**北京科技大学 计算机与通信工程学院**

**实 验 报 告**

实验名称： 数 字 逻 辑 实 验 二

学生姓名： 唐 誉 源

专 业： 计 算 机 科 学 与 工 程

班 级： 计 1503

学 号： 41503302

指导教师： 何 杰

实验成绩：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验地点： 机 电 楼 304

实验时间： 2017年3月8日

**一、实验目的与实验要求**

**1、实验目的**

. (1)  学习用verilog设计较复杂的组合逻辑电路

. (2)  进一步掌握vivado的仿真

**2、实验要求**

1. 在实验报告中提交Verilog代码、仿真代码、RTL级的实现、实验2.5的顶层 模块设计图(描述 8 位行逐位进位加法器、8 位超前进位加法器和 8 位选择 进位加法器分别由那些模块构成，模块的输入输出、连接关系)、RTL 级的 详细设计图、仿真结果图和表 1;
2. 提交实验报告和每个实验的完整工程文件。

**二、实验设备（环境）及要求**

实验室提供Windows 7操作系统下的Xilinx开发平台vivado2015.4环境，以及EGO开发板。

**三、实验内容与步骤**

**1、实验2.1**

（1）实验内容

. 实验 **2.1**——**4** 位乘法器:按照“4 实验步骤——4 位乘法器”完成 4 位乘法器的设计与仿真验证;

（2）主要步骤

* 创建项目
* 编写电路模块代码：

module mult4(

input [3:0]a,

input [3:0]b,

output reg[7:0] p

);

reg[7:0]pv;

reg[7:0]ap;

integer i;

always @(\*)

begin

pv=8'b00000000;

ap={4'b0000,a};

for(i=0;i<=3;i=i+1)

begin

if(b[i]==1)

pv=pv+ap;

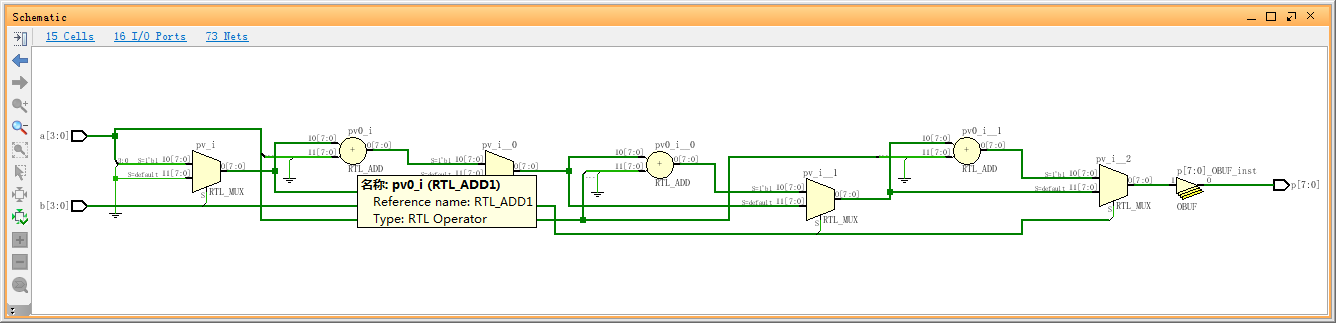
ap={ap[6:0],1'b0};

end

p=pv;

end

endmodule



RTL级设计图

* 生成RTL级设计图
* 编写测试模块代码：

module mult4\_tb();

reg[3:0] a;

reg[3:0] b;

wire[7:0]p;

initial begin

a=0;b=0;

repeat(10)

begin

#10 a={$random}%16;

b={$random}%16;

end

#10 $stop;

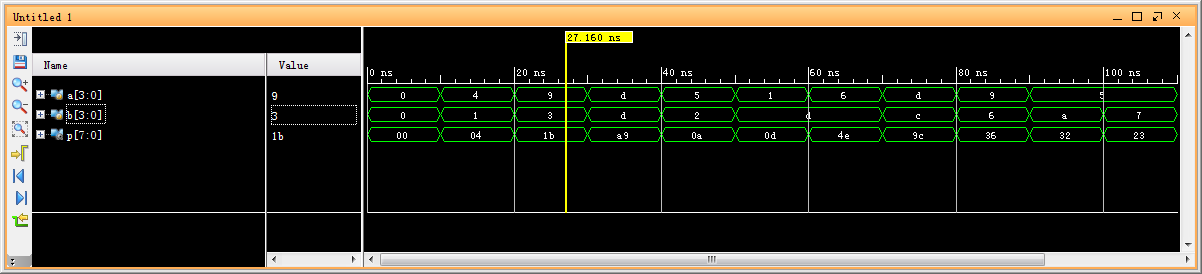
end

mult4 U1(.a(a),.b(b),.p(p));

endmodule

* 运行仿真

（3）实验结果与分析



仿真结果示意图

由图放大可看出，同一时刻的p都等于a与b相乘的结果。例如黄线所在的位置，a=9，b=3，c=1b（16）=27（10）符合乘法运算规律。且a，b均取最大值15时，相乘结果小于p的最大值256，所以均能正常运算并显示结果。

**2、实验2.2**

（1）实验内容

**8**位乘法器:新建工程，完成8位乘法器的设计与仿真验证。

（2）主要步骤

* 创建项目
* 编写电路模块代码：

module Mult8(

input [7:0]a,

input [7:0]b,

output reg[15:0] p

);

reg[15:0]pv;

reg[15:0]ap;

integer i;

always @(\*)

begin

pv=16'b0000000000000000;

ap={8'b00000000,a};

for(i=0;i<=7;i=i+1)

begin

if(b[i]==1)

pv=pv+ap;

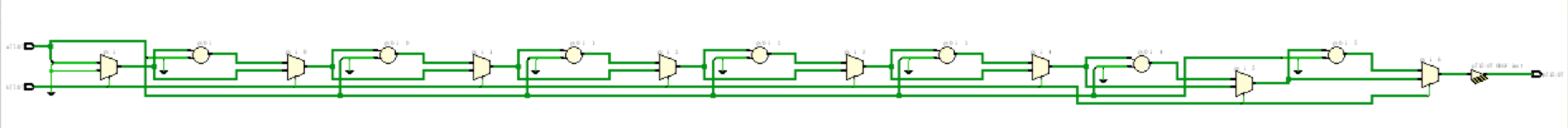
ap={ap[14:0],1'b0};

end

p=pv;

end

endmodule



RTL级设计图

* 生成RTL级设计图
* 编写测试模块代码：

module Mult8\_tb();

reg[7:0] a;

reg[7:0] b;

wire[15:0]p;

initial begin

a=0;b=0;

repeat(10)

begin

#10 a={$random}%256;

b={$random}%256;

end

#10 $stop;

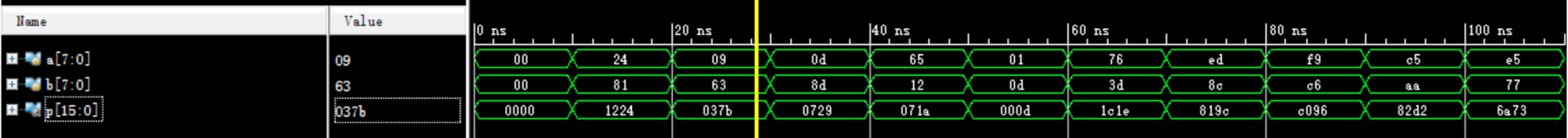
end

Mult8 U1(.a(a),.b(b),.p(p));

endmodule

* 运行仿真

（3）实验结果与分析

仿真结果图

由图放大可看出，同一时刻的p都等于a与b相乘的结果。例如黄线所在的位置，a=9（16），b=63（16）=99（10)，c=037b（16）=891（10）符合乘法运算规律。且a，b均取最大值255时，相乘结果小于p的最大值，所以均能正常运算并显示结果。

**3、实验2.3**

（1）实验内容

. **4** 位除法器:按照“5 实验步骤(4 位除法器)”完成 4 位除法器的设计与仿真验证;

（2）主要步骤

* 创建项目
* 编写电路模块代码：

module div84(

input [7:0] numerator,

input [3:0] denominator,

output [7:0] quotient,

output [3:0] remainder

);

wire [7:0] numerator;

wire [3:0] denominator;

reg [7:0] quotient;

reg [3:0] remainder;

reg [3:0] remH;

reg [3:0] remL;

reg [3:0] quotH;

reg [3:0] quotL;

always@(\*)

begin

div4({1'b0,numerator[7:4]},denominator,quotH,remH);

div4({remH,numerator[3:0]},denominator,quotL,remL);

quotient[7:4]=quotH;

quotient[3:0]=quotL;

remainder=remL;

end

task div4(

input[7:0] numer,

input[3:0] denom,

output[3:0] quot,

output[3:0] rem);

begin :D4

reg[4:0]d;

reg[4:0]n1;

reg[3:0]n2;

d={1'b0,denom};

n2=numer[3:0];

n1={1'b0,numer[7:4]};

repeat(4)

begin

n1={n1[3:0],n2[3]};

n2={n2[2:0],1'b0};

if(n1>=d)

begin

n1=n1-d;

n2[0]=1;

end

end

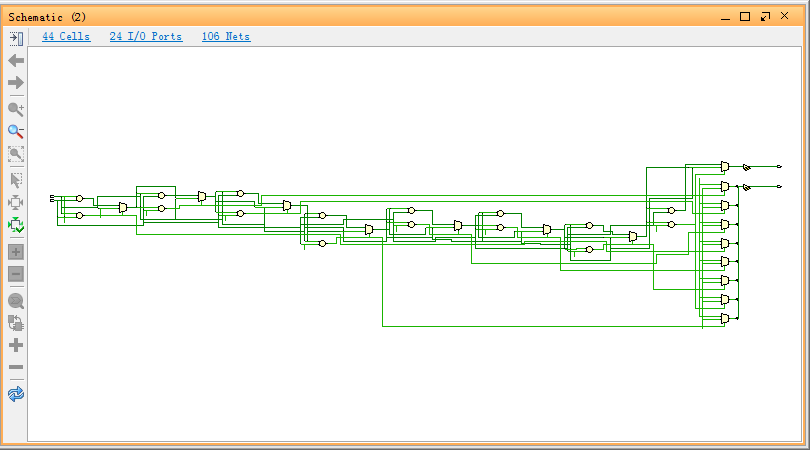
quot=n2;

rem=n1[3:0];

end

endtask

endmodule



RTL级设计图

* 生成RTL级设计图
* 编写测试模块代码：

module div84\_tb;

reg [7:0] numerator;

reg [3:0] denominator;

wire [7:0] quotient;

wire [3:0] remainder;

initial

begin

numerator=0;

denominator=1;

repeat(10)

begin

#10 numerator=($random)%16+7;

denominator=($random)%16+1;

end

#10 $stop;

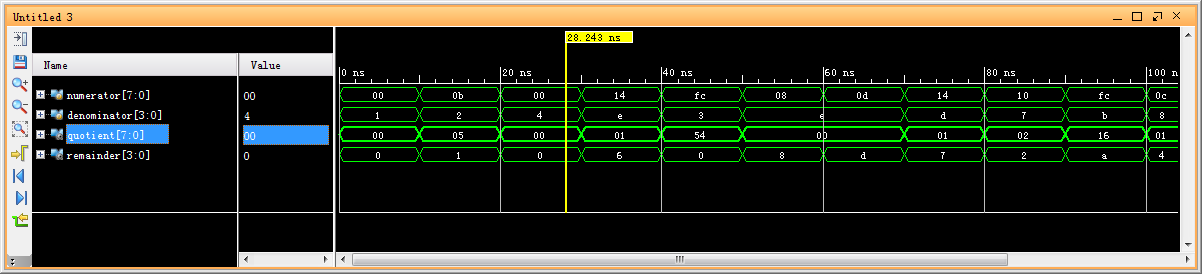
end

div84 U1(.numerator(numerator,.denominator (denominator),.quotient (quotient),.remainder(remainder));

endmodule

* 运行仿真

（3）实验结果与分析

仿真结果图

由图放大可看出，同一时刻的quotient都为被除数除以除数的商，remainder为余数。例如黄线所在的位置的下一时刻，numerator=14（16）=20（10)，dominator=e（16）=14（10)，quotient=1,remainder=6符合除法运算规律。电路模块及仿真模块代码实现正确。

**4、实验2.4**

（1）实验内容

. 实验**2.4**——**8**位除法器:新建工程，完成8位除法器的设计与仿真验证。

（2）主要步骤

* 创建项目
* 编写电路模块代码：

module div84\_tb;

reg [7:0] numerator;

reg [3:0] denominator;

wire [7:0] quotient;

wire [3:0] remainder;

initial

begin

numerator=0;

denominator=1;

repeat(10)

begin

#10 numerator=($random)%16+7;

denominator=($random)%16+1;

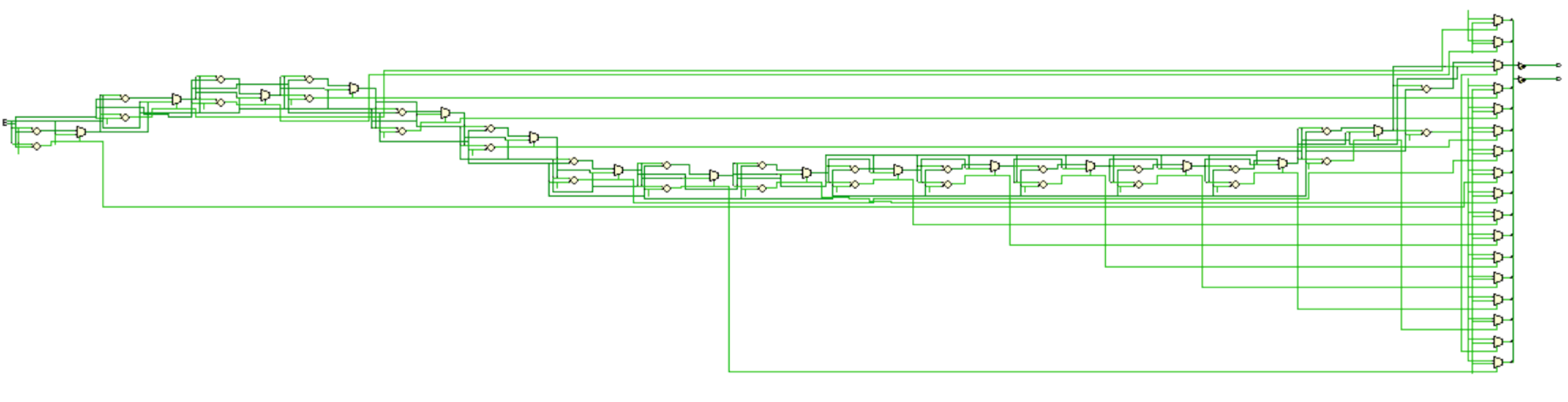
end

#10 $stop;

end

div84 U1(.numerator(numerator,.denominator (denominator),.quotient (quotient),.remainder(remainder));

endmodule



RTL级设计图

* 生成RTL级设计图
* 编写测试模块代码：

module div84\_tb;

reg [7:0] numerator;

reg [3:0] denominator;

wire [7:0] quotient;

wire [3:0] remainder;

initial

begin

numerator=0;

denominator=1;

repeat(10)

begin

#10 numerator=($random)%16+7;

denominator=($random)%16+1;

end

#10 $stop;

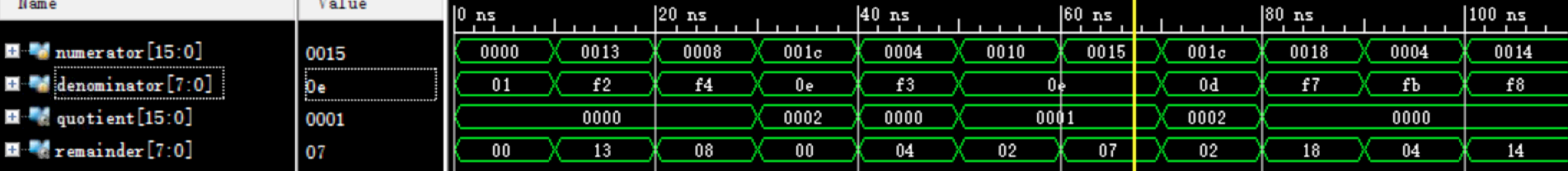
end

div84 U1(.numerator(numerator),.denominator (denominator),.quotient (quotient),.remainder(remainder));

endmodule

* 运行仿真

（3）实验结果与分析

仿真示意图

由仿真示意图可以看出，同一时刻的quotient（商）是numerator除以denominator的结果，remainder是其余数。例如黄线所在状态，numerator=0015（16）=21（10），denominator=0e（16）=14（10），quotient=1，remainder=7，符合数值除法计算规则。所以编写的电路模块和测试模块能实现要求的基本功能。

**5、实验2.5**

（1）实验内容

实验**2.5**——**8**位行逐位进位加法器、**8**位超前进位加法器(由两个**4**位超前进位加法器构成)、**8** 位选择进位加法器的性能比较。

a) 学习“组合电路模块.pptx”中逐位进位加法器、超前进位加法器(由两个 4 位超前进位加法器构成)、选择进位加法器的原理。

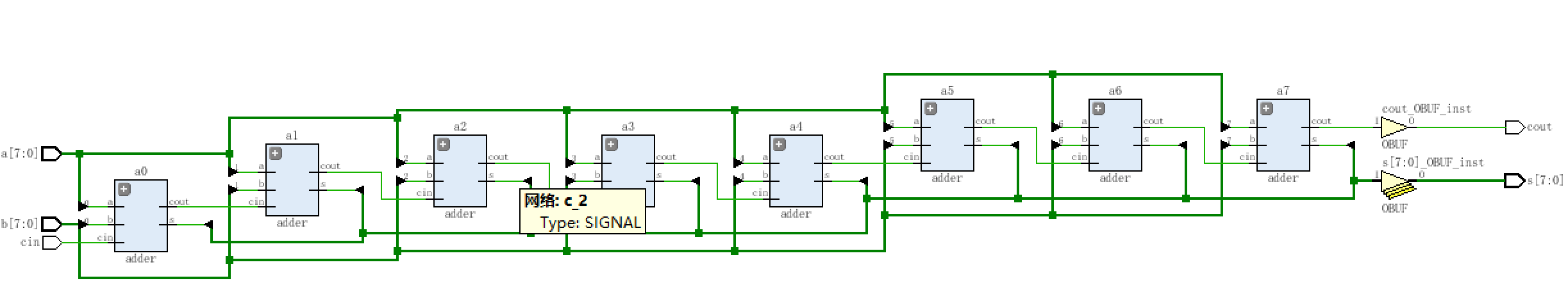
b)  新建工程完成8位行逐位进位加法器、8位超前进位加法器(由两个4位 超前进位加法器构成)、8 位选择进位加法器的设计与功能仿真验证。

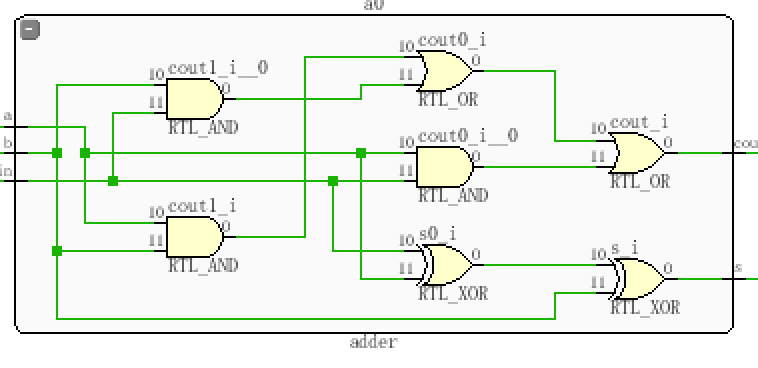
c)  设计仿真程序，通过输入 10 组数的加法运算(测试数据的设计要考虑体现加法器的性能)，比较三种加法器的性能。

（2）主要步骤

逐位进位加法器

逐位进位加法器是有八个全加器串行连接而组成的八位进位加法器。

系统设计图



单个全加器模块示意图

module RippleCarry(

input [7:0]a,b,

input cin,

output cout,

output [7:0]s

);

wire [7:1]c;

adder a0(a[0],b[0],cin,c[1],s[0]);

adder a1(a[1],b[1],c[1],c[2],s[1]);

adder a2(a[2],b[2],c[2],c[3],s[2]);

adder a3(a[3],b[3],c[3],c[4],s[3]);

adder a4(a[4],b[4],c[4],c[5],s[4]);

adder a5(a[5],b[5],c[5],c[6],s[5]);

adder a6(a[6],b[6],c[6],c[7],s[6]);

adder a7(a[7],b[7],c[7],cout,s[7]);

endmodule

module adder(

input a,b,cin,

output cout,s

);

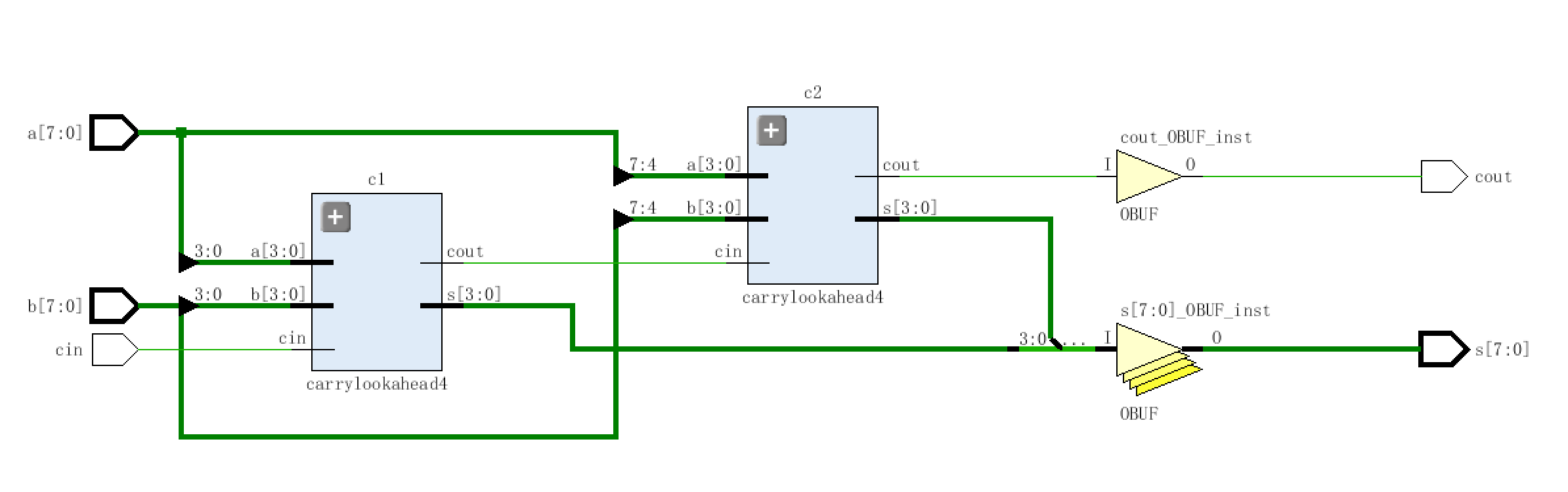
assign s=cin^a^b;

assign cout=a&b|b&cin|a&cin;

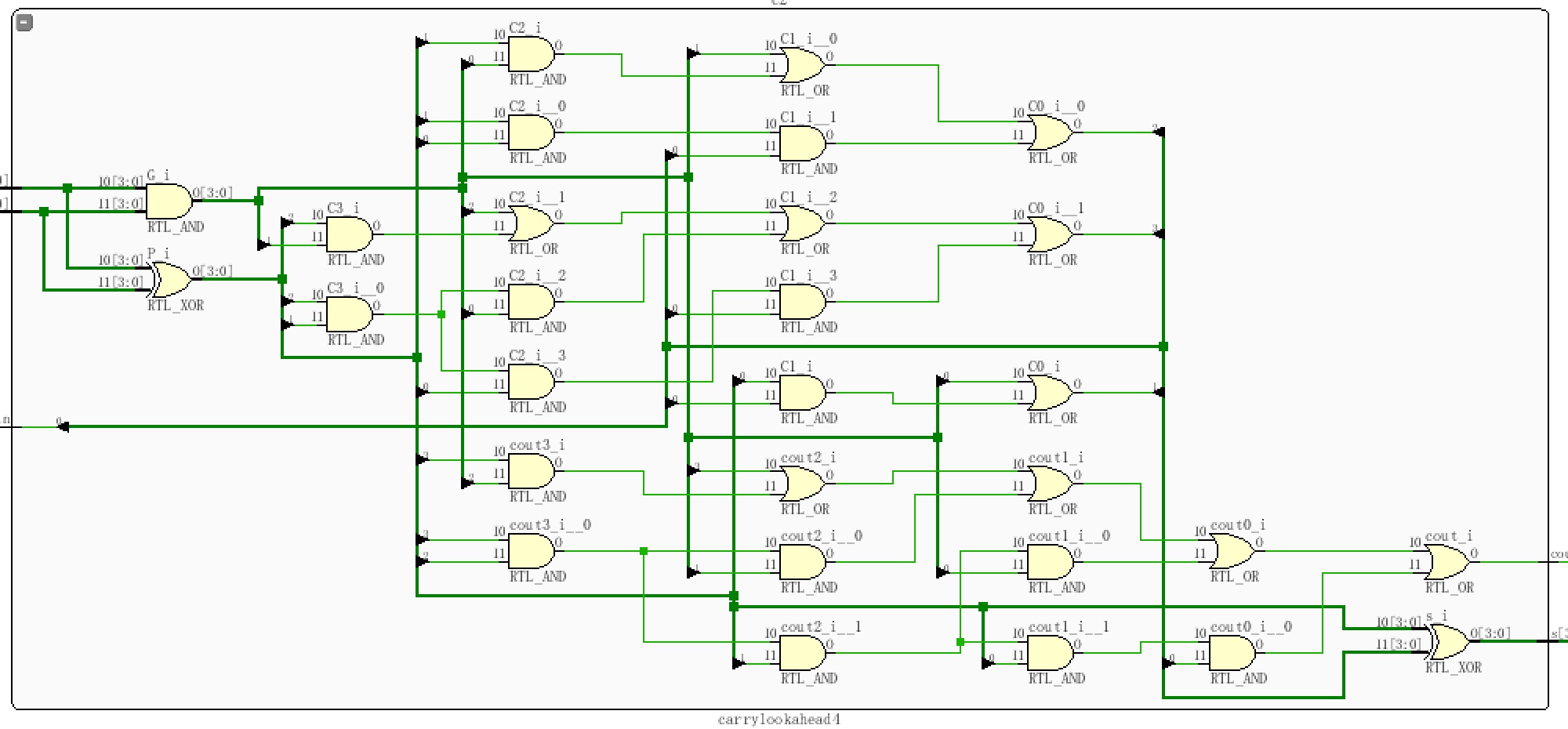
endmodule

超前进位加法器

八位超前进位加法器是由两个四位超前进位加法器构成的。超前进位加法器一般最多四位，过多位数会造成电路过于复杂，所占面积大。



系统级RTL图



单个4位超前进位加法器示意图

module carrylookahead4(

input [3:0]a,b,

input cin,

output cout,

output [3:0]s

);

wire [3:0]G,P,C;

assign G = a&b;

assign P = a^b;

assign C[0]=cin;

assign C[1] = G[0]|P[0]&C[0];

assign C[2] = G[1]|P[1]&G[0]|P[1]&P[0]&C[0];

assign C[3] = G[2]|P[2]&G[1]|P[2]&P[1]&G[0]|P[2]&P[1]&P[0]&C[0];

assign cout = G[3]|P[3]&G[2]|P[3]&P[2]&G[1]|P[3]&P[2]&P[1]&G[0]|P[3]&P[2]&P[1]&P[0]&C[0];

assign s = P^C;

endmodule

module carrylookahead8(

input [7:0]a,b,

input cin,

output cout,

output [7:0]s

);

wire c;

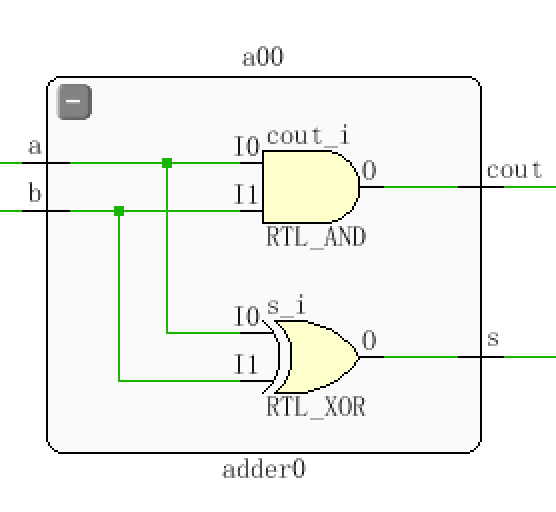
carrylookahead4 c1(a[3:0],b[3:0],cin,c,s[3:0]);

carrylookahead4 c2(a[7:4],b[7:4],c,cout,s[7:4]);

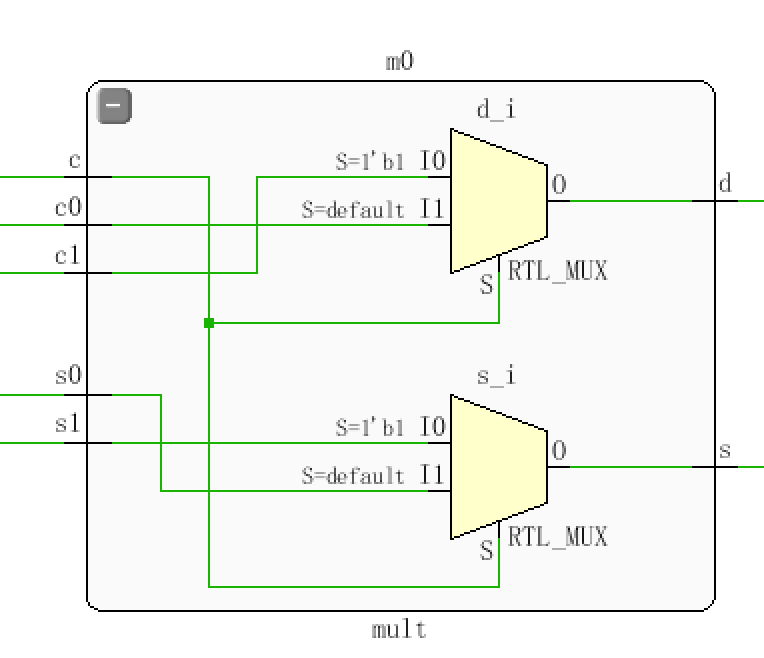
endmodule

选择进位加法器

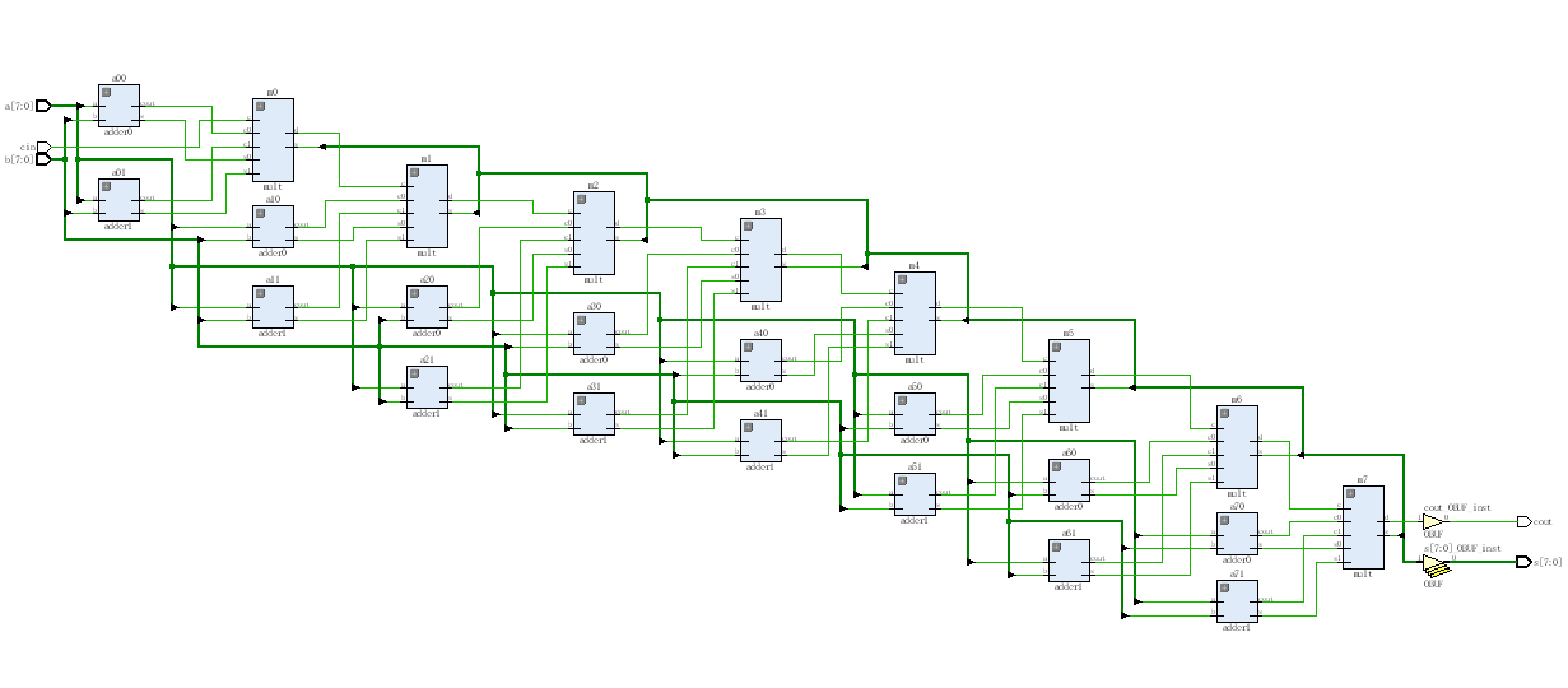
每一级都有两个加法器，内置进位分别是0和1，还有一个选择器，当上一级进位传给本级的选择器时，输出相应进位输出。



adder0模块示意图



选择模块



系统级RTL示意图

module carryselect(

input [7:0]a,b,

input cin,

output cout,

output [7:0]s

);

wire [7:0]c1;

wire [7:0]c0;

wire [7:1]c;

wire [7:0]s1,s0;

adder1 a01(a[0],b[0],c1[0],s1[0]);

adder0 a00(a[0],b[0],c0[0],s0[0]);

adder1 a11(a[1],b[1],c1[1],s1[1]);

adder0 a10(a[1],b[1],c0[1],s0[1]);

adder1 a21(a[2],b[2],c1[2],s1[2]);

adder0 a20(a[2],b[2],c0[2],s0[2]);

adder1 a31(a[3],b[3],c1[3],s1[3]);

adder0 a30(a[3],b[3],c0[3],s0[3]);

adder1 a41(a[4],b[4],c1[4],s1[4]);

adder0 a40(a[4],b[4],c0[4],s0[4]);

adder1 a51(a[5],b[5],c1[5],s1[5]);

adder0 a50(a[5],b[5],c0[5],s0[5]);

adder1 a61(a[6],b[6],c1[6],s1[6]);

adder0 a60(a[6],b[6],c0[6],s0[6]);

adder1 a71(a[7],b[7],c1[7],s1[7]);

adder0 a70(a[7],b[7],c0[7],s0[7]);

mult m0(c1[0],c0[0],s1[0],s0[0],cin,c[1],s[0]);

mult m1(c1[1],c0[1],s1[1],s0[1],c[1],c[2],s[1]);

mult m2(c1[2],c0[2],s1[2],s0[2],c[2],c[3],s[2]);

mult m3(c1[3],c0[3],s1[3],s0[3],c[3],c[4],s[3]);

mult m4(c1[4],c0[4],s1[4],s0[4],c[4],c[5],s[4]);

mult m5(c1[5],c0[5],s1[5],s0[5],c[5],c[6],s[5]);

mult m6(c1[6],c0[6],s1[6],s0[6],c[6],c[7],s[6]);

mult m7(c1[7],c0[7],s1[7],s0[7],c[7],cout,s[7]);

endmodule

module mult(

input c1,c0,s1,s0,c,

output d,s

);

assign d = c?c1:c0;

assign s = c?s1:s0;

endmodule

module adder1(

input a,b,

output cout,s

);

assign s=1^a^b;

assign cout=a&b|b|a;

endmodule

module adder0(

input a,b,

output cout,s

);

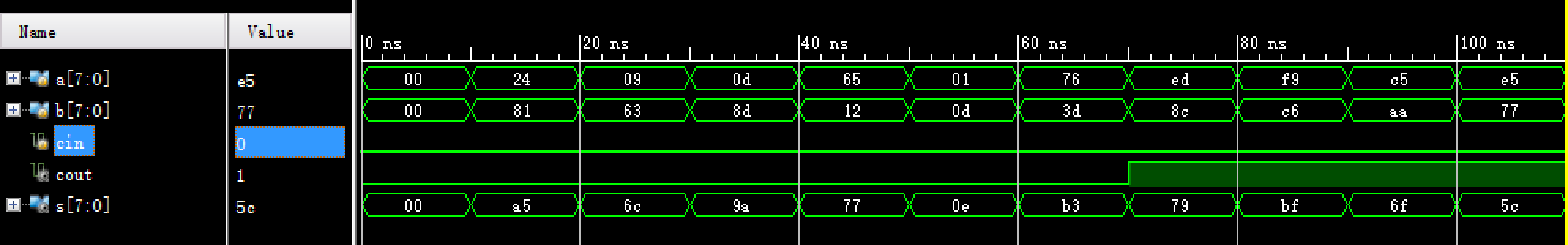
assign s=0^a^b;

assign cout=a&b;

endmodule

（3）实验结果与分析

所有加法器共用同一个测试文件的代码，所得结果如下图所示。

仿真模拟结果图

由仿真结果图可以看出，加法器工作正常能满足八位进位加法器的要求。例如25ns时，a=9,b=63(16)=99(10)，所得结果为{cout,s}=06c(16)=108(10);再例如85ns时，a=f9(16),b=c6(16)，所得结果为{cout,s}=1bf(16)。

测试文件代码：

module Carry\_tb;

reg [7:0]a,b;

reg cin;

wire cout;

wire [7:0]s;

initial begin

cin=0;

a=0;b=0;

repeat(10)

begin

#10 a=($random)%256;

b=($random)%256;

end

#10 $stop;

end

RippleCarry u0(a,b,cin,cout,s);

endmodule

**四：结论（讨论）**

**1、实验结论**

学习了用verilog设计较复杂的组合逻辑电路，并进一步掌握vivado的仿真。此外还掌握了4位乘法器和4位除法器的verilog实现代码和二进制算法，以及如何连接形成八位乘／除法器。更加详细了解了3种不同的加法器的工作原理，各个不同的模块如何连接，并通过实现verilog熟悉了verilog语言和仿真。

**2、讨论**

经过对3种不同的加法器的verilog实现，我对所需时间和所需面积之间的有了更切身的体会。为得到更短的时间，势必要以牺牲面积为代价，例如超前进位加法器和选择进位加法器比逐位进位加法器所需时间要短很多，性能好很多，但是电路复杂程度，电路所覆面积也增加很多。所以在实际设计电路中，要权衡性能和电路面积（集成电路大小和价格）才能设计出更加适合实际应用的电路。

**六、教师评审**

|  |  |
| --- | --- |
| **教师评语** | **实验成绩** |
| 签名：  日期： |  |