课程名称：EDA 技术综合设计

设计报告名称：设计一 流水灯

班级：通信 214

姓名： 王峤宇

学号： 214022

一、设计内容及原理

设计流水灯, 流水灯的设计是时序逻辑电路, 需要对输入的 100MHz 时钟进行分频, 通过计数器进行分频后驱动 LED 灯的控制寄存器移位, 实现流水灯的效果。流水灯实现模块框图如图 1所示。

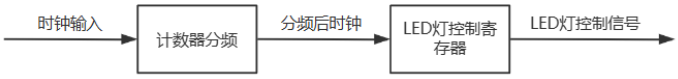

图 1: 流水灯模块框图

二、设计过程

流水灯源文件添加注释后如下所示:

Listing 1: 流水灯源文件

```
module flow_led (
    input clk,          /* 输入时钟 */
    input rst,          /* 复位信号 */
    output [15:0] led   /* 16个led的控制信号 */
    );
    reg [23:0] cnt_reg;     /* 计数器 */
    reg [15:0] light_reg;   /* led控制 */
    always @ (posedge clk) begin
        if(!rst)                    /* 同步复位 */
            cnt_reg <= 0;           /* 复位 */
        else                        /* 计数器计数 */
            cnt_reg <= cnt_reg + 1; /* 计数器++ */
    end
    always @ (posedge clk) begin
        if (!rst)                               /* 同步复位 */
            light_reg <= 16'h 0003;             /* 复位 */
        else if (cnt_reg == 24'hffffff) begin   /* 100MHz的2^24分频, 为5.9Hz */
            if (light_reg == 16'hc000)          /* 到达边缘复位 */
                light_reg <= 16'h0003;          /* 复位 */
            else                                /* 不到边缘则移位 */
                light_reg <= light_reg << 1;    /* 左移一位 */
        end
    end
    assign  led = light_reg; /* 输出led控制信号 */
endmodule
```

流水灯的仿真文件添加注释后如下所示:

Listing 2: 流水灯仿真文件

```
module flow_led_tb();
    reg clk;            /* 时钟信号 */
    reg rst;            /* 复位信号 */
    wire [15:0] led;    /* 16个LED */
    flow_led u0(        /* 例化流水灯控制模块 */
        .clk(clk),
        .rst(rst),
        .led(led)
    );
    parameter PERIOD = 10;  /* 指定时钟周期 */
    always begin            /* 生成时钟 */
        clk = 1'b0;
        #(PERIOD/2) clk = 1'b1;
        #(PERIOD/2);
    end
    initial begin           /* 完成复位初始化 */
        clk = 1'b0;
        rst = 1'b0;
        #100;
        rst = 1'b1;
        #100;
        rst = 1'b0;
        #100;
        rst = 1'b1;
    end
endmodule
```

流水灯的约束文件如下所示:

Listing 3: 流水灯约束文件

```
set_property PACKAGE_PIN K3 [get_ports {led[0]}]
set_property PACKAGE_PIN M1 [get_ports {led[1]}]
set_property PACKAGE_PIN L1 [get_ports {led[2]}]
set_property PACKAGE_PIN K6 [get_ports {led[3]}]
set_property PACKAGE_PIN J5 [get_ports {led[4]}]
set_property PACKAGE_PIN H5 [get_ports {led[5]}]
set_property PACKAGE_PIN H6 [get_ports {led[6]}]
set_property PACKAGE_PIN K1 [get_ports {led[7]}]
set_property PACKAGE_PIN K2 [get_ports {led[8]}]
set_property PACKAGE_PIN J2 [get_ports {led[9]}]
set_property PACKAGE_PIN J3 [get_ports {led[10]}]
set_property PACKAGE_PIN H4 [get_ports {led[11]}]
set_property PACKAGE_PIN J4 [get_ports {led[12]}]
set_property PACKAGE_PIN G3 [get_ports {led[13]}]
set_property PACKAGE_PIN G4 [get_ports {led[14]}]
set_property PACKAGE_PIN F6 [get_ports {led[15]}]
set_property PACKAGE_PIN P15 [get_ports rst]
set_property PACKAGE_PIN P17 [get_ports clk]
```

```
19 set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]
20 set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
21 set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
22 set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
23 set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
24 set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
25 set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
26 set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
27 set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
28 set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
29 set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
30 set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
31 set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
32 set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
33 set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
34 set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
35 set_property IOSTANDARD LVCMOS33 [get_ports clk]
36 set_property IOSTANDARD LVCMOS33 [get_ports rst]
```

三、仿真结果

图 2: 流水灯仿真结果

流水灯的仿真结果如图 2所示, 可以看到 16 位的 LED 灯不断流水循环移位。

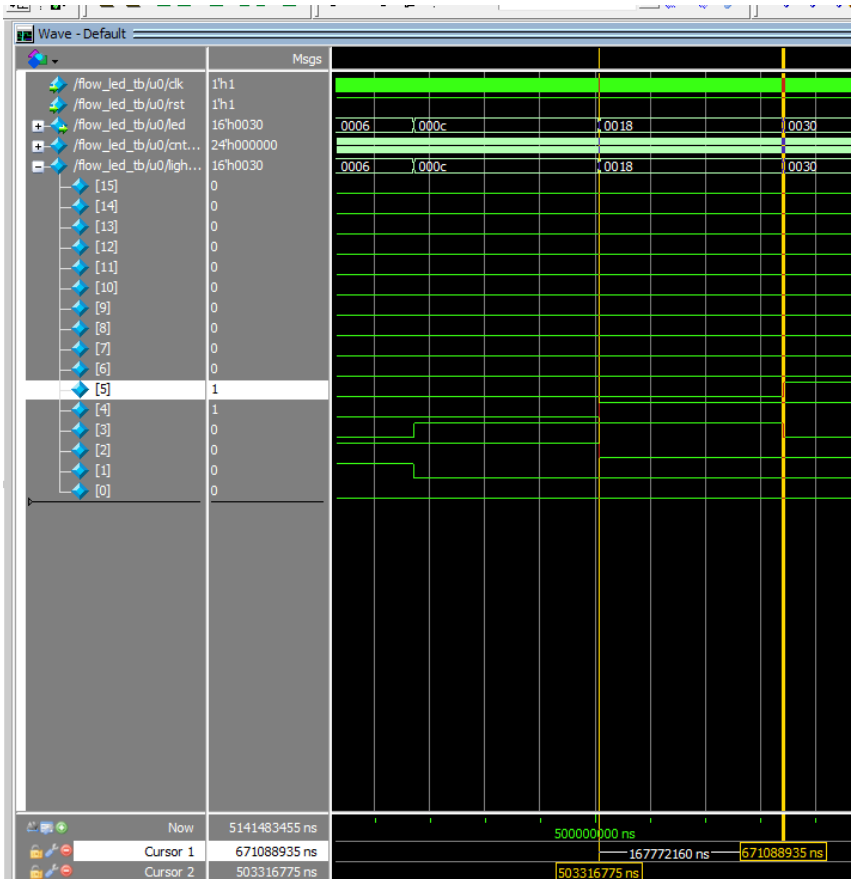

图 3: 流水灯仿真时序观察

针对流水灯时序结果的仿真观察如图 3所示, 设计的分频器为 24 位, 对 100MHz 进行 2^{24} 分频, 分频结果如下所示, 与仿真结果一致。

$$f = \frac{100 * 10^6}{2^{24}}\text{Hz} = 5.960\,46\,\text{Hz} \tag{1}$$

$$T = \frac{1}{f} = 0.167\,772\,16\,\text{s} \tag{2}$$

四、硬件验证结果

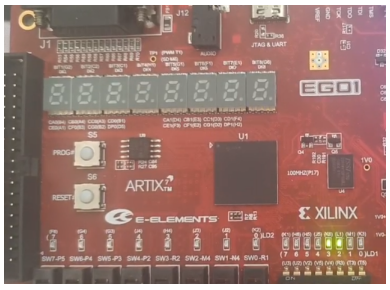

图 4: 硬件验证

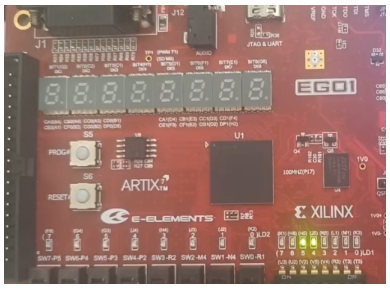

图 5: 硬件验证

硬件验证结果如图图 4和图 5所示, 其中图 5晚于图 4, 可以看到 LED 两个点亮, 每次步进一个灯的长度, 循环流水。

五、问题解决

程序固化到 Flash 中

解决: 查阅资料, 多次尝试。Vivado 默认的 program device 是通过 jtag 口下载程序, 将程序写入 FPGA 的 RAM 之中, 重新上电就会丢失。为了让 FPGA 重新上电后依然能够运行指定程序, 需要借助 FPGA 外部的 FLASH 固化程序, 上电时 FPGA 从其中读取程序。

利用 Vivado 将程序固化到 FLASH 需要生成 MCS(Memory Configuration File), 在 Vivado 的 tools 工具栏中, 这个文件主要用于指定 FLASH 类型, 以便 PC 使用特定的协议与 FLASH 通信。

查阅开发板手册, 手册中写到使用的 FLASH 为 N25Q32 3V3, 进一步查看 IC 手册, 确定 MCS 文件生成。

烧录失败, 提示检测到的芯片为 N25Q64, 检查芯片丝印, 确定为 N25Q64, 重新生成 MCS 固化, 查找原因, N25 常用的是 SPIx4, 但没办法直接设置, 因为默认综合产生的 bit 流文件为 SPIx1, 更改 SPI 位宽设置需要在综合界面的 Tcl 中执行如下命令完成:

```
1 set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
```

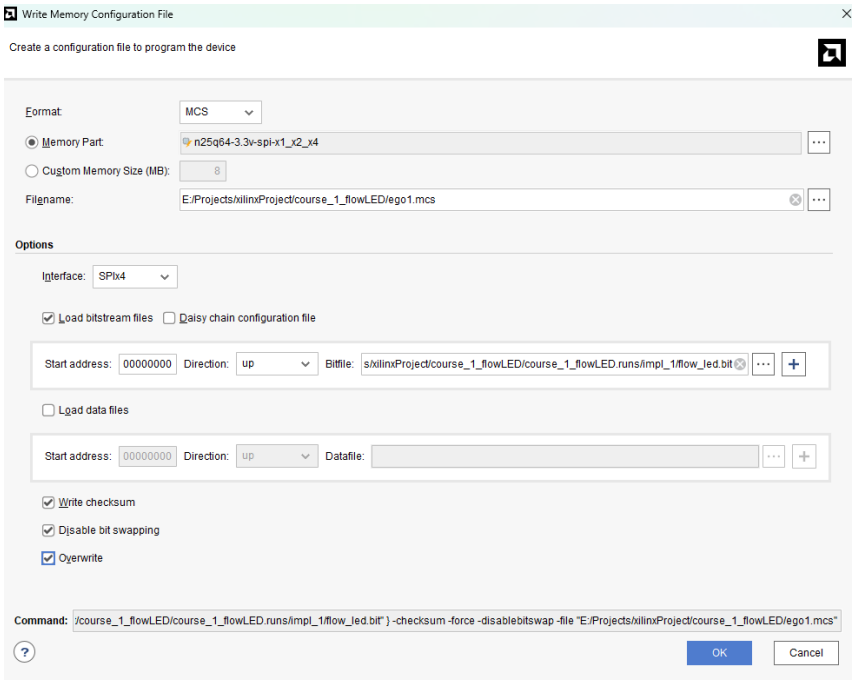

图 6: 正确的 MCS 配置

重新生成 MCS 文件如图 6所示, 最终烧录固化成功。

六、心得体会

通过一个简单的流水灯设计过程对数字电路基础所学内容进行复习回忆, 掌握 Vivado 工具的基本使用, 包括使用 Vivado 综合, 查看综合结果, 查看布局布线结果, 或是进行时序仿真, 通过 JTAG 接口进行硬件调试, 时序约束和 I/O 约束等等功能都被集成到了 Vivado 工具之中。其中 Vivado 还提供了 IP 封装以及 Xilinx 提供的 IP 核的配置和调用, 常见的有 FIFO, 时钟等 IP 核的提供。

了解到学习通中提供的众多资料中, 有很多是本课程多年积累的成果, 其中很多针对实际问题的案例和解析, 在学习的过程中可以利用这些资源辅助, 将工程实践的实际问题与理论学习相结合, 可以提高针对工程应用的学习效率。