**Verilog第 3 次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 设计一个简单预算的模块 | | | | |
| 学生姓名 | 邢清画 | 学号 | 2211999 | 指导老师 | 董前琨 |
| 实验地点 | 津南实验楼A区308 | | 实验时间 | 2023年11月15日 | |

1. **实验项目名称**

设计一个简单预算的模块

1. **实验目的**

让学生应用Verilog编程语言来实现一个基础的算术和逻辑运算模块，涉及二进制数的加法、移位、比较和异或等操作。通过这个实验，学生可以加深对数字逻辑和Verilog编程的理解。实践和加深对于：**数字逻辑基础**（了解和应用二进制数的基本操作）**Verilog编程技能；逻辑设计与实现；问题解决和调试技能等**几个关键概念的理解。

1. **必修或选修**

选修

1. **实验平台**

Vivado

1. **实验内容及步骤**

**实验内容：**

自行设计一个简单预算的模块，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按位异或之后的结果。

**实现过程**

**模块定义与输入输出声明：**

定义一个Verilog模块，命名为myadd。声明两个8位输入端口ina和inb。声明所需的输出端口，sumab（加法结果），sumflag（进位标志），leftshiftA（左移结果，位宽取决于左移位数的最大值），lessflag（小于标志），equalflag（等于标志），bitXorflag（异或结果，8位宽）等。

**加法运算实现：**

使用Verilog的加法操作符+来实现ina和inb的加法。将加法结果分配给sumab。对于进位标志sumflag，可以使用额外的位来捕获进位。

**逻辑移位实现：**

使用移位操作符<<来实现ina根据inb指定的位数向左移位。将结果分配给leftshiftA。

**比较运算实现：**

使用关系运算符<和==来比较ina和inb。根据比较结果，设置lessflag和equalflag。

**按位异或运算实现：**

使用异或运算符^来实现ina和inb的按位异或。将结果分配给bitXorflag。

module myadd (

input [7:0] ina,

input [7:0] inb,

output [7:0] sumab,

output sumflag,

output [7:0] leftshiftA,

output lessflag,

output equalflag,

output [7:0] bitXorflag

);

// 加法运算：将ina和inb相加，结果的最高位（进位）赋给sumflag，其余位赋给sumab。

assign {sumflag, sumab} = ina + inb;

// 逻辑移位运算：实现ina根据inb的值向左移位。

assign leftshiftA = ina << inb;

// 比较运算：使用三元运算符来设置lessflag，如果ina小于inb则为1，否则为0。

assign lessflag = (ina < inb) ? 1'b1 : 1'b0;

// 使用三元运算符来设置equalflag，如果ina等于inb则为1，否则为0。

assign equalflag = (ina == inb) ? 1'b1 : 1'b0;

// 按位异或运算：实现ina和inb的按位异或运算。

assign bitXorflag = ina ^ inb;

endmodule

除了主要模块的代码之外，还需要编写一个测试平台（testbench）来对该模块进行测试：

module tb();

// 输入和输出变量

reg [7:0] ina; // 声明两个寄存器型变量作为输入

reg [7:0] inb;

// 声明多个线网型变量作为输出

wire [7:0]sumab;

wire sumflag;

wire [7:0]leftshiftA;

wire lessflag;

wire equalflag;

wire[7:0]bitXorflag

// 模块的每个输入输出端口通过点语法连接到相应的信号

myadd utt(ina,inb,sumab,sumflag,leftshiftA,lessflag,equalflag,bitXorflag);

initial begin

// 初始化ina和inb为0

ina=8'b00000000;

inb=8'b00000000;

end

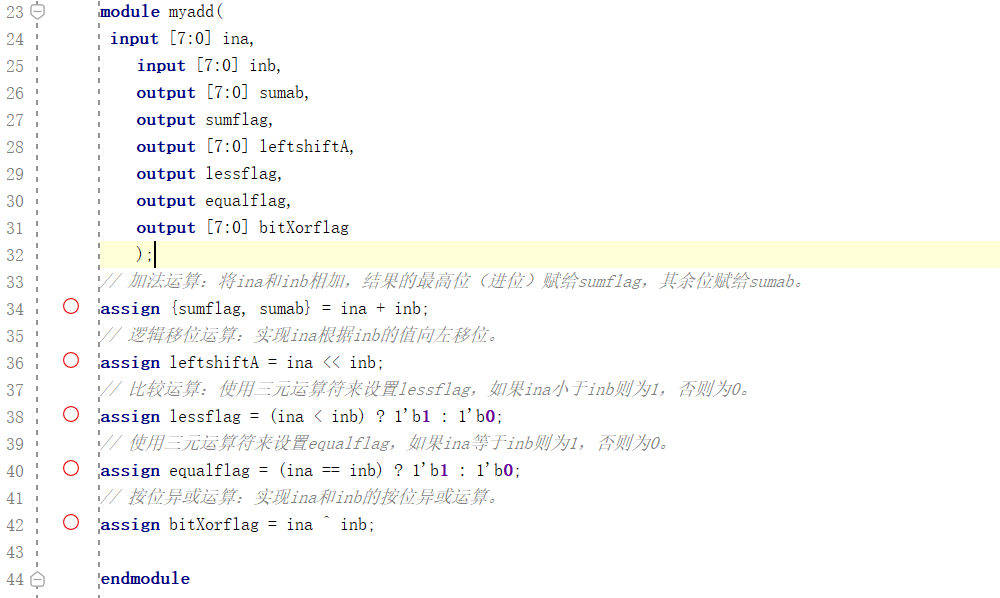
// always块定义了在仿真过程中重复执行的行为

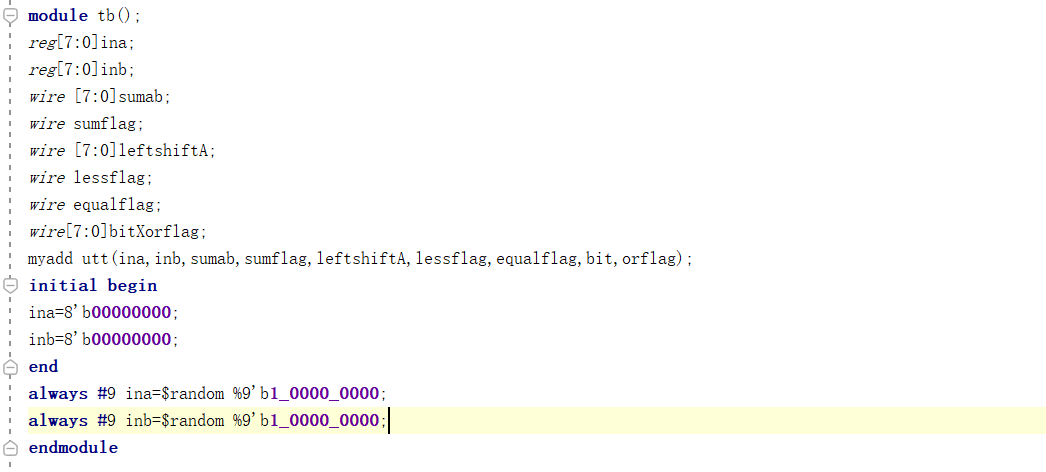
always #9 ina = $random % 256; // 每9个时间单位更新ina

always #9 inb = $random % 256; // 每9个时间单位更新inb

endmodule

1. **关键问题讨论**







如上波形图所示：

ina=01111001，inb=10111000，sumab=00110001，sumflag=1

leftshiftA=00000000，lessflag=1，equalflag=0,bitXorflag=00000001

1. **总结**

结合本次实验内容，不仅学习了如何使用Verilog编程语言来实现一个基于数字逻辑的预算模块，而且深入理解了数字逻辑设计的基本原理。通过设计包含加法、逻辑移位、比较和按位异或等操作的模块，实践如何将理论知识转化为实际的硬件设计。这个过程中不仅提高了编程技能，还增强了对数字电路设计流程、测试与调试策略的理解。此外，通过面对各种边界情况和异常情况的测试，进一步学会了如何分析和解决复杂的问题，这对于任何涉及硬件设计和计算机工程的领域都是至关重要的。总之，这次实验是在数字逻辑和硬件编程方面知识的一次全面应用。