计算机系统结构试验 Lab01: Flowing light

姓名：N/A

摘要

在 Lab01 中，我使用 Verilog 语言成功实现了 Flowing light 功能。通过本次实验，我学会了如何创建 module 和 simulation 文件，并初步理解了 Vivado 的语法、项目流程、仿真方法和调试技巧。这次实验给我带来了很多收获。

目录

1. 实验目的

（1）通过基础实验熟悉 Xilinx 逻辑设计工具 Vivado 开发环境；
（2）了解硬件描述语言 Verilog HDL 描述功能行为的逻辑；
（3）通过仿真检验电路设计是否预期；
（4）学习使用 I/O Planning 添加管脚约束；
（5）实现 Flowing light 的功能；
（6）熟悉系统硬件开发的基本实验流程。

2. 原理分析

2.1 Vivado 工程的基本组成

（1）design source.v 文件
（2）simulation source.v 文件
（3）constraints.xdc 文件（上板验证所需的管脚约束文件）

2.2 Flowing light 的原理

Flowing light 要求在一段时间内，8 个 LED 灯依次轮流亮灭，最后一个 LED 熄灭后，第一个 LED 循环亮起。这个功能可以使用移位来实现控制，每位对应一个 LED 灯。每次到达计数值时，light_reg 中的值循环左移一位。

```
1   `timescale 1ns / 1ps
2
3   module flowing_light (
4       input clock,
5       input reset,
6       output [7:0] led
7   );
8
9     reg [23:0] cnt_reg;
10    reg [ 7:0] light_reg;
11
12    always @(posedge clock) begin
13      if (reset) cnt_reg <= 0;
14      else cnt_reg <= cnt_reg + 1;
15    end
16
17    always @(posedge clock) begin
18      if (reset) light_reg <= 8'h01;
19      else if (cnt_reg == 24'hffffff) begin
20        if (light_reg == 8'h80) light_reg <= 8'h01;
21        else light_reg <= light_reg << 1;
22      end
23    end
24
25    assign led = light_reg;
26
27  endmodule
28
```

3. 功能实现

本实验比较简单，基于上述的原理容易实现 flowing light 的功能。在实现 flowing_light.v 后，生成 flowing_light_tb.v 的激励文件用以仿真测试，生成 flowing_light_xdc.xdc 的管脚约束用以练习。

4. 结果验证

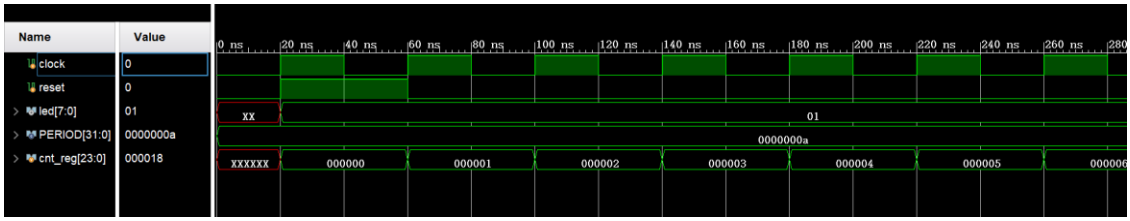
4.1 测试用激励文件

首先，按照实验导书上的要求，编写激励文件。设置时钟周期为 40ns，并设置各输入初值。代码如下：

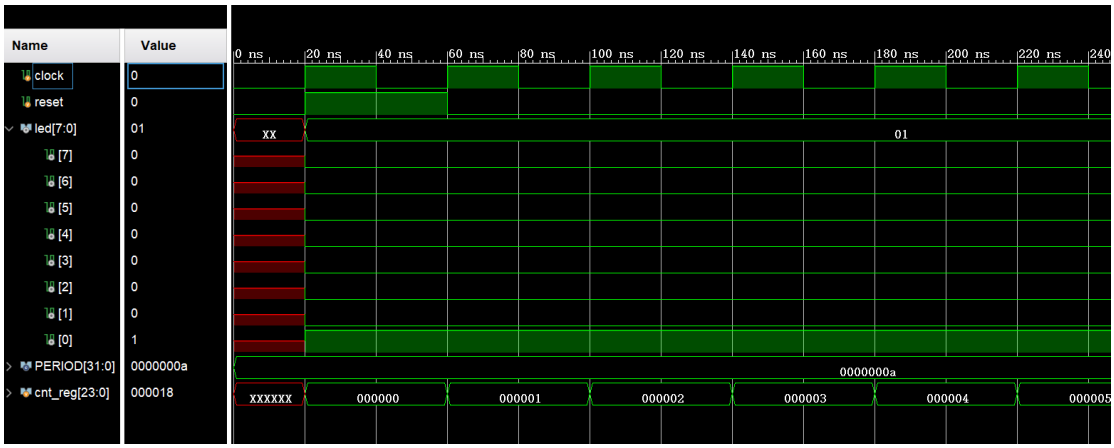
```
1   `timescale 1ns / 1ns
2
3   module flowing_light_tb;
4
5     reg clock;
6     reg reset;
7     wire [7:0] led;
8
9     flowing_light u0 (
10        .clock(clock),
11        .reset(reset),
12        .led  (led)
13    );
14
15    parameter PERIOD = 10;
16
17    always #(PERIOD * 2) clock = !clock;
18
19    initial begin
20      clock = 1'b0;
21      reset = 1'b0;
22      #(PERIOD * 2) reset = 1'b1;
23      #(PERIOD * 4) reset = 1'b0;
24    end
25
26  endmodule
27
```

4.2 reset 与基本逻辑的测试

首先进行仿真，结果如下所示：



上图中可以看到 reset，cnt_reg，light_reg 功能正常。

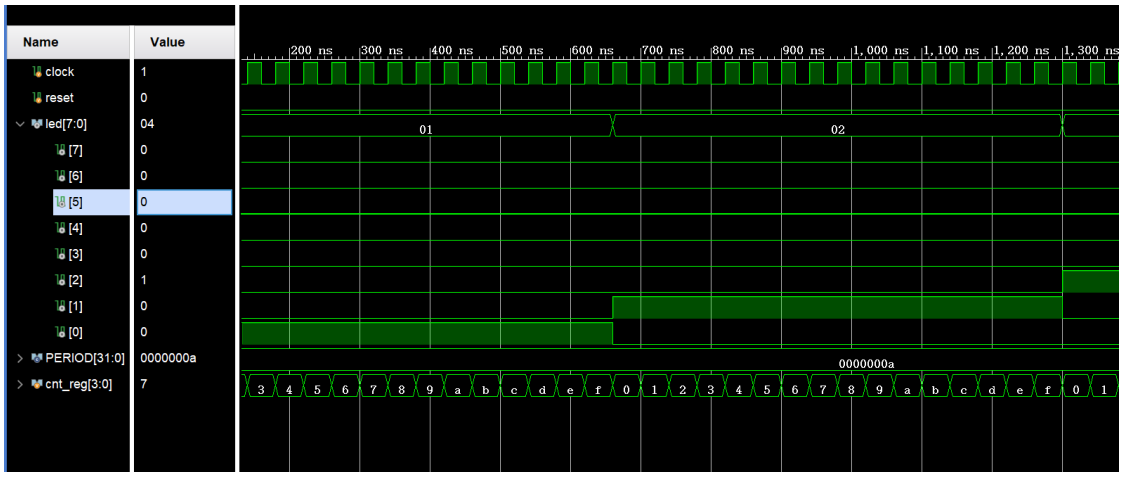


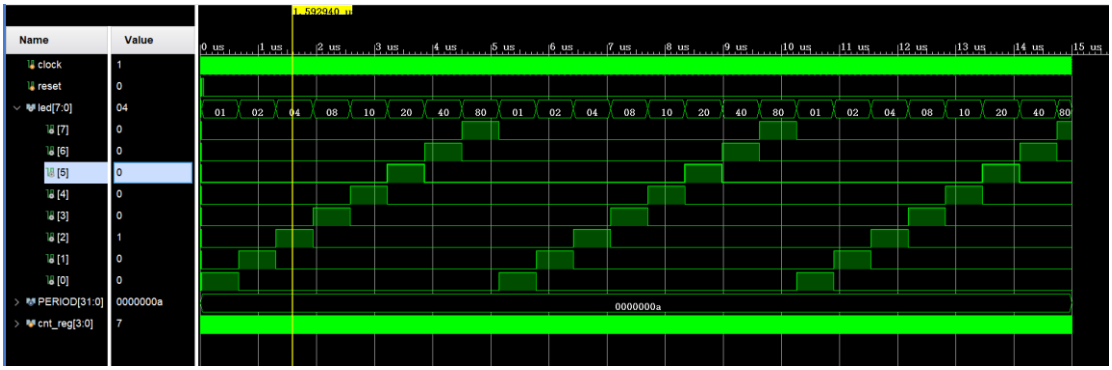
上图中可以看到 led 每一位的情况。

本仿真运行周期不够，计数器并没加到 0xffffff 而波形显示早已结束。因此更改计数器的目标值，以便较快速达到左移条件，结果记录于 4.3 中。

4.3 调整控制逻辑以观察移位

我将计数逻辑改为 cnt_reg==0xf 时移位，即经过 16 个时钟周期就移位，并相应把 cnt_reg 改为 4 位。在仿真模拟中观察到了移位，如下图所示：





这样，我通过改变计数器的控制逻辑在仿真中观察到了移位。

5. 管脚约束

由于实验板板载了 200MHz 时钟振荡器，属高频时钟，做下载验证时则需用到差分时钟以更好适应工程上的需要。修改代码如下：

```
29  module flowing_light (
30      input clock_p,
31      input clock_n,
32      input reset,  // active low
33      output [7:0] led
34  );
35
36    reg [23:0] cnt_reg;
37    reg [ 7:0] light_reg;
38
39    IBUFGDS IBUFGDS_inst(
40      .O  (CLK_i),
41      .I  (clock_p),
42      .IB (clock_n)
43    )
44
45    always @(posedge CLK_i) begin
46      if (!reset) cnt_reg <= 0;
47      else cnt_reg <= cnt_reg + 1;
48    end
49
50    always @(posedge CLK_i) begin
51      if (!reset) light_reg <= 8'h01;
52      else if (cnt_reg == 24'hffffff) begin
53        if (light_reg == 8'h80) light_reg <= 8'h01;
54        else light_reg <= light_reg << 1;
55      end
56    end
57
58    assign led = light_reg;
59
60  endmodule
```

同时设置管脚约束如下：

```
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]
```

6. 总结与反思

在工科创中我曾使用过 Vivado 通过这次实验，我再次熟悉了 Vivado 的开发环境，因此上手并不困难。在这个实验中，我不仅复习了 Verilog HDL 的基本语法和项目开发流程，还学习了仿真激励文件的写法。

我要感谢课程组为我们准备的详细指导书。在接下来的学习中，我计划进一步巩固 Verilog HDL 的知识，尝试更复杂的项目，并探索其他开发工具和技术，以提升我的硬件设计能力。