中国科学技术大学计算机学院

《数字电路实验》报告

实验题目：FPGA 原理及 Vivado 综合

学生姓名：舒文炫

学生学号：PB18000029

完成日期：2021.11.25

计算机实验教学中心制

2020 年 09 月

【实验题目】

FPGA 原理及 Vivado 综合

【实验目的】

- 了解 FPGA 工作原理
- 了解 Verilog 文件和约束文件在 FPGA 开发中的作用
- 学会使用 Vivado 进行 FPGA 开发的完整流程

【实验环境】

- FPGAOL 实验平台：fpgaol.ustc.edu.cn
- Logisim
- Vivado 工具

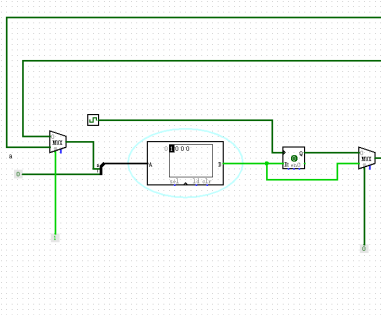
【实验练习】

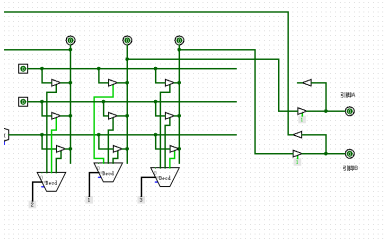
题目 1

请通过实验中给出的可编程逻辑单元、交叉互连矩阵及 IOB 电路图，实现如下代码，并将其输出到引脚 B 上。给出配置数据和电路截图。

```
module test(input clk,output reg a);
  always@(posedge clk)
    a <= a ^ 1'b1;
endmodule
```

这里电路图比较长，截图分成两部分



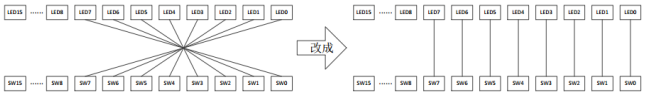


这里的配置数据，需要输出 a 异或 1, 所以 a 为 0 时要输出 1，a 为 1 时要输出 0，这里我使用的 RAM，地址宽 2bit，数据宽 1bit, 使用分线器，分线器第一位为 0，所以是选择 RAM 前两个地址的值，对应为 1，0. 后接 D 触发器，选择器选择的是触发器的输出，保证代码里面是在时钟上升沿变化。

第二张截图展示了交叉互连矩阵的配置，主要让第三个 Decd 选择 3，这样可以把 a 异或 1 的信号接到引脚 B，前两个 Decd 选不同的线防止冲突，然后通过 B 反馈到输入 a, 这样就能实现当时钟上升沿到来时 a 都会变成上一次的 b，这样就能实现上面的代码

题目 2

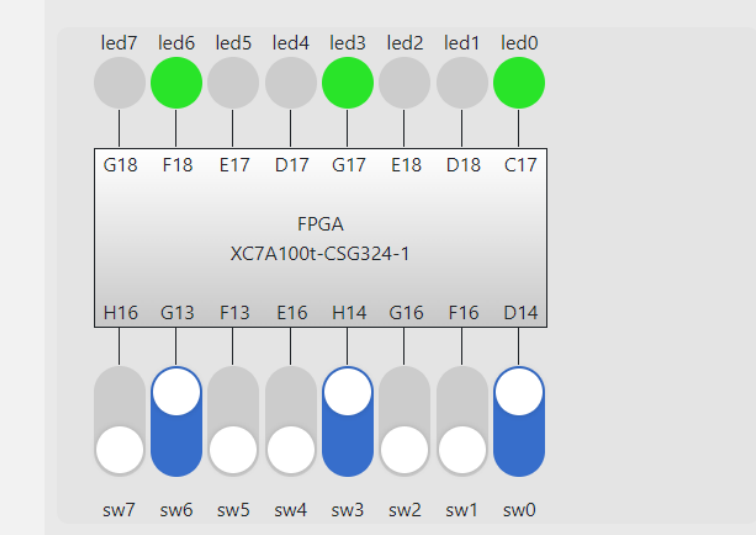
实验中的开关和 LED 的对应关系是相反的，即最左侧的开关控制最右侧的 LED，最右侧的开关控制最左侧的 LED，请修改实验中给出的 XDC 文件，使开关和 LED 一一对应（最左侧的开关控制最左侧的 LED），如下图所示。



这里只要注意到管脚的对应关系即可，下面是我的约束文件

```
set_property -dict { PACKAGE_PIN E3  IOSTANDARD LVCMOS33 } [get_ports { clk }];
set_property -dict { PACKAGE_PIN B18  IOSTANDARD LVCMOS33 } [get_ports { rst }];
set_property -dict { PACKAGE_PIN G18  IOSTANDARD LVCMOS33 } [get_ports { led[0] }];
set_property -dict { PACKAGE_PIN F18  IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN E17  IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
set_property -dict { PACKAGE_PIN D17  IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
set_property -dict { PACKAGE_PIN G17  IOSTANDARD LVCMOS33 } [get_ports { led[4] }];
set_property -dict { PACKAGE_PIN E18  IOSTANDARD LVCMOS33 } [get_ports { led[5] }];
set_property -dict { PACKAGE_PIN D18  IOSTANDARD LVCMOS33 } [get_ports { led[6] }];
set_property -dict { PACKAGE_PIN C17  IOSTANDARD LVCMOS33 } [get_ports { led[7] }];
set_property -dict { PACKAGE_PIN D14  IOSTANDARD LVCMOS33 } [get_ports { sw[0] }];
set_property -dict { PACKAGE_PIN F16  IOSTANDARD LVCMOS33 } [get_ports { sw[1] }];
set_property -dict { PACKAGE_PIN G16  IOSTANDARD LVCMOS33 } [get_ports { sw[2] }];
set_property -dict { PACKAGE_PIN H14  IOSTANDARD LVCMOS33 } [get_ports { sw[3] }];
set_property -dict { PACKAGE_PIN E16  IOSTANDARD LVCMOS33 } [get_ports { sw[4] }];
set_property -dict { PACKAGE_PIN F13  IOSTANDARD LVCMOS33 } [get_ports { sw[5] }];
set_property -dict { PACKAGE_PIN G13  IOSTANDARD LVCMOS33 } [get_ports { sw[6] }];
set_property -dict { PACKAGE_PIN H16  IOSTANDARD LVCMOS33 } [get_ports { sw[7] }];
```

这里将 led 的位连接管脚的顺序与实验步骤里面的顺序反过来即可，下面是在 FPGA 平台的结果



可以看到这里确实是开关和 LED 一一对应了

题目 3

设计一个 30 位计数器，每个时钟周期加 1，用右侧的 8 个 LED 表示计数器的高 8 位，观察实际运行结果。将该计数器改成 32 位，将高 8 位输出到 LED，与前面的运行结果进行对比，分析结果及时钟信号在其中所起的作用。

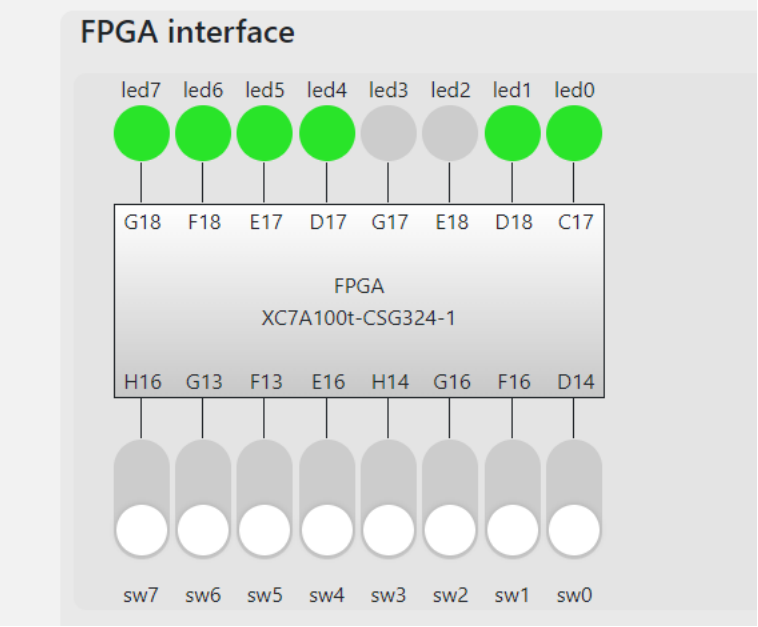
30 位计数器就需要一个 30 位的寄存器变量 count，然后将高 8 位，也就是 29,28,27,26,25,24,23,22 位传给 led，led 是一个 8 位的寄存器变量然后将 led 与管脚对应上，这样就可以了。具体代码如下

```
module p3(
input clk,
input rst,
output reg [7:0] led
    );
reg [29:0] count=30'h00000000;
always@(posedge clk or posedge rst)
begin
    if(rst)
        count<=30'h00000000;
    else
        count<=count+1;
    led<={count[29],count[28],count[27],count[26],count[25],count[24],count[23],count[22]};
end
endmodule
```

下面是对应的约束文件

```
set_property -dict { PACKAGE_PIN E3  IOSTANDARD LVCMOS33 } [get_ports { clk }];
set_property -dict { PACKAGE_PIN B18  IOSTANDARD LVCMOS33 } [get_ports { rst }];
set_property -dict { PACKAGE_PIN G18  IOSTANDARD LVCMOS33 } [get_ports { led[0] }];
set_property -dict { PACKAGE_PIN F18  IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN E17  IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
set_property -dict { PACKAGE_PIN D17  IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
set_property -dict { PACKAGE_PIN G17  IOSTANDARD LVCMOS33 } [get_ports { led[4] }];
set_property -dict { PACKAGE_PIN E18  IOSTANDARD LVCMOS33 } [get_ports { led[5] }];
set_property -dict { PACKAGE_PIN D18  IOSTANDARD LVCMOS33 } [get_ports { led[6] }];
set_property -dict { PACKAGE_PIN C17  IOSTANDARD LVCMOS33 } [get_ports { led[7] }];
```

下面是运行的截图

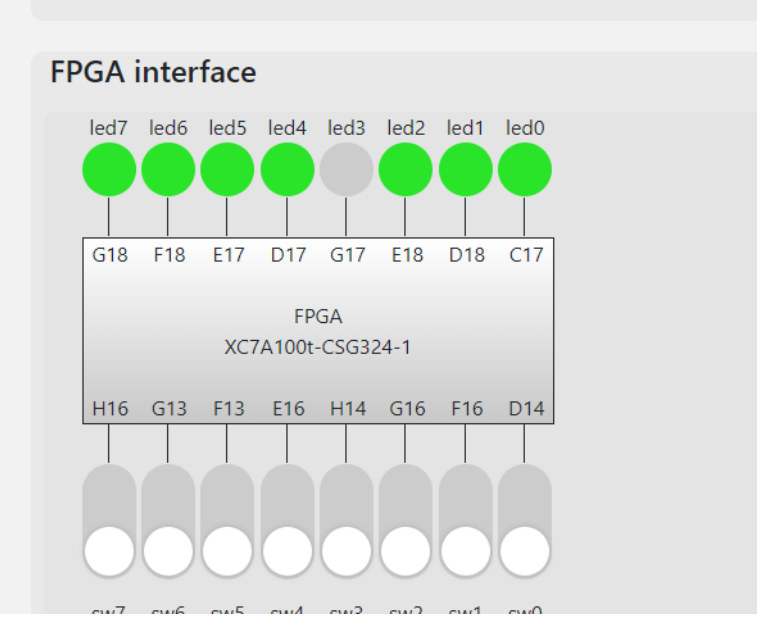


不过截图看不出这个 led 的变化

如果改成 32 位，就将 count 变成 32 位，然后将 31，30，29，28，27，26，25，24 赋给 led 即可，代码如下

```
module p3(
input clk,
input rst,
output reg [7:0] led
    );
reg [31:0] count=32'h00000000;
always@(posedge clk or posedge rst)
begin
    if(rst)
        count<=32'h00000000;
    else
        count<=count+1;
    led<={count[31],count[30],count[29],count[28],count[27],count[26],count[25],count[24]};
end
endmodule
```

由于输入输出接口都一样，这里约束文件与 30 位计数器的相同，下面是运行的截图



由于变成了 32 位，计数器来前八位的变化会变慢，所以这里 led 变化会慢很多，至于时钟信号的作用，这里计数器 +1 是在时钟上升沿进行的

【总结与思考】

1. 请总结本次实验的收获

本次实验学习了如何使用 FPGA 在线开发平台，了解了约束文件的写法，同时也对 FPGA 的原理有了一定的了解，亲手用 logisim 实现了简单的 vivado 代码，收获很多

2. 请评价本次实验的难易程度

本次实验难度不高，知道了管脚的对应关系，相应的修改约束文件就可以了，本身要实现的功能也没有特别复杂的逻辑，所以做起来很快

3. 请评价本次实验的任务量

任务量不大，很快就能做完

4. 请为本次实验提供改进建议

感觉可以在实验步骤里面多介绍一点关于这个 FPGA 在线开发平台的东西，就是突然整一个约束文件对应引脚什么的，同学们可能只会照搬，但是做出来依然一头雾水，这个内容貌似是放到了第七次实验中具体介绍，不过我觉得可以放在这里，还有时钟频率，一般看到 30 位，感觉会很大，不过这个板子能做到 100MHz，这个数据在做这个实验时并没有给出，可能会让有些同学想这个习题三时觉得很复杂，然后想歪了。其他的感觉挺好