北京科技大学 计算机与通信工程学院

实 验 报 告

实验名称： 实验二 组合逻辑实验

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

（1） Xilinx Ego1 实验平台。

（2） OS：Win7 64 位

（3） Software：Vivado15.4 开发工具

三、实验内容与步骤

1、实验 1

（1）实验内容

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

a) 实验步骤详见实验文件

b) 模块代码如下：

```
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
```

c) RTL 门级结构图：

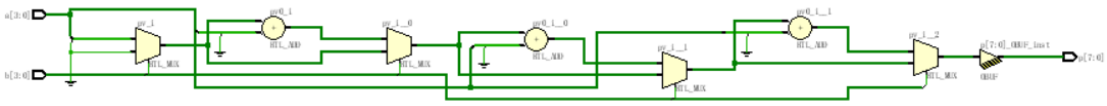

图 1:　4 位乘法器 RTL 图

d) 测试激励代码：

```
module mult4_tb;
    reg[3:0] a;
    reg[3:0] b;
    wire[7:0] p;

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

    initial
        begin
            a=0;
            b=0;
        repeat(8)
```

```
        begin
                #10 a={$random}%5;
                    b={$random}%5;
                #10 a={$random}%10;
                    b={$random}%10;
                #10 a={$random}%15;
                    b={$random}%15;

        end
        end

endmodule
```

e) 进行行为仿真，得到波形图

2、实验 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

a) 新建工程，完成 8 位乘法器的设计与仿真验证

b) 模块代码如下：

```
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'b00000000_00000000;
            ap={8'b0000_0000,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
```

c) RTL 门级结构图

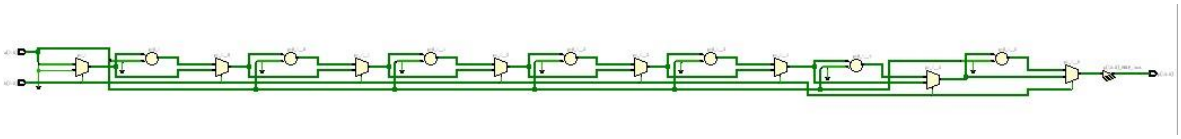

图 2： 4 位乘法器 RTL 结构图

d) 测试激励代码：

```
module mult4_tb;
  reg[3:0] a;
  reg[3:0] b;
  wire[7:0] p;

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

  initial
    begin
      a=0;
      b=0;
    repeat(8)
      begin
            #10 a={$random}%5;
                b={$random}%5;
            #10 a={$random}%10;
                b={$random}%10;
            #10 a={$random}%15;
                b={$random}%15;

      end
      end

endmodule
```

e) 进行行为仿真，得到并验证波形图。

f)

3、实验 3

（1）实验内容

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

a) 新建工程，div4 即为 4 位除法器

b) 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
```

c) RTL 门级结构图

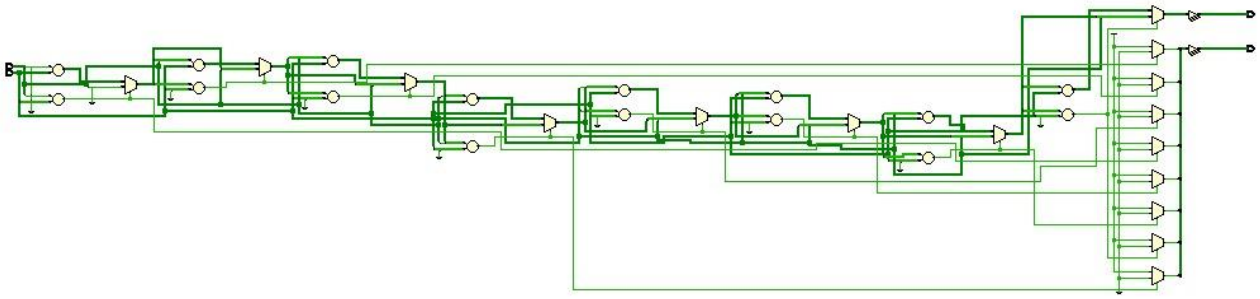

图 3： 4 位除法器 RTL 门级结构图

d) 测试激励代码：

```
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={$random}%10+7;
                denominator={$random}%7;
                end
                #10
                 numerator={$random}%16+6;
                 denominator={$random}%6;
                end
                div84 U1(. numerator( numerator),.denominator(denominator),.
quotient( quotient),.remainder(remainder));



endmodule
```

e) 进行行为仿真，得到并验证波形图。

4、实验 4

（1）实验内容

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

a) 新建工程，div8 即为 8 位除法器

b) 模块代码：

```
`timescale 1ns / 1ps

module div16_8(
    input [15:0] numerator,
    input [7:0] denominator,
    output [15:0] quotient,
    output [7:0] remainder
    );
    wire[15:0] numerator;
        wire[7:0] denominator;
        reg[15:0]    quotient;
        reg [7:0]    remainder;
        reg[7:0] remH;
         reg[7:0] remL;
           reg[7:0] quotH;
            reg[7:0] quotL;
            always@(*)
                begin
```

```
div16_8({1'b0,numerator[15:8]},denominator,quotH,remH);
div16_8({remH,numerator[7:0]},denominator,quotL,remL);
quotient[15:8]=quotH;
quotient[7:0]=quotL;
remainder=remL;
end
task div16_8(
input[15:0] numer,
input[7:0] denom,
output[7:0] quot,
output[7:0] rem);
begin : D8
reg[8:0] d;
reg[8:0] n1;
reg[7:0] n2;
d={1'b0,denom};
n2=numer[7:0];
n1={1'b0,numer[15:8]};
repeat(8)
begin
  n1={n1[7:0],n2[7]};
  n2={n2[6:0],1'b0};
  if(n1>=d)
  begin
  n1=n1-d;
  n2[0]=1;
  end
  end
  quot=n2;
  rem=n1[7:0];
  end
  endtask
endmodule
```

c) RTL 门级结构图：

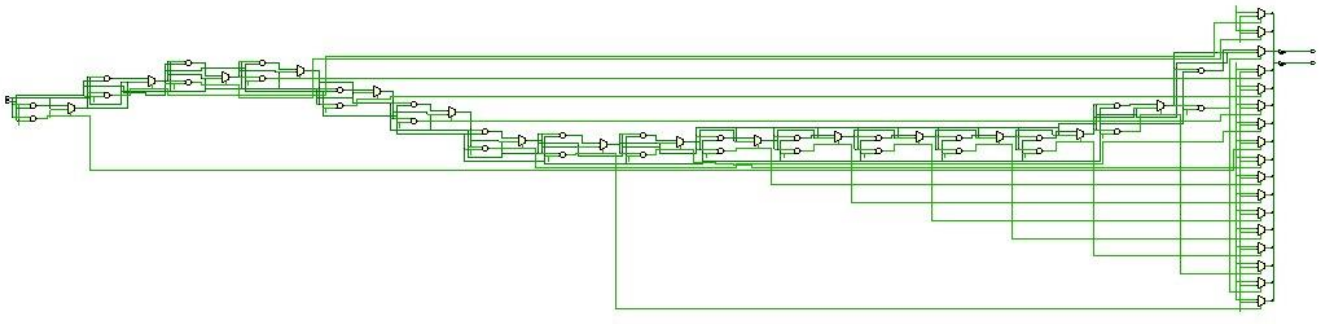

图 4： 16 位除法器 RTL 门级结构

d) 测试激励代码：

```
`timescale 1ns / 1ps

module div16_8_tb(

    );
  reg [15:0] numerator;
        reg [7:0] denominator;
        wire [15:0]    quotient;
        wire [7:0]    remainder;
        initial
          begin
          numerator=0;
          denominator=1;                repeat(10)
          begin
            #10
            numerator={$random}%16+15;
            denominator={$random}%15;
            #10
            numerator={$random}%10+15;
            denominator={$random}%7;
            #10
            numerator={$random}%5+15;
            denominator={$random}%3;

            end

            end
            div16_8                                                    U1(.
numerator(                          numerator),.denominator(denominator),.
quotient( quotient),.remainder(remainder));

endmodule
```

e) 进行行为仿真，得到并验证波形图。

5、实验 5

（1）实验内容

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。

（2）主要步骤

a) **8 位逐位进位加法器**

i. 新建工程，ripple 即为行波进位加法器

ii. 行波进位加法器模块代码：

```
module ripple(ain,bin,cin,sum,cout);
   input [7:0] ain,bin;
   input cin;
   output [7:0] sum;
   wire [6:0] cinreg;
   output cout;
      FullAdder   uo(ain[0], bin[0], cin, sum[0], cinreg[0]);
      FullAdder   u1(ain[1], bin[1], cinreg[0], sum[1], cinreg[1]);
      FullAdder   u2(ain[2], bin[2], cinreg[1], sum[2], cinreg[2]);
      FullAdder   u3(ain[3], bin[3], cinreg[2], sum[3], cinreg[3]);
      FullAdder   u4(ain[4], bin[4], cinreg[3], sum[4], cinreg[4]);
      FullAdder   u5(ain[5], bin[5], cinreg[4], sum[5], cinreg[5]);
      FullAdder   u6(ain[6], bin[6], cinreg[5], sum[6], cinreg[6]);
      FullAdder   u7(ain[7], bin[7], cinreg[6], sum[7], cout);
  endmodule
```

iii. 测试激励代码

```
module test;
reg [7:0] a;
reg [7:0] b;
reg cin;
wire [7:0] sum;
```

```
    wire cout;
    ripple U1(a,b,cin,sum,cout);
    initial
      begin
        a=0;b=1;cin=1;
      end
    always
      begin
        #20 a=2;b=3;
        #20 a=20;b=44;
        #20 a=25;b=55;
        #20 a=30;b=66;
        #20 a=35;b=77;
      end
    always #22 cin=~cin;
  endmodule
```

iv. 行波进位加法器 RTL 结构图：

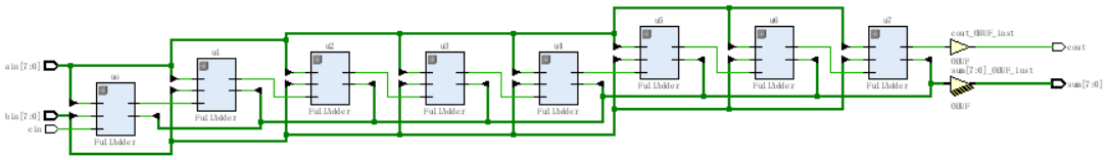

图 5： 行波进位加法器 RTL 结构图

v. 进行仿真激励观察，得到图像验证结果，图像见下。

b) **8 位超前进位加法器**

i. 新建工程，add_mux 即为选择进位加法器。

ii. 选择加法器模块代码：

```
    `timescale 1ns / 1ps
    module add_mux( a,b,sum,ovf);
    input [7:0]a;
```

```
input [7:0]b;
output [7:0]sum;
output    ovf;
wire [7:0]jin;
wire [7:0]temp1;
add_mux_2 U1(a[1:0],b[1:0],0,temp1[1:0],jin[0]);
add_mux_2 U2(a[3:2],b[3:2],jin[0],temp1[3:2],jin[1]);
add_mux_2 U3(a[5:4],b[5:4],jin[1],temp1[5:4],jin[2]);
add_mux_2 U4(a[7:6],b[7:6],jin[2],temp1[7:6],jin[3]);
assign {ovf,sum}={jin[3],temp1};

endmodule

module    add_mux_2(A,B,cin,s,cout);
input [1:0]A;
input [1:0]B;
input cin;
output reg [1:0]s;
output reg cout;
reg sel;
reg [3:0]temp;
always @(*)
          begin
     fork
       s[0]=A[0]^B[0]^cin;
       sel=(A[0]&B[0])|(A[0]^B[0])&cin;

         temp[1:0]=A[1]+B[1];
         temp[3:2]=A[1]+B[1]+1;
     join
```

```
        if(sel)
            {cout,s[1]}=temp[3:2];
            else
            {cout,s[1]}=temp[1:0];
        end
    endmodule
```

iii. 8 位选择进位加法器 RTL 结构图：

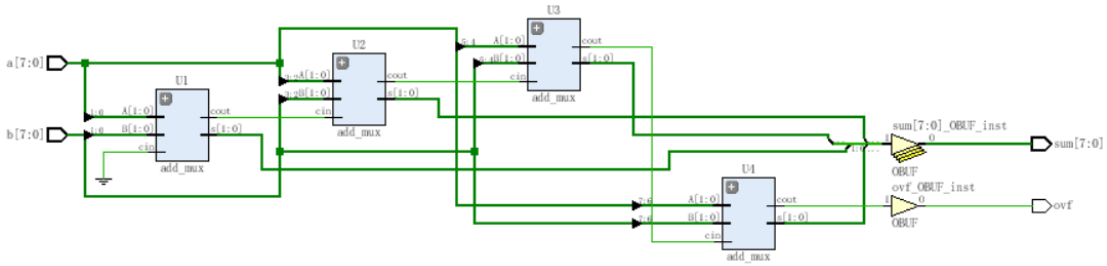

图 6： 进位选择加法器 RTL 结构图

iv. 测试激励模块代码：

```
module test;
    reg [7:0] a;
    reg [7:0] b;
    wire [7:0] sum;
    wire cout;
    add_mux4 U1(a,b,sum,cout);
    initial
      begin
        a=0;b=1;
      end
    always
      begin
        #20 a=2;b=3;
```

```
        #20 a=20;b=44;
        #20 a=25;b=55;
        #20 a=30;b=66;
        #20 a=35;b=77;
    end
Endmodule
```

v. 仿真测试激励图像，见实验结果与分析。

c) 8 位超前进位加法器

i. 新建工程，即 max_speed 为 8 位超前进位加法器

ii. 8 位超前进位加法器模块代码：

```
module max_speed(a,b,cin,sum,cout);
    input [7:0]a,b;
    input cin;
    output [7:0]sum;
    output cout;
    wire cout1;
        add u0(a[3:0],b[3:0],cin,sum[3:0],cout1);
        add u1(a[7:4],b[7:4],cout1,sum[7:4],cout);
endmodule

module add(a,b,ci,s,co);
    input [3:0]a,b;
    input ci;
    output [3:0]s;
    output co;
    wire [3:0]c_tmp;
    wire [3:0]g;
    wire [3:0]p;
        assign co=c_tmp[3];
```

```
        assign g[0]=a[0]&b[0],g[1]=a[1]&b[1], g[2]=a[2]&b[2], g[3]=a[3]&b[3];
        assign p[0]=a[0]|b[0],p[1]=a[1]|b[1], p[2]=a[2]|b[2], p[3]=a[3]|b[3];
        assign c_tmp[0]=g[0]|(p[0]&ci);
        assign c_tmp[1]=g[1]|(p[1]&g[0])|(p[1]&p[0]&ci);
        assign
c_tmp[2]=g[2]|(p[2]&g[1])|(p[2]&p[1]&g[0])|(p[2]&p[1]&p[0]&ci);
        assign
c_tmp[3]=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]&ci);
        assign s[3:0]=a[3:0]^b[3:0]^{c_tmp[2:0],ci};
endmodule
```

iii. 8 位超前进位加法器 RTL 结构图

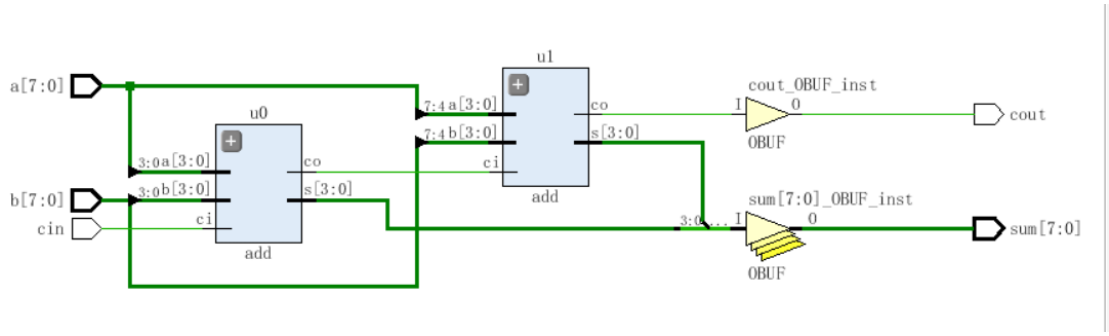

图 7： 超前进位加法器 RTL 结构图

iv. 测试激励模块代码

```
module test;
    reg [7:0] a;
    reg [7:0] b;
    reg cin;
    wire [7:0] sum;
    wire cout;
    max_speed U1(a,b,cin,sum,cout);
    initial
```

```
    begin
        a=0;b=1;cin=1;
    end
always
    begin
        #20 a=2;b=3;
        #20 a=20;b=44;
        #20 a=25;b=55;
        #20 a=30;b=66;
        #20 a=35;b=77;
    end
always #22 cin=~cin;
```

v. endmodule 得到仿真结果图像，并对图像进行分析，见实验结果与分析。

四：实验结果与分析

（1） 如下图所示，为 4 位乘法器仿真测试波形，从图中可知，仿真结果正确，电路设计有效：

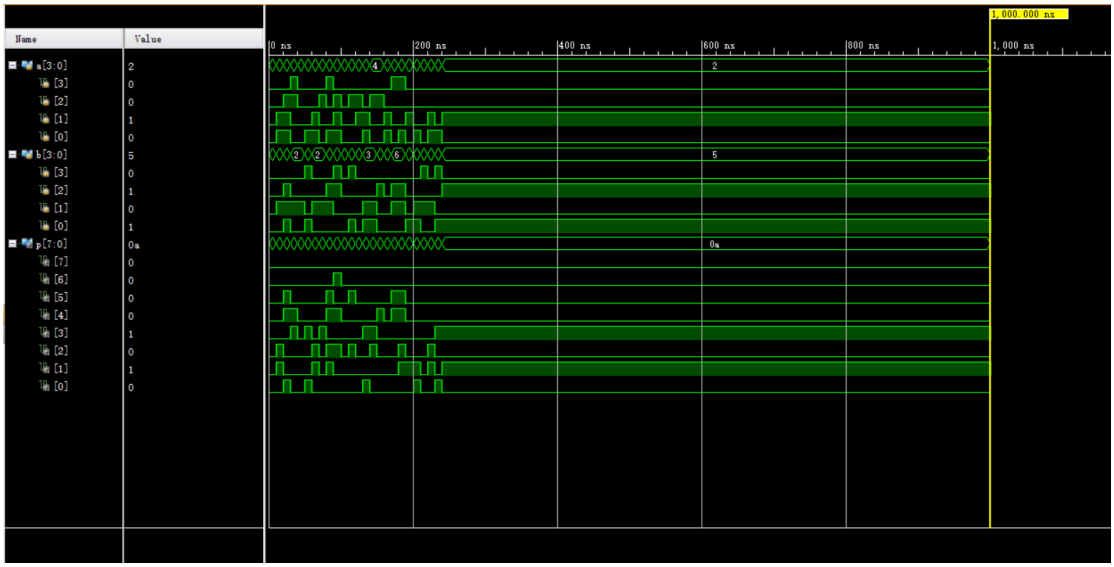

图 8： 4 位乘法器仿真波形

（2） 如下图所示，为 4 位除法器仿真测试波形，从图中可知，仿真结果正确，电路设计有效：

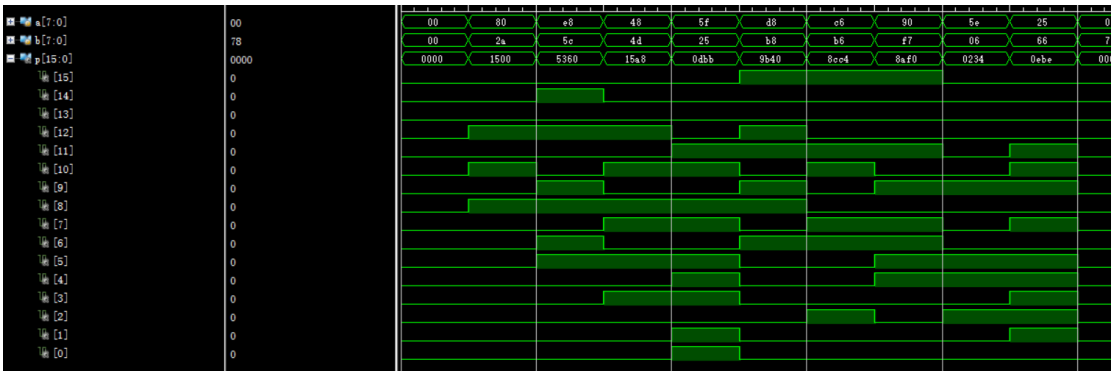

图 9： 4 位除法器仿真波形

（3） 如下图所示，为 8 位乘法器仿真测试波形，从图中可知，仿真结果正确，电路设计有效：

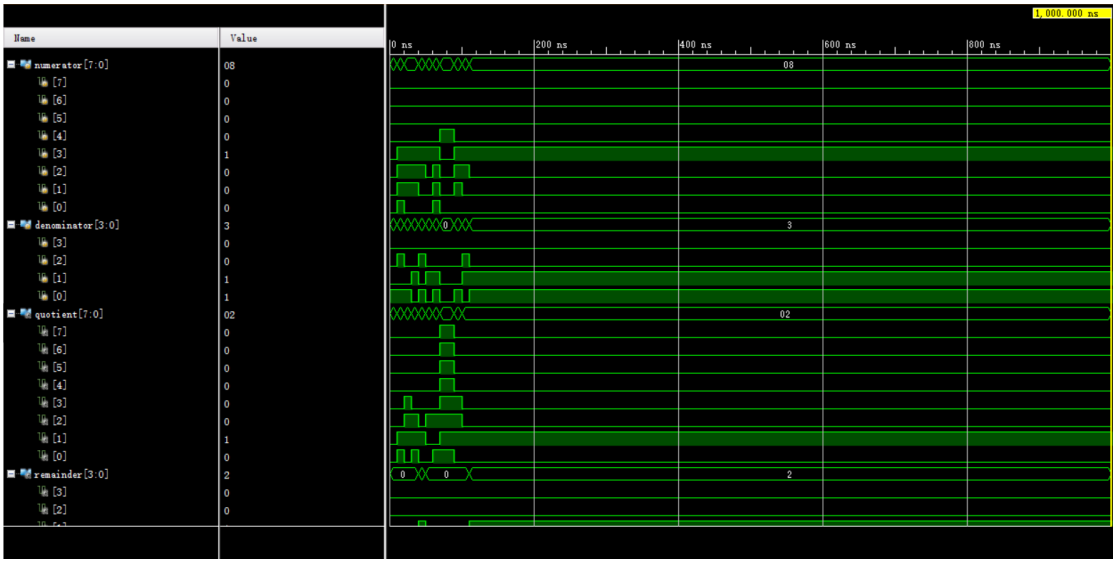

图 10： 8 位乘法器仿真波形

（4） 如下图所示，为 8 位除法器仿真测试波形，从图中可知，仿真结果正确，电路设计有效：

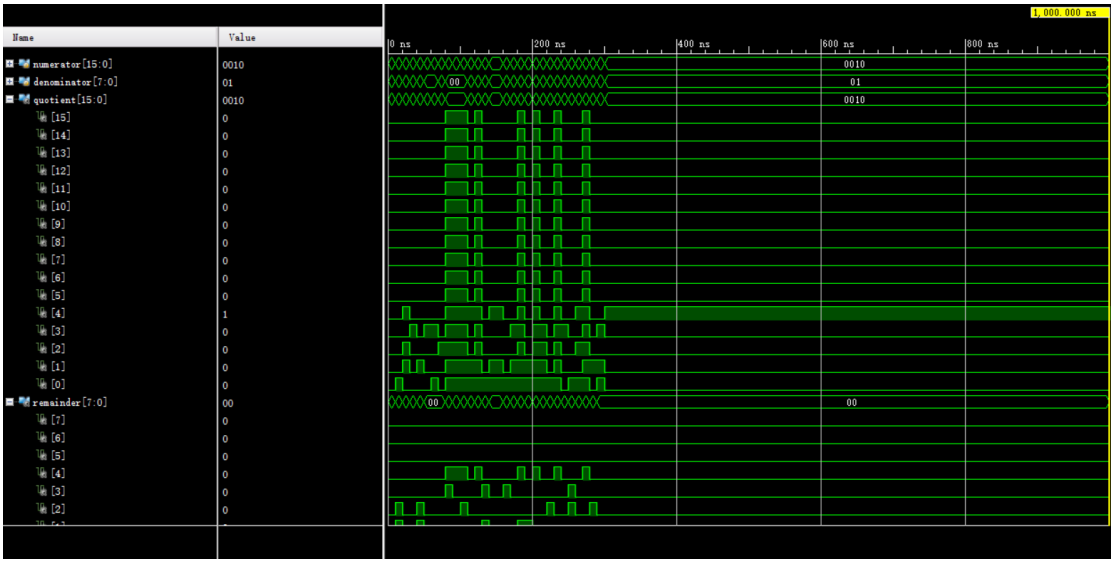

图 9：8 位除法器仿真波形

（5） 如下图所示，为 8 位进位加法器仿真激励图像，从图中可知，仿真结果正确，电路设计有效：

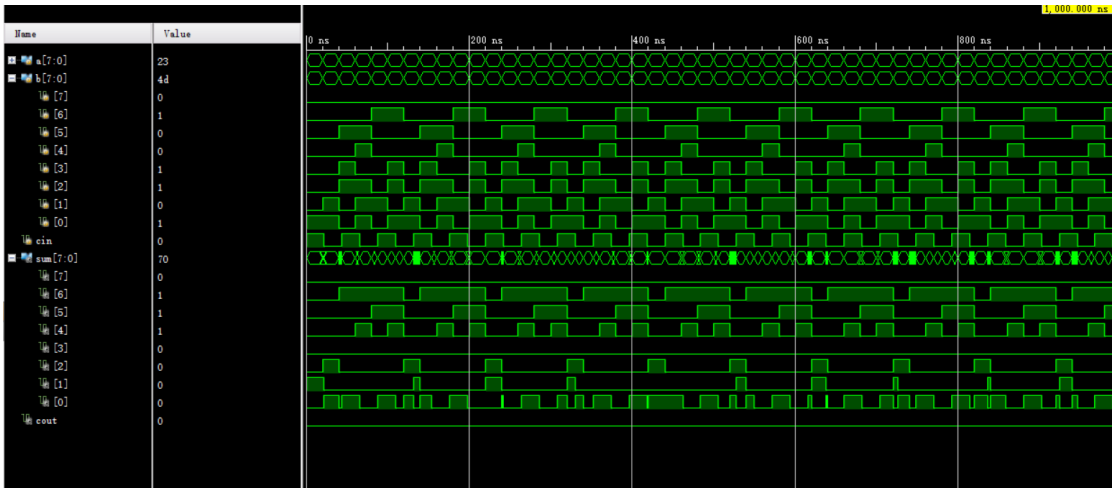

图 11：8 位行波加法器仿真波形

（6） 如下图所示，为 8 位选择进位加法器仿真激励图像，从图中可知，仿真结果正确，电路设计有效：

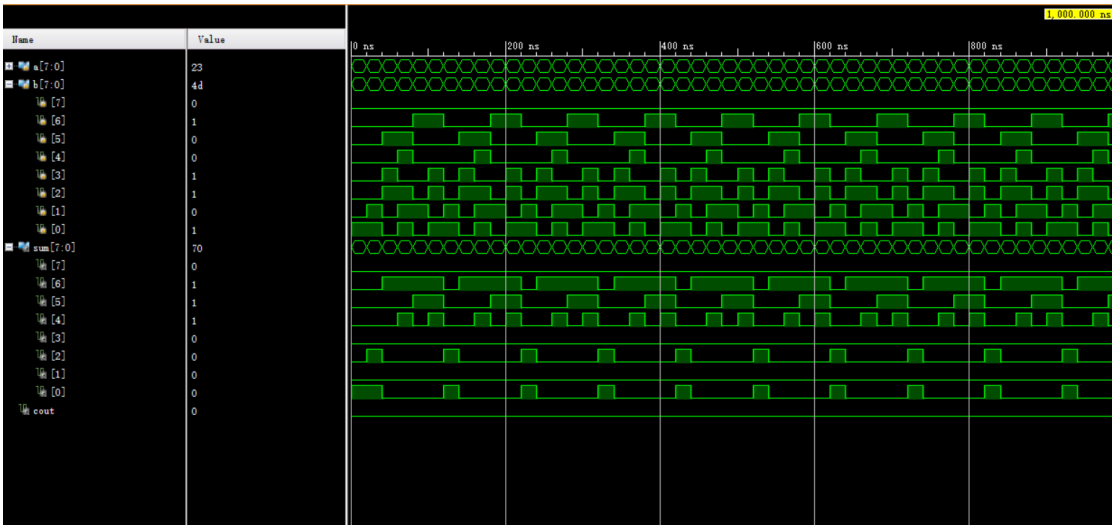

图 12：8 位选择进位加法器仿真波形

（7） 如下图所示，为 8 位选择进位加法器仿真激励图像，从图中可知，仿真结果正确，电路设计有效：

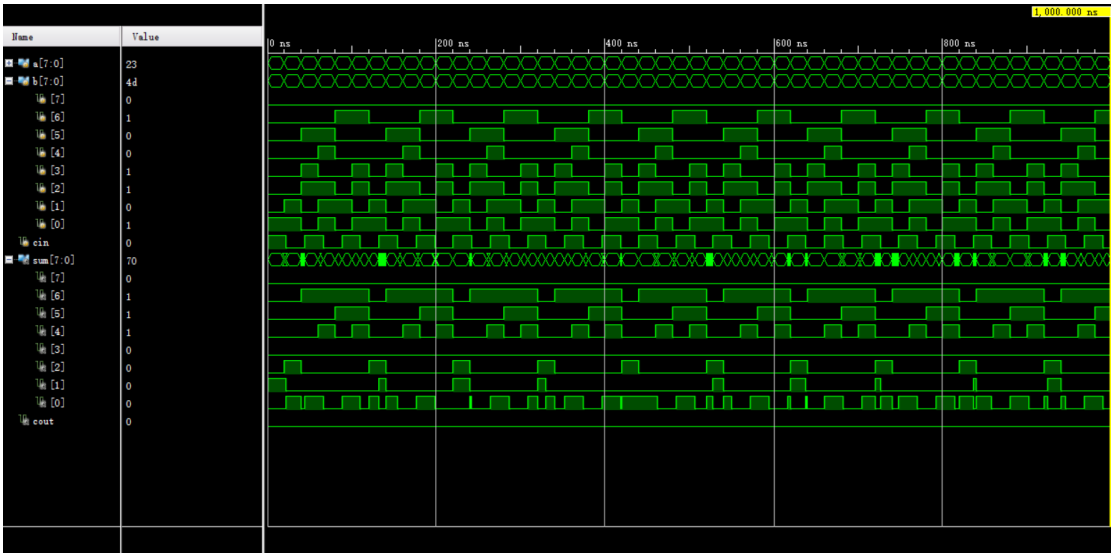

图 12：8 位超前进位加法器仿真波形

（8） 8 位行波进位加法器顶层模块图：

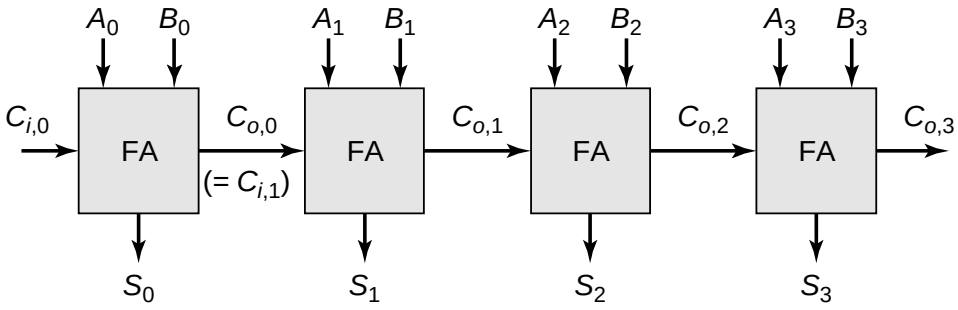

图 12：8 位超前进位加法器模块图

（9） 8 位选择进位加法器顶层模块图

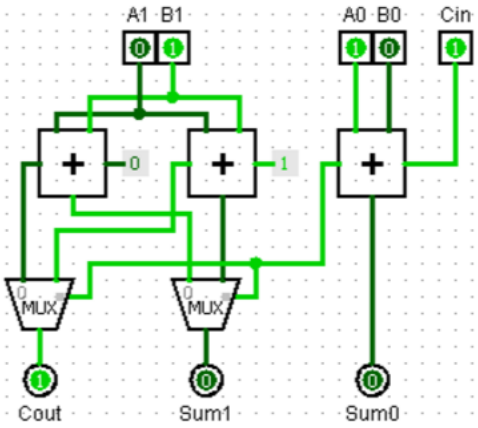

图 13： 8 位选择进位加法器顶层模块图

（10） 8 位超前进位加法器顶层模块图

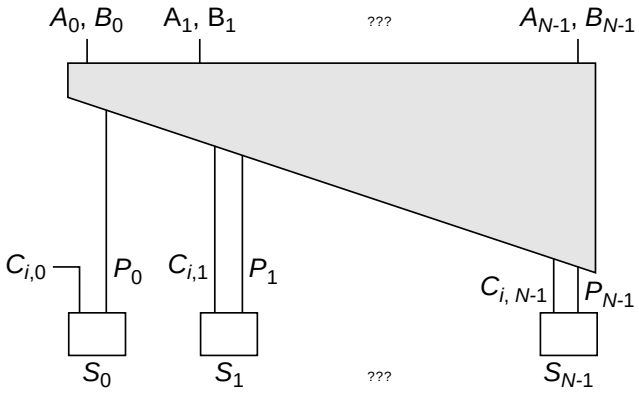

图 14：N 位超前进位加法器顶层模块图

五：结论（讨论）

1、实验结论

数字电路根据逻辑功能的不同特点，可以分成两大类，一类叫组合逻辑电路（简称组合电路），另一类叫做时序逻辑电路（简称时序电路）。组合逻辑电路在逻辑功能上的特点是任意时刻的输出仅仅取决于该时刻的输入，与电路原来的状态无关。而时序逻辑电路在逻辑功能上的特点是任意时刻的输出不仅取决于当时的输入信号，而且还取决于电路原来的状态，或者说，还与以前的输入有关。

在一般的设计中，组合逻辑电路设计的最简化是很重要的，在设计时常要求用最少的逻辑门或导线实现。与逻辑表示只有在决定事物结果的全部条件具备时，结果才发生。输出变量为 1 的某个组合的所有因子的与表示输出变量为 1 的这个组合出现、所有输出变量为 0 的组合均不出现，因而可以表示输出变量为 1 的这个组合。 通常通过逻辑电路图，输出端的逻辑表达式，真值表方式来解决。

而在 Vivado 环境下，Verilog HDL 描述语言给了我们一个简便的方式，通过高级语言的方式，行为描述语句，然后编译器自动的将代码转化成我们想要的电路，简单方便，在实现大规模电路的时候，显得尤为简便。

2、讨论

实验中依然存在着，过多的依赖于高级语言描述，而很少关注电路本身内部的门级关系这一问题，在学习的过程中，我们也应该注重电路内部的结构，因为只有对底层的更加了解，才能更好的运用和优化电路。同时，2.5 实验中，进位选择加法器的设计，其实优化的效果并不明显，而且理解上也最为难设计，我认为应加大对超前进位选择加法器的理解和使用，做到加法器中的最优情况。

六、教师评审

教师评语	实验成绩
签名： 日期：	