中国科学技术大学计算机学院

《数字电路实验》报告

实验题目：FPGA 实验平台及 IP 核使用

学生姓名：舒文炫

学生学号：PB18000029

完成日期：2021.12.7

计算机实验教学中心制

2020 年 09 月

【实验题目】

FPGA 实验平台及 IP 核使用

【实验目的】

- 熟悉 FPGAOL 在线实验平台结构及使用
- 掌握 FPGA 开发各关键环节
- 学会使用 IP 核（知识产权核）

【实验环境】

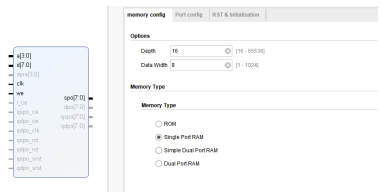
- win10 操作系统
- FPGAOL 平台：fpgaol.ustc.edu.cn
- Vivado
- Logisim

【实验练习】

题目 1

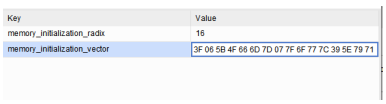
例化一个 16*8bit 的 ROM, 并对其进行初始化, 输入端口由 4 个开关控制, 输出端口连接到七段数码管上（七段数码管与 LED 复用相同的一组管脚), 控制数码管显示与开关相对应的十六进制数字, 例如四个开关输入全为零时, 数码管显示“0”, 输入全为 1 时, 数码管显示“F”。

这里只需要熟悉 ROM 如何生成就可以了, 主要应用 IP 核中的 Distributed Memory, 下面是我的配置截图



这里 16*8bit 需要 4 位的地址每个地址对应一个 8 位的数据，a 是输入的地址，d 是输入数据，we 是写使能，当 we 为高电平时将 d 写入到对应地址，spo 是读取对应地址的输出。

下面是对应初始化文件



这里的 16 个 8 位宽十六进制数分别对应在七段数码管上显示的十六进制数

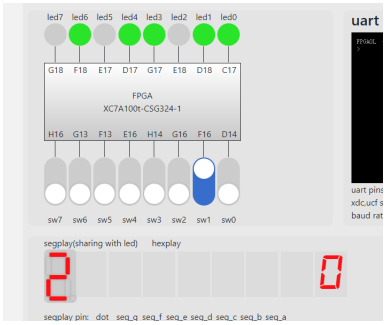
下面是 verilog 代码，主要就是调用这个生成的 IP 核，因为这里只是一个 ROM，写使能直接赋值为 0，表示不能写入即可

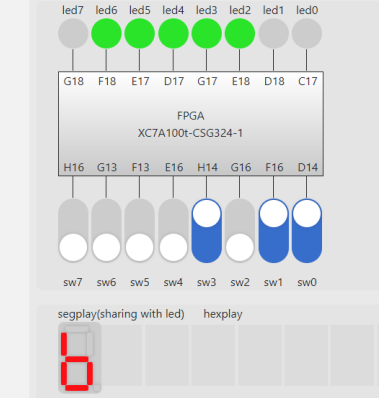
```
module plqwq(
input [3:0] a,
input clk,
output reg [7:0] led
    );
wire we;
wire [7:0] d;
assign we=0;
assign d=8'h00;
wire [7:0] spo;
dist_mem_gen_0 dist_mem_gen_0(
.a (a),
.d (d),
.clk (clk),
.we (we),
.spo (spo));
always@(posedge clk)
    begin
    led<={spo[7],spo[6],spo[5],spo[4],spo[3],spo[2],spo[1],spo[0]};
    end

endmodule
```

这个模块对应的约束文件，将输入输出接入到管脚上，led 接到了七段数码管，a 接到了前四个开关

```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { clk }];
## leds
set_property -dict { PACKAGE_PIN G18 IOSTANDARD LVCMOS33 } [get_ports { led[7] }];
set_property -dict { PACKAGE_PIN F18 IOSTANDARD LVCMOS33 } [get_ports { led[6] }];
set_property -dict { PACKAGE_PIN E17 IOSTANDARD LVCMOS33 } [get_ports { led[5] }];
set_property -dict { PACKAGE_PIN D17 IOSTANDARD LVCMOS33 } [get_ports { led[4] }];
set_property -dict { PACKAGE_PIN G17 IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
set_property -dict { PACKAGE_PIN E18 IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
set_property -dict { PACKAGE_PIN D18 IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN C17 IOSTANDARD LVCMOS33 } [get_ports { led[0] }];
set_property -dict { PACKAGE_PIN D14 IOSTANDARD LVCMOS33 } [get_ports { a[0] }];
set_property -dict { PACKAGE_PIN F16 IOSTANDARD LVCMOS33 } [get_ports { a[1] }];
set_property -dict { PACKAGE_PIN G16 IOSTANDARD LVCMOS33 } [get_ports { a[2] }];
set_property -dict { PACKAGE_PIN H14 IOSTANDARD LVCMOS33 } [get_ports { a[3] }];
```

下面是在 FPGA 在线平台跑的情况





上面是任选的两个例子，可以看到这个功能正常实现了

题目 2

采用 8 个开关作为输入，两个十六进制数码管作为输出，采用时分复用的方式将开关的十六进制数值在两个数码管上显示出来，例如高四位全为 1，低四位全为 0 时，数码管显示“F0”。

这里理解了 FPGA 在线平台数码管的显示方式就可以了，一个数码管要有 50Hz，所以 2 个需要给一个 100Hz 的时钟，但是板载时钟是 100MHz，所以这里需要手动降频，这里我使用 count 方式，计数到一个值生成一个脉冲信号，这样达到降频的目的，代码如下

```
module p2(
input [3:0]a, b,
input clk,
output reg [2:0] an,
output reg [3:0] d1
    );
wire we;
wire [3:0] spo1, spo2, d;
reg cont=0;
integer cont1=0;
wire clk_100hz;
assign we=0;
assign d=0;
pulse inst_qwq( .clk(clk), .clk_100hz(clk_100hz));
always@(posedge clk_100hz)
    begin
    cont=cont+1;
    if(cont==0)
        begin
        an<=3'b000;
        d1<=spo1;
        end
    else
        begin
        an<=3'b001;
```

```
        d1<=spo2;
        end
    end
dist_mem_gen_0 inst_1( .a (a),
.d (d),
.clk (clk),
.we (we),
.spo (spo1));
dist_mem_gen_0 inst_2( .a (b),
.d (d),
.clk (clk),
.we (we),
.spo (spo2));
endmodule
```

这里我使用 pulse 模块生成 100Hz 的时钟，代码如下

```
module pulse(
input clk,
output reg clk_100hz
    );
integer count=0;
wire pulse;
always@(posedge clk)
    begin
    if(count<500000)
        count=count+1;
    else
        count=0;
    end
assign pulse=(count==0);
always@(posedge clk)
begin
    if(pulse)
        clk_100hz = ~clk_100hz;
end
endmodule
```

这里我用了 ROM 来储存数据，也就是 0 到 f 这 16 个十六进制数，实际上可以直接赋值，没必要这么麻烦，因为这里数码管就是直接反应出输入的值，不像七段数码管，需要知道每一位对应到七段数码管上的位置。这是初始化的文件

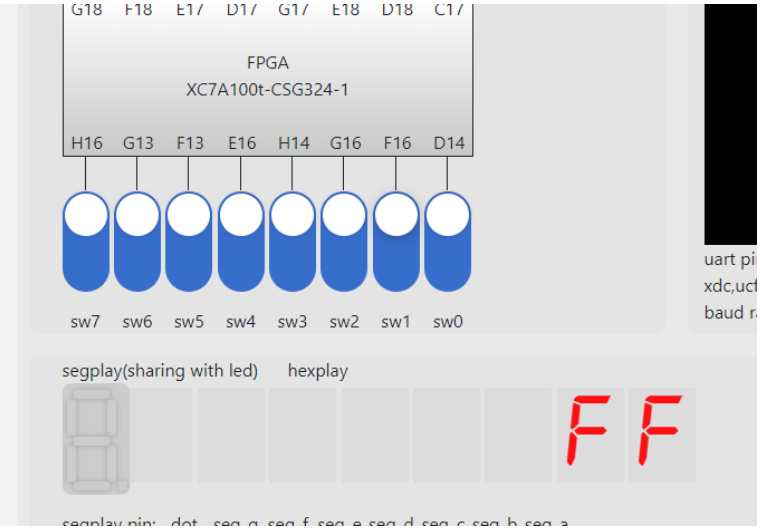
```
memory_initialization_radix=16;
memory_initialization_vector=0 1 2 3 4 5 6 7 8 9 A B C D E F;
```

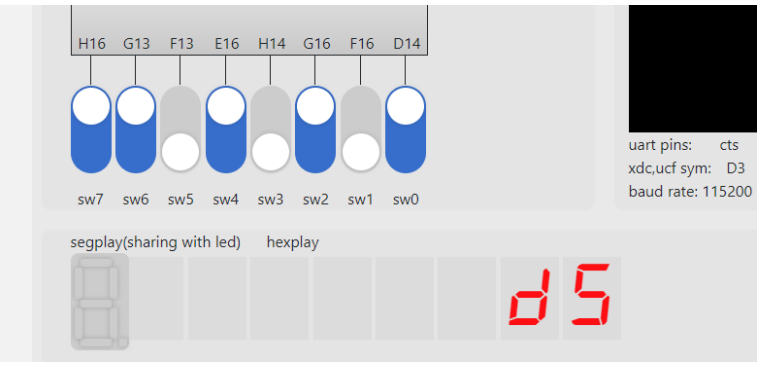
约束文件如下

```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { clk }];
## leds
set_property -dict { PACKAGE_PIN A15 IOSTANDARD LVCMOS33 } [get_ports { d1[3] }];
set_property -dict { PACKAGE_PIN A16 IOSTANDARD LVCMOS33 } [get_ports { d1[2] }];
set_property -dict { PACKAGE_PIN A13 IOSTANDARD LVCMOS33 } [get_ports { d1[1] }];
set_property -dict { PACKAGE_PIN A14 IOSTANDARD LVCMOS33 } [get_ports { d1[0] }];
set_property -dict { PACKAGE_PIN D14 IOSTANDARD LVCMOS33 } [get_ports { a[0] }];
set_property -dict { PACKAGE_PIN F16 IOSTANDARD LVCMOS33 } [get_ports { a[1] }];
set_property -dict { PACKAGE_PIN G16 IOSTANDARD LVCMOS33 } [get_ports { a[2] }];
set_property -dict { PACKAGE_PIN H14 IOSTANDARD LVCMOS33 } [get_ports { a[3] }];
set_property -dict { PACKAGE_PIN E16 IOSTANDARD LVCMOS33 } [get_ports { b[0] }];
set_property -dict { PACKAGE_PIN F13 IOSTANDARD LVCMOS33 } [get_ports { b[1] }];
set_property -dict { PACKAGE_PIN G13 IOSTANDARD LVCMOS33 } [get_ports { b[2] }];
set_property -dict { PACKAGE_PIN H16 IOSTANDARD LVCMOS33 } [get_ports { b[3] }];
set_property -dict { PACKAGE_PIN B17 IOSTANDARD LVCMOS33 } [get_ports { an[0] }];
set_property -dict { PACKAGE_PIN B16 IOSTANDARD LVCMOS33 } [get_ports { an[1] }];
set_property -dict { PACKAGE_PIN A18 IOSTANDARD LVCMOS33 } [get_ports { an[2] }];
```

d1 输出到数码管，an 进行位置的选择，a，b 分别对应两组开关，用来对应两个十六进制数

在 FPGA 在线平台上跑的结果如下





可以看到这里我正常实现了题目要求的功能

题目 3

利用本实验中的时钟管理单元或周期脉冲技术，设计一个精度为 0.1 秒的计时器，用 4 位数码管显示出来，数码管从高到低，分别表示分钟、秒钟十位、秒钟个位、十分之一秒，该计时器具有复位功能（可采用按键或开关作为复位信号），复位时计数值为 1234，即 1 分 23.4 秒

精度 0.1 秒的话，需要的是 10Hz 的时钟，然后这里要显示 4 个数码管，所以还要一个 200hz 的时钟用来扫描数码管，这里分钟、秒钟十位、秒钟个位、十分之一秒这四个数我分别用四个寄存器去储存，后两个逢 10 进 1，第二个逢 6 进 1，具体实现代码如下

```
module timer(
input clk,rst,
output reg [3:0] d,
output reg [2:0] an
 );
wire pulse_10hz,pulse_200hz;
reg [1:0] count=2'b00;
reg [3:0] d1,d2,d3,d4;
pulse inst_1(.clk(clk),.clk_10hz(pulse_10hz));
pulse1 inst_2(.clk(clk),.clk_200hz(pulse_200hz));
always@(posedge pulse_10hz or posedge rst)
    begin
    if(rst)
        begin
        d1<=4'b0100;
        d2<=4'b0011;
        d3<=4'b0010;
        d4<=4'b0001;
        end
    else
        begin
        if(d1<4'b1001)
            d1<=d1+4'b0001;
        else
```

```
        else
            begin
            d1<=4'b0000;
            if(d2<4'b1001)
                d2<=d2+4'b0001;
            else
                begin
                d2<=4'b0000;
                if(d3<4'b0101)
                    d3<=d3+4'b0001;
                else
                    begin
                    d3<=4'b0000;
                    d4<=d4+4'b0001;
                    end
                end
            end
        end
    end
```

这两张图是计时器主要部分实现

```
always@(posedge pulse_200hz)
    begin
    if(count==2'b00)
            begin
            d<=d1;
            an<=3'b000;
            end
     else if(count==2'b01)
            begin
            d<=d2;
            an<=3'b001;
            end
     else if(count==2'b10)
            begin
            d<=d3;
            an<=3'b010;
            end
      else
            begin
            d<=d4;
            an<=3'b011;
            end
       count=count+2'b01;
    end
```

这一张图是用 200Hz 时钟扫描数码管

```
module pulse(
input clk,
output reg clk_10hz
    );
integer count=0;
wire pulse;
always@(posedge clk)
    begin
    if(count<5000000)
        count=count+1;
    else
        count=0;
    end
assign pulse=(count==0);
always@(posedge clk)
begin
    if(pulse)
        clk_10hz = ~clk_10hz;
end
endmodule
```

```
module pulse1(
input clk,
output reg clk_200hz
    );
integer count=0;
wire pulse;
always@(posedge clk)
    begin
    if(count<250000)
        count=count+1;
    else
        count=0;
    end
assign pulse=(count==0);
always@(posedge clk)
begin
    if(pulse)
        clk_200hz = ~clk_200hz;
end
endmodule
```

这两张图是生成 10Hz 与 200Hz 时钟的模块
下面是所需要的约束文件

```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { clk }];
set_property -dict { PACKAGE_PIN B18 IOSTANDARD LVCMOS33 } [get_ports { rst }];
## leds
set_property -dict { PACKAGE_PIN A15 IOSTANDARD LVCMOS33 } [get_ports { d[3] }];
set_property -dict { PACKAGE_PIN A16 IOSTANDARD LVCMOS33 } [get_ports { d[2] }];
set_property -dict { PACKAGE_PIN A13 IOSTANDARD LVCMOS33 } [get_ports { d[1] }];
set_property -dict { PACKAGE_PIN A14 IOSTANDARD LVCMOS33 } [get_ports { d[0] }];
set_property -dict { PACKAGE_PIN B17 IOSTANDARD LVCMOS33 } [get_ports { an[0] }];
set_property -dict { PACKAGE_PIN B16 IOSTANDARD LVCMOS33 } [get_ports { an[1] }];
set_property -dict { PACKAGE_PIN A18 IOSTANDARD LVCMOS33 } [get_ports { an[2] }];
```

生成比特文件后，在 FPGA 平台上跑的结果如下，按下按钮复位成 1234

放开按钮，过一段时间，计时器显示如下

可以看到题目要求的功能实现了

【总结与思考】

1. 请总结本次实验的收获

本次实验学习了 IP 核的使用，这是很强大的工具，很多功能都能给你先实现好，比较方便，学会了如何从一个高频的时钟，得到一个特定的低频的时钟，本次实验的代码量比前几次高出了不少，也复杂了很多，更好的锻炼了我的代码能力。

2. 请评价本次实验的难易程度

本次实验难度比较高，体现在代码量很大，然后涉及到很多 FPGA 管脚，需要弄清楚这些管脚的作用

3. 请评价本次实验的任务量

任务量也较大，每一道题都需要思考一段时间

4. 请为本次实验提供改进建议

相比前几次实验，本次实验难度突然增大，我认为这很不合理，应该使得每次实验难度线性增大可以让学生们更舒服，可以考虑重新组织一下实验内容，将某些部分分散到前几次实验中，像第五六次实验就很简单，没什么东西，这两次实验可以稍微多加一些本次实验的预备内容，这样使得学生容易接受.