

《数字电路实验》实验报告

实验名称: LED 灯流水实验与按键控制 LED 灯实验 指导教师: 高兴宇

姓名: 尹超 学号: 2023K8009926003 专业: 人工智能 班级: 2313 组号: 12

实验日期: 2025.4.23 实验地点: 阶一 2 是否调课/补课: 否 成绩:

小组成员: 尹超 张硕 李雨恒

目录

1 实验仪器与用具	2
2 实验内容与步骤	3
2.1 概述	3
2.2 实验步骤	3
3 实验结果	12
3.1 实验结果 1: LED 流水灯实验	12
3.2 实验结果 2: 按键控制 LED 灯实验基础问题	12
3.3 实验结果 3: 按键控制 LED 灯实验变式 1	12
3.4 实验结果 4: 按键控制 LED 灯实验变式 2	12
4 实验总结	12
4.1 实验中的问题与感想	12
5 附录: Verilog 代码	13

§1 实验仪器与用具

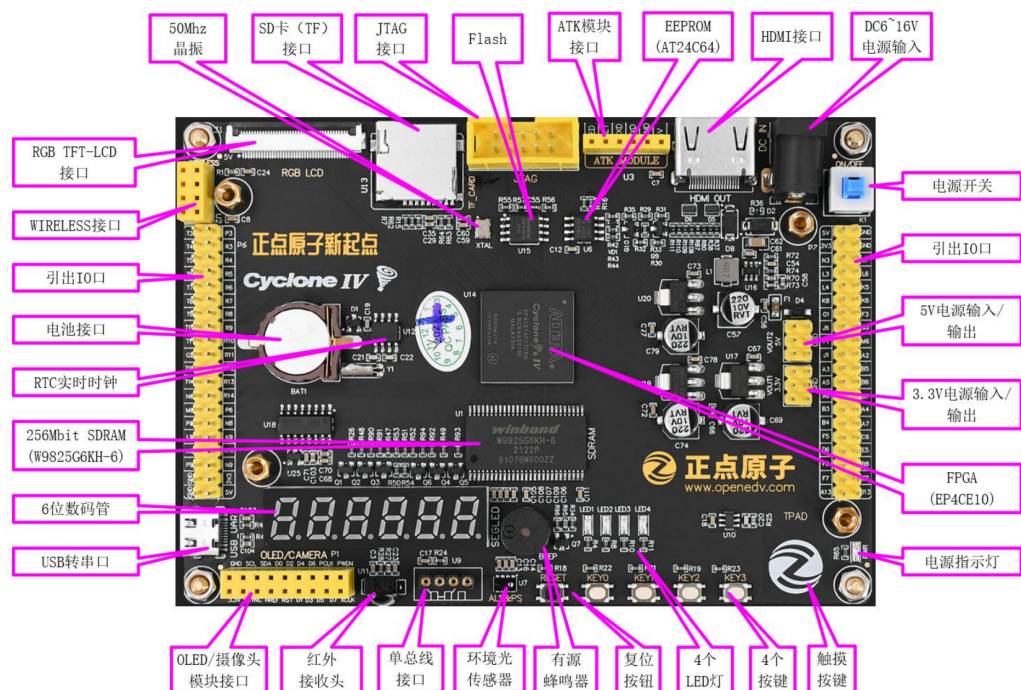


图 1: 实验仪器与用具

新起点 FPGA 开发板板载资源如下:

- 主控芯片: EP4CE10F17C8N, 封装: BGA256
- 晶振: 50 MHz
- FLASH: W25Q16, 容量: 16 Mbit (2 M 字节)
- SDRAM: W9825G6KH-6, 容量: 256 Mbit (32 M 字节)
- EEPROM: AT24C64, 容量: 64 Kbit (8 K 字节)
- 1 个电源指示灯 (蓝色)
- 4 个状态指示灯 (DS0 DS3: 红色)
- 1 个红外接收头, 并配备一款小巧的红外遥控器
- 1 个无线模块接口, 支持 NRF24L01 无线模块
- 1 路单总线接口, 支持 DS18B20/DHT11 等单总线传感器
- 1 个 ATK 模块接口, 支持正点原子蓝牙/GPS/MPU6050/RGB 灯模块
- 1 个环境光传感器, 采用 AP3216C 芯片
- 1 个标准的 RGB TFT-LCD 接口
- 1 个 OLED/摄像头模块接口
- 1 个 USB 串口
- 1 个有源蜂鸣器
- 1 个 SD 卡接口 (在板子背面)
- 1 个 HDMI 接口
- 1 个标准的 JTAG 调试下载口
- 1 组 5V 电源供应/接入口
- 1 组 3.3V 电源供应/接入口
- 1 个直流电源输入接口 (输入电压范围: DC6 16V)

- 1 个 RTC 后备电池座，并带电池（在板子背面）
- 1 个 RTC 实时时钟，采用 PCF8563 芯片
- 1 个复位按钮，可作为 FPGA 程序执行的复位信号
- 4 个功能按钮
- 1 个电容触摸按键
- 1 个电源开关，控制整个开发板的电源
- 两个 20×2 扩展口，共 72 个扩展 IO 口（除去电源和地）

§ 2 实验内容与步骤

2.1 概述

LED 流水灯实验：

- A. 按照指南开发指南第八章（P198）复现 LED 流水灯实验，仿真部分暂时不用做
- B. 固化程序到开发板中，实现断开下载器、电源后重启仍可实现 LED 流水灯

按键控制 LED 灯实验：

- A. 按照指南开发指南第九章（P206）复现按键控制流水灯实验，仿真部分暂时不用做
- B. 修改原实验的按键逻辑，使其分别实现下述功能：
 - (1) 按键 1, 2, 4 保持原功能不变，按下按键 3 会使 LED 灯按照从左向右的顺序流水，时间间隔设置为 1s
 - (2) 按下对应按键，松开按钮 LED 也会保持按键对应的逻辑运行（用寄存器存储上一个按下的按钮状态）

2.2 实验步骤

(1) 新建 Project

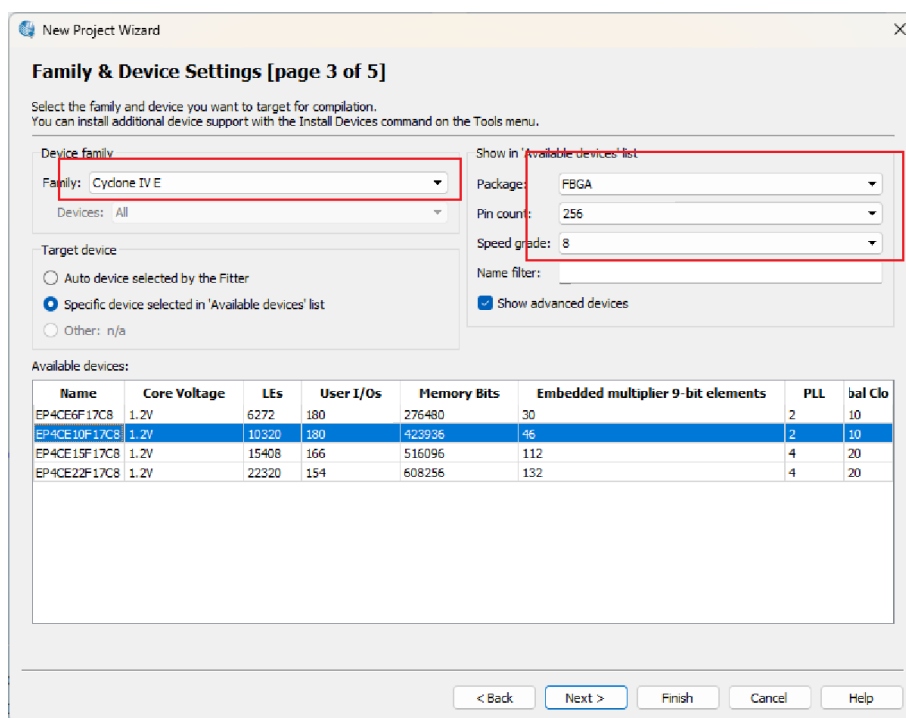


图 2: 新建 Project 的操作界面

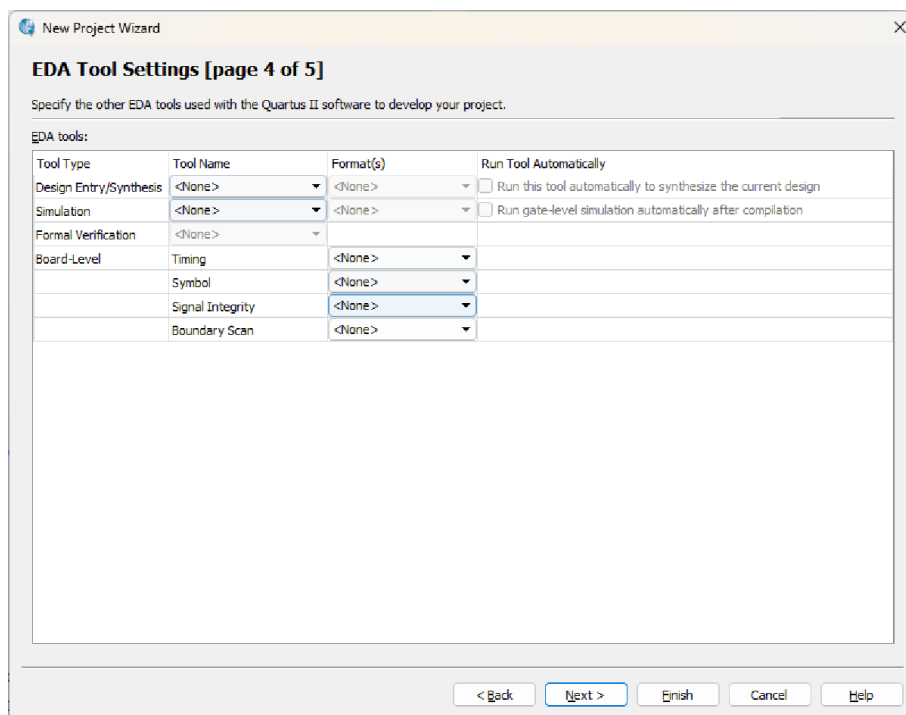
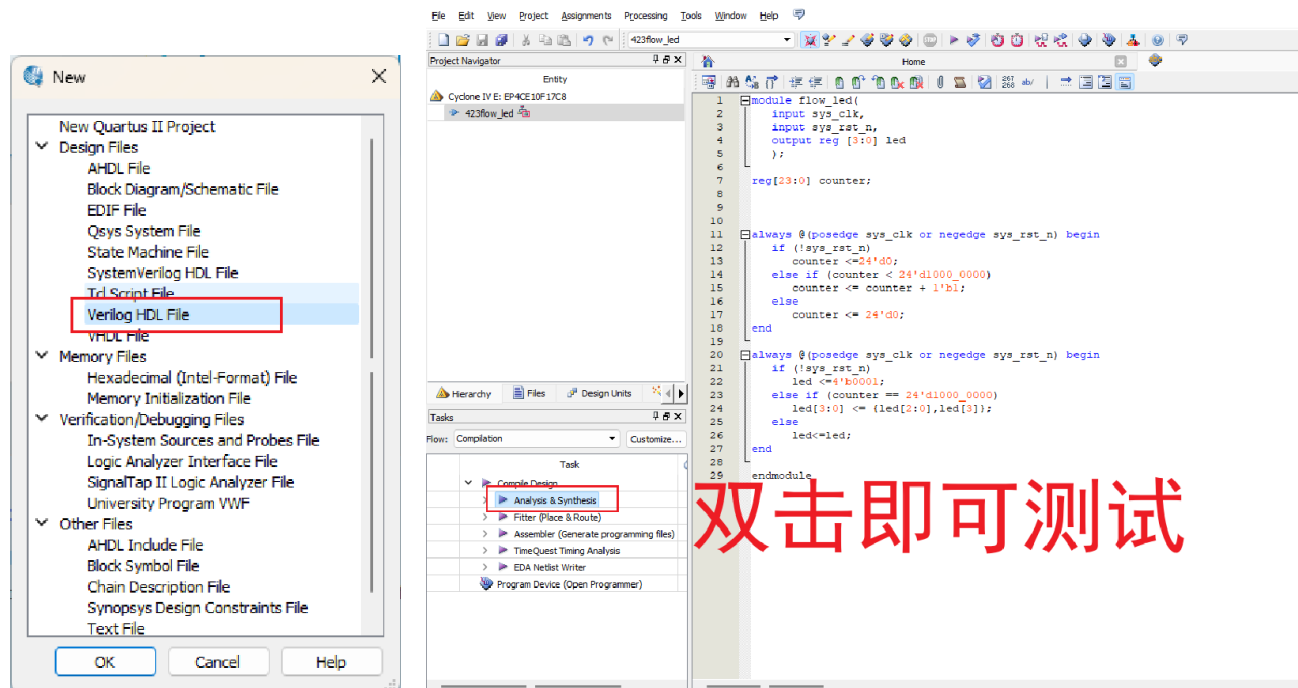


图 3: 新建 Project 的操作界面

说明: (此处填写新建工程的详细步骤和注意事项)

(2) 新建.v 文件, 编写代码并测试编译



(a) 新建 Verilog 文件

(b) 代码编写界面

说明: (此处填写代码编写、保存及测试编译的详细说明)

(3) 预编译, 确认无误

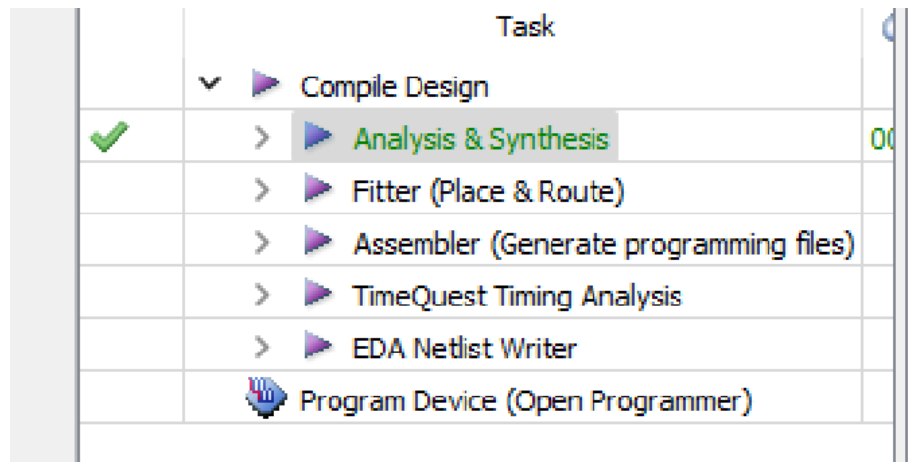


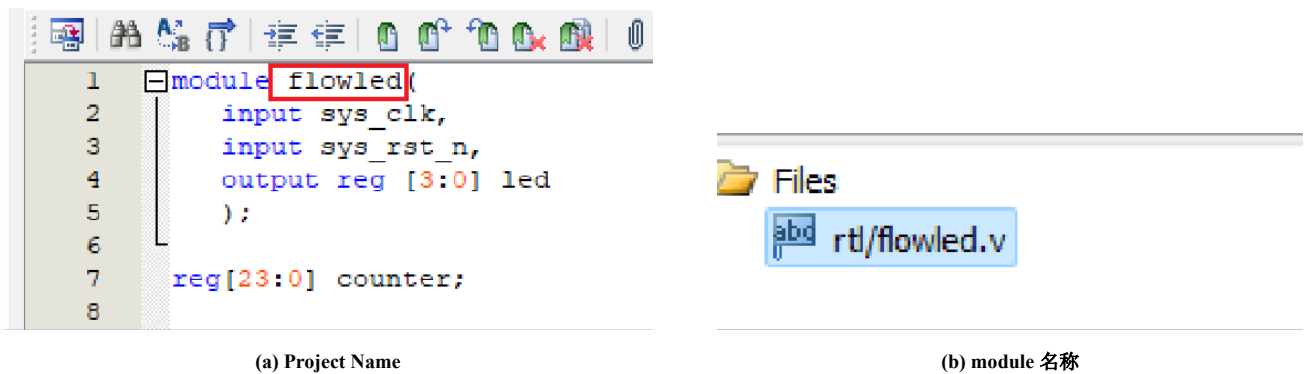
图 5: 预编译结果界面 1

```
> 286030 Timing-Driven Synthesis is running
> 16010 Generating hard_block partition "hard_block:auto_generated_inst"
> 21057 Implemented 48 device resources after synthesis - the final resource count might be different
> Quartus II 64-Bit Analysis & Synthesis was successful. 0 errors, 0 warnings
```

图 6: 预编译结果界面 2

说明: (此处填写预编译通过的判断方法及常见问题)

(4) 注意 Project Name 和代码中的 module 名称一致



说明: (此处说明为何需要一致及如何检查)

(5) 保存代码文件至 rtl 文件夹内

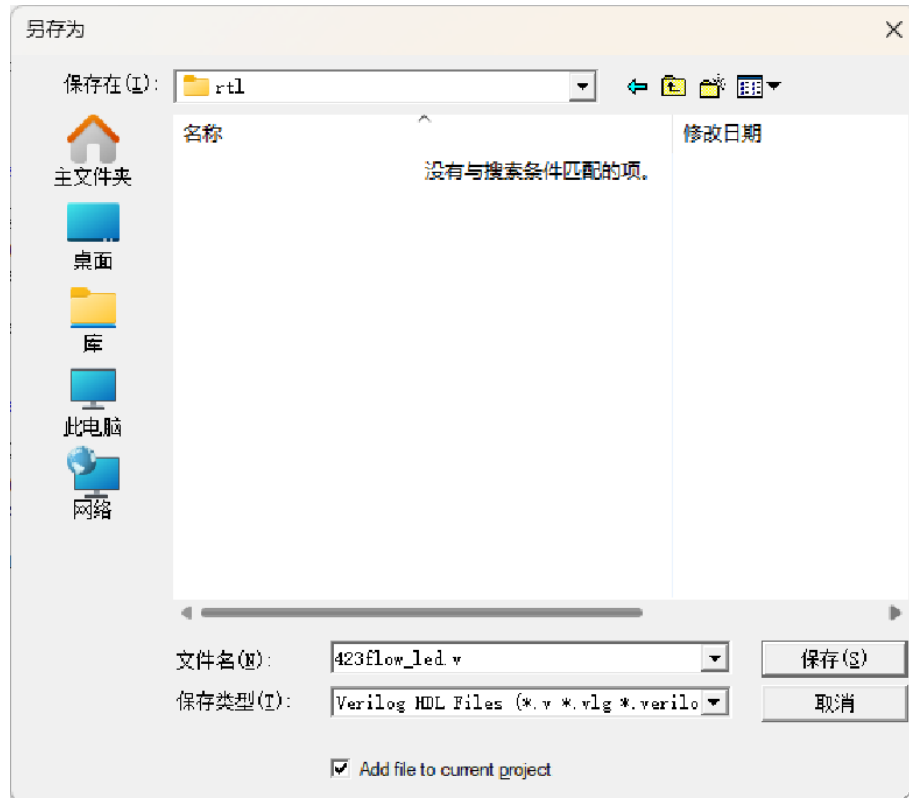


图 8: 保存代码文件到指定文件夹

说明: (此处说明文件管理规范)

(6) 打开 Pin Planner 分配引脚

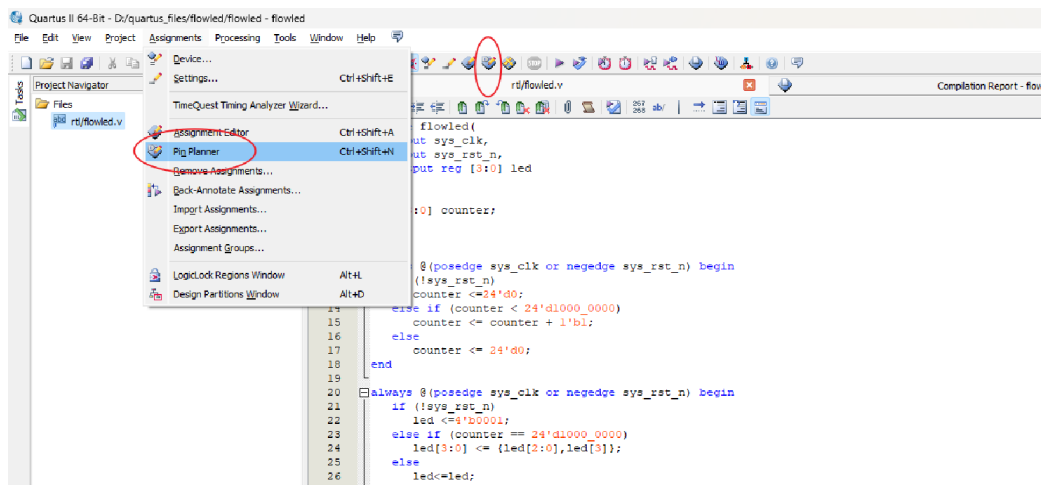


图 9: Pin Planner 分配引脚界面

说明: (此处填写分配引脚的详细步骤)

(7) 手动输入/使用 tcl 脚本文件

信号名	方向	管脚	端口说明
sys_clk	input	M2	系统时钟, 50M
sys_rst_n	input	M1	系统复位, 低有效
led[0]	output	D11	LED0
led[1]	output	C11	LED1
led[2]	output	E10	LED2
led[3]	output	F9	LED3

图 10: tcl 脚本分配引脚示意

tcl 脚本示例:

```

1      set_location_assignment PIN_M2 -to sys_clk
2      set_location_assignment PIN_M1 -to sys_rst_n
3      set_location_assignment PIN_D11 -to led[0]
4      set_location_assignment PIN_C11 -to led[1]
5      set_location_assignment PIN_E10 -to led[2]
6      set_location_assignment PIN_F9 -to led[3]
7

```

说明: (此处说明两种方式的区别和操作方法)

(8) 完整编译工程

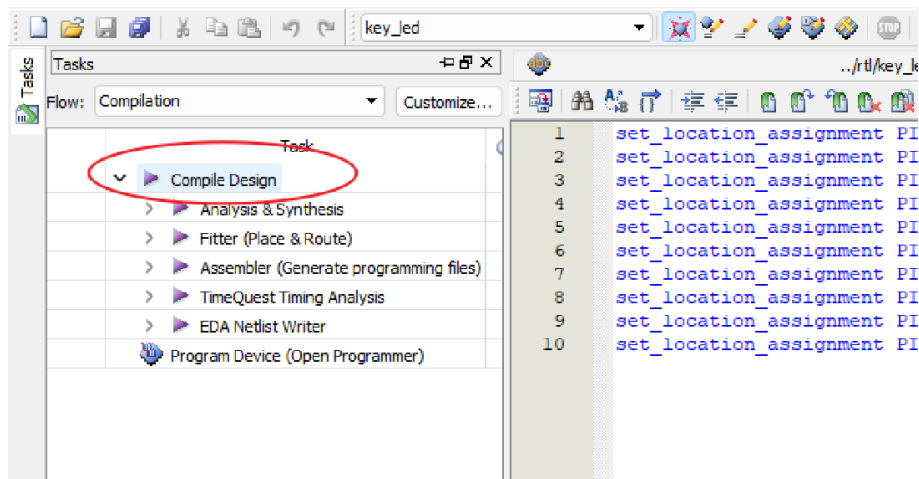


图 11: 完整编译工程界面

说明: (此处填写完整编译的流程和注意事项)

(9) 下载程序至开发板

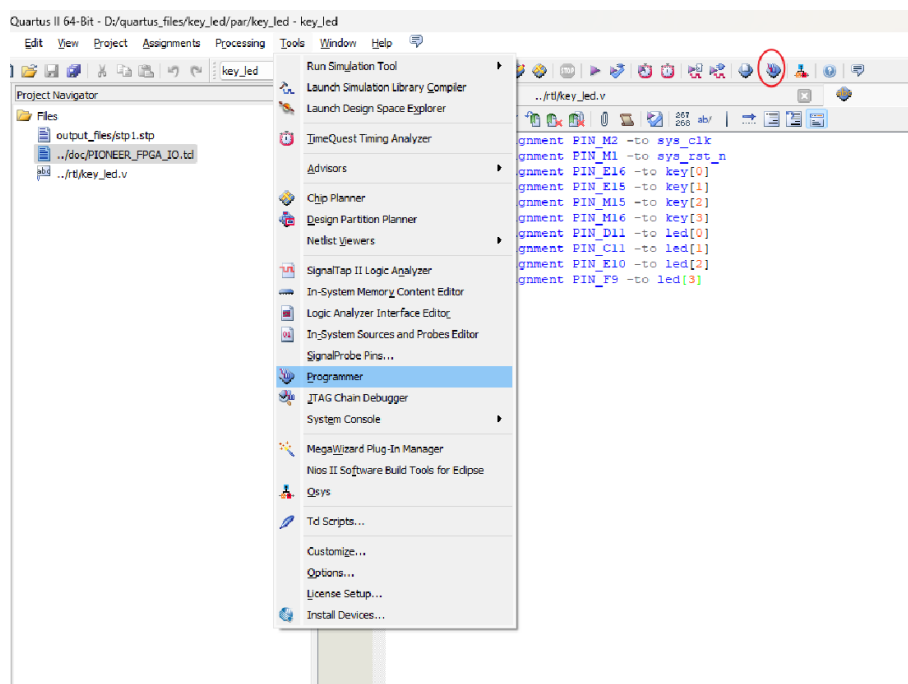


图 12: 下载程序到开发板界面 1

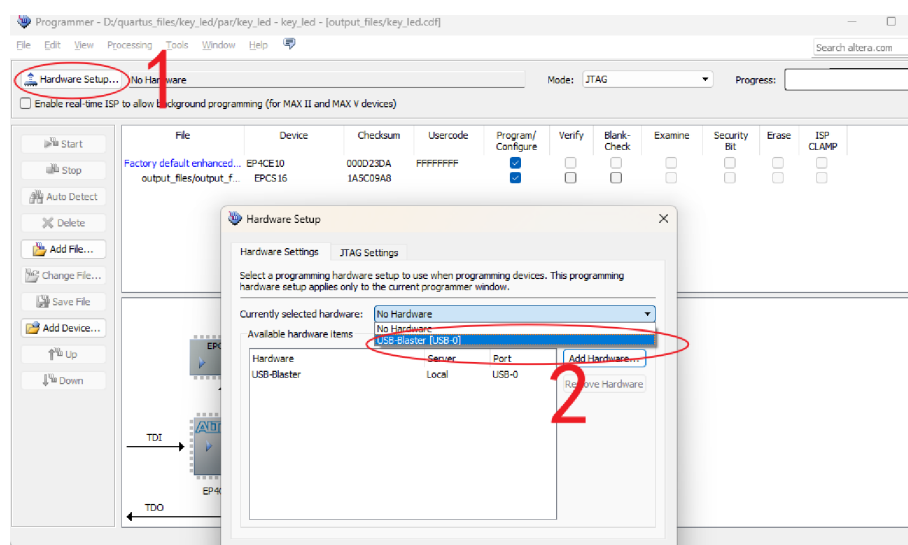


图 13: 下载程序到开发板界面 2

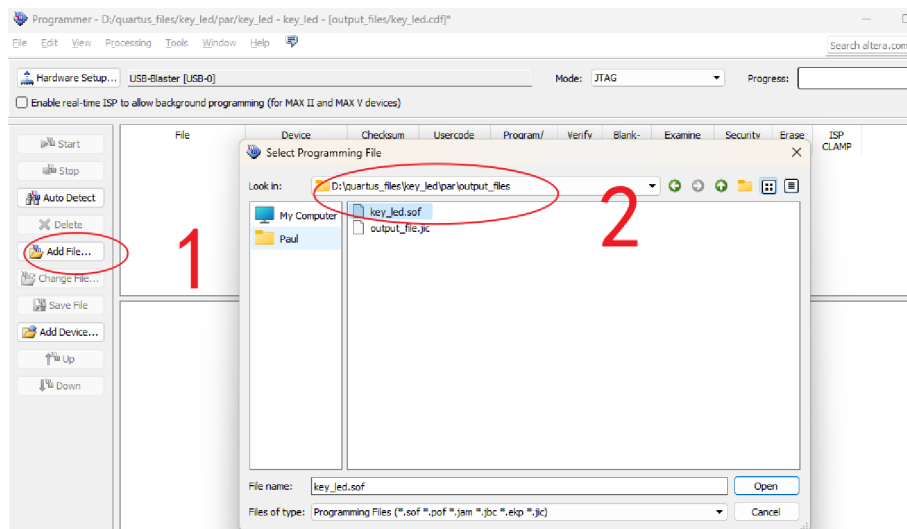


图 14: 下载程序到开发板界面 3

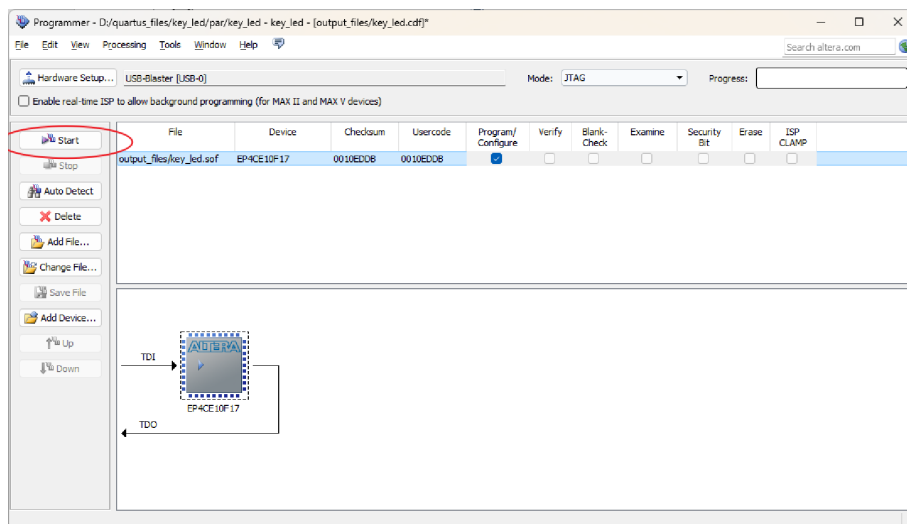


图 15: 下载程序到开发板界面 4

说明: (此处填写下载步骤和常见问题)

(10) 导出 jic 文件, 固化程序

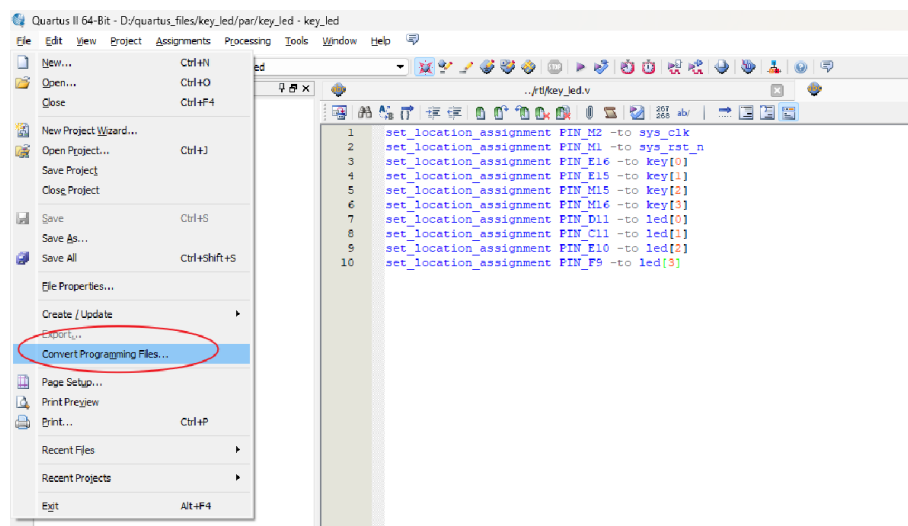


图 16: 导出 jic 文件界面

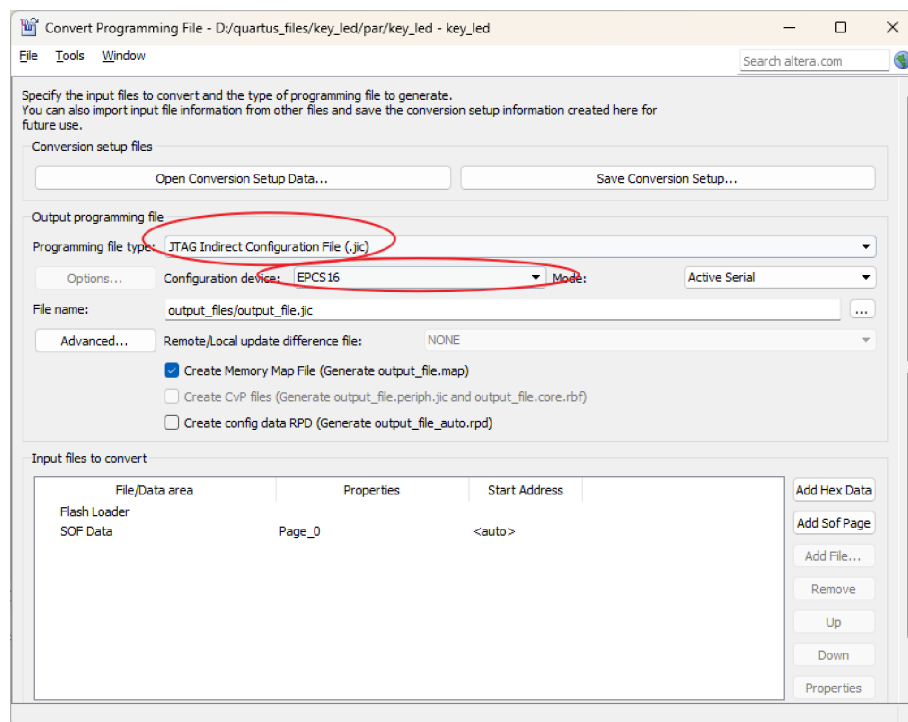


图 17: 导出 jic 文件界面 2

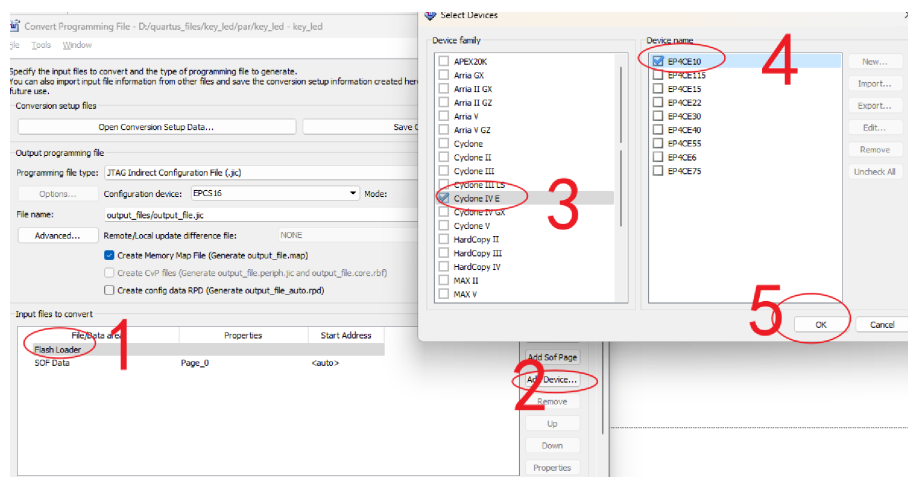


图 18: 导出 jic 文件界面 3

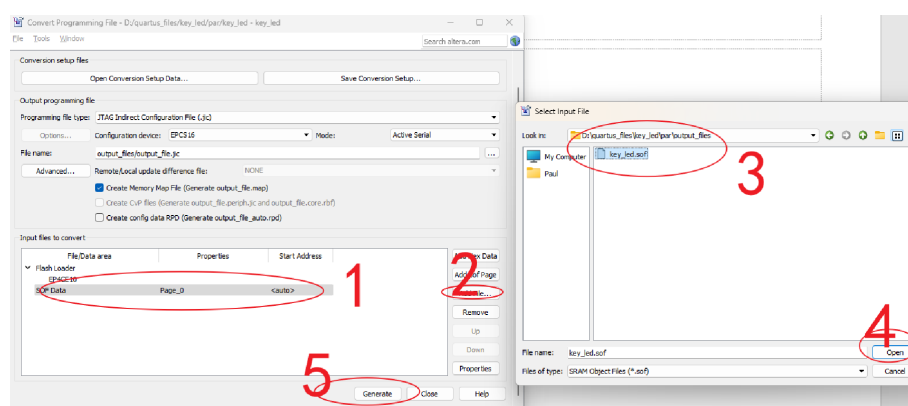


图 19: 导出 jic 文件界面 4

说明：（此处填写固化流程和注意事项）

§3 实验结果

3.1 实验结果 1: LED 流水灯实验

演示视频: [点击查看演示视频](#)

3.2 实验结果 2: 按键控制 LED 灯实验基础问题

演示视频: [点击查看演示视频](#)

3.3 实验结果 3: 按键控制 LED 灯实验变式 1

演示视频: [点击查看演示视频](#)

3.4 实验结果 4: 按键控制 LED 灯实验变式 2

演示视频: [点击查看演示视频](#)

§4 实验总结

4.1 实验中的问题与感想

- (1) 在实验过程中, 遇到了一些问题, 例如: 编译错误、引脚分配不当等。通过查阅资料和请教老师, 最终解决了这些问题。
- (2) 通过本次实验, 我对 FPGA 开发流程有了更深入的理解, 尤其是在代码编写和调试方面。
- (3) 实验中使用的 Verilog 语言让我对硬件描述语言有了更直观的认识, 增强了我的编程能力。
- (4) 通过对 LED 流水灯和按键控制 LED 灯的实验, 我对数字电路的基本原理有了更深入的理解。
- (5) 在实验中, 我还学会了如何使用 Quartus 软件进行 FPGA 开发和调试, 这对我今后的学习和工作都非常有帮助。

§5 附录: Verilog 代码

Verilog 代码: LED 流水灯实验

```
1  module flow_led(  
2      input sys_clk,  
3      input sys_rst_n,  
4      output reg [3:0] led  
5  );  
6  
7      reg [23:0] counter;  
8  
9      always @(posedge sys_clk or negedge sys_rst_n) begin  
10         if (!sys_rst_n)  
11             counter <= 24'd0;  
12         else if (counter < 24'd1000_0000)  
13             counter <= counter + 1'b1;  
14         else  
15             counter <= 24'd0;  
16     end  
17  
18  
19     always @(posedge sys_clk or negedge sys_rst_n) begin  
20         if (!sys_rst_n)  
21             led <= 4'b0001;  
22         else if(counter == 24'd1000_0000)  
23             led[3:0] <= {led[2:0],led[3]};  
24         else  
25             led <= led;  
26     end  
27  
28     endmodule
```

Verilog 代码: 按键控制 LED 灯实验基础

```
1  module key_led(  
2      input sys_clk,  
3      input sys_rst_n,  
4      input [3:0] key,  
5      output reg [3:0] led  
6  );  
7  
8  
9      reg [23:0] cnt;  
10     reg [1:0] led_control;  
11  
12     always @ (posedge sys_clk or negedge sys_rst_n) begin  
13         if (!sys_rst_n)  
14             cnt<=24'd9_999_999;  
15         else if(cnt<24'd9_999_999)  
16             cnt<=cnt+1;  
17         else  
18             cnt<=0;  
19     end  
20  
21  
22     always @ (posedge sys_clk or negedge sys_rst_n) begin  
23         if (!sys_rst_n)  
24             led_control <= 2'b00;  
25         else if(cnt == 24'd9_999_999)  
26             led_control <= led_control + 1'b1;  
27         else  
28             led_control <= led_control;  
29     end  
30  
31  
32     always @ (posedge sys_clk or negedge sys_rst_n) begin  
33         if (!sys_rst_n) begin  
34             led <= 4'b 0000;  
35         end  
36         else if (key[0]==0)  
37             case (led_control)  
38                 2'b00 : led<=4'b1000;  
39                 2'b01 : led<=4'b0100;  
40                 2'b10 : led<=4'b0010;  
41                 2'b11 : led<=4'b0001;  
42                 default : led<=4'b0000;  
43             endcase  
44         else if (key[1]==0)  
45             case (led_control)  
46                 2'b00 : led<=4'b0001;  
47                 2'b01 : led<=4'b0010;  
48                 2'b10 : led<=4'b0100;  
49                 2'b11 : led<=4'b1000;  
50                 default : led<=4'b0000;  
51             endcase
```

```
52     else if (key[2]==0)
53         case (led_control)
54             2'b00 : led <=4'b1111;
55             2'b01 : led <=4'b0000;
56             2'b10 : led <=4'b1111;
57             2'b11 : led <=4'b0000;
58             default : led <=4'b0000;
59         endcase
60     else if (key[3]==0)
61         led<=4'b1111;
62     else
63         led<=4'b0000;
64 end
65
66 endmodule
```

Verilog 代码: 按键控制 LED 灯实验变式 1

```
1  module key_leda(  
2      input sys_clk,  
3      input sys_rst_n,  
4      input [3:0] key,  
5      output reg [3:0] led  
6  );  
7  
8  reg [23:0] cnt;  
9  reg [25:0] cnt_1s; // New counter for 1-second interval  
10 reg [1:0] led_control;  
11 reg [1:0] flow_control; // New control for flowing sequence  
12  
13 // Counter for 0.2-second interval (original timing)  
14 always @ (posedge sys_clk or negedge sys_rst_n) begin  
15     if (!sys_rst_n)  
16         cnt <= 24'd9_999_999;  
17     else if (cnt < 24'd9_999_999)  
18         cnt <= cnt + 1;  
19     else  
20         cnt <= 0;  
21 end  
22  
23 // Counter for 1-second interval (for key[2])  
24 always @ (posedge sys_clk or negedge sys_rst_n) begin  
25     if (!sys_rst_n)  
26         cnt_1s <= 26'd0;  
27     else if (cnt_1s < 26'd49_999_999)  
28         cnt_1s <= cnt_1s + 1;  
29     else  
30         cnt_1s <= 0;  
31 end  
32  
33 // led_control increments every 0.2 seconds  
34 always @ (posedge sys_clk or negedge sys_rst_n) begin  
35     if (!sys_rst_n)  
36         led_control <= 2'b00;  
37     else if (cnt == 24'd9_999_999)  
38         led_control <= led_control + 1'b1;  
39     else  
40         led_control <= led_control;  
41 end  
42  
43 // flow_control increments every 1 second  
44 always @ (posedge sys_clk or negedge sys_rst_n) begin  
45     if (!sys_rst_n)  
46         flow_control <= 2'b00;  
47     else if (cnt_1s == 26'd49_999_999)  
48         flow_control <= flow_control + 1'b1;  
49     else  
50         flow_control <= flow_control;  
51 end
```



```
52
53 // LED control logic
54 always @ (posedge sys_clk or negedge sys_rst_n) begin
55     if (!sys_rst_n) begin
56         led <= 4'b0000;
57     end
58     else if (key[0] == 0) // Original function: shift right every 0.2s
59         case (led_control)
60             2'b00 : led <= 4'b1000;
61             2'b01 : led <= 4'b0100;
62             2'b10 : led <= 4'b0010;
63             2'b11 : led <= 4'b0001;
64             default : led <= 4'b0000;
65         endcase
66     else if (key[1] == 0) // Original function: shift left every 0.2s
67         case (led_control)
68             2'b00 : led <= 4'b0001;
69             2'b01 : led <= 4'b0010;
70             2'b10 : led <= 4'b0100;
71             2'b11 : led <= 4'b1000;
72             default : led <= 4'b0000;
73         endcase
74     else if (key[2] == 0) // Modified function: flow left to right every 1s
75         case (flow_control)
76             2'b00 : led <= 4'b1000;
77             2'b01 : led <= 4'b0100;
78             2'b10 : led <= 4'b0010;
79             2'b11 : led <= 4'b0001;
80             default : led <= 4'b0000;
81         endcase
82     else if (key[3] == 0) // Original function: all LEDs on
83         led <= 4'b1111;
84     else
85         led <= 4'b0000;
86 end
87
88 endmodule
```

Verilog 代码: 按键控制 LED 灯实验变式 2

```
1  module key_ledb(  
2      input sys_clk,  
3      input sys_rst_n,  
4      input [3:0] key,  
5      output reg [3:0] led  
6  );  
7  
8  reg [23:0] cnt;  
9  reg [1:0] led_control;  
10 reg [1:0] mode;  
11 reg active;  
12  
13 // Counter logic: counts up to 9,999,999 for timing  
14 always @ (posedge sys_clk or negedge sys_rst_n) begin  
15     if (!sys_rst_n)  
16         cnt <= 24'd0; // Initialize to 0 for standard counter behavior  
17     else if (cnt < 24'd9_999_999)  
18         cnt <= cnt + 1;  
19     else  
20         cnt <= 0;  
21 end  
22  
23 // LED control logic: increments every time cnt reaches 9,999,999  
24 always @ (posedge sys_clk or negedge sys_rst_n) begin  
25     if (!sys_rst_n)  
26         led_control <= 2'b00;  
27     else if (cnt == 24'd9_999_999)  
28         led_control <= led_control + 1'b1;  
29     else  
30         led_control <= led_control;  
31 end  
32  
33 // Mode and active logic: stores the last pressed key and activates the pattern  
34 always @ (posedge sys_clk or negedge sys_rst_n) begin  
35     if (!sys_rst_n) begin  
36         mode <= 2'b00; // Initial mode (arbitrary)  
37         active <= 0; // Inactive until a key is pressed  
38     end  
39     else if (key[0] == 0) begin  
40         mode <= 2'b00; // Key[0] pressed  
41         active <= 1; // Activate pattern  
42     end  
43     else if (key[1] == 0) begin  
44         mode <= 2'b01; // Key[1] pressed  
45         active <= 1;  
46     end  
47     else if (key[2] == 0) begin  
48         mode <= 2'b10; // Key[2] pressed  
49         active <= 1;  
50     end  
51     else if (key[3] == 0) begin
```

```
52         mode <= 2'b11; // Key[3] pressed
53         active <= 1;
54     end
55     else begin
56         mode <= mode; // Retain last mode
57         active <= active; // Retain active state
58     end
59 end
60
61 // LED output logic: sets LED pattern based on mode when active
62 always @ (posedge sys_clk or negedge sys_rst_n) begin
63     if (!sys_rst_n) begin
64         led <= 4'b0000; // Reset LEDs to off
65     end
66     else if (active) begin
67         case (mode)
68             2'b00: begin // Key[0] pattern
69                 case (led_control)
70                     2'b00 : led <= 4'b1000;
71                     2'b01 : led <= 4'b0100;
72                     2'b10 : led <= 4'b0010;
73                     2'b11 : led <= 4'b0001;
74                 endcase
75             end
76             2'b01: begin // Key[1] pattern
77                 case (led_control)
78                     2'b00 : led <= 4'b0001;
79                     2'b01 : led <= 4'b0010;
80                     2'b10 : led <= 4'b0100;
81                     2'b11 : led <= 4'b1000;
82                 endcase
83             end
84             2'b10: begin // Key[2] pattern
85                 case (led_control)
86                     2'b00 : led <= 4'b1111;
87                     2'b01 : led <= 4'b0000;
88                     2'b10 : led <= 4'b1111;
89                     2'b11 : led <= 4'b0000;
90                 endcase
91             end
92             2'b11: begin // Key[3] pattern
93                 led <= 4'b1111;
94             end
95         endcase
96     end
97     else begin
98         led <= 4'b0000; // LEDs off when no key has been pressed
99     end
100 end
101
102 endmodule
```