Verilog第二次作业

2211290 姚知言 计算机学院

1.用自己的话总结always语句和assign语句的区别

always语句，可以理解为触发器的结构，一般用在时序逻辑中，包括一个触发条件列表，当触发条件中任意一个变量发生改变的时候更改，在always块中，begin和end包裹的内容需要按顺序执行，所以可以包括if-else，case语句等。

assign语句，类似于把一根导线连过去，等号左边会实时获得等号右边的结果，不具有触发器结构，一般用在组合逻辑中，同样，所有assign语句都是实时计算的，也就不能够有if-else等结构。

2.用自己的话总结reg类型变量和wire类型变量的区别

reg类型变量：触发器，具有数据存储的作用，不能够通过assign赋值，一般要通过块语句来赋值，在重新赋值之前一直保持原来的结果，在赋值后刷新结果。

wire类型变量：一种常用的nets型变量，输出随输入实时变化而变化，不能够存储数据，一般用来表示以assign语句赋值的组合逻辑信号。

3.完成第三页PPT中的真值表

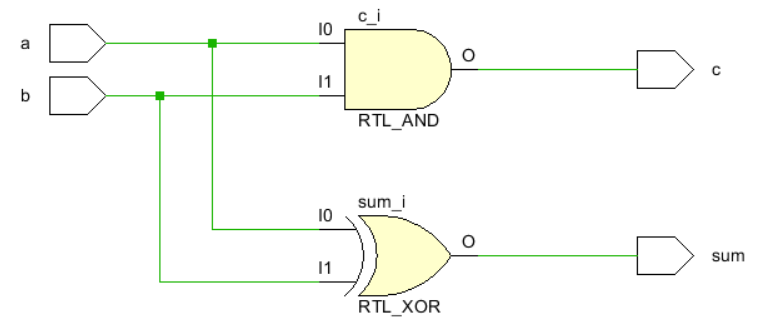
如图所示，仅当a==b，即a[1]==b[1]且a[0]==b[0]的时候，equal=1，其他时候equal=0。



4.请使用vivado依次编写半加器、全加器、8位加法器并验证正确性

a.半加器halfadder，设计及电路图如下：

module halfadder(

 input a,

input b,

output sum,

output c

);

assign sum = a ^ b ;

assign c = a & b;

endmodule

仿真文件testbench.v：

设计思路：当a和b一个0一个1时sum为1，所以用异或，a和b都为1的时候c为1，所以用与；

tempb = 1'b0;

end

always #5 tempa = ~tempa;

always #3 tempb = ~tempb;

endmodule

module testbench();

reg tempa,tempb;

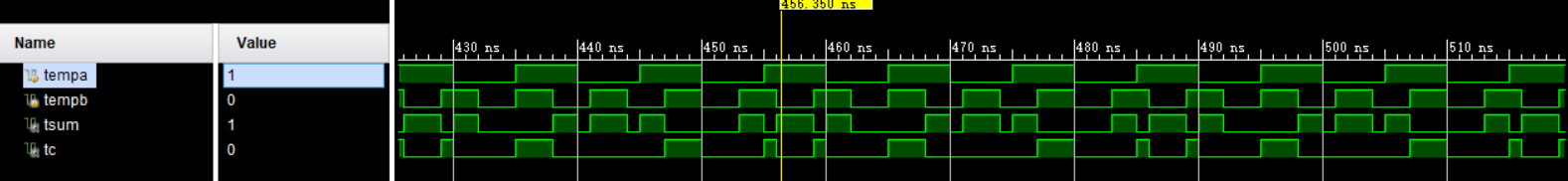
wire tsum,tc;

halfadder myha(tempa,tempb,tsum,tc);

initial begin

tempa = 1'b0;

仿真结果：如下图所示，可以看出tempa和tempb都为0的时候tsum和tc为0，有一个为1则tsum=1，tc=0，均为1则tc为1，tsum为0，验证成功。



b.全加器，调用半加器，fulladder.v和testbench.v设计如下：

module testbench();

reg tempa,tempb,tempci;

wire tsum,tco;

fulladder myfu(tempa,tempb,tempci,tsum,tco);

initial begin

tempa = 1'b0;

tempb = 1'b0;

tempci = 1'b0;

end

always #5 tempa = ~tempa;

always #3 tempb = ~tempb;

always #7 tempci = ~tempci;

endmodule

module fulladder(

input a,

input b,

input ci,

output sum,

output co

);

wire t1,t2,t3;

halfadder h1(a,b,t1,t2);

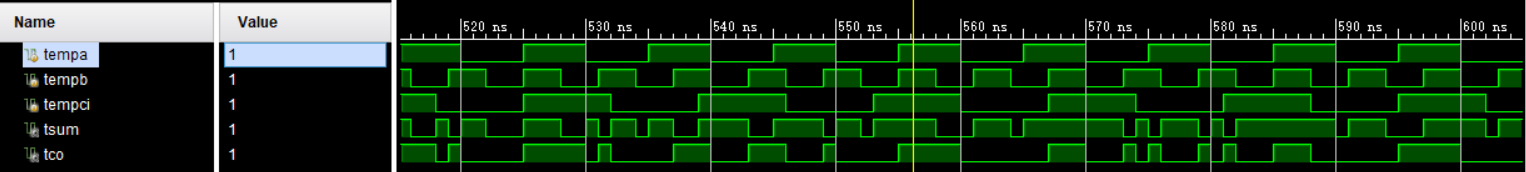
halfadder h2(t1,ci,sum,t3);

assign co=t2^t3;

endmodule

设计思路：调用两个半加器，两两相加。

仿真结果：如下图所示，可以看到，加法器满足当tempa，tempb，tempci均为0的时候，tsum和tco都为0；其中一个输入为1的时候，tsum为1，tco为0；其中两个输入为1的时候，tsum为0，tco为1；三个输入均为1的时候，tsum和tco均为1。



c.8位加法器，调用全加器，\_8bitadder.v和testbench.v设计如下：

module testbench3();

reg [7:0] tempa,tempb;

reg tempci;

wire [7:0]tsum;

wire tco;

\_8bitadder my8bit(tempa,tempb,tempci,tsum,tco);

initial begin

tempa = 8'b0;

tempb = 8'b0;

tempci = 1'b0;

end

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

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

always #5 tempci=~tempci;

endmodule

module \_8bitadder(

input [7:0] a,

input [7:0] b,

input ci,

output [7:0] sum,

output co

);

wire [6:0] temp;

fulladder f1(a[0],b[0],ci,sum[0],temp[0]);

fulladder f2(a[1],b[1],temp[0],sum[1],temp[1]);

fulladder f3(a[2],b[2],temp[1],sum[2],temp[2]);

fulladder f4(a[3],b[3],temp[2],sum[3],temp[3]);

fulladder f5(a[4],b[4],temp[3],sum[4],temp[4]);

fulladder f6(a[5],b[5],temp[4],sum[5],temp[5]);

fulladder f7(a[6],b[6],temp[5],sum[6],temp[6]);

fulladder f8(a[7],b[7],temp[6],sum[7],co);

endmodule

设计思路：调用8个全加器，按位计算，低位的co作为高一位的ci，最高位的co作为输出。

仿真思路：tempa和tempb使用random取余生成8位随机数，tempci则使用翻转的方式模拟。

仿真结果如图所示，取400-410ns的两例分析：

0xf7+0x69+0x0=0x160,tsum为60，tco为1；0xb4+0x88+0x1=0x13d，tsum为3d，tco为1，验证成功。

