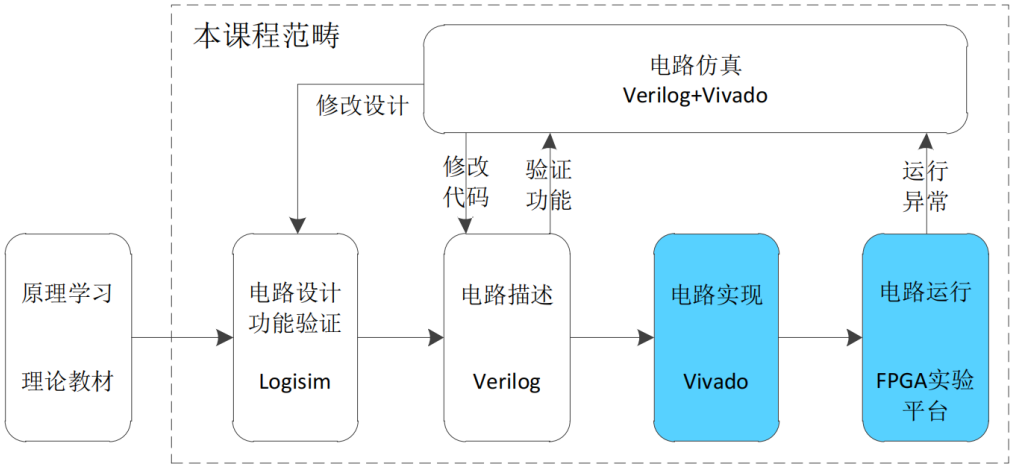
数电实验7

实验题目

FPGA 实验平台及 IP 核使用

实验介绍

简介



前面的实验中，我们完成了一个简单的电路设计，并顺利的烧写到 FPGA 实验平台上。但可能会有部分读者对于其中的一些步骤并不了解，本次实验中，我们将对所使用的 FPGA 实验平台进行介绍，以帮助读者加深对 FPGA 开发流程各环节的理解。此外，我们还会介绍到如何使用 IP 核进行电路设计。

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

- VLAB 平台：vlab.ustc.edu.cn
- FPGAOL 平台：fpgaol.ustc.edu.cn
- Vivado（虚拟机中）

实验步骤

Step1. FPGAOL 实验平台设备显示

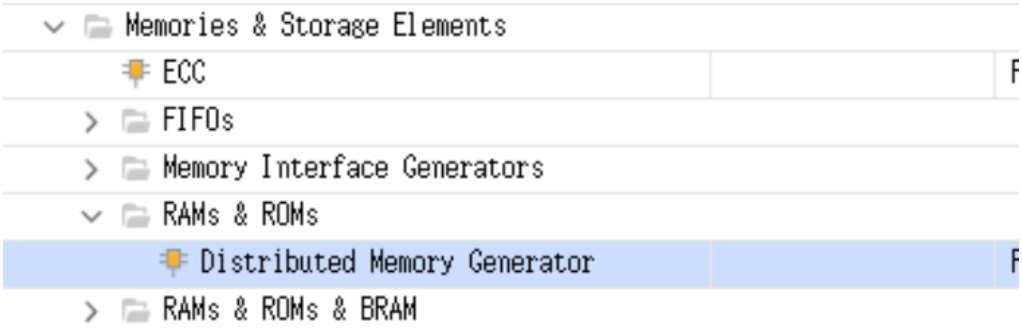
在这个过程中了解七段数码管、LED灯以及开关sw之间的联系，通过在适当的xdc文件影响下，用来实现七段数码管表示相应的信号。

Step2. 外设工作原理介绍

- FPGAOL 实验平台的开发板上有一个 100MHz 的时钟晶振，该晶振与 FPGA 芯片的 E3 管脚直接相连，在开发板上电后，会持续的为 FPGA芯片提供一个 100MHz 频率的时钟信号，不需要用户进行操作。
- 拨动开关作为输入设备，当开关推到上方后，对应 FPGA 管脚输入高电平，当开关拉下来之后，对应 FPGA 管脚输入低电平。
- 我们对七段数码管与 LED 的管脚进行了复用。数码管本质上是由 8 个 LED（发光二极管）构成，发光二极管的阴极共同连接到一端，并接到地，8 个阳极分别由 FPGA的 8 个输出管脚控制，当输出管脚为高电平时，对应的 LED 亮起。通过控制 8 个 LED 的亮灭，便能显示出不同的字符。

Step3. 使用时钟管理单元 IP 核

- 明确时钟信号和时钟脉冲的含义。
- 如果设计中确实需要不同频率的时钟信号应该通过时钟管理单元IP 核生成。首先，点击“IP catalog”。
- 在本次实验中我们只需要会生成ROM和RAM，进行如下图的选择



- 在其中定义位宽和深度，选择存储类型。

Step4. 使用片内存储单元

- 学会使用ROM和RAM
- 会编辑coe文件

实验练习

T1

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”。

- **设计文件**

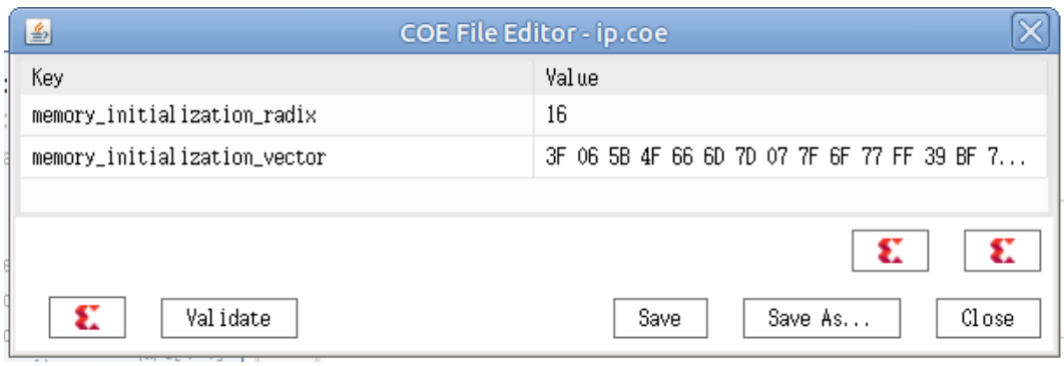
```
module test(
    input [3:0] sw,
    output [7:0] led
);
    dist_mem_gen_0 dist(.a(sw),.spo(led));
endmodule
```

- **约束文件**

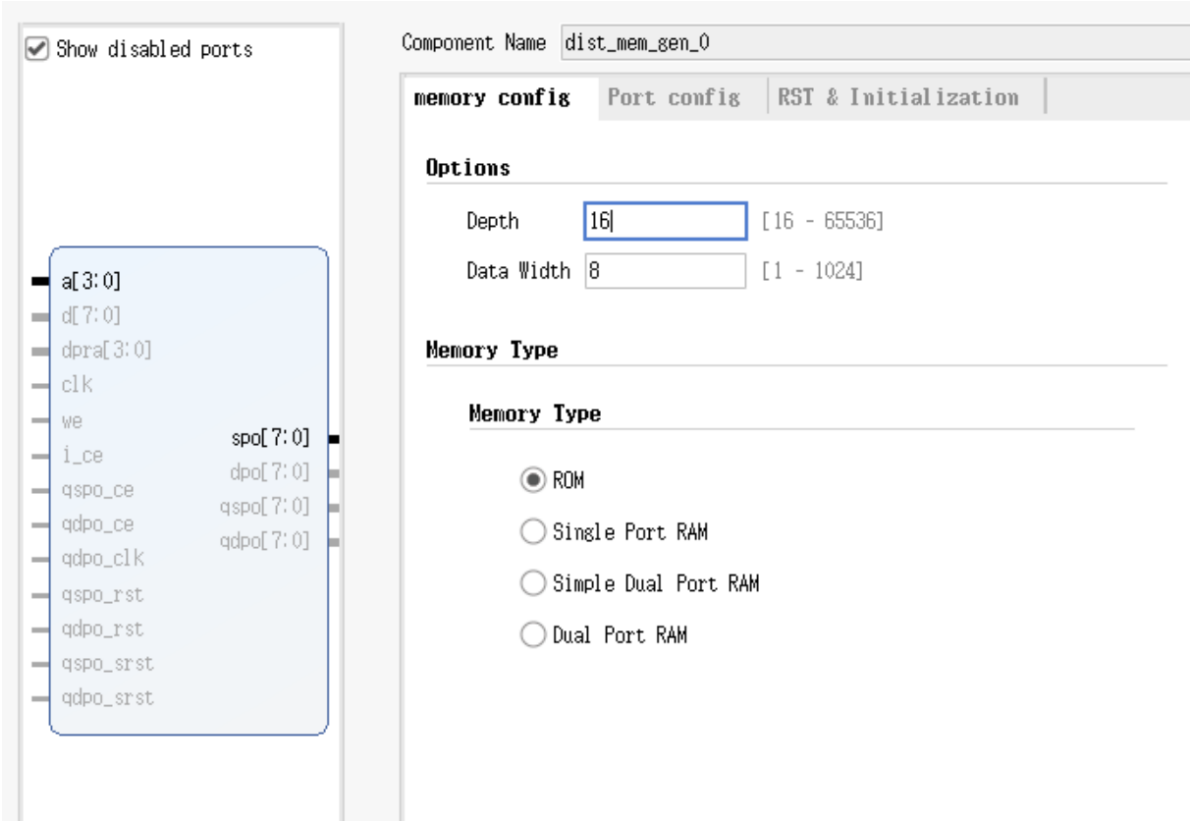
```
set_property -dict { PACKAGE_PIN C17    IOSTANDARD LVCMOS33 } [get_ports { led[0]} ];
set_property -dict { PACKAGE_PIN D18    IOSTANDARD LVCMOS33 } [get_ports { led[1]} ];
set_property -dict { PACKAGE_PIN E18    IOSTANDARD LVCMOS33 } [get_ports { led[2]} ];
set_property -dict { PACKAGE_PIN G17    IOSTANDARD LVCMOS33 } [get_ports { led[3]} ];
set_property -dict { PACKAGE_PIN D17    IOSTANDARD LVCMOS33 } [get_ports { led[4]} ];
set_property -dict { PACKAGE_PIN E17    IOSTANDARD LVCMOS33 } [get_ports { led[5]} ];
set_property -dict { PACKAGE_PIN F18    IOSTANDARD LVCMOS33 } [get_ports { led[6]} ];
set_property -dict { PACKAGE_PIN G18    IOSTANDARD LVCMOS33 } [get_ports { led[7]} ];

set_property -dict { PACKAGE_PIN D14    IOSTANDARD LVCMOS33 } [get_ports { sw[0]} ];
set_property -dict { PACKAGE_PIN F16    IOSTANDARD LVCMOS33 } [get_ports { sw[1]} ];
set_property -dict { PACKAGE_PIN G16    IOSTANDARD LVCMOS33 } [get_ports { sw[2]} ];
set_property -dict { PACKAGE_PIN H14    IOSTANDARD LVCMOS33 } [get_ports { sw[3]} ];
```

- **coe文件**

```
memory_initialization_radix 16
memory_initialization_vector 3F 06 5B 4F 66 6D 7D 07 7F 6F 77 FF 39 BF 79 71
```



- **IP设计**



- **实验结果实例**

T2

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”。

- **设计文件**

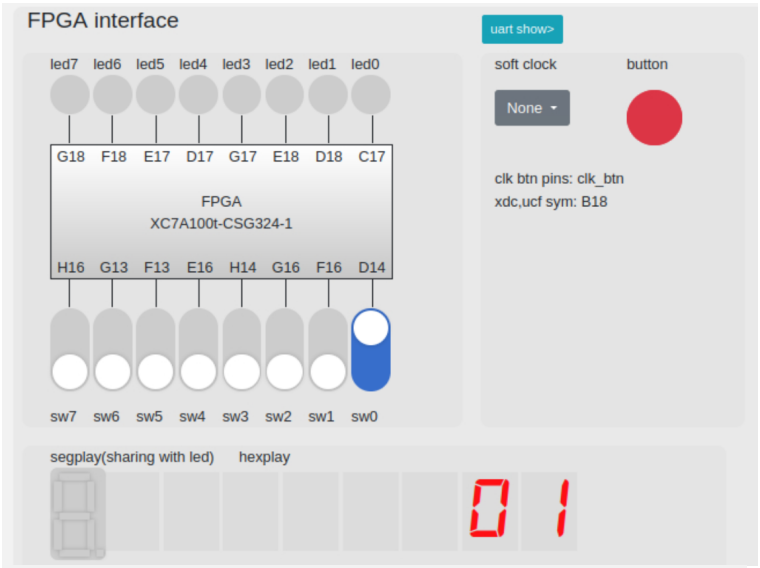
```
module design2(
input clk,
input [7:0] sw,
output reg [3:0] d,
```

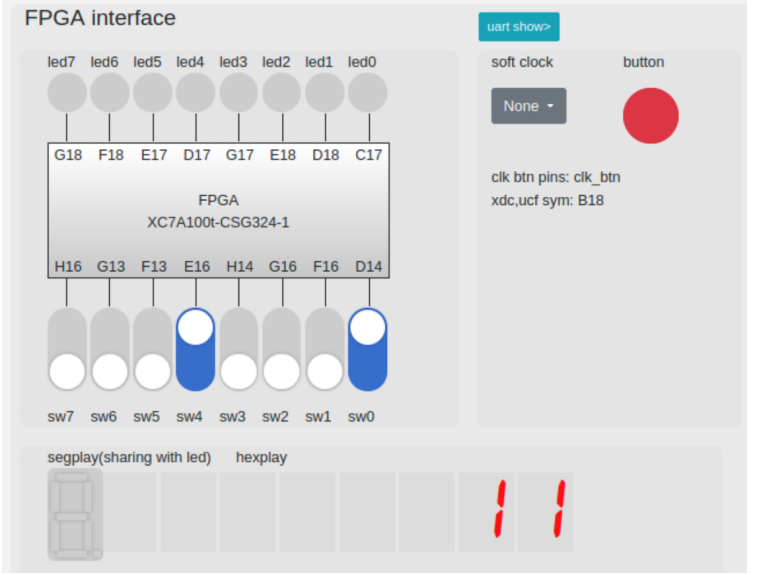
```
output reg an
);
    reg [2:0] cnt;
    always@(posedge clk)
    begin
        an <= cnt[2];
        d<=sw[3:0];//分时复用
        if(an==1)
            d<=sw[7:4];
        cnt<=cnt+1;
    end
endmodule
```

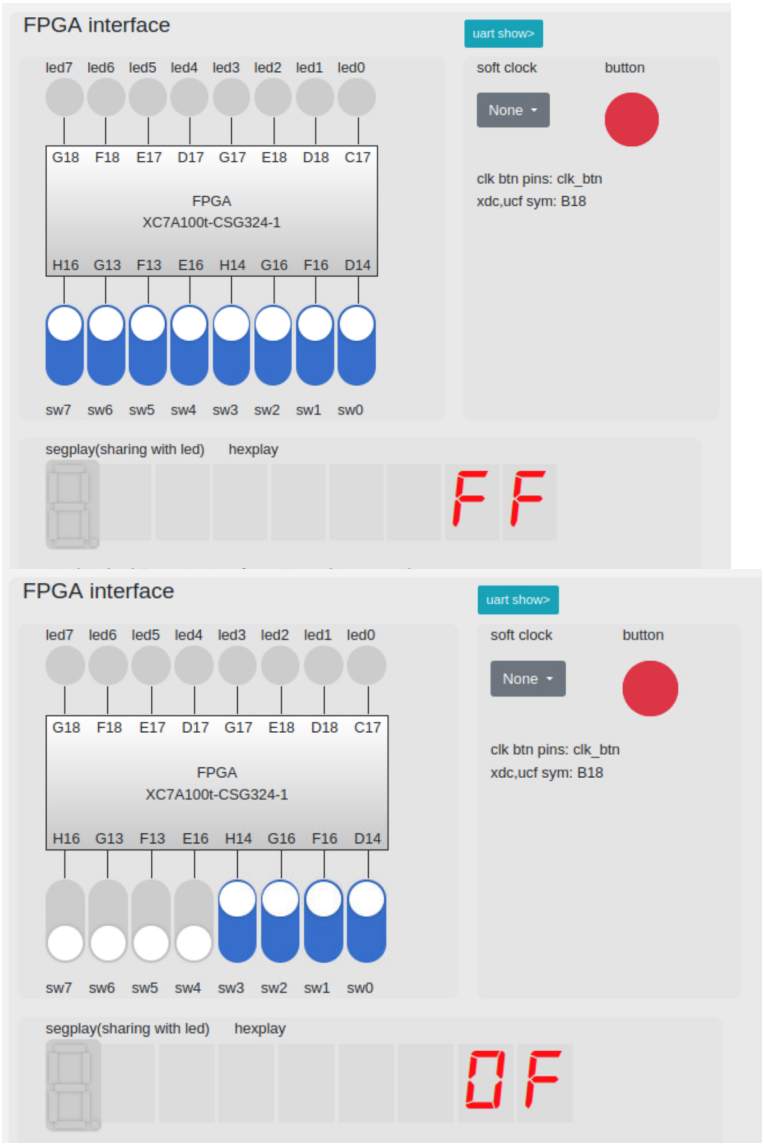
- **约束文件**

```
set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }];
set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }];
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }];
set_property -dict { PACKAGE_PIN E16   IOSTANDARD LVCMOS33 } [get_ports { sw[4] }];
set_property -dict { PACKAGE_PIN F13   IOSTANDARD LVCMOS33 } [get_ports { sw[5] }];
set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { sw[6] }];
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { sw[7] }];

set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports { d[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { d[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { d[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { d[3] }];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports { an }];

set_property -dict { PACKAGE_PIN E3   IOSTANDARD LVCMOS33 } [get_ports { clk }];
```

- **实验结果示例**


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
soft clock
None
button
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay(sharing with led)
hexplay
0 1


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
soft clock
None
button
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay(sharing with led)
hexplay
1 1



T3

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。

- 生成10HZ频率

```
module generate_10hz(//生成10hz时钟
input clk,
output clk_10hz
);
reg [23:0] cnt;
always@(posedge clk)
    begin
    cnt <= 24'h0;
    if(cnt>=9999999)
    cnt <= 24'h0;
    else
    cnt <= cnt + 24'h1;
    end
assign clk_10hz = (cnt==24'h1);
endmodule

```

- 设计文件

```
module T3(
input CLK,
input RST,
output reg [3:0] d,
output reg [1:0] an
);
reg [3:0] N3,N2,N1,N0;
wire clk_10hz;
reg [4:0] cnt;
generate_10hz CLK_change1(CLK,clk_10hz);
always@(posedge CLK or posedge RST)
begin
    if(RST) begin//置位1234
    N3 <= 4'h1; N2 <= 4'h2; N1 <= 4'h3; N0 <= 4'h4; end
    else if(clk_10hz)
        begin
        N0 <= N0 + 4'h1;
            if(N0==9)
            begin
                N0 <= 4'h0;
                N1<=N1 + 4'h1;
                if(N1==9)
                begin
                    N1<=4'h0;
                    N2 <= N2 + 4'h1;
                    if(N2==5)
                    begin
                        N2<=4'h0;//1分=60秒
                        N3 <=N3 + 4'h1;
                        if(N3==9)
                            if(N3 == 0 && N2 == 0 && N1 == 0 && N0==0)
                            begin
                                N3<= 4'h0; N2 <= 4'h0; N1 <= 4'h0;N0 <=
4'h0;
                            end
                    end
                end
            end
        end
end
always@(posedge CLK)
    begin
    cnt=cnt + 1;
    an<=cnt[4:3];
        if(cnt[4:3]==2'b11)d<=N3;
        if(cnt[4:3]==2'b10)d<=N2;
        if(cnt[4:3]==2'b01)d<=N1;
        if(cnt[4:3]==2'b00) d<=N0;
    end
endmodule
```

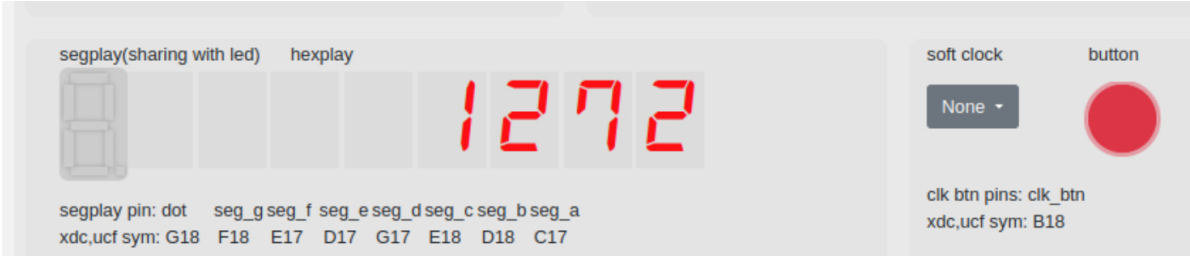
- 约束文件

```
set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports { d[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { d[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { d[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { d[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 E3   IOSTANDARD LVCMOS33 } [get_ports { CLK }];

set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports { RST }];
```

实验结果示例

- 按住按钮时

- 不按按钮时会开始计时，从123.4s向后计时



- 再次按住按钮会重置为123.4s

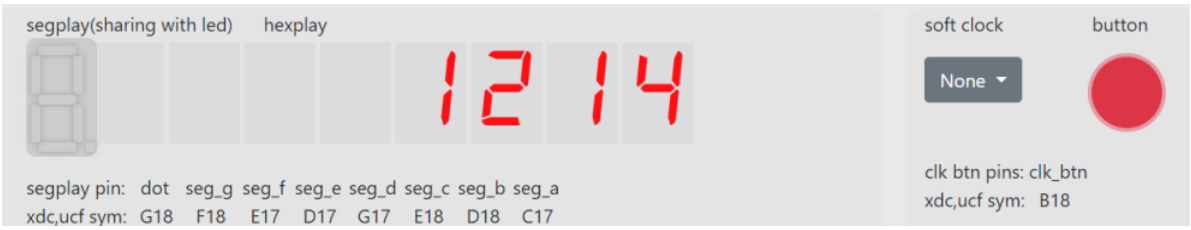
（PS：本题一开始写成了尴尬的倒计时，设计文件的代码与现在设计文件的代码几乎一致，无非是借位与进位有区别，毕竟debug也是实验过程的一部分，还是展示出来吧~~）

- 这是倒计时计数器代码

```
module T3(
input CLK,
input RST,
output reg [3:0] d,
```

```
output reg [1:0] an
);
reg [3:0] N3,N2,N1,N0;
wire clk_10hz;
reg [4:0] cnt;
generate_10hz CLK_change1(CLK,clk_10hz);
always@(posedge CLK or posedge RST)
begin
    if(RST) begin//置位1234
    N3 <= 4'h1; N2 <= 4'h2; N1 <= 4'h3; N0 <= 4'h4; end
    else if(clk_10hz)
        begin
        N0 <= N0 - 4'h1;
            if(N0==0)
            begin
                N0 <= 4'h9;
                N1<=N1 + 4'h1;
                if(N1==0)
                begin
                    N1<=4'h9;
                    N2 <= N2 - 4'h1;
                    if(N2==0)
                    begin
                        N2<=4'h5;//1分=60秒
                        N3 <=N3 - 4'h1;
                        if(N3==0)
                            if(N3 == 0 && N2 == 0 && N1 == 0 && N0==0)
                            begin
                                N3<= 4'h0; N2 <= 4'h0; N1 <= 4'h0;N0 <=
4'h0;
                            end
                    end
                end
            end
        end
end
always@(posedge CLK)
    begin
    cnt=cnt + 1;
    an<=cnt[4:3];
        if(cnt[4:3]==2'b11)d<=N3;
        if(cnt[4:3]==2'b10)d<=N2;
        if(cnt[4:3]==2'b01)d<=N1;
        if(cnt[4:3]==2'b00) d<=N0;
    end
endmodule
```

这是它的运行结果



总结与思考

- 收获：对FPGAOL的使用更熟练、更了解，了解了IP核相关知识。
- 难易程度：难！
- 任务量：巨大，写了一天，debug了一天。
- 改进建议：手册里能再多给点提示吗，真的绷不住~