19302010101 王海伟

这个lab因为没搞清楚reg和wire浪费了很多时间，干脆记录一下

wire V.S. reg

wire(组合逻辑)

wire用来连接模块实例化的输入和输出端口；
wire用作实际模块声明中输入和输出；
wire 元素必须由某些东西驱动，并且在没有被驱动的情况下，无法存储值；
wire 元素不能用在 always模块中 = 或者 <= 的左边；
wire元素是assign语句左侧 唯一的合法类型；
wire 元素是在基于Verilog的设计中连接两片的无状态方式;
wire 只能用在 组合逻辑；

reg(组合和时序逻辑)

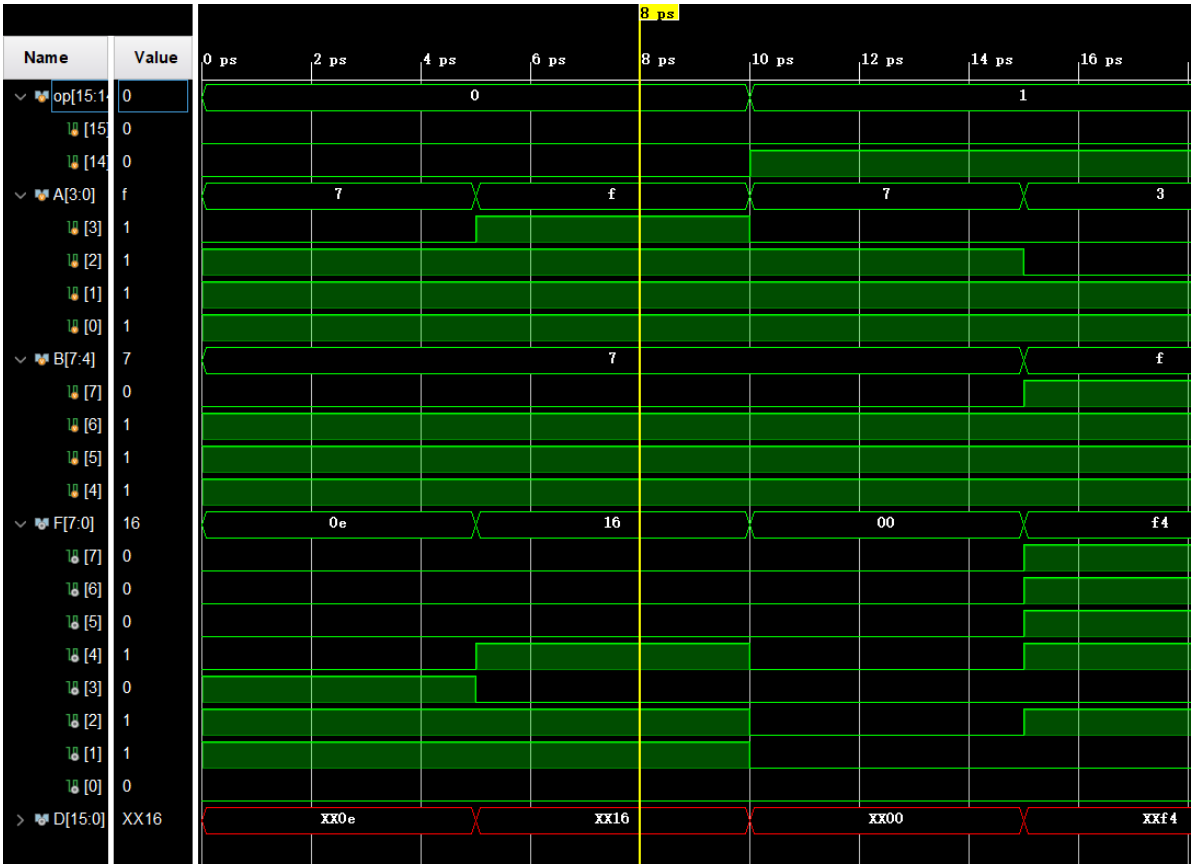
reg 和 wire类似，但它可以用来存储信息（状态），就像寄存器
reg可以连接到模块实例化的输入端口；
reg 不能连接到模块实例化的输出端口；
reg 可以用作实际模块声明中的输出；
reg 不能 用作实际模块声明中的输入；
reg是always 模块中 = 或者 <= 左侧的唯一正确类型；
reg是initial模块中 = 左侧唯一的合法类型；
reg不能用在assign的左边；
reg 当与always @（posedge Clock）块结合使用时，reg可用于创建寄存器。

什么时候wire 和reg 可以互换?

1. 两者都可以出现在assign语句和 always 模块中=或<=的右侧;
2. 两者都可以连接到模块实例的输入端口;

ALU

alu允许使用verilog自己的运算符其实减少了很多工作量，对于运算直接用运算符就好了，为了直观，我把输出扩展到了8位，但是对于进位符依旧是对于四位输出而言的，使用了最后一位led作为进位符号，使用前8位LED来记录输出，然后通过top module把数据传入数码管来负责显示，数码管的部分没做仿真，直接上板测试，ALU进行了简要仿真，以下为波形图。(截图放不下，省去了led的部分)



7段数码管

这个比较复杂，关键是需要时序逻辑，大概的思路就是输入一个时钟信号，然后每次收到时钟信号计数器加一，然后根据计数器的后三位（值是0~7）来选择要显示的数位，然后刷新A2G，当然刷新前需要根据输入计算得到每一位需要显示的数字。因为一个时钟周期非常短，这就导致了两个数字看上去仿佛是同时显示的。