長安大學

数字逻辑实验报告

课程名称： 数字逻辑

实验名称： 门电路的 FPGA 实现

专业名称： 计算机科学与技术/卓越工程师

小组成员： 刘文越、王子卓、孙翔、康旭

指导教师： 马峻岩

一、实验目的

1. 了解 Vivado 开发环境及基本操作；

2. 掌握 Basys3 开发板实验的基本流程；

3. 掌握电路的综合和实现；

4. 掌握电路仿真与时序分析；

主要仪器和设备：计算机，Basys3 开发板。

二、实验要求

1. Verilog 实现基本门电路

（1）功能要求

① Verilog 实现反相器，sw0 输入，led0 输出。

② 2 输入与门，sw0、sw1 输入，led1 输出。

③ 2 输入或门，sw0、sw1 输入，led2 输出。

④ 2 输入与非门，sw0、sw1 输入，led3 输出。

⑤ 2 输入或非门，sw0、sw1 输入，led4 输出。

⑥ 2 输入异或门，sw0、sw1 输入，led5 输出。

⑦ 2 输入同或门，sw0、sw1 输入，led6 输出。

（2）其他要求

① 使用**数据流级建模**方式实现。

② 要求撰写仿真程序，对代码进行仿真测试。

③ 将仿真后的 Verilog 代码进行综合与实现，并下载到 Basys3 上验证。

2. Verilog 实现 4 选 1 数据选择器

（1）功能要求

① 数据选择控制信号 s0-s1，sw0, sw1。

② 数据输入信号 d0-d3，sw2 – sw5。

③ 数据输出信号 y，led0。

（2）其他要求

① 使用**门级建模**方式实现 1 个 2 选 1 数据选择器，然后通过实例化 3 个 2 选 1 数据选择器实现 4 选 1 数据选择器。

② 要求撰写仿真程序，对代码进行仿真测试。

③ 将仿真后的 Verilog 代码进行综合与实现，并下载到 Basys3 上验证。

三、相关外设接口及原理

1. Basys3 开发板拨码开关原理图及简介

拨码开关的电路如图 3-1 所示。当开关打到下档时，FPGA 对应引脚输入为低电平；当开关打到上档时，FPGA 对应引脚输入为低电平。

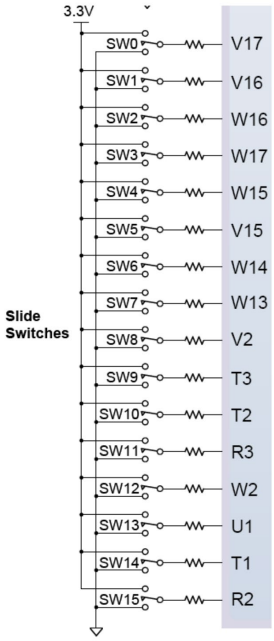

图 3-1 板拨码开关原理图

2. Basys3 开发板 LED 原理图及简介

LED 部分的电路如图 3-2 所示。当 FPGA 对应引脚输出为高电平时，相应的 LED 点亮；当 FPGA 对应引脚输出为低电平时，相应 LED 熄灭。

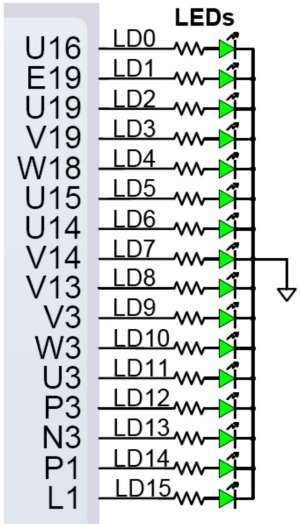

图 3-2 LED 原理图

四、实验步骤及关键代码

1. Vivado 中开发 FPGA 的流程

（1）在 Vivado 中新建项目，选择对应的 FPGA 型号。（BASYS3 即选择 xc7a35tcpg236-1）

（2）编写 Verilog 模块代码，实现对应的功能。

（3）对于模块代码添加对应的顶层文件，注意顶层文件中的输入输出接口需和约束文件中的引脚编号保持一致。

（4）创建仿真文件，编写 initial 时序仿真块，并将其设置为顶层文件，运行仿真并查看波形是否符合预期。

（5）运行综合，查看电路图是否符合预期。

（6）添加约束文件，使用开发板提供的官方约束文件，通过注释/取消注释的方式选择对应的引脚。

（7）运行实现，将综合后的基本单元转为对应芯片上的单元。

（8）生成比特流文件，并对开发板进行编程。

（9）验证开发板。

2. Verilog 实现基本门电路

（1）主要模块代码

①设计文件：

basic_gates.v

```
`timescale 1ns / 1ps

module basic_gates(
    input D0, D1,
    output [6: 0] LED
    );
    assign LED[0] = ~D0;
    assign LED[1] = D0 & D1;
    assign LED[2] = D0 | D1;
    assign LED[3] = ~(D0 & D1);
    assign LED[4] = ~(D0 | D1);
    assign LED[5] = D0 ^ D1;
    assign LED[6] = ~(D0 ^ D1);

endmodule
```

②顶层文件：

basic_gates_top.v

```
`timescale 1ns / 1ps

module basic_gates_top(
    input [1: 0] sw,
    output [6: 0] led
    );

    basic_gates(
        .D0(sw[0]),
        .D1(sw[1]),
        .LED(led)
        );

endmodule
```

（2）仿真测试代码

basic_gates_simulation.v

```
`timescale 1ns / 1ps

module basic_gates_simulation(
    );
    reg D0, D1;
    wire [6: 0] LED;

    basic_gates bg(
        .D0(D0),
        .D1(D1),
        .LED(LED)
        );

    initial begin
        #0
        D0 = 0;
        D1 = 0;
        #10
        D0 = 1;
        D1 = 0;
        #10
        D0 = 0;
        D1 = 1;
```

```
        #10
        D0 = 1;
        D1 = 1;
        #10
        $finish;
    end

endmodule
```

（3）仿真波形图

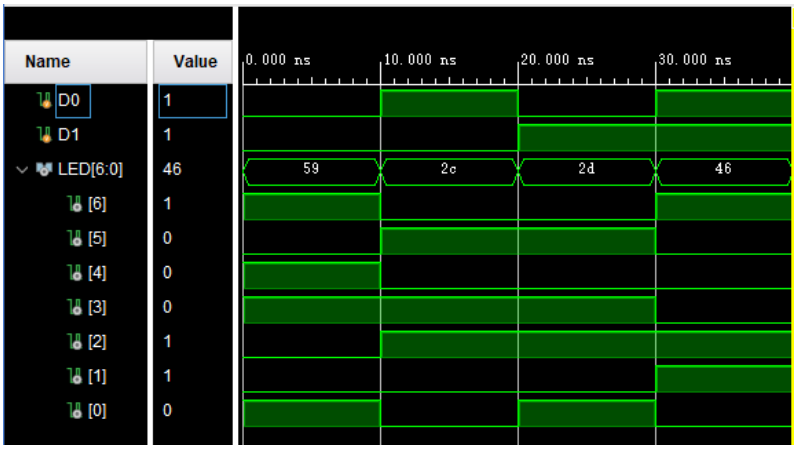

图 4-1 基本门电路仿真波形图

（4）开发板测试照片

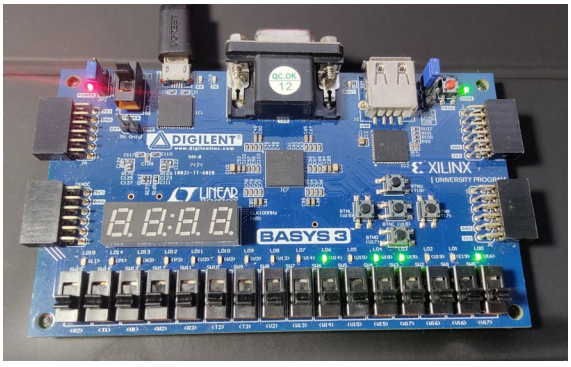

图 4-2 sw[0]为 0，sw[1]为 0

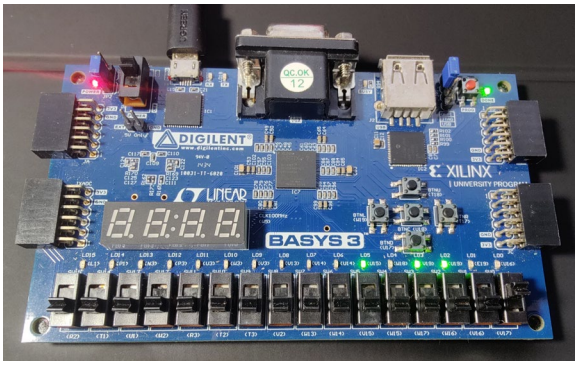

图 4-3 sw[0]为 1，sw[1]为 0

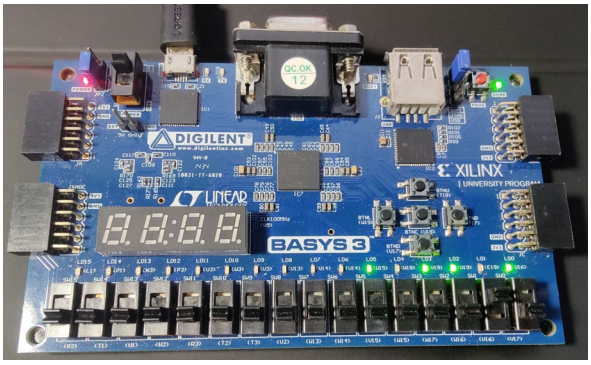

图 4-4 sw[0]为 0，sw[1]为 1

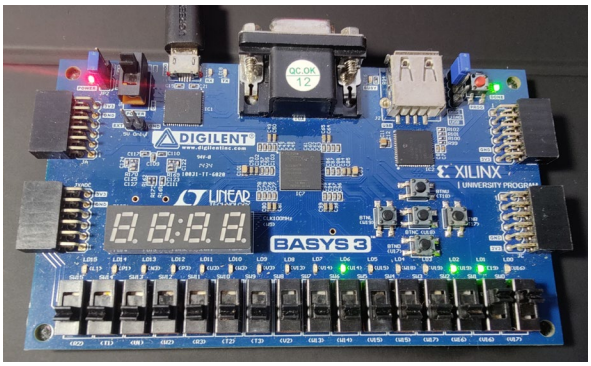

图 4-5 sw[0]为 1，sw[1]为 1

3. Verilog 实现 4 选 1 数据选择器

（1）主要模块代码

①设计文件：

mux21.v

```
`timescale 1ns / 1ps

module mux21(
    input D0, D1, SEL,
    output Y
    );
    not (a, SEL);
    and (b, D0, a);
    and (c, D1, SEL);
    or (Y, b, c);

endmodule
```

mux41.v

```
`timescale 1ns / 1ps

module mux41(
    input D0, D1, D2, D3,
```

```
    input [1: 0] SEL,
    output Y
    );

    wire x, y;
    mux21 a(.D0(D0), .D1(D1), .SEL(SEL[0]), .Y(x));
    mux21 b(.D0(D2), .D1(D3), .SEL(SEL[0]), .Y(y));
    mux21 c(.D0(x), .D1(y), .SEL(SEL[1]), .Y(Y));
endmodule
```

②顶层文件

mux41_top.v

```
`timescale 1ns / 1ps

module mux41_top(
    input [5: 0] sw,
    output [0: 0] led
    );

    mux41(
        .D0(sw[2]),
        .D1(sw[3]),
        .D2(sw[4]),
        .D3(sw[5]),
        .SEL(sw[1: 0]),
        .Y(led)
    );
endmodule
```

（2）仿真测试代码

mux21_simulation.v

```
`timescale 1ns / 1ps

module mux21_simulation(
    );

    reg D0, D1, SEL;
    wire Y;
    mux21 m21(
        .D0(D0),
        .D1(D1),
        .SEL(SEL),
        .Y(Y)
```

```
    );

    initial begin
        #0
        D0 = 0;
        D1 = 0;
        SEL = 0;
        #10
        D0 = 1;
        D1 = 0;
        SEL = 0;
        #10
        D0 = 0;
        D1 = 1;
        SEL = 0;
        #10
        D0 = 1;
        D1 = 1;
        SEL = 0;
        #10
        D0 = 0;
        D1 = 0;
        SEL = 1;
        #10
        D0 = 1;
        D1 = 0;
        SEL = 1;
        #10
        D0 = 0;
        D1 = 1;
        SEL = 1;
        #10
        D0 = 1;
        D1 = 1;
        SEL = 1;
        #10
        $finish;
    end
endmodule
```

mux41_simulation.v

```
`timescale 1ns / 1ps

module mux41_simulation(
    );
```

```
    reg D0, D1, D2, D3;
    reg [1: 0] SEL;
    wire Y;

    mux41 m41(
        .D0(D0),
        .D1(D1),
        .D2(D2),
        .D3(D3),
        .SEL(SEL),
        .Y(Y)
    );

    initial begin
        #0
        D0 = 0;
        D1 = 0;
        D2 = 0;
        D3 = 0;
        SEL[0] = 0;
        SEL[1] = 0;
        #10
        D0 = 1;
        D1 = 0;
        D2 = 0;
        D3 = 0;
        SEL[0] = 0;
        SEL[1] = 0;
        #10
        D0 = 0;
        D1 = 0;
        D2 = 0;
        D3 = 0;
        SEL[0] = 1;
        SEL[1] = 0;
        #10
        D0 = 0;
        D1 = 1;
        D2 = 0;
        D3 = 0;
        SEL[0] = 1;
        SEL[1] = 0;
        #10
        D0 = 0;
```

```
        D1 = 0;
        D2 = 0;
        D3 = 0;
        SEL[0] = 0;
        SEL[1] = 1;
        #10
        D0 = 0;
        D1 = 0;
        D2 = 1;
        D3 = 0;
        SEL[0] = 0;
        SEL[1] = 1;
        #10
        D0 = 0;
        D1 = 0;
        D2 = 0;
        D3 = 0;
        SEL[0] = 1;
        SEL[1] = 1;
        #10
        D0 = 0;
        D1 = 0;
        D2 = 0;
        D3 = 1;
        SEL[0] = 1;
        SEL[1] = 1;
        #10
        $finish;
    end

endmodule
```

（3）仿真波形图

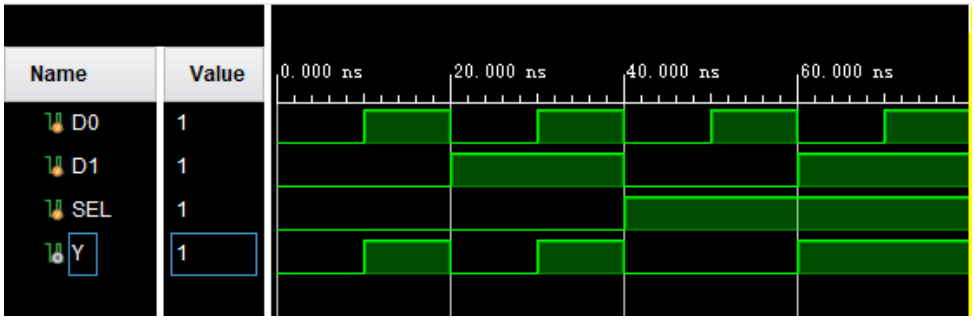

图 4-6 mux21 仿真波形图

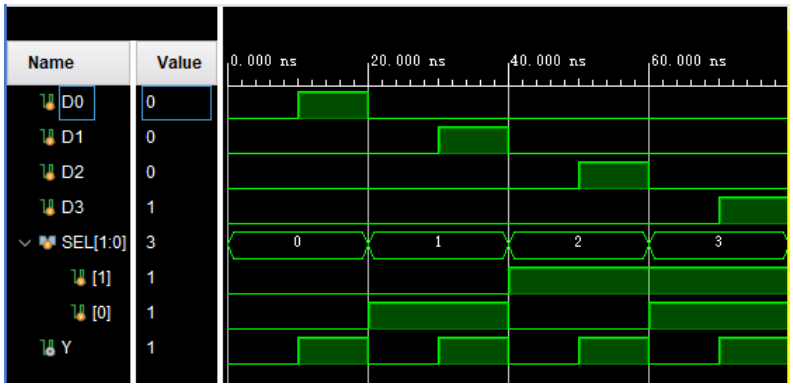

图 4-7 mux41 仿真波形图

（4）开发板测试照片

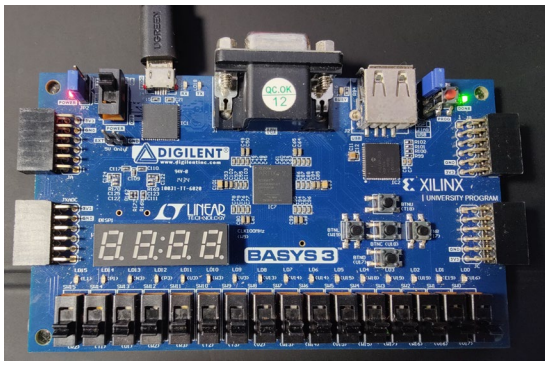

图 4-8 sw[0]为 0， sw[1]为 0，sw[2]为 0，其余均为 0

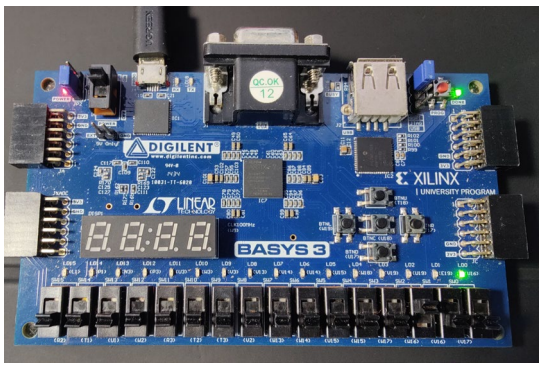

图 4-9 sw[0]为 0， sw[1]为 0，sw[2]为 1，其余均为 0

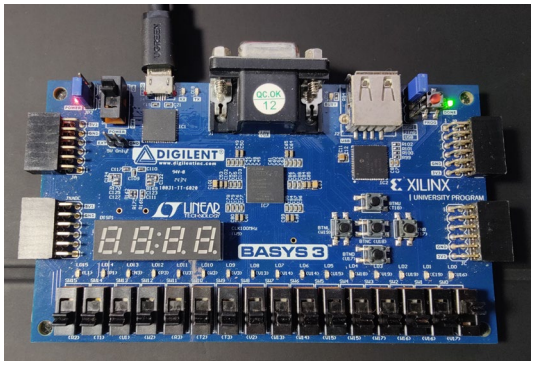

图 4-10 sw[0]为 1， sw[1]为 0，sw[3]为 0，其余均为 0

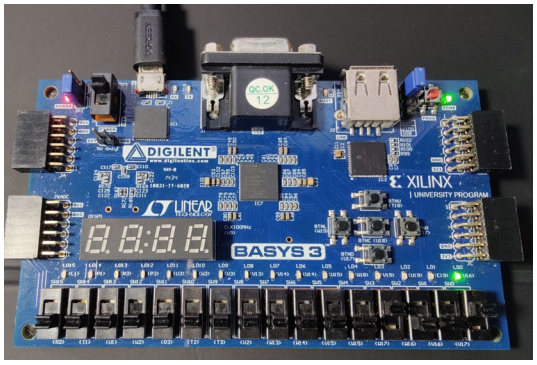

图 4-11 sw[0]为 1， sw[1]为 0，sw[3]为 1，其余均为 0

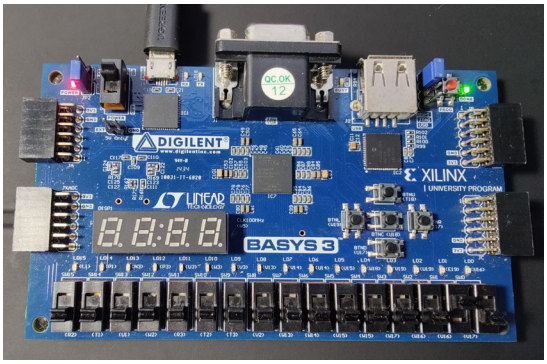

图 4-12 sw[0]为 0， sw[1]为 1，sw[4]为 0，其余均为 0

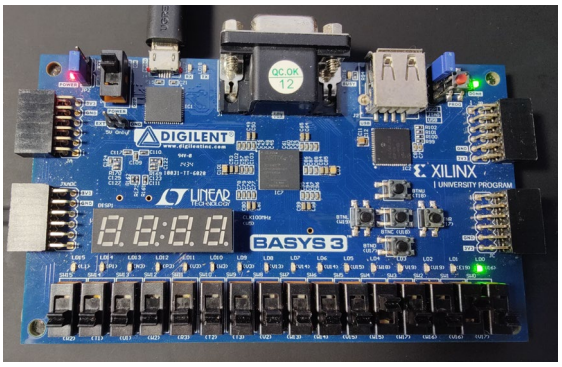

图 4-13 sw[0]为 0， sw[1]为 1，sw[4]为 1，其余均为 0

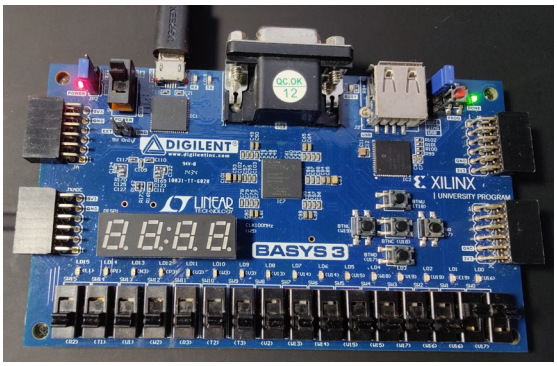

图 4-14 sw[0]为 1， sw[1]为 1，sw[5]为 0，其余均 0

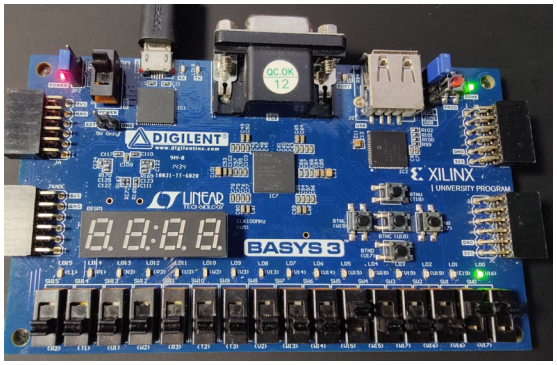

图 4-15 sw[0]为 1， sw[1]为 1，sw[5]为 1，其余均 0

五、思考问题

1. Verilog 名词解释：网线，寄存器，向量，连续赋值语句。

（1）网线：

表示物理元件之间的连线；

（2）寄存器：

表示抽象的数据存储元件；

（3）向量：

①向量表示： reg [7:0] 向量名。

②向量的调用：在向量名后添加对应索引即可调用对应位，可以直接对整个向量赋值，也可以只对向量中的某些位赋值。

（4）连续赋值语句：

连续性赋值语句是数据流建模的基本语句，用于对线网（wire）进行赋值。连续性赋值语句，只要输入端操作数的值发生变化，该语句就重新计算并刷新赋值结果，通常可以使用连续性赋值语句来描述组合逻辑电路，而不需要用门级电路和互联线。连续赋值的目标主要是标量线网和向量线网。不能是标量或向量寄存器。标量线网如“wire a, b;”，向量线网如“wire [3:0] a, b;”。

2. Verilog 中端口与外部信号连接的两种方式。

端口列表中的所有端口必须在模块中进行声明，verilog 中的端口具有以下三种了类型：input、output、和 inout。在 verilog 中，所有的端口隐含地声明为 wire 类型，因此如果希望端口具有 wire 数据类型，将其声明为三种类型之一即可：如果输出类型的端口需要保存数值，则必须将其显式的声明为 reg 数据类型。

不能将 input 和 inout 类型的端口声明为 reg 数据类型，这是因为 reg 类型的变量是用于保存数值的，而输入端口只反映与其相连的外部信号的变化，并不能保存这些信号的值。

3. 约束文件的作用。

定义管脚约束、时钟约束，以及其他时序约束。可以将文件中的输入输出与真实的开发板上的引脚相连。

4. 为什么要进行仿真。

进行仿真测试。在描述完电路之后，我们需要进行对代码进行验证，主要是进行功能验证，在仿真测试后能够有效验证功能准确性。