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：傅申

学生学号：PB20000051

完成日期：2021 年 12 月 2 日

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】

- Windows PC
- Microsoft Visual Studio Code
- Logisim
- Xilinx Design Tools Vivado HL Design Edition 2019.1

【实验练习】

题目 1: Verilog 代码中为时序电路逻辑, 且有 `a^1'b1 = a`, 因此可以在 Logisim 中画出如下图 1 的电路图. 其中, RAM / MUX 内存放的数据分别为 `1, 0, 1, 0`.

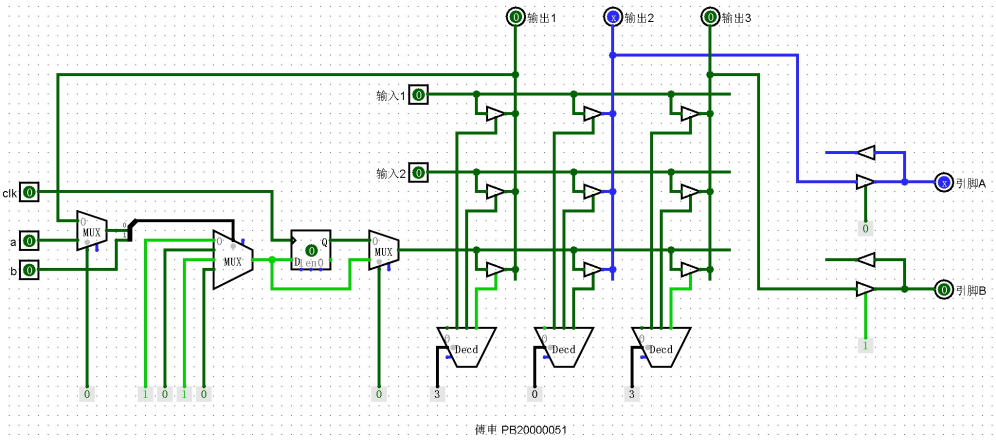

图 1: 实现题目 1. 代码的电路图

题目 2: 将 xdc 文件中约束开关的语句中端口信号逆置, 即 `sw[7]` 与 `sw[0]` 互换, 以此类推, 即可使开关与 LED 一一对应. 部分 xdc 文件中代码如下.

Verilog 代码 1: 修改后的 xdc 文件

```
## FPGAOL LED (single-digit-SEGPLAY)
set_property -dict { PACKAGE_PIN C17   IOSTANDARD LVCMOS33 } [get_ports { led[0] }];
set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
```

```
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
set_property -dict { PACKAGE_PIN D17   IOSTANDARD LVCMOS33 } [get_ports { led[4] }];
set_property -dict { PACKAGE_PIN E17   IOSTANDARD LVCMOS33 } [get_ports { led[5] }];
set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports { led[6] }];
set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports { led[7] }];

## FPGAOL SWITCH
set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { sw[7] }];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { sw[6] }];
set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { sw[5] }];
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { sw[4] }];
set_property -dict { PACKAGE_PIN E16   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }];
set_property -dict { PACKAGE_PIN F13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }];
set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }];
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }];
```

生成 bit 文件后, 烧写入 FPGA 内, 效果如下图 2.

图 2: FPGA 烧写后的效果

题目 3: 设计的 30 位计数器模块如下代码 2, xdc 文件如下代码 3.

Verilog 代码 2: 30 位计数器模块代码

```
module counter_30bit(
    input  clk, rst,
    output [7:0] led );
reg [29:0] count;
assign led = count[29:22];
always @(posedge clk or posedge rst) begin
    if (rst || count == 30'h3fffffff)
        count <= 30'h00000000;
    else
```

```
        count <= count + 1;
end
endmodule
```

Verilog 代码 3: 计数器 xdc 文件

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }];
## FPGAOL LED (signle-digit-SEGPLAY)
set_property -dict { PACKAGE_PIN C17   IOSTANDARD LVCMOS33 } [get_ports { led[0] }];
set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
set_property -dict { PACKAGE_PIN D17   IOSTANDARD LVCMOS33 } [get_ports { led[4] }];
set_property -dict { PACKAGE_PIN E17   IOSTANDARD LVCMOS33 } [get_ports { led[5] }];
set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports { led[6] }];
set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports { led[7] }];
## FPGAOL BUTTON & SOFT_CLOCK
set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports { rst }];
```

将生成的 bit 文件烧写入 FPGA 内, 可以看到 8 个 LED 在有规律地闪烁, 即为计数器的效果, 如下图 3. 如果将计数器改为 32 位的, 那么 LED 闪烁的频率会变为 30 位的四分之一.

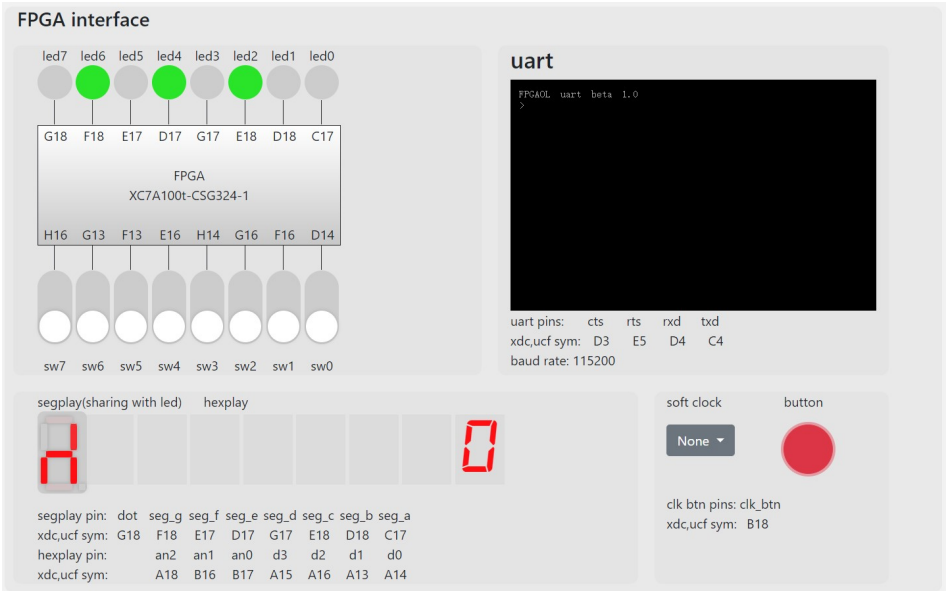

图 3: FPGA 烧写后的效果

【总结与思考】

收获 学习了如何进行简单的 FPGA 开发

难易程度 中等

任务量 轻松

建议 暂无