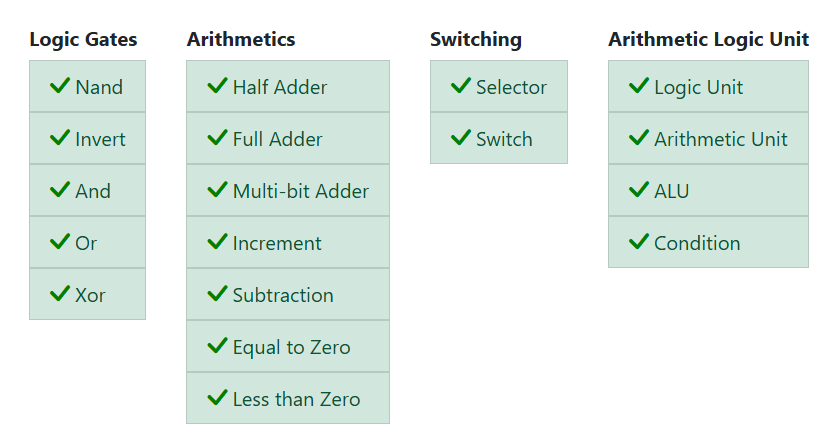
Verilog第三次作业

2211290 姚知言 计算机学院

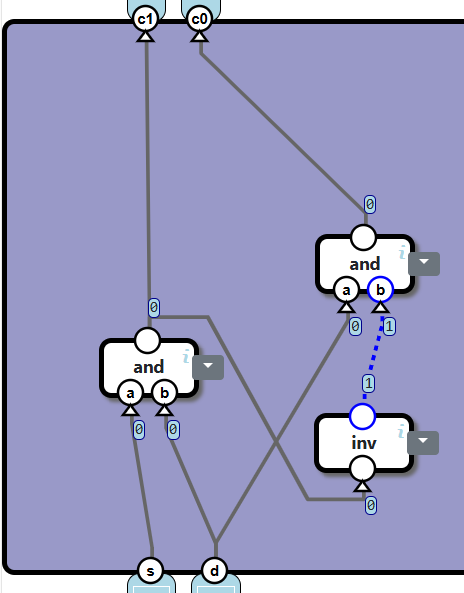
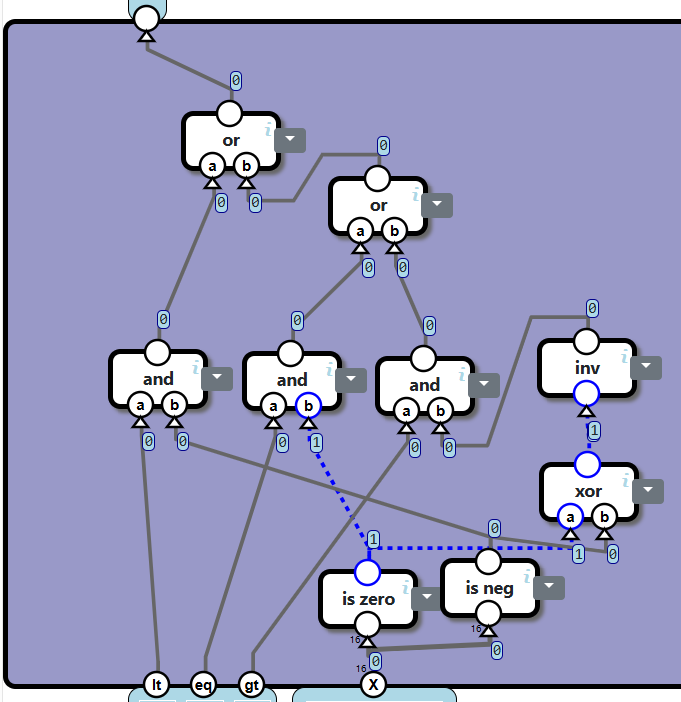
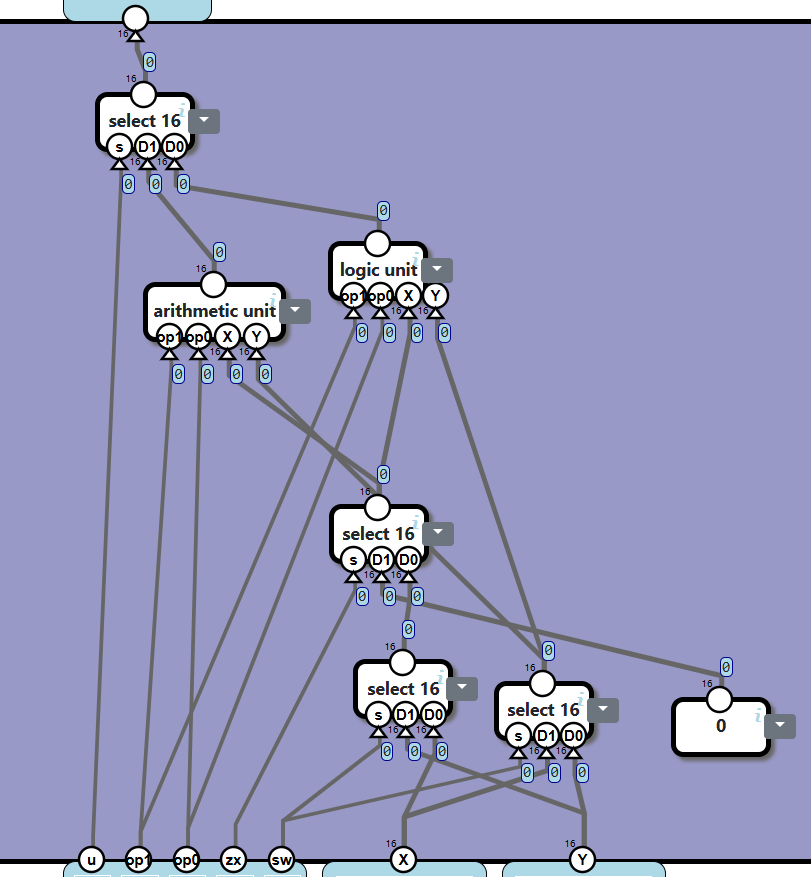
**1.继续完成nandgame里面的Switching和Arithmetic Logic Unit，记录耗时并总结收获。**

Switching部分已经在第一节课做完，ALU部分四个实验用时大约30分钟。在这两个部分的所有实验中，均实现了最简方案或者使用最少组件（但还可以使用更少的nandgates）的方案。

通过结果如下：



部分实验截图：（从左到右：ALU Condition Switch



在Switching部分中我主要搭建了选择逻辑电路，这一逻辑电路的意义在于能够根据额外的输入来选择把哪一个输入通向输出。选择电路的意义是比较大的，毕竟在日常生活中，我们经常会遇到分支情况。

在ALU部分我完成了逻辑计算模块，算数计算模块，并通过对前两个模块的应用完成了ALU，此外我还完成了条件判断。这些练习为我后续编写Verilog代码打下了基础。

**2.自行设计一个简单运算模块，要求如下：**

**（1）两个输入（ina，inb），分别为8位的二进制数**

**（2）多个输出，分别是这两个二进制数的运算结果，包括**

**sumab:两个输入之和，sumflag: 两个输入相加之后的进位**

**leftshiftA：把ina向左逻辑移位，移动的位数为inb，得到的结果**

**lessflag: ina小于inb时返回1，否则返回0**

**equalflag: ina等于inb时返回1，否则返回0**

**bitXorflag: 把ina按位缩减异或之后的结果**

以下是我的verilog实现：

multi\_alu.v:

module multi\_alu(

input [7:0] ina,

input [7:0] inb,

output [7:0] sumab,

output sumflag,

output [15:0] leftshiftA,

output lessflag,

output equalflag,

output bitXorflag

);

\_8bitadder my8bitadder(ina,inb,0,sumab,sumflag);

assign leftshiftA = ina << (inb % 4'd9);

assign lessflag = ina < inb;

assign equalflag = ina == inb;

assign bitXorflag = ^ina;

endmodule

对于加法的实现，调用上节课设计的8位加法器（\_8bitadder），设置两个输入ina，inb，输入进位为0，输出为sumab两数之和和sumflag向上进位。

ina左移inb，直接用inb%9平替inb小于9的情况。

lessflag和equalflag直接用小于和等于的判断赋值，按位异或使用对a的异或运算符。

我的testbench设计如下：

testbench.v:

module alu\_testbench();

reg [7:0] tina,tinb;

wire [7:0] tsumab;

wire tsumflag,tlessflag,tequalflag,tbitXorflag;

wire [15:0] tleftshiftA;

multi\_alu myalu(tina,tinb,tsumab,tsumflag,tleftshiftA,tlessflag,tequalflag,tbitXorflag);

initial begin

tina = 8'b0;

tinb = 8'b0;

end

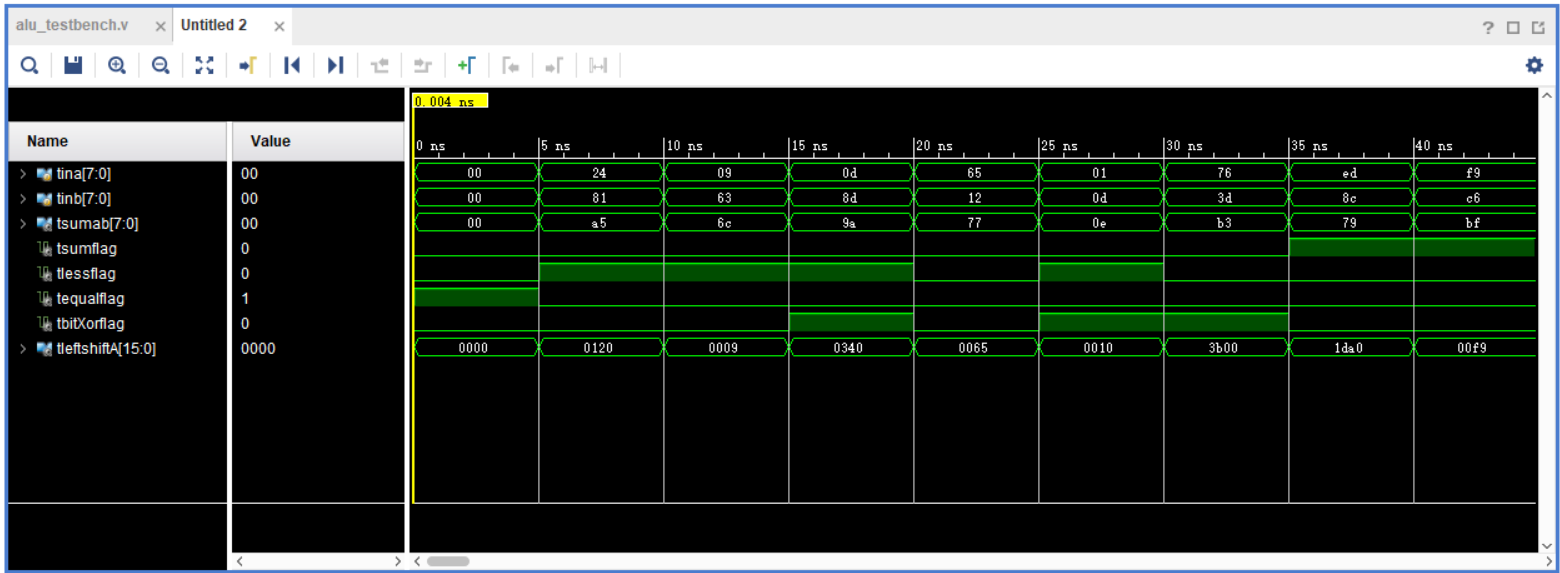
always #5 tina=$random % 9'b1\_0000\_0000;

always #5 tinb=$random % 9'b1\_0000\_0000;

endmodule

将ina和inb设计为，初始化为0，每5ns刷新一次的8位随机数。

仿真结果如下：



对于加法，能够正常完成计算，例如：

0x0+0x0=0x0(0-5ns),0x24+0x81=0xa5(5-10ns),0xed+0x8c=0x179(输出0x79，进位flag1,35=40ns)

对于相等运算，可以看到0x0=0x0(0-5ns)，flag=1，其他均不等，flag=0

对于小于运算，可以看到所有lessflag为1的点tina都小于tinb，为0的点都大于等于。

对于按位异或运算，例如：0x24=>0010 0001b(5-10ns)

0^0=0 0^1=1 1^0=1 1^0=1 1^0=1 1^0=1 1^1=0，所以结果为0

或者更直接的，按位异或本身就是统计1的个数是否为奇数。

同样的，0x76=>0111 0110b(30-35ns)，显然结果为1

对于左移运算，因为直接用inb模9平替（显然在inb小于9可以得到相同的结果），对于10-15ns，0x63=99，99%9=0，左移0位，结果等于ina，对于15-20ns，0x8d=141，141%9=6，左移6位，0x0d=>1101b，左移6位应为11 0100 0000b，即0x0340，完成验证。

至此，所有模块都得到的正确验证。