实验二 半加器、全加器的设计及应用

（注：由于本人实验图片不慎丢失，某些图片借鉴申程宇同学的图片）

一、实验目的

1. 掌握Verilog语言框架，编程及调试的方法；
2. 熟悉Verilog的基本语法；
3. 掌握Vivado开发平台及FPGA开发板的使用。

二、实验内容

1. 完成1位半加器、1位全加器模块的实现与测试；
2. 利用1位全加器实现2位全加器，并烧录到开发板进行验证；
3. 实现16/32位全加器，观察、分析仿真波形和RTL电路图。

三、实验要求

1. 画出模块的电路图（RTL分析）。
2. 分析电路的仿真波形
3. 记录设计和调试过程。

四、实验过程及结果分析

对实验代码、仿真结果等进行分析说明，不能只贴截图。

1. 一位半加器

代码：

module hafladder(S,C,A,B);

input A,B;

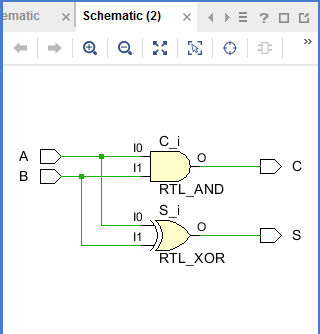
output S,C;

xor(S,A,B);

and(C,A,B);

endmodule

RTL电路图：



代码分析：

通过二输入与门和异或门实现半加器功能，输出为进位C和本位结果S

2.全加器

代码：

设计文件：

module fulladder(S,Co,A,B,Ci)

input A,B,Ci;

output S,Co;

wire S1,D1,D2;

halfadder HA1(S1,D1,A,B);

halfadder HA2(.A(S1),.B(Ci),.S(S),.C(D2));

or G1(Co,D2,D1);

endmodule

仿真文件：

module sim\_addr;

reg A,B;

reg Ci;

wire S;

wire Co;

initial begin

A=1’b0;B==1’b0;Ci=0;

end

always begin

#20 Ci=1’b1;

#20 B=B+1’b1;Ci=~Ci;

#20 {A,B,Ci}=3’b011;

#20{A,B,Ci}=3’b100;

repeat(3) begin

#20 {A,B,Ci}={A,B,Ci} +1’b001;

end

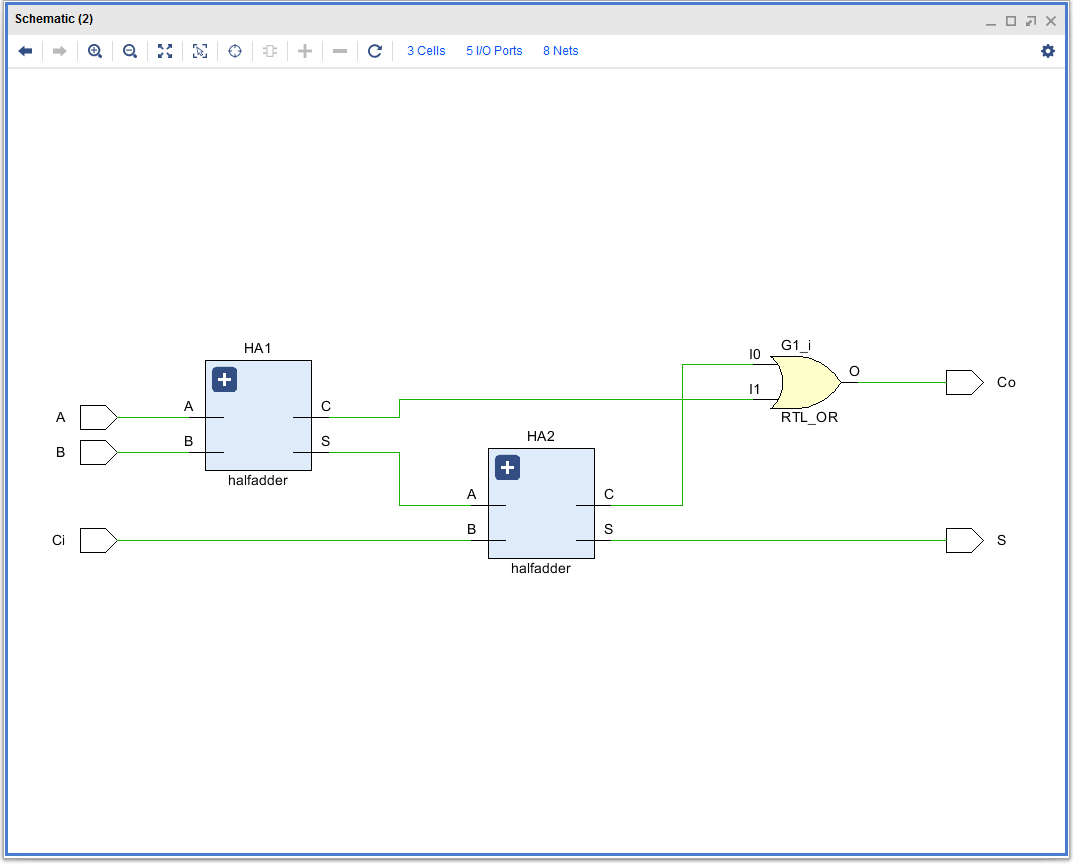
end

fulladder T(S,Co,A,B,Ci);

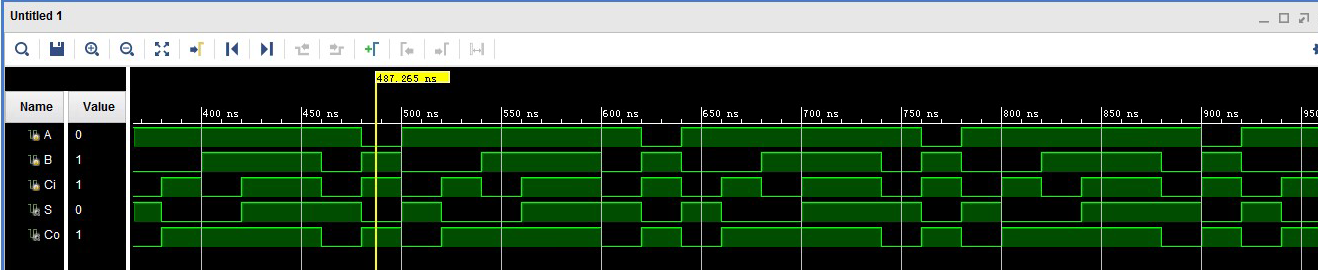
endmodule

仿真文件

RTL电路图：



仿真结果：



代码分析：

Desgin设计模块：

通过两个半加器实现全加器功能

三个输入信号：A、B、Ci

两个输出信号：S、Co

调用两个半加器HA1,HA2

HA1半加器对A、B进行运算，得到S与Ci进行运算，两个半加器D1、D2进行或运算作为最后的Co。

Sim仿真模块：

声明：

寄存器变量:A、B、Ci

线变量：S、Co

Initial对寄存器变量初始化,A，B位二位变量初始化位1’b0,一位Ci初始化为0。

接下来的always begin表示：

#20表示每20个时间单位：

Ci=1’b1表示Ci设置为1；

B=B+1’B1表示B在原来基础上加上1；

Ci=~Ci表示Ci位对原本取反

{A,B,Ci}=3’b100表示将A、B、Ci都设置为二进制数100

Repeat(3)用来将{A,B,Ci}={A,B,Ci}+1’b001语句重复三次

End结束reapeat模块

Fullader T(S,Co,A,B,Ci)实例化全加器模块

3.加法器（2位）

代码：

module addr\_2bit(S, Co, A, B, Ci);

input [1:0] A, B;

input Ci;

output [1:0] S;

output Co;

wire C;

fulladder FAO(S[0], C, A[0], B[0], Ci);

fulladder FA1(S[1], Co, A[1], B[1], C);

endmodule

设计模块

module sim\_addr\_2bit;

reg [1:0] A, B;

reg Ci;

wire [1:0] S;

wire Co;

initial begin

A = 2'b00;

B = 4'b00;

Ci = 1;

end

always begin

#10 A = A\_1'b1;

#20 B = B + 1'b1;

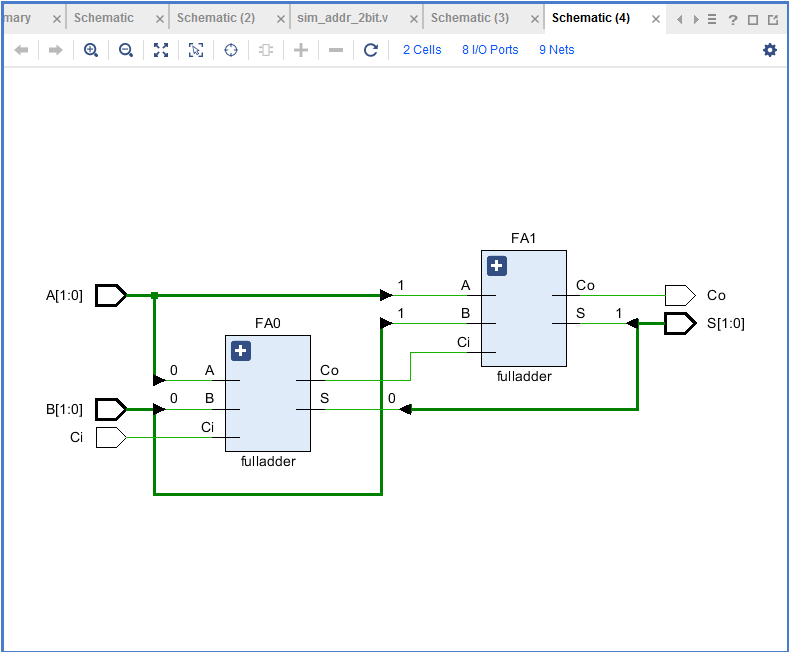
end

addr\_2bit T(S, Co, A, B, Ci);

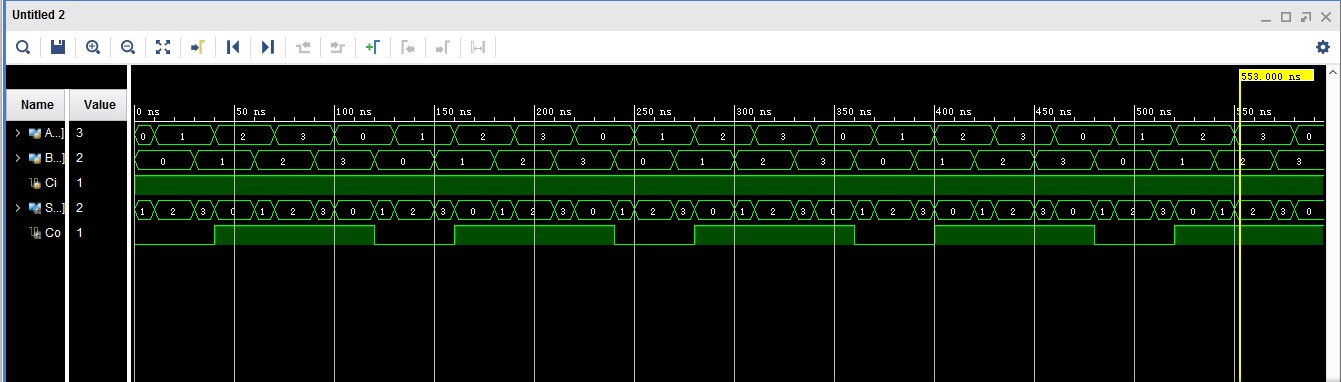
endmodule

仿真模块

RTL电路分析：



波形仿真图：



代码分析：

Design模块：

定义addr\_2bit的模块

声明两个输入端口A、B，都是2位向量，表示要相加的两个数

声明一个输入端口Ci，只有一位表示进位信号

声明输出端口S，为2位向量

声明进位Co，只有1位输出的进位信号

然后主要内容为调用2个一位全加器板块，命名为F0，F1，每个全加器三个输入端口和两个输出端口，输入位和来自上一个全加器的进位信号相加，产生进位信号和这一位的输出

Sim模块：

定义名为sim\_addr\_2bit的模块

声明寄存器变量A,B，都是2位宽度的向量，存储输入信号

声明寄存器变量Ci，1位宽度，存储进位信号

声明线变量S，2位宽度向量，连接输出结果

声明线变量Co，用于连接加法器的进位输出

Initial语句初始化，A,B初始化位二进制0，Ci初始化为1；

Always语句循环，每10个单位A设置为1，每20个时间单位B自增1

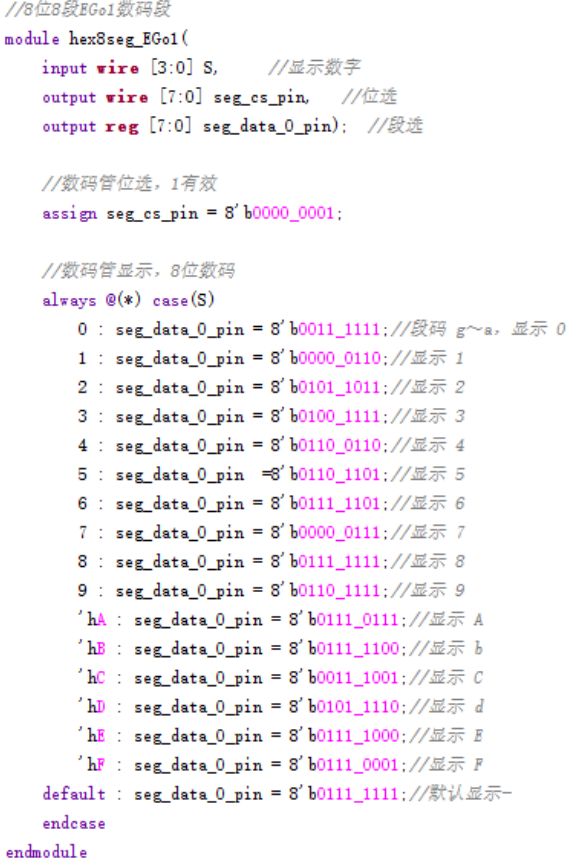
addr\_2bit T(S,Co,A,B,Ci)实例化名为T的2位加法器模块，并且把各个信号对应到相应的输出端口

 2位全加器EGo1开发封装代码：

module addr2b\_ hex8seg EGo1 (  
input vire Ci;  
input wire [1:0] AB;  
output wire [1:0] s;   
output wire Co;  
output vire [7:0] seg cs, pin, output vire [7:0] seg data\_0. pin):   
addr\_ 2bit addr2(S, Co,A, B，Ci):  
hex8seg EGol he({&apos; b0,Co,S}, seg cs. pin, seg \_data. 0. pin):

endmodule

EGo1数码管显示模块：



1. 约束文件



开发板验证：



通过左下方按键控制增量，可以实现两位的加法器功能。

1. 加法器（16位）

代码：

module addr\_16bit(S, Ch, A, B, Cl);

input [15:0] A, B;

input Cl;

output [15:0] S;

output Co;

wire C0, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14;

fulladder FAO(S[0], Co, A[0], B[0], C1),

fulladder FA1(S[1], C1, A[1], B[1], C0),

fulladder FA2(S[2], C2, A[2], B[2], C1),

fulladder FA3(S[3], C3, A[3], B[3], C2),

fulladder FA4(S[4], C4, A[4], B[4], C3),

fulladder FA5(S[5], C5, A[5], B[5], C4),

fulladder FA6(S[6], C6, A[6], B[6], C5),

fulladder FA7(S[7], C7, A[7], B[7], C6),

fulladder FA8(S[8], C8, A[8], B[8], C7),

fulladder FA9(S[9], C9, A[9], B[9], C8),

fulladder FA10(S[10], C10, A[10], B[10], C9),

fulladder FA11(S[11], C11, A[11], B[11], C10),

fulladder FA12(S[12], C12, A[12], B[12], C11),

fulladder FA13(S[13], C13, A[13], B[13], C12),

fulladder FA14(S[14], C14, A[14], B[14], C13),

fulladder FA15(S[15], Co, A[15], B[15], C14);

设计文件

module sim\_addr\_16bit;

reg [15:0] A, B;

reg Ci;

wire [15:0] S;

wire Co;

initial begin

A = 16'b0000000000000000;

B = 16'b0000000000000000;

Ci = 1;

end

always begin

#10 A = A + 1'b1;

#20 B = B + 1'b1;

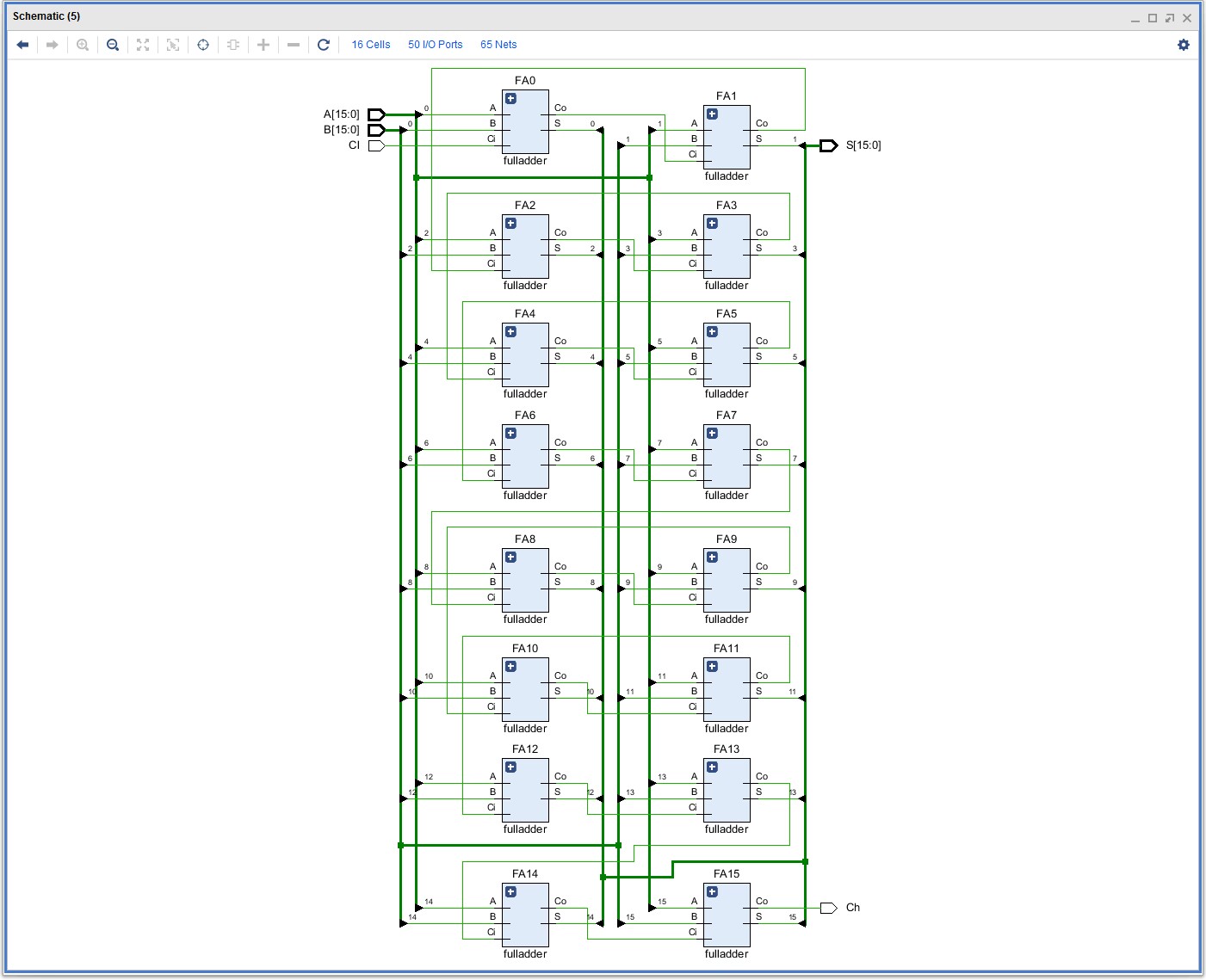
end

addr\_16bit T(S, Co, A, B, Ci);

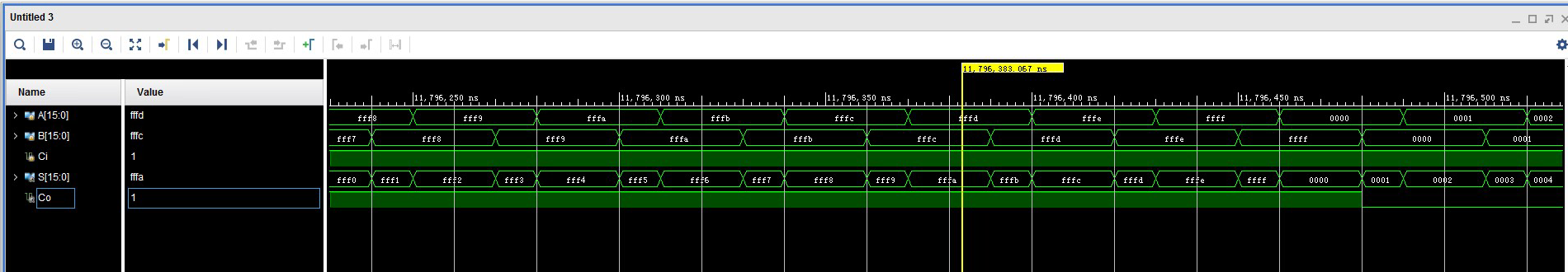
endmodule

仿真文件

RTL电路图：



仿真分析：



代码分析：

Design模块：

定义addr\_16bit的模块

声明两个输入端口A、B，都是16位向量，表示要相加的两个数

声明一个输入端口Ci，只有一位表示进位信号

声明输出端口S，为16位向量

声明进位Co，只有一位输出的进位信号

然后主要内容为调用16个一位全加器板块，命名为F0到F15，每个全加器三个输入端口和两个输出端口，输入位和来自上一个全加器的进位信号相加，产生进位信号和这一位的输出(与2位加法器原理相同)

Sim模块：

声明两个16位寄存器变量A和B，存储输入信号

声明1位寄存器变量Ci，存储进位信号

声明16位向量线变量S，连接存储输出信号

声明1位线变量Co，连接进位输出

Initial进行变量初始化，A,B赋0，Ci初始化为1

always进行循环，每10个时间单位A增加1，每20个时间单位B增加1。

addr\_16bit T(S,Co,A,B,Ci)实例化名为T的16位加法器模块，把A,B,C,Ci,Co连接到对应端口。

五、调试和心得体会

记录实验过程中遇到的问题和调试解决过程，心得体会等。

通过这次实验，我有如下心得体会：

1. 本次实验学习了半加器和全加器的设计，让我更加深入了了解了半加器和全加器的原理。
2. 更加深入了解了reg变量和wire变量的不同。
3. 学会了自己编写16位加法器，实际上16位加法器与2位加法器的原理基本相同，只是中间变量的个数更多，递归关系都是相同的。本次实验我通过16次调用一位加法运算其实现此功能，实际上可以先设计4位全家器，再通过调用4位全加器，实现16全加器的功能，代码可能会更加简洁。