电子技术基础实验

# 实验9 数字设计2

**祝尔乐**

***2020013020***

***2021/12/20***

# 一.实验目的

- 熟悉 Quartus Prime 开发环境。
- 熟悉用硬件描述语言开发数字系统的方法。
- 熟悉 FPGA 硬件平台。

# 二.实验内容

## 1、设计电路，使 LED1~LED4 分别以 0.5Hz、1Hz、2Hz 和 4Hz 的频率闪烁。

### 1.1 设计思路

通过计数器的方式实现分频器，用`div_reg`记录时间上升沿次数，25000000对应1s，故可以求出各个频率对应的阈值，从而产生不同频率的时钟控制小灯。

### 1.2 设计代码

设计文件的代码如下：

```
module flashing(
    input clk,
    input reset,
    output reg[3:0]clk_div
);

reg [27:0] div1_reg;
reg [27:0] div2_reg;
reg [27:0] div3_reg;
reg [27:0] div4_reg;
always @ (posedge clk, posedge reset)
if (reset)
begin
    div1_reg <= 0;
    div2_reg <= 0;
    div3_reg <= 0;
    div4_reg <= 0;
    clk_div <= 0;
end

else
begin

if (div1_reg < 12500000)
    div1_reg <= div1_reg + 1;
else
```

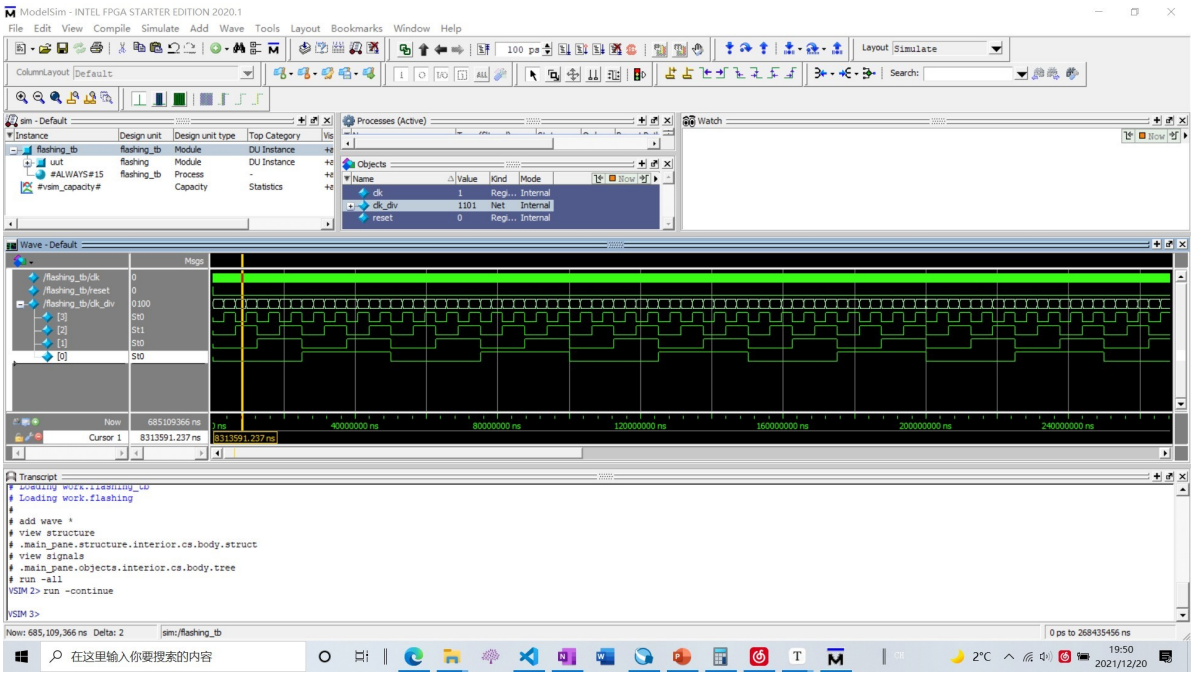
```
begin
    div1_reg <= 0;
    clk_div[0] <= ~ clk_div[0];
end

if (div2_reg < 6250000)
    div2_reg <= div2_reg + 1;
else
begin
    div2_reg <= 0;
    clk_div[1] <= ~ clk_div[1];
end

if (div3_reg < 3125000)
    div3_reg <= div3_reg + 1;
else
begin
    div3_reg <= 0;
    clk_div[2] <= ~ clk_div[2];
end

if (div4_reg < 1562500)
    div4_reg <= div4_reg + 1;
else
begin
    div4_reg <= 0;
    clk_div[3] <= ~ clk_div[3];
end
end
endmodule
```

### 1.3 仿真验证

为了验证设计的正确性，我们先对设计文件进行仿真，仿真文件的代码如下：

```
`timescale 1ns/1ps
module flashing_tb;

reg clk;
reg reset;

wire [3:0]clk_div;

flashing uut(
    .clk(clk),
    .reset(reset),
    .clk_div(clk_div)
);

always #1 clk = ~clk;
initial begin
    clk = 1;
    reset = 1;
    #20 reset = 0;
end

endmodule
```

我们可以得到仿真波形：




由仿真我们可以看出，我们实现了四种频率的时钟信号，当调整`clk`引脚为`FPGA`的时钟时，我们就可以分别实现$0.5Hz, 1Hz, 2Hz, 4Hz$频率的时钟，将时钟引脚与LED相连，即可实现四种频率闪烁的灯。

将代码烧录进入`FPGA`，观察现象，与预期相符，验证了我们电路设计的正确性。

## 2、设计电路，实现至少两种形式的流水灯（LED1~LED8）。可用按键或拨码 开关切换。

### 2.1 设计思路

使用分频器`divider_new`产生$4Hz$频率的时钟，在`sled_comtrol_1`，`sled_control_2`通过状态机的方式实现两种控制模式，再通过`sled_swich`实现模式选择。

### 2.2 设计代码

`stream_led`：总体框架。

```
module stream_led(
    input clk,
    input reset,
    input sel,
    output wire[7:0] led_io
);

wire clk_div;
wire [7:0]led_io_1, led_io_2;

divider_new uut0(
    .clk(clk),
    .reset(reset),
    .clk_div(clk_div)
);

sled_control_1 uut1(
    .clk(clk_div),
    .reset(reset),
```

```
    .led_io(led_io_1)
);  //流水灯1

sled_control_2 uut2(
    .clk(clk_div),
    .reset(reset),
    .led_io(led_io_2)
);  //流水灯2

sled_switch uut3(
    .sel(sel),
    .led_io_1(led_io_1),
    .led_io_2(led_io_2),
    .led_io(led_io)
);  //选择

endmodule
```

`divider_new`：分频器。

```
module divider_new(
    input clk,
    input reset,
    output reg clk_div
);
reg [27:0] div_reg;
always @ (posedge clk, posedge reset)
if (reset)
begin
    div_reg <= 0;
    clk_div <= 0;
end
else
begin
    if (div_reg < 3125000)
    div_reg <= div_reg + 1;
    else
    begin
        div_reg <= 0;
        clk_div <= ~ clk_div;
    end
end
endmodule
```

`sled_comtrol_1`：实现八个小灯沿一个方向依次闪烁。

```
module sled_control_1(
    input clk,
    input reset,
    output reg [7:0] led_io
);
reg [2:0] state,nextstate;
parameter
S0 = 3'b000, S1 = 3'b001,S2 = 3'b010,S3 = 3'b011,
S4 = 3'b100, S5 = 3'b101,S6 = 3'b110,S7 = 3'b111;
```

```
//state register
always @ (posedge clk, posedge reset)
if (reset) state <= S0;
else state <= nextstate;

//next state logic
always @ (*)
case (state)
S0: begin
    led_io = 8'b00000001;
    nextstate = S1;
end
S1: begin
    led_io = 8'b00000010;
    nextstate = S2;
end
S2: begin
    led_io = 8'b00000100;
    nextstate = S3;
end
S3: begin
    led_io = 8'b00001000;
    nextstate = S4;
end
S4: begin
    led_io = 8'b00010000;
    nextstate = S5;
end
S5: begin
    led_io = 8'b00100000;
    nextstate = S6;
end
S6: begin
    led_io = 8'b01000000;
    nextstate = S7;
end
S7: begin
    led_io = 8'b10000000;
    nextstate = S0;
end
default: nextstate = S0;
endcase
endmodule
```

`sled_control_2`：实现八个小灯从两侧向中间依次闪烁。

```
module sled_control_2(
    input clk,
    input reset,
    output reg [7:0] led_io
);
reg [1:0] state,nextstate;
parameter S0 = 2'b00,S1 = 2'b01,S2 = 2'b10,S3 = 2'b11;
//state register
always @ (posedge clk, posedge reset)
    if (reset) state <= S0;
    else state <= nextstate;
```

```
//next state logic
always @ (*)
case (state)
S0: begin
    led_io = 8'b10000001;
    nextstate = S1;
end
S1: begin
    led_io = 8'b01000010;
    nextstate = S2;
end
S2: begin
    led_io = 8'b00100100;
    nextstate = S3;
end
    S3: begin
    led_io = 8'b00011000;
    nextstate = S0;
end
default: nextstate = S0;
endcase
endmodule
```

`sled_swich`：通过拨码开关可切换流水灯模式。

```
module sled_switch(
    input [7:0] led_io_1, led_io_2,
    input sel,
    output wire [7:0] led_io
);

assign led_io = sel ? led_io_1 : led_io_2;

endmodule
```

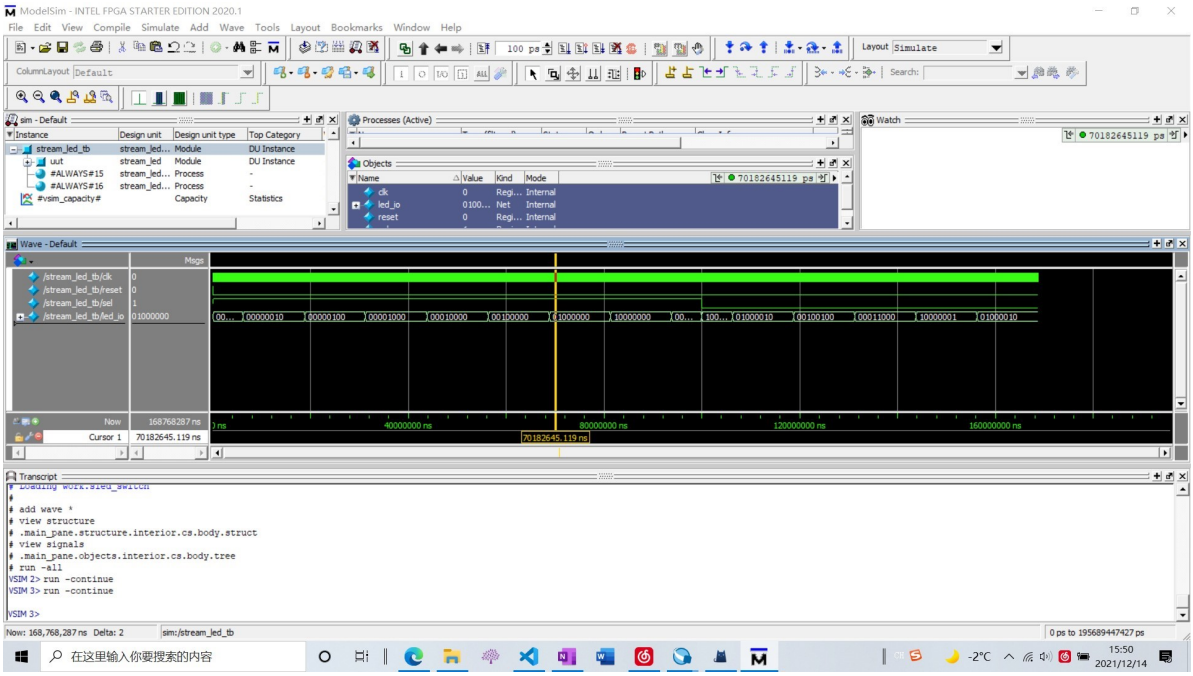
### 2.3 仿真验证

编写仿真代码如下：

```
`timescale 1ns/1ps
module stream_led_tb;

reg clk;
reg reset;
reg sel;
wire [7:0]led_io;

stream_led uut(
    .clk(clk),
    .reset(reset),
    .sel(sel),
    .led_io(led_io)
);
always #1 clk = ~clk;
always #100000000 sel = ~sel;
initial begin
```

```
    clk = 1;
    reset = 1;
    sel  = 1;
    #2;
    reset = 0;
end

endmodule
```

观察到仿真的波形为：



可以看到开始是第一种流水灯的输出，当开关`sel`变化的时候输出变为第二种流水灯的输出，这与我们设计的预想是一致的。

将代码烧录进入`FPGA`，通过调整拨码开关，观察到两种形式的流水灯，验证了我们设计的正确性。

## 3、选做：设计电路，实现秒表计数（0~60 秒），计数值用数码管显示。

### 3.1 设计思路

采用计数器进行计数，当计数达到`25000000`时则为1s, 用数码管实现时间的显示。为了方便显示，我们用`sec_0`,`sec_1`来储存时间的第0位和第1位，显示过程中我们通过`1kHz`的频率来回切换显示的位数，使得两位显示出不同的数字。

### 3.2 设计代码

编写设计代码如下：

```
module timer (
    input clk,
    input reset,

    output reg[3:0] dig,
    output reg[7:0] segments
);

reg [31:0] cnt; //1Hz计数
reg [31:0] cnt_k;   //1kHz计数
reg clk_1kHz;
```

```
reg [3:0] sec_0;
reg [3:0] sec_1;

initial
begin
    cnt = 0;
    cnt_k = 0;
    clk_1kHz = 0;
    sec_0 = 0;
    sec_1 = 0;
end

/* 1秒计数 */
always @ (posedge clk, posedge reset)
begin
    if (reset)
    begin
        cnt <= 0;
        sec_0 <= 0;
        sec_1 <= 0;
    end
    else
    begin
        if (cnt < 25_000_000)
            cnt <= cnt + 1;
        else
        begin
            cnt <= 0;
            sec_0 <= sec_0 + 1;
        end

        if (sec_0 > 9)
        begin
            sec_0 <= 0;
            sec_1 <= sec_1 + 1;
        end

        if (sec_1 > 5)
        begin
            sec_1 <= 0;
        end
    end
end

/* 产生1kHz频率 */
always @ (posedge clk, posedge reset)
if (reset)
    begin
    cnt_k <= 0;
    clk_1kHz <= 0;
    end
else
    begin
    if (cnt_k < 12_500)
    cnt_k <= cnt_k + 1;
    else
        begin
        cnt_k <= 0;
```

```
        clk_1kHz <= ~ clk_1kHz;
    end
end

/* 显示时钟 */
reg [7:0] seg_set[15:0];   //储存各个数字对应的引脚设置
reg switch;

initial
begin
    seg_set[0] = 8'hc0;
    seg_set[1] = 8'hf9;
    seg_set[2] = 8'ha4;
    seg_set[3] = 8'hb0;
    seg_set[4] = 8'h99;
    seg_set[5] = 8'h92;
    seg_set[6] = 8'h82;
    seg_set[7] = 8'hf8;
    seg_set[8] = 8'h80;
    seg_set[9] = 8'h90;
    seg_set[10] = 8'h88;
    seg_set[11] = 8'h83;
    seg_set[12] = 8'hc6;
    seg_set[13] = 8'ha1;
    seg_set[14] = 8'h86;
    seg_set[15] = 8'h8e;
end

always @(*) begin
    dig <= (clk_1kHz) ? 4'b1110 : 4'b1101;
    segments <= (clk_1kHz) ? seg_set[sec_0] : seg_set[sec_1];
end

endmodule
```

### 3.3 验证

由于本部分时间部分跨度较大，通过编写仿真文件观察波形效果不易验证结果的正确性，于是我直接将代码烧录进入`FPGA`进行验证。

观察到数码管的实现了时间的显示，与手机秒表速率一致，同时到60秒后自动归零，按下`reset`后重新计数，通过现象验证了我们设计的正确性。