` 每过 384 个时钟周期进行一次位移。

</> CODE 1: flowing light 实现

```
always @ (posedge clock)
begin
    if (reset)
        light_reg <= 8'h01;
    else if (cnt_reg == 24'h0ffffff)
        begin
            if (light_reg == 8'h80)
                light_reg <= 8'h01;
            else
                light_reg <= light_reg << 1;
        end
end
```

3 功能实现

基于上述即可完成 flowing light 的功能。完整代码如 </>CODE 3所示。

</> CODE 2: flowing_light.v

```
`timescale 1ns / 1ps

module flowing_light(
    input clock,
    input reset,
    output [7:0] led
    );
    reg [23 : 0] cnt_reg;
    reg [7 : 0] light_reg;
    always @ (posedge clock)
        begin
            if (reset)
                cnt_reg <= 0;
            else
                cnt_reg <= cnt_reg + 1;
        end
    always @ (posedge clock)
        begin
            if (reset)
                light_reg <= 8'h01;
            else if (cnt_reg == 24'h0ffffff)
                begin
                    if (light_reg == 8'h80)
                        light_reg <= 8'h01;
                    else
                        light_reg <= light_reg << 1;
                end
        end
    assign led = light_reg;
endmodule
```

实现 flowing_light.v 后，生成 flowing_light_tb.v 的激励文件用以仿真测试，生成 lab01_xdc.xdc 的管脚约束用以练习。

4 结果验证

4.1 测试用激励文件

按照实验指导书的要求编写 flowing_light_tb.v 文件，代码如 </>CODE 4 所示。

</> CODE 3: flowing_light_tb.v

```
`timescale 1ns / 1ps

module flowing_light_tb(

    );
    reg clock;
    reg reset;
    wire [7 : 0] led;

    flowing_light u0(
        .clock(clock),
        .reset(reset),
        .led(led));

    parameter PERIOD = 10;

    always #(PERIOD*2) clock = !clock;

    initial begin
        clock = 1'b0;
        reset = 1'b0;
        #(PERIOD*2) reset = 1'b1;
        #(PERIOD*4) reset = 1'b0;

        //#580; reset = 1'b1;
    end
endmodule
```

4.2 reset 与基本逻辑的测试

获得的仿真结果如图1所示。从图中可以看到 reset，cnt_reg，light_reg 功能正常。

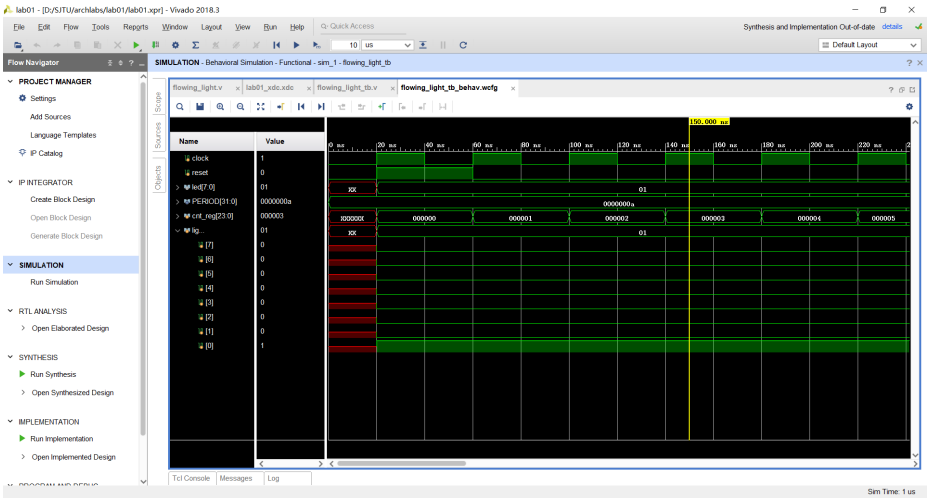

图 1: 实验结果 1

4.3 调整控制逻辑以观察位移

上述仿真运行周期不够，计数器并没加到 24 位全是 1 波形显示已结束。将 `cnt_reg` 改为两位后可观察到位移，代码如 </>CODE 4 所示，仿真结果如图 2 所示。

</> CODE 4: 修改后的 flowing light

```
reg [1 : 0] cnt_reg;
reg [7 : 0] light_reg;
always @ (posedge clock)
    begin
        if (reset)
            cnt_reg <= 0;
        else
            cnt_reg <= cnt_reg + 1;
    end
always @ (posedge clock)
    begin
        if (reset)
            light_reg <= 8'h01;
        else if (cnt_reg == 2'b11)
            begin
                if (light_reg == 8'h80)
                    light_reg <= 8'h01;
                else
                    light_reg <= light_reg << 1;
            end
    end
assign led = light_reg;
```

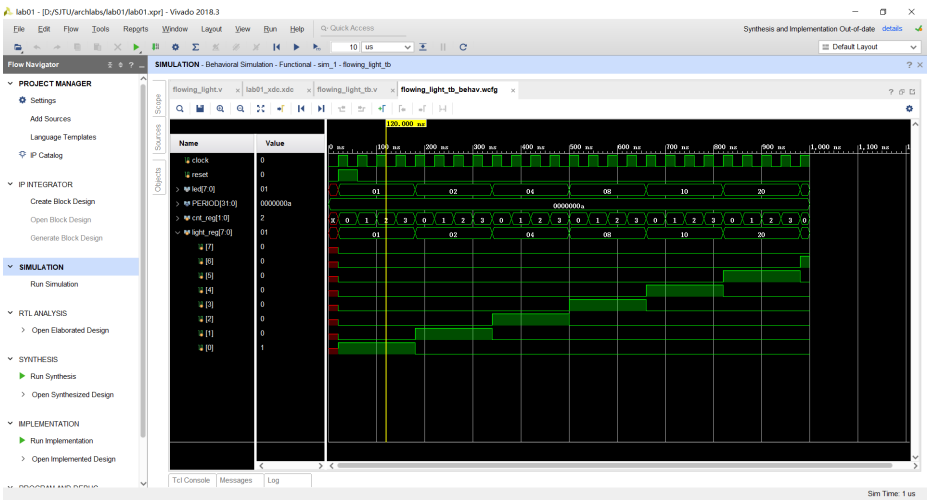

图 2: 实验结果 2

5 管脚约束

管脚约束文件为 lab01_xdc.xdc，代码如 </>CODE 5 所示。在上板前，需要将 flowing_light.v 文件也进行修改，代码如 </>CODE 6 所示。

</> CODE 5: lab01_xdc.xdc

```
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 LVCMOS18 [get_ports reset]
```

</> CODE 6: flowing_light.v 最终代码

```
`timescale 1ns / 1ps
module flowing_light(
    input clock_p,
    input clock_n,
    input reset,
    output [7:0] led
    );
    reg [23 : 0] cnt_reg;
    reg [7 : 0] light_reg;

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

    always @ (posedge CLK_i)
```

```
        begin
            if (!reset)
                cnt_reg <= 0;
            else
                cnt_reg <= cnt_reg + 1;
        end
    always @ (posedge CLK_i)
        begin
            if (!reset)
                light_reg <= 8'h01;
            else if (cnt_reg == 24'h0ffffff)
                begin
                    if (light_reg == 8'h80)
                        light_reg <= 8'h01;
                    else
                        light_reg <= light_reg << 1;
                end
        end
    assign led = light_reg;
endmodule
```

6 总结与反思

这是我第一次接触 Vivado 开发环境和 Verilog 语言，刚上手时闹了不少笑话，比如把代码写到 I/O 说明中。此外由于对工程流程不熟悉，Lab01 花了我大量的时间。好在经过一个上午的摸索之后我还是成功地完成了本次实验，受益匪浅。感谢老师和助教的指导！