北京航空航天大學

BEIHANG UNIVERSITY

实验一、初识实验板——流水灯与计数器

2023 年 10 月 19 日

学号：20231023

姓名：龙熙

网络空间安全学院

目录

实验一、初识实验板——流水灯与计数器

1 实验指导

通过一个实际的数字逻辑电路的实现案例，熟悉型号为 ALINX XILINX Artix7 A7 XC7A35T HDMI 的 FPGA 实验硬件实验平台（以下简称“实验板”），以及按键、八段数码管、LED 灯等外设；练习通过 Verilog HDL 语言实现 FPGA 可综合代码，以及运用测试程序进行功能仿真的方法；熟悉 FPGA 综合和加载的过程，并实践相应的软硬件调试。

1.1 实验板的硬件资源

实验板核心的 FPGA 芯片型号为 XC7A35T-2FGG484I，具有约 4 万个可配置逻辑单元（CLB），1800kb（“b”表示比特，“B”表示字节，下同）的块 RAM，以及 90 个 DSP 处理单元（DSP Slices）。实验板上集成了编程芯片，通过 JTAG 以菊花链的形式连接 PROM 和 FPGA 芯片，可以通过 USB 串口直接编程。

对于实验板编程有两种方式，(1)直接下载*.bit 文件对 FPGA 编程，这种情况下如果掉电则配置丢失；(2)下载*.bin 文件，对 Flash 编程，形成非易失实现。

1.2 背景知识——流水灯与计数器

1.2.1 流水灯

在“实验板”的右下角有 7 个 LED 指示灯（低电平点亮），其中 1 个是电源指示灯（PWR），1 个是配置指示灯（DONE），2 个是 USB Uart 的数据接收和发送指示灯， 4 个是用户 LED 灯 （LED1~LED4）。FPGA 可以通过引脚的高或低电平控制 LED 的亮灭状态。在本次实验中，通过 FPGA 内部的定时器，循环点亮 4 个用户 LED 灯（LED1~LED4），达到流水灯的效果。如图 1 所示，4 个 LED 指示灯，依次给它们赋值，每次只有一个 LED 点亮，每次点亮某个 LED 的时间一定（固定延时）。4 个 LED 依次被点亮一次，如此循环便实现了流水灯的效果。

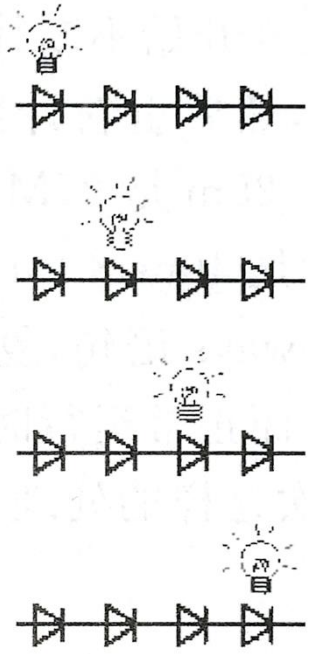

图 1 流水灯示意图

1.2.2 计数器

通过开发板实现计数功能是 FPGA 最基本的操作之一，主要分为**自动计数（比如通过对时钟上升沿计数）**和**手动计数（通过按键计数）**两种方式。在本实验中，我们将通过板上按键 Key1 实现计数，并借助此实验告诉大家数码管的工作原理。

数码管是工程设计中使用很广的一种显示输出器件。一个 7 段数码管（如果包括右下的小点可以认为是 8 段）分别由 a、b、c、d、e、f、g 位段和表示小数点的 dp 位段组成。实际是由 8 个 LED 灯组成的，控制每个 LED 的点亮或熄灭实现数字显示。通常数码管分为共阳极数码管和共阴极数码管，结构如下图所示：

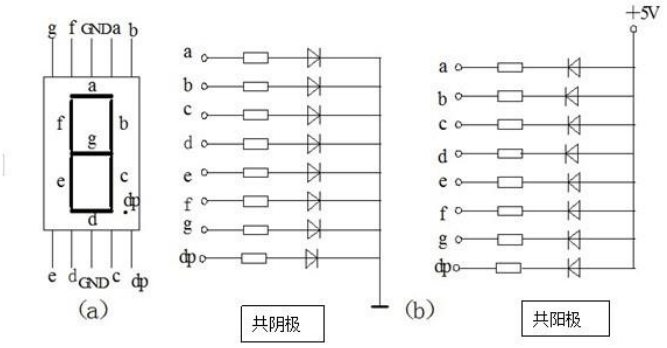

图 2 数码管原理

数码管所有的信号都连接到 FPGA 的管脚，作为输出信号控制。FPGA 只要输出这些信号就能够控制数码管的那一段 LED 亮或者灭。通过组合逻辑的输出来控制数码管

显示数字，下面是数码管显示的表格:

输出码（共阳极）								字型
dp	g	f	e	d	c	b	a	
1	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	1	1
1	0	1	0	0	1	0	0	2
1	0	1	1	0	0	0	0	3
1	0	0	1	1	0	0	1	4
1	0	0	1	0	0	1	0	5
1	0	0	0	0	0	1	0	6
1	1	1	1	1	0	0	0	7
1	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	0	9

图 3 数码管数字对应表

除了显示 0-9 的数字之外，我们还可以通过小数点将输出范围扩大到 0.1-99，这就需要我们使用两个 8 段 led 数码管来执行。我们的开发板通过 SEL0 和 SEL1 两个引脚（共有 6 个数码管）来对两段数码管进行使能，每次只能使能一段数码管，通过快速切换使能的数码管让其高速交替发光，从而达到两段数码管同时发光的效果。

1.3 基本实验要求

本实验的目的主要以熟悉和掌握实验板为主，学有余力的同学们，可以在基本和扩展要求下，充分发挥设计能力进行改进，或者甚至是全面修正。

基本实验要求为：

a) 结合指导书中提供的背景知识，设计出流水灯模块，在《实验报告》中给出关键设计步骤；

b) 编写测试用例，对流水灯模块进行功能仿真测试；

c) 对流水灯模块电路进行综合(Synthesis)和实现(Implementation)，并把代码下载到“实验板”上，包括烧录易失性的程序和非易失性的程序。

提示：原始时钟的频率高达 50MHz，如果 LED 灯切换的频率与时钟频率相同，实

验现象可能会不明显；建议用一个计数器降低时钟的频率（时钟分频），让 LED 灯切换的频率降下来，使实验现象更明显。

请注意，所谓的思考和改进在一定程度上并没有唯一解，往往是多种解决方案的权衡和比较。

由于本实验阶段为“初识”，因此仅以实验报告为考核形式，具体**截止期限**请注意任课教师的通知，实验报告格式请参考本文档。

后续的实验将采取**实验报告和现场汇报演示相结合**的形式，特此广而告之。

1.4 扩展实验要求

扩展的实验要求（根据完成的程度可酌情提高评价的等级）

d) 结合指导书中提供的背景知识和提供的按键消抖代码（消抖模块将按键波形整合为**一个时钟周期的高电平脉冲**），设计出用按键（KEY4）控制的**手动计数**计数器模块，计数器要求使用 SEL0 对应的数码管 1（**共阳极**）。（有兴趣的同学可以扩展计数器为两位数码管（选用 SEL0 和 SEL1），计数计到 19）在《实验报告》中给出关键设计步骤；

e) 编写测试用例，对计数器模块进行功能仿真测试；

f) 对计数器模块电路进行综合(Synthesis)和实现(Implementation)，并把代码下载到“实验板”上，包括烧录易失性的程序和非易失性的程序。

1.5 已有软件资源

1.5.1 时钟分频

实验板可以产生 50MHz 的晶振时钟，过快的时钟会过多耗能，因此设置了时钟分频的过程块，Vivado 自带时钟 Clock Wizard 的 IP 核，可以根据需要调整时钟频率。

1.5.2 数码管

两个数码管采用动态刷新，需要用寄存器将它们显示的内容寄存起来，通过选通端 `SEL[1:0]`，交替刷新显示。

1.5.3 按键去抖

按键需要去抖，提供按键去抖模块 Key 可供利用。

1.5.4 引脚

请结合附件中《AX7035 开发板用户手册 REV1.1》第 29 页查对数码管引脚分配，第 38 页查对按键 FPGA 引脚分配，第 39 页查对 LED 灯 FPGA 引脚分配。

1.5.5 实验板的 UART

提供一个只能每次读写一个字节的UART 的样例源代码，具有固定的 9600 波特率、8 位数据位、偶校验、1 位停止位。更好的 UART（注：更稳定可靠、波特率可配置、有 Telnet 等上层协议）需要用户自行开发。

实验一、初识实验板——流水灯与计数器

2 实验报告

2.1 实验背景与需求分析

流水灯和计数器。

2.2 系统设计

2.2.1 总体设计思路

流水灯设计：每次亮一个 led 灯，一秒后跳到下一个 led 灯，循环亮。

计数器设计：从 0 计数到 19，共两个数字显示，按键 key 实现加 1 功能，按键 reset 提供重置功能，计数归零。

2.2.2 接口设计

流水灯

lead_water.v

信号名	方向	功能描述
clk	I	时间信号
rst	I	重置信号
led_on[3:0]	O	控制四个 led 灯

计数器

count.v

信号名	方向	功能描述
clk	I	时间信号
rst	I	重置信号
key	I	按键计数信号

led[7:0]	O	控制显示数字的形状
sel[5:0]	O	控制哪个数字亮

debounce.v

信号名	方向	功能描述
clk	I	时间信号
rst	I	重置信号
key[N-1:0]	I	按键信号
key_pulse [N-1:0]	O	消抖后的按键信号

2.2.3 模块设计

led_water.v

```
module led_water(
    input wire clk,//时钟信号，50MHz
    input wire rst_n,//复位信号，下降沿有效
    output wire   [3:0] led_on
);

parameter MAXS = 26'd50_000_000;
reg [25:0] cnt1s = 0;//计数寄存器器 0.5s
reg [3:0]     led_r = 4'b1110;//led 信号寄存器

always @(posedge clk or negedge rst_n) begin
    if (!rst_n ) begin//复位，重新计数
        cnt1s<=26'd0;
    end
    else if (cnt1s == MAXS)begin//记到最大数,重新计数
        cnt1s<=26'd0;
    end
```

```
        else begin//其他情况+1
            cnt1s <= cnt1s + 1'd1;
        end
end

always @(posedge clk or negedge rst_n) begin
        if (!rst_n ) begin//复位
                led_r = 4'b1110;
        end
        else if (cnt1s == MAXS)begin//记到最大数,翻转
                led_r = {led_r[2:0],led_r[3]};
        end
        else begin
                led_r = led_r;
        end
end

assign led_on = led_r;
endmodule
```

count.v

```
module count(
        input clk,
        input key,
        input rst,
        output reg [7:0] led,
        output reg [5:0] sel
        );
        reg [12:0] t;
```

```
reg op = 0; //时间分频
reg [4:0] cnt = 0; //计数
reg [3:0] num_1, num_0;   //提取个位、十位
wire key_pulse; //  消抖后
reg [7:0] led_r [0:9];

initial
    begin
        sel = 60;
        led_r[0] = 192;
        led_r[1] = 249;
        led_r[2] = 164;
        led_r[3] = 176;
        led_r[4] = 153;
        led_r[5] = 146;
        led_r[6] = 130;
        led_r[7] = 248;
        led_r[8] = 128;
        led_r[9] = 144;
    end

debounce u1 (
    .clk(clk),
    .rst(rst),
    .key(key),
    .key_pulse(key_pulse)
    );

//计数
```

```
always @(posedge clk or negedge rst) begin
    if(!rst) begin
        cnt <= 0;
    end
    else begin
        if(key_pulse)begin
            if(cnt == 5'd19) begin
                cnt <= 0;
            end
            else begin
                cnt <= cnt + 1;
            end
        end
        else begin
            cnt <= cnt;
        end
    end
end

//显示
always @(posedge clk) begin
    num_1 <= cnt / 10;
    num_0 <= cnt % 10;
    t <= t + 1;
    if(t == 0) begin
        op = ~op;
    end
    if(op) begin
        led <= led_r[num_1];
```

```
                sel[1] <= 0;
                sel[0] <= 1;
            end
            else begin
                led <= led_r[num_0];
                sel[0] <= 0;
                sel[1] <= 1;
            end
        end

endmodule
```

2.3 功能仿真测试

2.3.1 测试程序设计

流水灯 led_water_sim.v

原文件为了达到 1s 切换 led 的效果，MAXS 设置为 50_000_000，为了仿真更好观察，仿真时候设置为 5。

```
module led_water_sim;
    reg clk;
    reg rst;
    wire [3:0] led;

defparam uut.MAXS = 5;

    led_water uut(
        .clk(clk),
        .rst_n(rst),
```

```
        .led_on(led)
        );

    initial begin
        clk = 0;
        rst = 1;
    end

    always #1 clk = ~clk;
endmodule
```

计数器 count_sim.v

同理为了计数器更方便调试，调试时需要消抖模块的时间调小。

```
module count_sim;
    reg clk, rst, key;
    wire [7:0] led;
    wire [5:0] sel;

    count uut(
        .clk(clk),
        .key(key),
        .rst(rst),
        .led(led),
        .sel(sel)
        );

    initial begin
        rst = 0;
        clk = 0;
```

```
        key = 1;


        #10
        rst = 1;
    end


    always #1 clk = !clk;
    always #20 key = !key;



endmodule
```

2.3.2 功能仿真过程

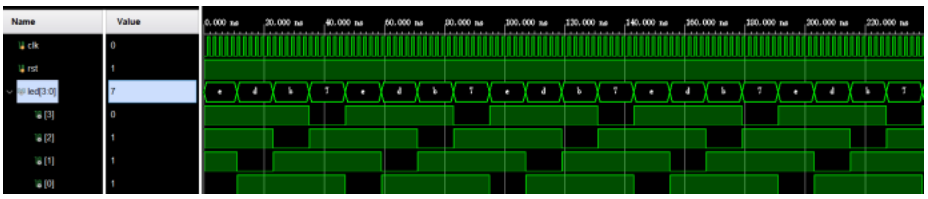

图 4 流水灯仿真

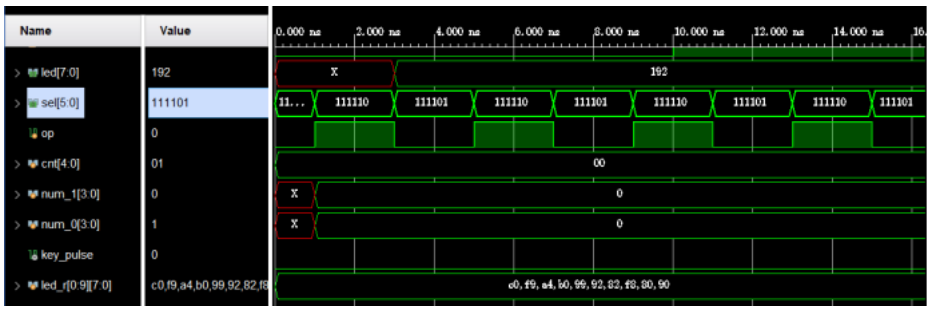

图 5 计数器仿真

2.4 设计实现

2.4.1 综合和下载过程

略

2.4.2 实验关键结果及其解释

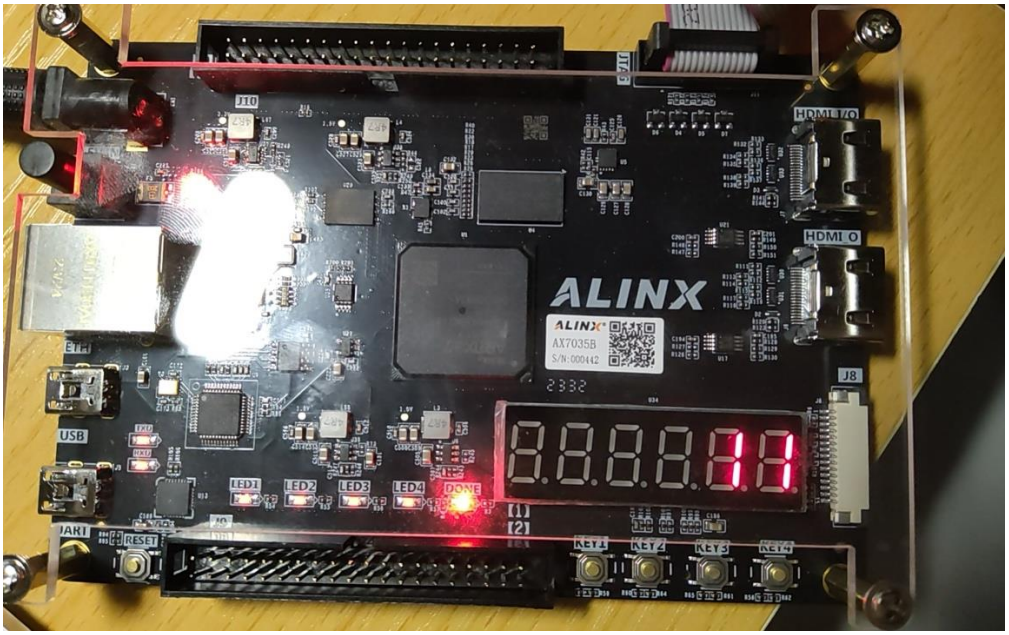

图 6 计数器实现

2.5 小结

参考文献

[1] （如果有）