**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] }];

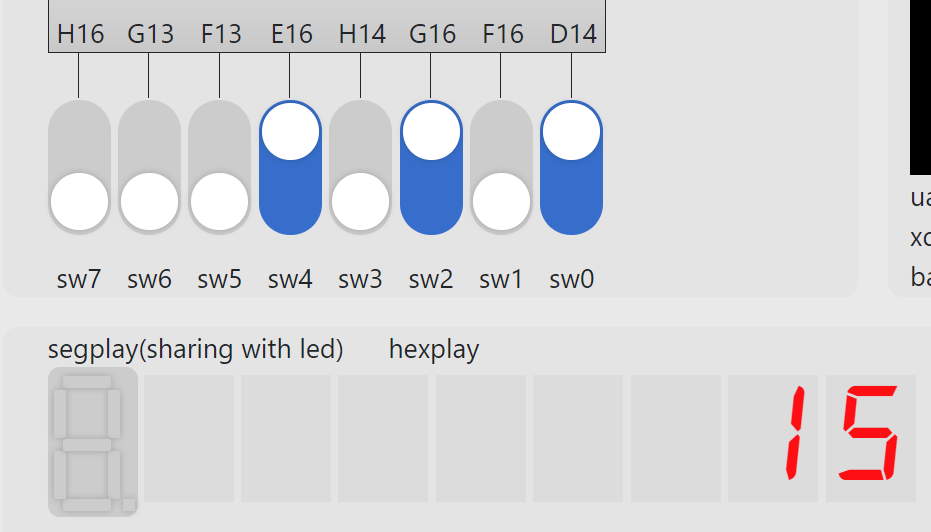
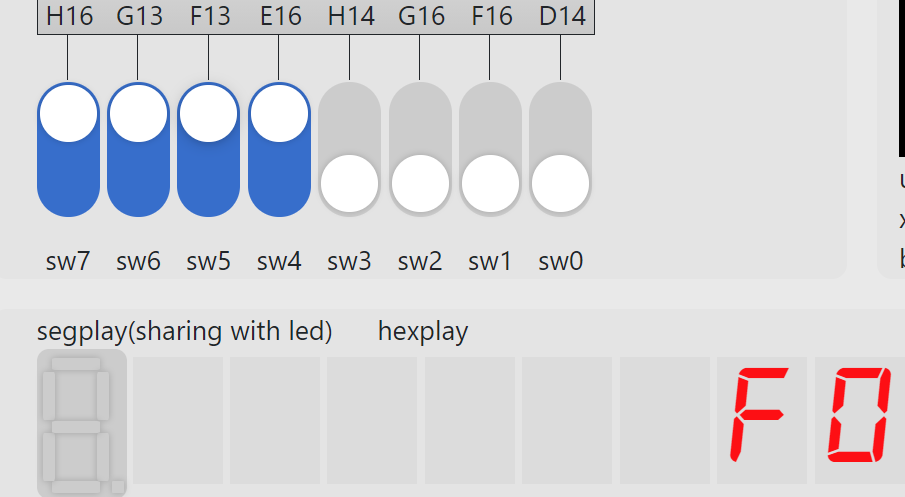
set\_property -dict { PACKAGE\_PIN A15 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[3] }];

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] }];

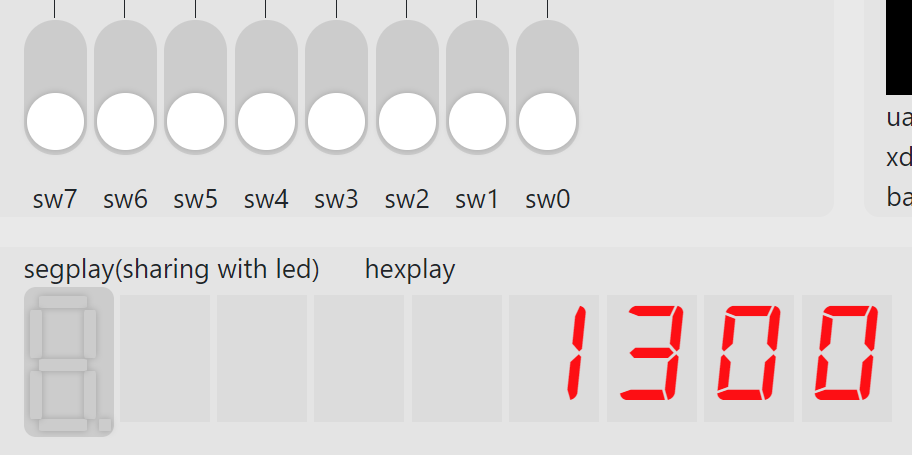
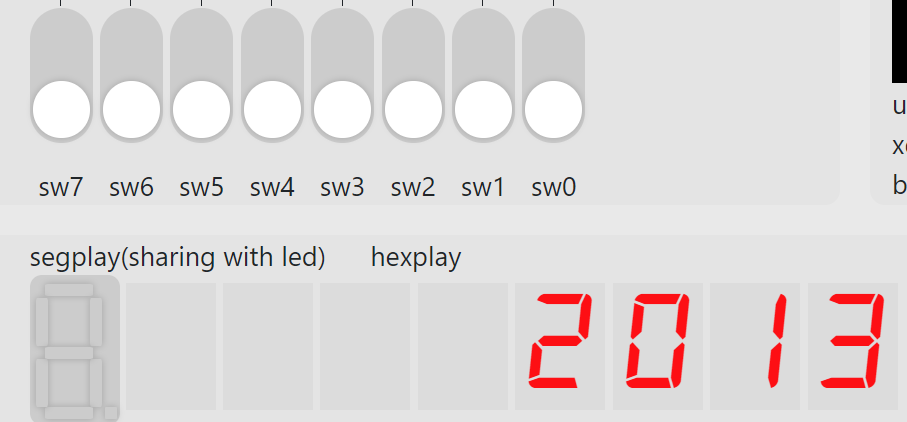
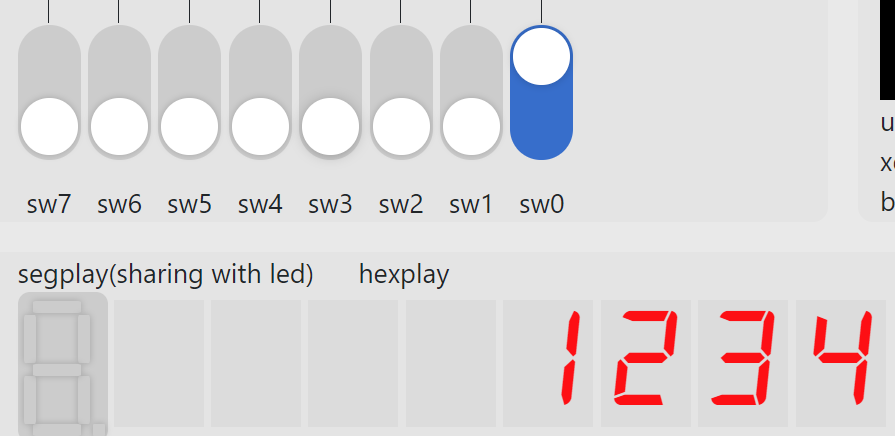
set\_property -dict { PACKAGE\_PIN A15 IOSTANDARD LVCMOS33 } [get\_ports { hexplay\_data[3] }];

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。