线上自学复习实验 01_简单组合逻辑

简介

通过系列实验，希望能够帮助数字电路基础较为薄弱的同学快速掌握完成计算机组成原理实验所需的逻辑电路设计基础。

本次实验中，我们将学习简单组合逻辑电路的设计方法。

实验目的

了解在线实验平台结构

学习使用远程在线实验平台

掌握简单组合逻辑的设计方法

实验环境

PC 一台（网络流畅）

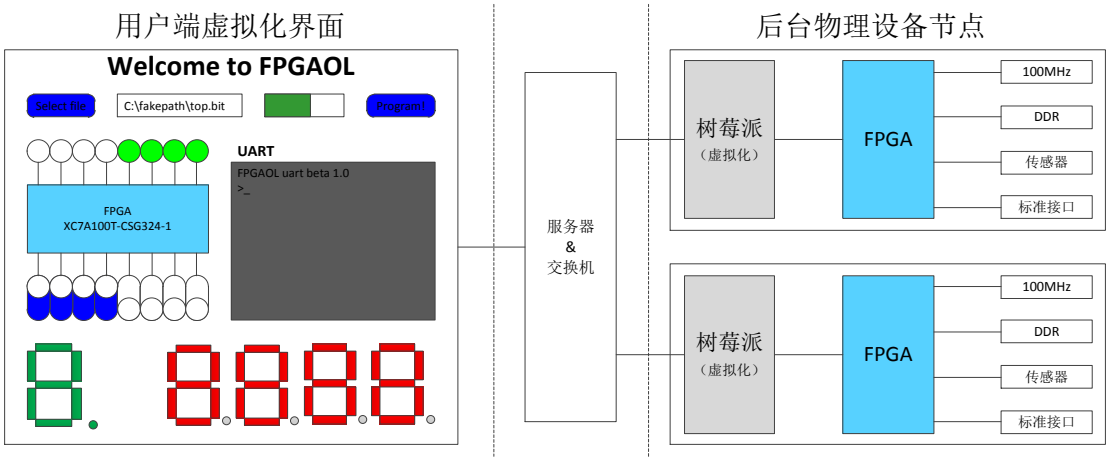
远程桌面环境（vlab）

FPGA 远程实验平台（FPGAOL）

实验步骤

Step1： FPGAOL 在线实验平台介绍

下图为 FPGAOL 在线实验平台的结构图，包含用户端和后台两部分。



用户端可通过浏览器访问，包含：申请设备节点、烧写 FPGA、通

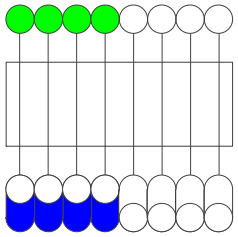
过开关和 LED 等虚拟外设接口与 FPGA 通信等功能。用户端的虚拟接口与 FPGA 的物理接口一一对应，因此对虚拟接口的操作实际上就是对 FPGA 物理接口的操作。

用户除了可以操作虚拟接口外，还可以直接利用后台物理设备节点上与 FPGA 直接相连的各种外设，如 100MHz 时钟、DDR 内存颗粒、各类传感器等。

注意：FPGAOL 平台使用的 FPGA 型号为 XC7A100T-CSG324-1（与 Nexys4DDR 开发板一致），因此在 Vivado 中建立工程时应该选择该型号作为目标芯片才可以生成能够在 FPGAOL 平台上运行的电路。

Step2： 简单直连组合逻辑设计

首先，我们将通过实验平台完成一个最简单的组合逻辑电路：用 8 个虚拟开关来控制 8 个虚拟 LED 灯。即在 FPGA 内部将开关和 LED 两两相连，当开关推上去时，对应 LED 亮，否则不亮，如下图所示。



其 Verilog 代码为：

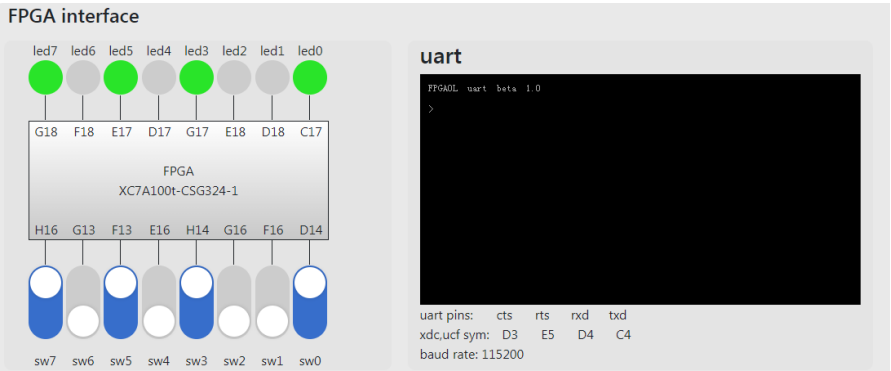
```
module top(
input   [7:0] sw,
output  [7:0] led);
    assign  led = sw;
endmodule
```

其管脚约束文件为：

```
#Switch
set_property -dict { PACKAGE_PIN H16    IOSTANDARD LVCMOS33 } [get_ports { sw[7] }];
set_property -dict { PACKAGE_PIN G13    IOSTANDARD LVCMOS33 } [get_ports { sw[6] }];
```

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

在 Vivado 中建立工程，将设计文件和约束文件导入工程，并进行综合、实现，将生成的 bit 文件烧写到 FPGAOL 实验平台，观察运行结果，可得如下图所示结果。



可以看出，LED 灯的状态与开关状态一一对应，每个开关控制一个与其相对应的 LED 灯的亮灭。

Step3：具有优先级的 8-3 编码器

在这一步骤中，我们将设计一个具有优先级的 8-3 编码器，其真值表如下所示，其中 error 信号输出到最左侧的 led 灯（led[7]）上：

输入　sw[7:0]	输出　led[2:0]	输出　error
8'b1xxx_xxxx	3'b111	1'b0
8'b01xx_xxxx	3'b110	1'b0
8'b001x_xxxx	3'b101	1'b0

8'b0001_xxxx	3'b100	1'b0
8'b0000_1xxx	3'b011	1'b0
8'b0000_01xx	3'b010	1'b0
8'b0000_001x	3'b001	1'b0
8'b0000_0001	3'b000	1'b0
8'b0000_0000	3'b000	1'b1

其 Verilog 代码为：

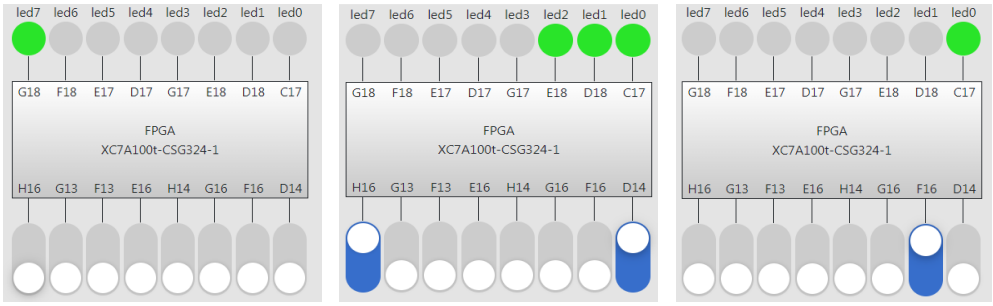
```
module top(
input       [7:0] sw,
output reg  [2:0] led,
output            error);
assign error = ~(|sw);
always@(*)
begin
    if(sw[7])       led = 3'b111;
    else if(sw[6])  led = 3'b110;
    else if(sw[5])  led = 3'b101;
    else if(sw[4])  led = 3'b100;
    else if(sw[3])  led = 3'b011;
    else if(sw[2])  led = 3'b010;
    else if(sw[1])  led = 3'b001;
    else if(sw[0])  led = 3'b000;
    else            led = 3'b000;
end
endmodule
```

其管脚约束文件为：

```
#Switch
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { sw[7] }];
set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { sw[6] }];
set_property -dict { PACKAGE_PIN F13   IOSTANDARD LVCMOS33 } [get_ports { sw[5] }];
set_property -dict { PACKAGE_PIN E16   IOSTANDARD LVCMOS33 } [get_ports { sw[4] }];
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }];
set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }];
set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }];
#Led
set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports { error }];
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] }];
```

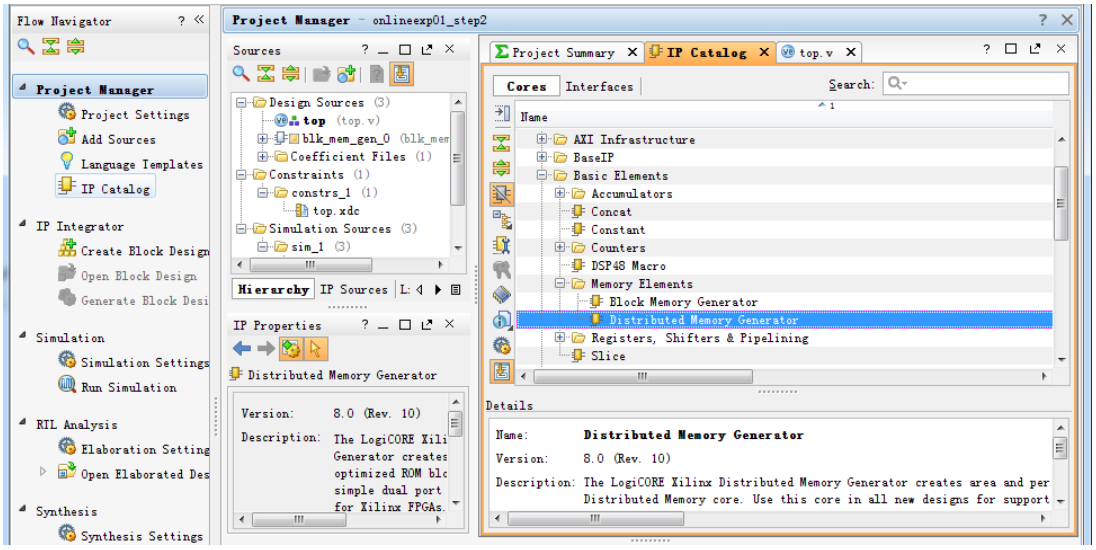
将生成的 bit 文件烧写到 FPGAOL 实验平台，其运行结果如下图所

示，运行结果与预期一致。：

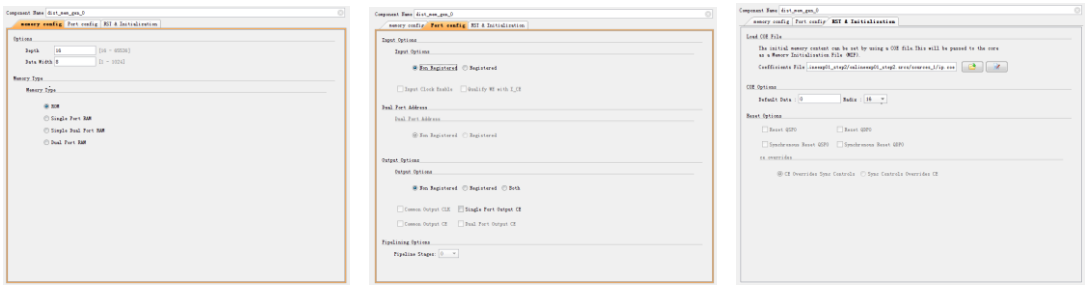


step4：从两个 ROM 中读取数据，相加并输出

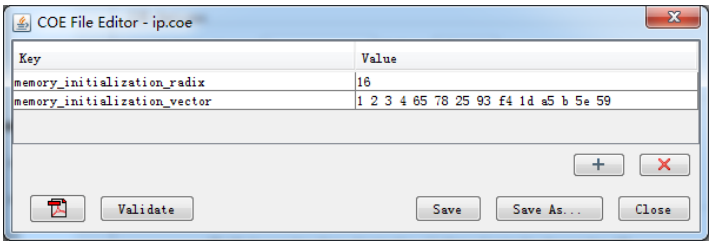
在 Vivado 中新建一个工程，点击“IP Catalog”按钮，创建一个只读存储器的 IP 核，如下图所示：



在参数设置界面使用如下图所示的参数（深度为 16，位宽为 8，类型为 ROM，使用初始化文件）：



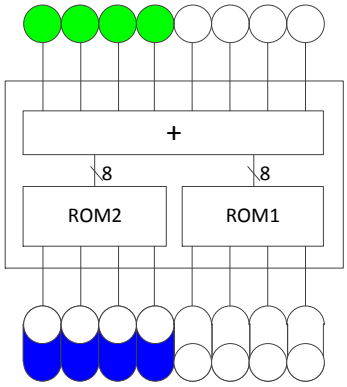
其初始化文件如下图所示（初始化数据可自行修改）：



IP 核例化完成中，将在“*工程目录\工程名.srcs\sources_1\ip\IP 核名*”目录下生成该 IP 核的接口文件，用户可直接依照该文件对 IP 核进行例化使用。本例中，ROM IP 核的接口文件内容如下所示：

```
(* x_core_info = "dist_mem_gen_v8_0_10,Vivado 2016.2" *)
module dist_mem_gen_0(a, spo)
/* synthesis syn_black_box black_box_pad_pin="a[3:0],spo[7:0]" */;
  input [3:0]a;
  output [7:0]spo;
endmodule
```

在顶层设计文件中将该 ROM IP 核例化两次，用分别用 4 个开关控制 ROM 的地址，并将两个 ROM 的输出数据相加，结果输出到 8 个 LED 上，如下图所示



下面是提供的顶层设计文件模板，请试着将代码补充完整：

```
module top(
input   [3:0] rom1_addr,rom2_addr,
output  [7:0] led);
wire    [7:0] rom1_data,rom2_data;
dist_mem_gen_0 rom1(
.a    (rom1_addr),
.spo  (rom1_data));
dist_mem_gen_0 rom2(
```

```
    //to be added
));
assign led =  ; //to be added
endmodule
```

其管脚约束文件为：

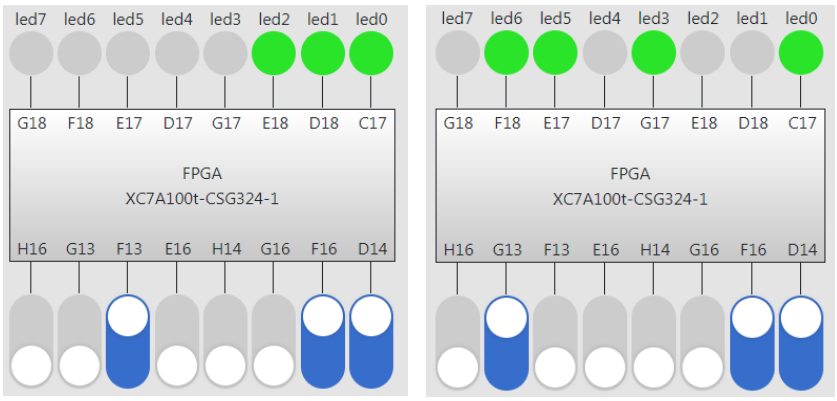
```
#Switch
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { rom2_addr[3] }];
set_property -dict { PACKAGE_PIN G13   IOSTANDARD LVCMOS33 } [get_ports { rom2_addr[2] }];
set_property -dict { PACKAGE_PIN F13   IOSTANDARD LVCMOS33 } [get_ports { rom2_addr[1] }];
set_property -dict { PACKAGE_PIN E16   IOSTANDARD LVCMOS33 } [get_ports { rom2_addr[0] }];
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { rom1_addr[3] }];
set_property -dict { PACKAGE_PIN G16   IOSTANDARD LVCMOS33 } [get_ports { rom1_addr[2] }];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { rom1_addr[1] }];
set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { rom1_addr[0] }];
#Led
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] }];
```

ROM 初始化文件内容为：

```
memory_initialization_radix=16;
memory_initialization_vector=1 2 3 4 65 78 25 93 f4 1d a5 b 5e 59 ;
```

其中第一个参数表示 16 进制显示，第二个参数为从 0 地址开始的初始化数据序列。

将生成的 bit 文件烧写到 FPGAOL，观察运行结果是否与预期相同，如下图所示，



试修改 ROM 的初始化内容，并重新综合实现烧写，观察运行结果。

总结与思考

1. 请总结本次实验的收获

2. 请评价本次实验的难易程度

3. 请评价本次实验的任务量

4. 请为本次实验提供改进建议