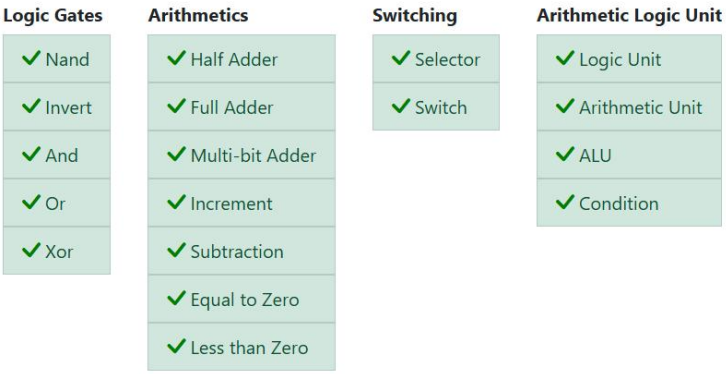
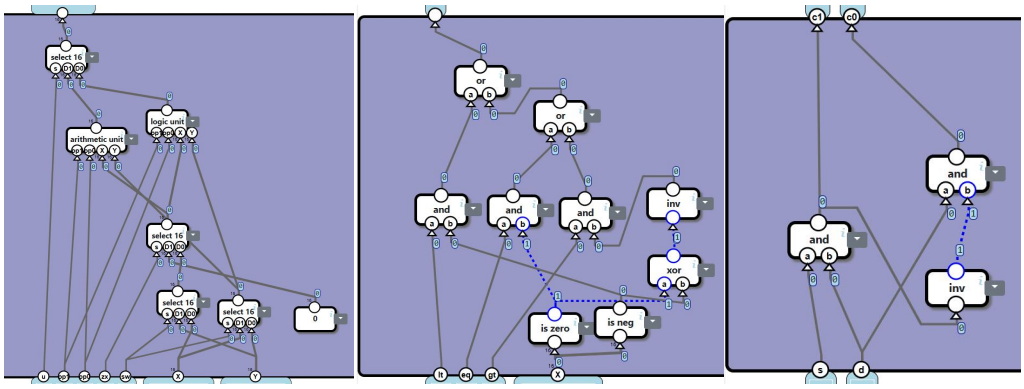
Verilog 第三次作业
2211290 姚知言 计算机学院

**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
```

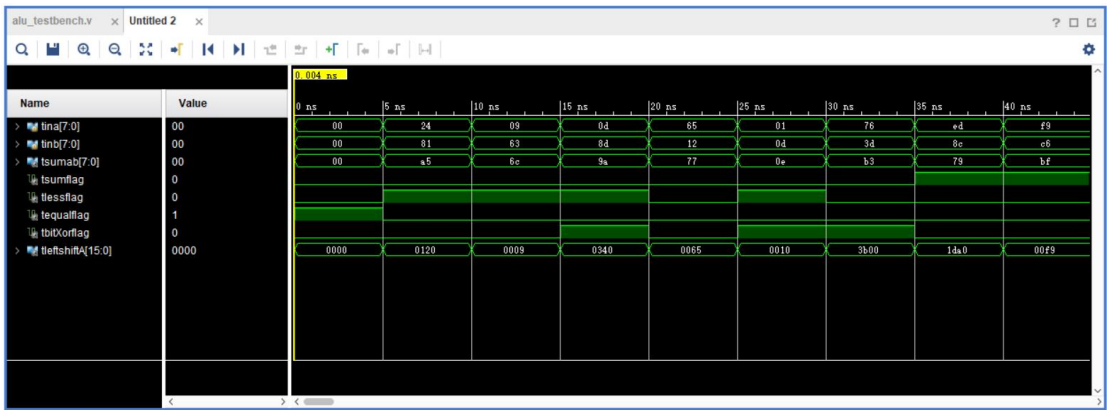
对于加法的实现，调用上节课设计的 8 位加法器（_8bitadder），设置两个输入 ina，inb，输入进位为 0，输出为 sumab 两数之和和 sumflag 向上进位。
ina 左移 inb，直接用 inb%9 平替 inb 小于 9 的情况。
lessflag 和 equalflag 直接用小于和等于的判断赋值，按位异或使用对 a 的异或运算符。

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。