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：黄瑞轩

学生学号：PB20111686

完成日期：2021.12.01

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

- VLAB 平台： vlab.ustc.edu.cn
- FPGAOL 实验平台：fpgaol.ustc.edu.cn
- Logisim
- Vivado 工具

实验练习

【题目 1】

- 首先，根据指导手册中的示例实例化一个 16*8 bit 的 ROM，命名为 dist_men_gen_0，初始化文件（coe 文件）内容如下所示：

```
memory_initialization_radix=16;
memory_initialization_vector=3f 06 5b 4f 66 6d 7d 07 7f 6f 77 7c 39 5e 79 71;
```

第二行的数字代表的是 0~F 对应的七段数码管的输入。

- 创建设计文件 test.v 如下：

```
module test(input [3:0] sw, output [7:0] led);
    dist_mem_gen_0 dist_mem_gen_0_in(.a(sw),.spo(led));
endmodule
```

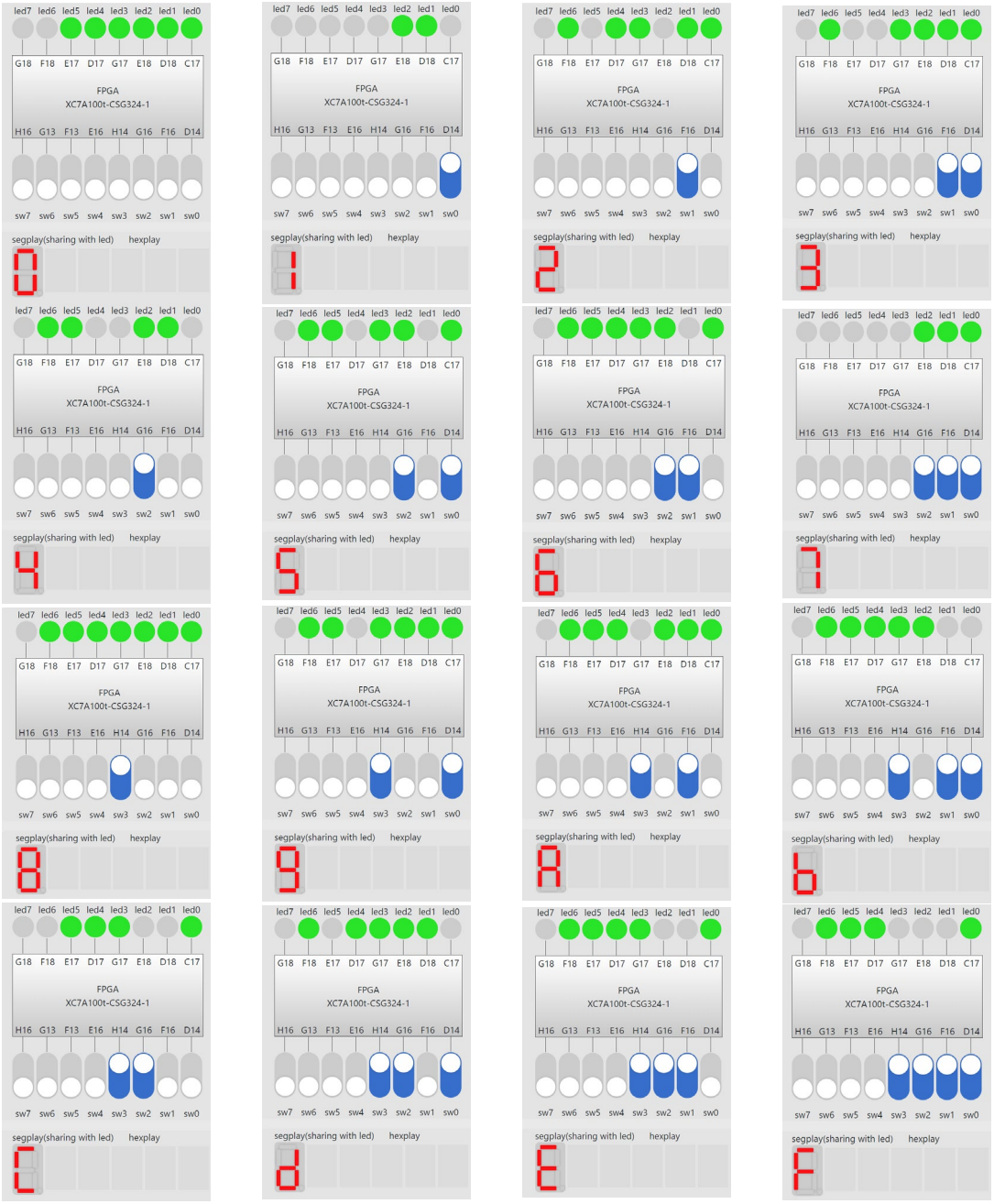
这里的.spo 端口是将 ROM 中已选择的内存译码输出端口。

- 创建约束文件 test_xdc.xdc 文件如下：

```
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] }];
```

- 生成 bitstream 文件，上传到 fpgaol 平台烧写，测试结果如下。



【题目 2】

- 创建设计文件 test.v 如下：

```
module test(
    input CLK100MHZ,
    input [7:0] sw,
    output reg [2:0] hexplay_an,
    output reg [3:0] hexplay_data
);
reg [32:0] hexplay_cnt;
// 时分复用
always@(posedge CLK100MHZ) begin
    if (hexplay_cnt >= (2000000 / 8))
        hexplay_cnt <= 0;
    else
        hexplay_cnt <= hexplay_cnt + 1;
end
always@(posedge CLK100MHZ) begin
    if (hexplay_cnt == 0) begin
        if (hexplay_an == 1)
            hexplay_an <= 0;
        else
            hexplay_an <= hexplay_an + 1;
    end
end
// 更新 hexplay_data 的值
always@(*) begin
    case(hexplay_an)
        1: hexplay_data = {sw[7],sw[6],sw[5],sw[4]};
        0: hexplay_data = {sw[3],sw[2],sw[1],sw[0]};
    endcase
end
endmodule
```

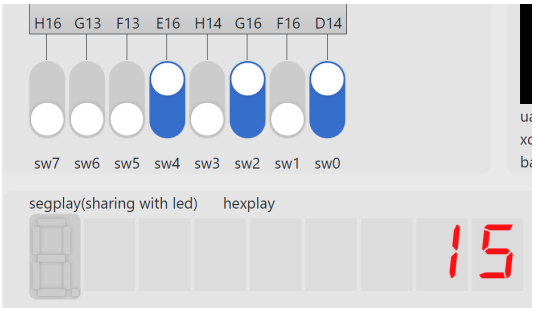
- 创建约束文件 test.xdc 文件如下：

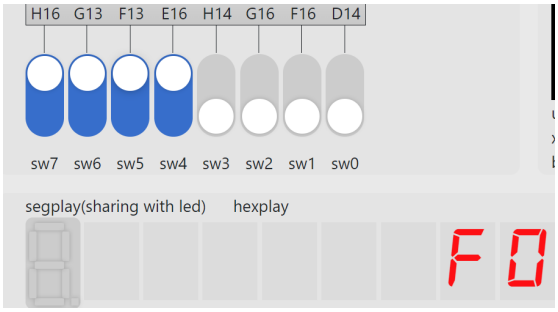
```
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { CLK100MHZ }];

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 { hexplay_data[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { hexplay_data[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { hexplay_data[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { hexplay_data[3] }];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports { hexplay_an[0] }];
set_property -dict { PACKAGE_PIN B16   IOSTANDARD LVCMOS33 } [get_ports { hexplay_an[1] }];
set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports { hexplay_an[2] }];
```

- 按实验指导手册上的实验过程生成.bit 文件，上传到 fpgaol 平台烧写，结果如下所示。





【题目 3】

- 此处应注意到我们的数据只有 2、4、8、16 之类的进制，题目要求实现一个分、秒和 1/10 秒的计数器，应设法将计数器修改为 10 进制/6 进制。
- 创建设计文件 test.v 如下：

```
module test(
    input CLK100MHZ,
    input rst,
    output reg [2:0] hexplay_an,
    output reg [3:0] hexplay_data
);
reg [15:0] data;
reg [32:0] hexplay_cnt;

always@(posedge CLK100MHZ) begin
    if (hexplay_cnt >= (2000000 / 8))
        hexplay_cnt <= 0;
    else
        hexplay_cnt <= hexplay_cnt + 1;
end

always@(posedge CLK100MHZ) begin
    if (hexplay_cnt == 0)begin
        if (hexplay_an == 3)
            hexplay_an <= 0;
        else
            hexplay_an <= hexplay_an + 1;
    end
end

always@(*) begin
    case(hexplay_an)
        0: hexplay_data = data[3:0];
        1: hexplay_data = data[7:4];
        2: hexplay_data = data[11:8];
        3: hexplay_data = data[15:12];
    endcase
end

reg [26:0] timer_cnt;
always@(posedge CLK100MHZ) begin
    if (timer_cnt >= 10000000)
        timer_cnt <= 0;
    else
        timer_cnt <= timer_cnt + 1;
end
```

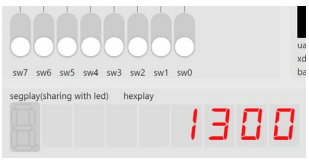
```
// 下面是给计数器自增部分的代码，为了避免出现闪动，这里直接对每一位是否应该进位进行判断
always@(posedge CLK100MHZ) begin
    if (timer_cnt == 0) begin
        if(!rst) begin
            if(data[3:0] < 4'b1001) data <= data + 1;
            if(data[3:0] == 4'b1001) begin
                data[3:0] <= 4'b0000;
                if(data[7:4] == 4'b1001) begin
                    data[7:4] <= 4'b0000;
                    if(data[11:8] < 4'b0101) begin
                        data[11:8] <= data[11:8] + 1;
                    end
                    else begin
                        data[11:8] <= 4'b0000;
                        data[15:12] <= data[15:12] + 1;
                    end
                end
                else data[7:4] <= data[7:4] + 1;
            end
        end
        else data[15:0] <= 16'b0001_0010_0011_0100;//复位 1234
    end
end

endmodule
```

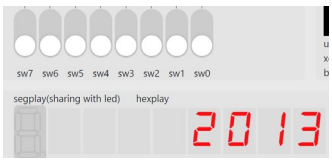
- 创建约束文件 test.xdc 如下：

```
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { CLK100MHZ }];
set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { rst }];
set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports { hexplay_data[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { hexplay_data[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { hexplay_data[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { hexplay_data[3] }];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports { hexplay_an[0] }];
set_property -dict { PACKAGE_PIN B16   IOSTANDARD LVCMOS33 } [get_ports { hexplay_an[1] }];
set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports { hexplay_an[2] }];
```

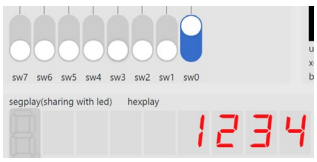
- 按实验指导手册上的实验过程生成.bit 文件，上传到 fpgaol 平台烧写，结果如下所示。



经过 1 分 30 秒



经过 2 分 1.3 秒



复位

总结与思考

- 本次实验中在前面实验的基础上了解了 Vivado 也可以像 C 语言那样调用库函数的方法来调用 IP 核来设计电路，并且知道了时分复用的 FPGA 设计方法，总体来说收获很大；
- 本次试验我借助 fpgaol 平台的 examples 了解了 hexplay 的具体用法，结合前面实验的经验完成了本次实验。我认为本次实验难度适中。
- 本次实验只有 3 小题，任务量不大；Vivado 的 Generate Bitstream 每次需要消耗的时间较长，会导致多出很多调试时间；
- 改进建议：
  在 fpgaol 平台上提供一些示例文件以供烧写；适当延长 fpgaol 每个节点的单次可使用时间。