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

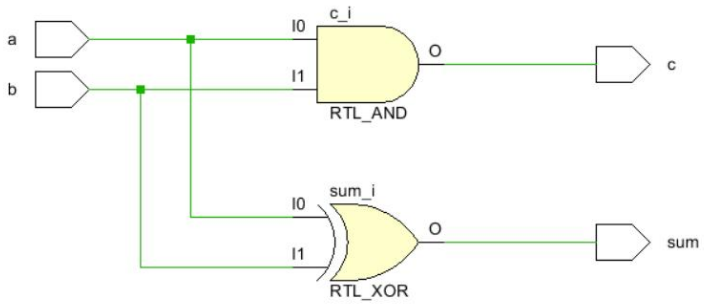
如图所示，仅当 a==b，即 a[1]==b[1]且 a[0]==b[0]的时候，equal=1，其他时候 equal=0。

a[0]	a[1]	b[0]	b[1]	equal	a[0]	a[1]	b[0]	b[1]	equal
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	1
0	0	1	1	1	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	1	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

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
```

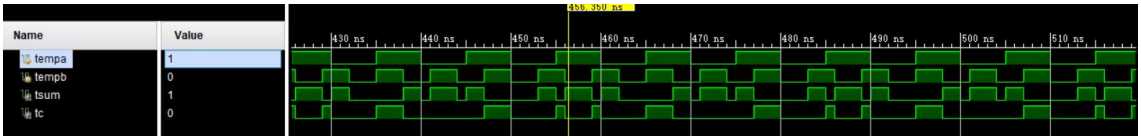


仿真文件 testbench.v：

```
module testbench();
reg tempa,tempb;
wire tsum,tc;
halfadder myha(tempa,tempb,tsum,tc);
initial begin
    tempa = 1'b0;
    tempb = 1'b0;
end
always #5 tempa = ~tempa;
always #3 tempb = ~tempb;
endmodule
```

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；

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。

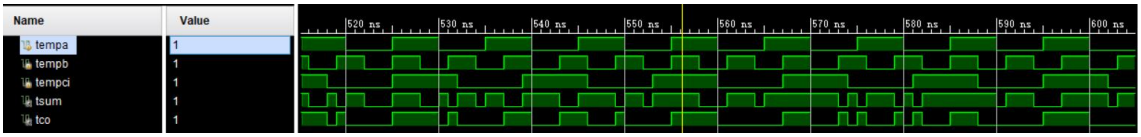


b.全加器，调用半加器，fulladder.v 和 testbench.v 设计如下：

```
module 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
```

```
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
```

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 _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
```

```
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
```

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。