实验平台使用手册

本学期的计算机组成原理课程系列实验需要学生：

1）使用 **Verilog** 语言设计 RISC-V 指令集 CPU 相关电路；

2）通过 **Vivado** 进行仿真、综合，生成正确的电路文件；

3）将电路文件烧写到 **FPGA** 平台上进行验证。

针对上述三个步骤，本实验中心开发了三个平台供同学们使用，分别是：

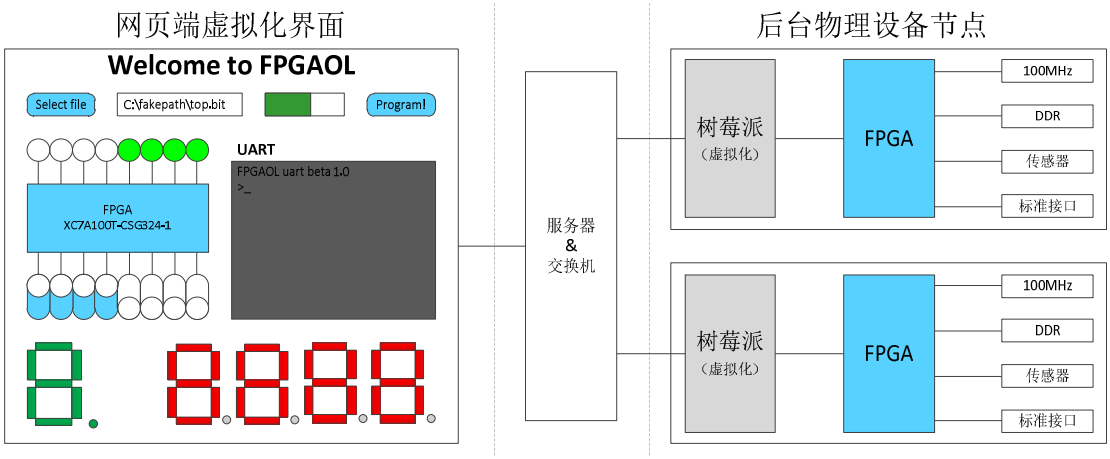
1) Verilog OJ 在线测评系统, (平台网址为：vlab.ustc.edu.cn/oj)支持在线编写、仿真 Verilog 代码，帮助同学们快速掌握 Verilog 语言，目前该平台已经在试运行，Verilog 题库正在完善中；

2）VLAB 平台（网址为：vlab.ustc.edu.cn，支持统一身份认证登录）。组成原理实验中用到的 Vivado 软件占用空间非常大，且系统兼容性差。往年学生做实验需要花费大量时间来安装此软件。为节省同学的时间，本实验中心开发了 VLAB 平台，为每一位同学提供一台远程的 Linux 虚拟机，预装了 Vivado、Logisim 等必备软件，同学们可登陆平台网站进行体验。

3）FPGAOL 平台（网址为：fpgaol.ustc.edu.cn,支持统一身份认证登录和游客登录）。我们搭建了一套在线的 FPGA 实验平台，同学们可以在线获取 FPGA 设备节点进行硬件实验。

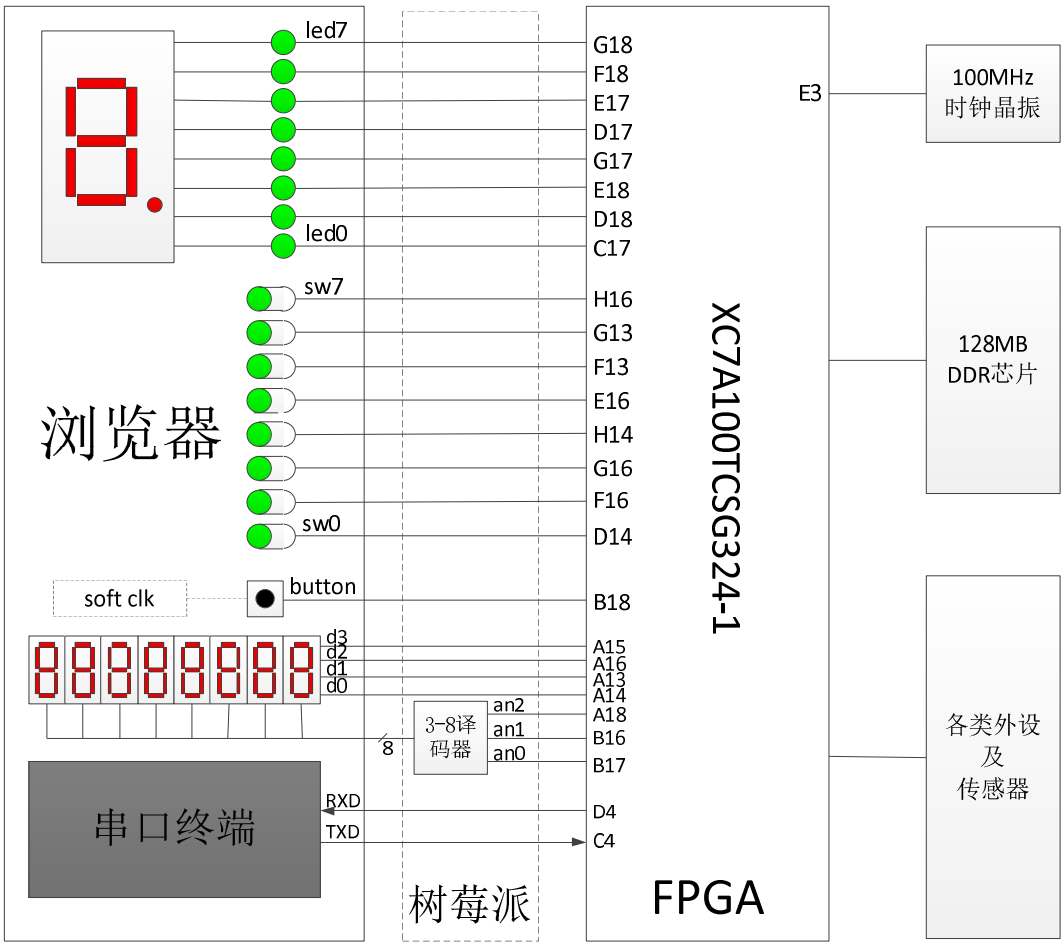
FPGAOL 在线平台介绍

FPGA online（简称 FPGAOL）是中国科学技术大学计算机学院实验中心自主开发的一套在线实验平台，该平台通过分时复用的方式为用户提供 FPGA 设备节点的在线服务，目前 FPGAOL 一代系统已正式对外提供服务，用户可通过网址 fpgaol.ustc.edu.cn 访问平台，并使用统一身份认证登陆。



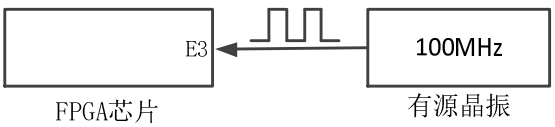
FPGAOL 平台的每个设备节点都包含了一个树莓派和一个 FPGA 板卡，树莓派与 FPGA 芯片的 26 个 IO 管脚直接相连，并最终映射到网页端，其中包括 8 个对应 LED（同时与七段数码管复用），8 个对应开关，7 个对应 8 位 16 进制数码管，1 个对应按键，2 个对应串口 UART。用户在网页端对虚拟化外设进行的操作都会通过树莓派映射到对应的 FPGA 管脚上。同理，FPGA 管脚的电平变化也会通过树莓派反应到网页端的虚拟化外设上。因此，用户是通过网页端的虚拟化外设实际地控制物理设备节点。

除了网页端的虚拟化外设，FPGA 板卡上还带有板载 100MHz 时钟，128MB DDR 内存芯片，以及各类传感器及接口外设，如温度传感器、麦克风等，具体管脚对应关系可参考下图或网站提供的 XDC 文件。

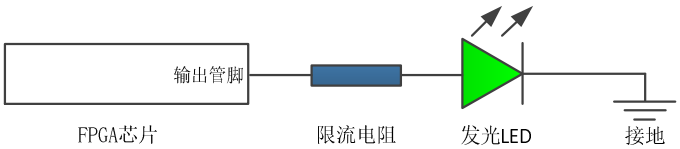


外设工作原理介绍

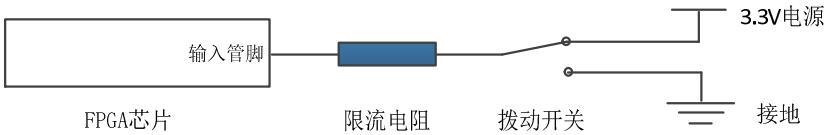
FPGAOL 实验平台的开发板上有一个 100MHz 的时钟晶振，该晶振与 FPGA 芯片的 E3 管脚直接相连，在开发板上电后，会持续的为 FPGA 芯片提供一个 100MHz 频率的时钟信号，不需要用户进行操作，如下图所示。



FPGAOL 实验平台中 LED 的工作方式为：FPGA 管脚为高电平时对应 LED 点亮，为低电平时则熄灭，其原理图如下所示：

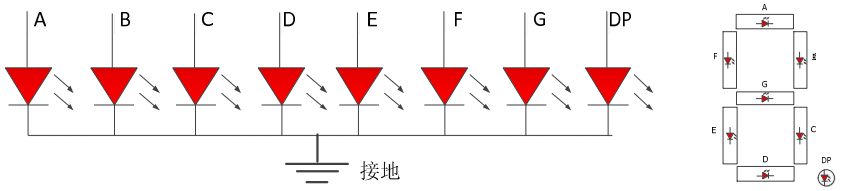


拨动开关作为输入设备，当开关推到上方后，对应 FPGA 管脚输入高电平，当开关拉下来之后，对应 FPGA 管脚输入低电平，其工作原理如下图所示：

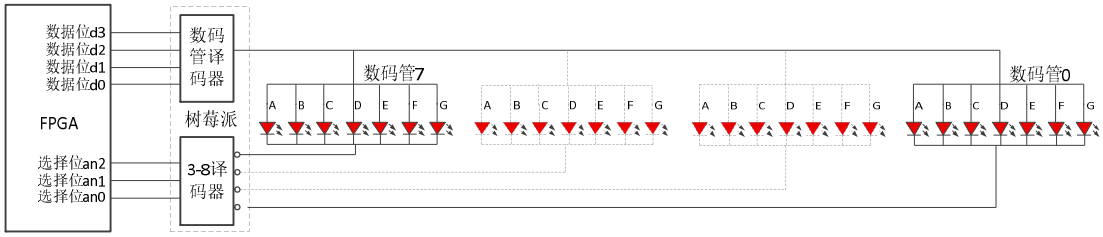


按键的工作原理与拨动开关类似，区别在于：按键按下时向 FPGA 芯片输入高电平信号，松开时输入低电平信号，按键默认处于松开状态。FPGAOL 1.0 实验平台中只有一个按键，且与 soft clock 功能复用（实际上 soft clock 功能暂未对用户开放）。

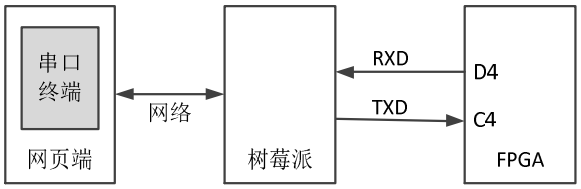
因平台可用的 FPGA 管脚数量有限，我们对七段数码管与 LED 的管脚进行了复用。数码管本质上是由 8 个 LED（发光二极管）构成，发光二极管的阴极共同连接到一端，并接到地，8 个阳极分别由 FPGA 的 8 个输出管脚控制，当输出管脚为高电平时，对应的 LED 亮起。通过控制 8 个 LED 的亮灭，便能显示出不同的字符，例如，当 A~F 6 个 LED 亮，G、DP 两个 LED 不亮时，显示的便是字符“0”。



平台包含一个 8 位的十六进制数码管，该数码管共用 4bit 位宽的数据位，同时由 3bit 的选择位经译码后生成 8 个选择信号，控制 8 个数码管的使能，数据位个选择位信号都是高电平有效。其工作原理如下图所示。



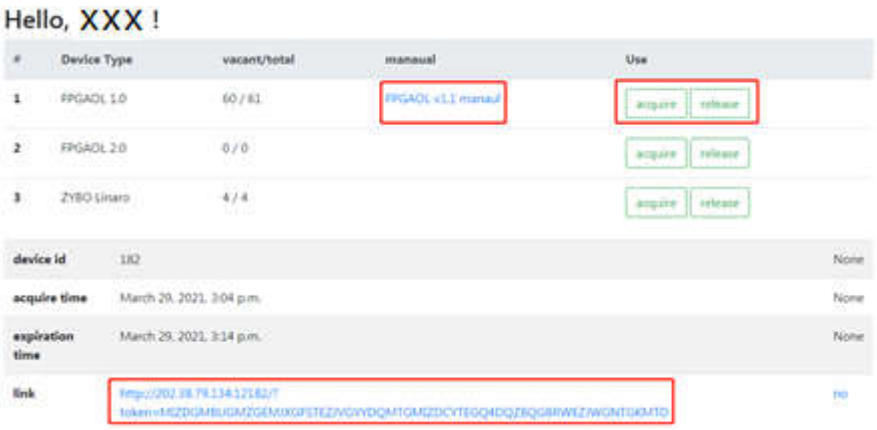
此外，FPGAOL 平台还集成了一个串口终端，FPGA 芯片上的两个 IO 直接与树莓派上的串口相连，在 FPGA 端编写程序，将 RXD 信号直接与 TXD 信号直连，然后在网页端的串口终端发送数据，便能够实时接收到从 FPGA 侧环回的数据了，感兴趣的同学可以实际测试一下。



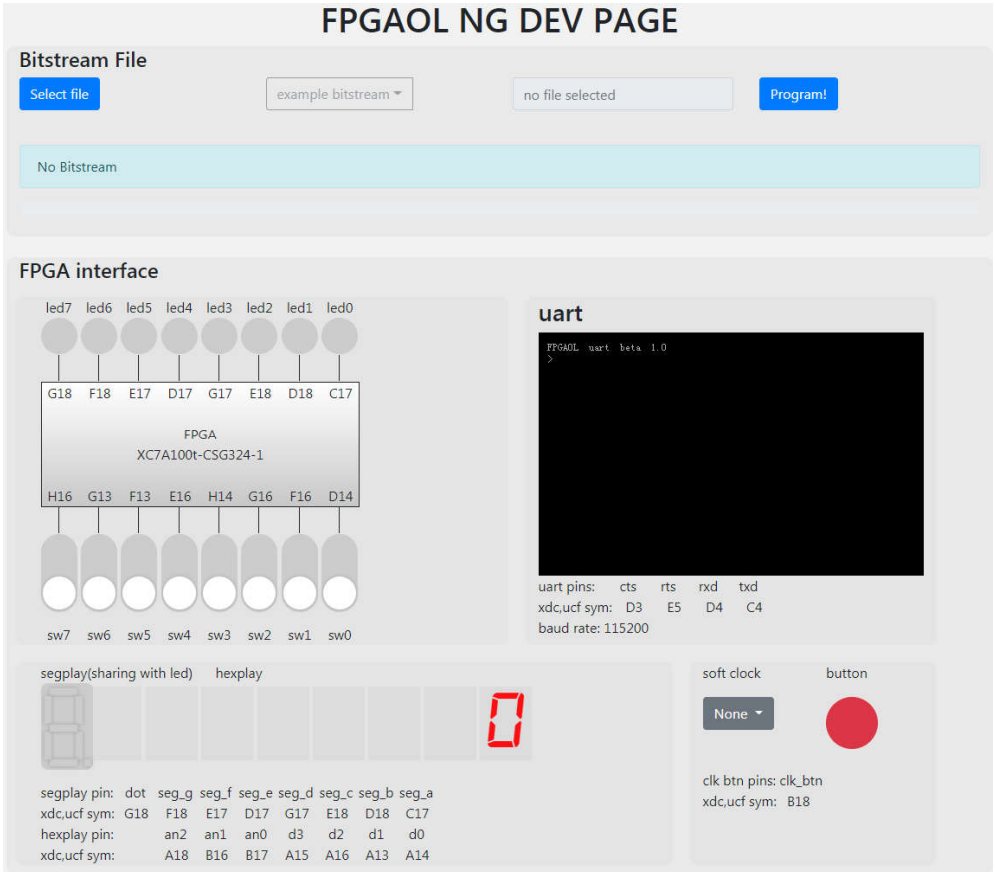
FPGAOL 在线平台使用流程

第一步：登录平台网站 fpgaol.ustc.edu.cn，并使用统一身份认证登录，或者直接以游客身份登录

第二步：进入设备获取界面，点击“acquire”按钮获取一个 FPGA 结点，成功获取后，将在下方 link 栏显示链接，用户可通过链接进入设备操作界面。每次默认使用时长为 10 分钟，10 分钟后自动释放结点，用户也可以点击“release”按钮手动释放。



第三步：进入设备操作界面烧写 FPGA，每个 FPGA 结点网页内包括 8 个 LED、8 个开关、8 个数码管、1 个按键、1 个串口。每种设备都与 FPGA 芯片的物理管脚相对应，用户操作的都是实际的物理设备。用户可点击“Select file”按钮，选择需要烧写的 bit 文件，点击“Program!”进行烧写（在网站首页的 Download 栏下有测试用的 bit 文件、样例工程以及管脚约束文件）。



附录：

FPGAOL 管脚约束文件（使用对应的管脚时，应将对应行开头的#删掉，并修改端口名称，以与 Verilog 设计文件一致）：

```
## Clock signal
#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}];

## FPGAOL LED (signle-digit-SEGPLAY)
#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] }];

## FPGAOL SWITCH
#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] }];

## FPGAOL HEXPLAY
#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] }];

## FPGAOL BUTTON & SOFT_CLOCK
#set_property -dict { PACKAGE_PIN B18 IOSTANDARD LVCMOS33 } [get_ports { btn }];

##USB-RS232 Interface
#set_property -dict { PACKAGE_PIN C4 IOSTANDARD LVCMOS33 } [get_ports { UART_TXD_IN }];
#set_property -dict { PACKAGE_PIN D4 IOSTANDARD LVCMOS33 } [get_ports { UART_RXD_OUT }];
#set_property -dict { PACKAGE_PIN D3 IOSTANDARD LVCMOS33 } [get_ports { UART_CTS }];
#set_property -dict { PACKAGE_PIN E5 IOSTANDARD LVCMOS33 } [get_ports { UART_RTS }];
```