**实验七 16位乘法器实验**

**一、实验目的**

1、了解16位有符号、无符号乘法器的实现原理。

2、使用Verilog实现16位无符号乘法器和有符号乘法器。

**二、实验设备**

1、装有vivado的计算机 1台

2、EGO1开发板 1块

**三、实验任务**

1、设计无符号乘法器，将两个16位无符号数相乘,得到一个32位无符号数。

2、设计有符号乘法器，将两个16位有符号数相乘,得到一个32位有符号数。

3、设计键盘输入控制电路，设计七段显示器显示电路。通过按键输入两个乘数，乘积的结果送给七段显示器。

**四、实验原理**

乘法的基本概念

被乘数x为1000，乘数y为1001，下面的乘法过程是手工运算的一个步骤,而计算机在做乘法时就是模拟手工运算的执行过程。如图7.1所示

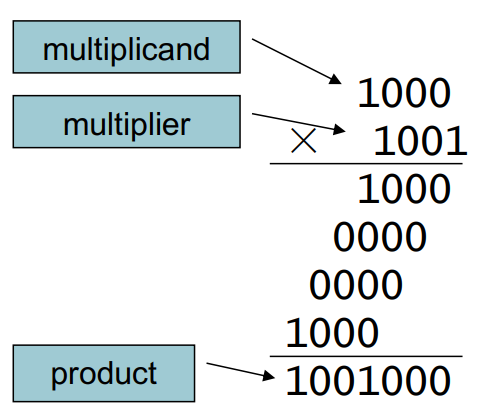
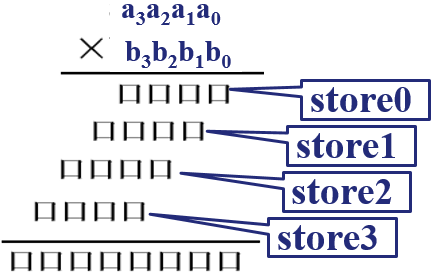
 

图7.1 手工乘法的执行过程

根据上面的手工计算过程，画出与之对应的硬件电路，如图7.2所示。

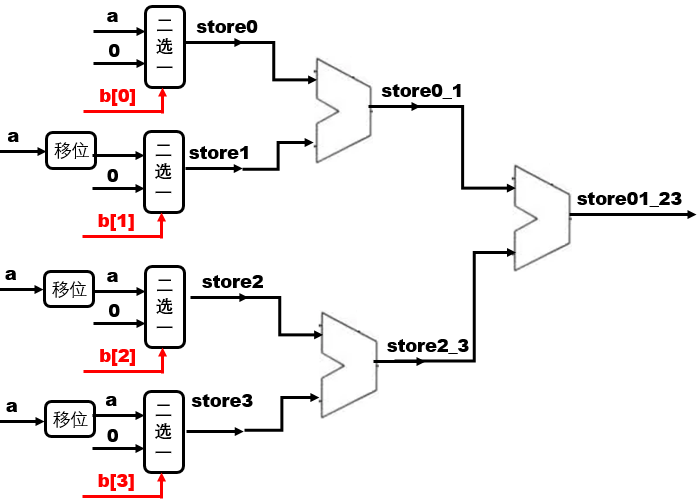


图7.2

1、无符号乘法器的功能为:将两个16位无符号数相乘,得到一个32位无符号数﹐如图7.3所示。

接口定义:

module MULTU(

input clk, //乘法器时钟信号

input reset, //复位信号,低电平有效

input [ 15:0] a, //输入数a(被乘数)

input [ 31:0] b, //输入数b(乘数)

output [ 31:0] z //乘积输出z

);

2、有符号乘法器的功能为:将两个16位有符号数相乘,得到一个32位有符号数,如图7.4所示。

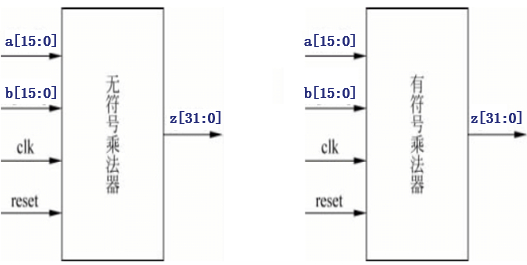


图7.3无符号乘法器 图7.4有符号乘法器

接口定义:

module MULT(

input clk, //乘法器时钟信号

input reset, //复位信号,低电平有效

input [15:0] a, //输入数a(被乘数)

input [ 15:0] b, //输入数b(乘数)

output [ 31:0] z //乘积输出z

);

3、无符号乘法器功能为:将两个16位无符号数相乘,得到一个32位无符号数。有符号乘法器功能为:将两个16位有符号数相乘,得到一个32位有符号数。将低16位存放在专用寄存器lo中,高16位存放在寄存器hi中。执行乘法指令过程中不产生异常。

本实验建议不要使用行为级描述方式实现对应的电路。

4、下面提供几种实验实现思路,仅供参考。

**思路(1)**：两个二进制数a和 b相乘,可以认为是a和 b的每一位相乘移位后的结果相加。关于a与b的每一位相乘产生的中间结果,如果b那位是0,那么中间结果就是0;如果是1 ,那么中间结果就是在a前后补上相应位数的零，通过字符拼接的方式表示。然后将这些中间乘积相加就是最后的结果。

**思路(2）**：二进制的乘法可以用加法和移位操作完成，可以循环迭代的方法实现。每次循环时,判断b的值是否为1,然后决定是否将中间值加上a。每次循环,a左移一位,b右移一位。循环结束,最后的中间值就是最后的乘积。

**思路(3）**：可以从Wallace Tree乘法算法的角度出发实现。有兴趣实现的读者可以自行查阅Wallace Tree算法实现的特点。

下面按照上面思路（1）来举例说明8位无符号数乘8位无符号数的一种实现方式。

对应的verilog参考如下：

module MULTU(

input clk, //乘法器时钟信号

input reset,

input [ 7:0] a, //输入a(被乘数)

input [ 7:0] b, //输入b(乘数)

output [ 15:0] z //乘积输出z

);

reg [ 15:0 ] temp; //申请寄存器

reg [ 15:0 ] stored0;

reg [ 15:0 ] stored1;

reg [ 15:0 ] stored2;

reg [ 15:0 ] stored3;

reg [15 :0] stored4;

reg [15 :0] stored5 ;

reg [ 15:0 ] stored6;

reg [ 15:0 ] stored7;

reg [ 15:0 ] add0\_1;

reg [ 15:0 ] add2\_3;

reg [ 15:0 ] add4 \_5;

reg [ 15:0 ] add6\_7;

reg [ 15:0 ] add0t1\_2t3;

reg [ 15:0 ] add4t5\_6t7;

reg [ 15:0 ] add0t3\_4t7;

always @( posedge clk or negedge reset)

begin

if(reset) begin //reset 置零

temp <= 0;

stored0 <= 0 ;

stored1 <= 0 ;

stored2<= 0 ;

stored3<= 0 ;

stored4 <= 0 ;

stored5<= 0 ;

stored6<= 0 ;

stored7 <= 0 ;

addO\_1 <= 0;

add2\_3<= 0;

add4\_5 <= 0;

add6\_7 <= 0 ;

add0t1\_2t3<= 0;

add4t5\_6t7<= 0;

end

else begin //通过字符拼接方式表示出中间相乘值,并相加

stored0<= b[0] ? { 8' b0,a} : 16'b0 ;

stored1 <= b[1] ? { 7’b0,a,1'b0} :16' b0 ;

stored2<= b[2] ? {6' b0,a,2'b0} :16' b0 ;

stored3<= b[3] ? { 5'b0,a,3 'b0} :16' b0 ;

stored4 <= b[4] ? {4' b0,a,4 'b0} :16' b0 ;

stored5<= b[5] ? { 3' b0,a,5 'b0} :16' b0;

stored6<= b[6] ? { 2'b0,a,6'b0} :16' b0 ;

stored7 <= b[7] ? { 1' b0,a,7'b0} :16' b0 ;

add0\_1 <= stored1 + stored0 ;

add2\_3<= stored2 + stored3 ;

add4\_5<= stored4 + stored5 ;

add6\_7<= stored6 + stored7;

add0t1\_2t3<= add0\_1 + add2\_3;

add4t5\_6t7 <= add4\_5 + add6\_7;

temp <=add0t1\_2t3 + add4t5\_6t7;

end

end

assign z = temp;

endmodule

5、写完乘法电路模块,可以采用以下数据进行测试。

a = 0, b = 0; a = 0,b = 8'b11111111;

a = 8' b10110011,b = 0 ; a = 8'b11111111, b = 8'b11111111;

a = 8'b10000000, b = 8'b10101010; a = 8'b10101010，b = 8'b1000000o;

a = 8'b101101 ; b= 8'b1101000; a = 8' b1000111, b = 8'b1110

(注意:在写32位乘法器的时候，需要用32位的数据进行测试。)

6、至于有符号数乘法器的实现只需要进行简单的变动,即可实现。请大家自己思考。本实验不允许使用行为级(乘号)方式实现。

**五、实验步骤**

**（一）设计无符号乘法器:将两个16位无符号数相乘,得到一个32位无符号数﹐参考图7.2和图7.3类似的结构图。（实验室完成和只进行仿真验收）**

1、**新建Vivado 工程**

**参照实验五的步骤创建新工程。**

由2个16位无符号数相乘,得到一个32位无符号数，接口定义类似如下:

module MULTU(

input clk, //乘法器时钟信号

input reset,

input [ 15:0] a, //输入a(被乘数)

input [15:0] b, //输入b(乘数)

output [ 31:0] z //乘积输出z

);

填写下面乘法器模块的verilog代码。

**乘法器模块**：

module MULTU(…… )

请在此处补充对应的verilog代码，要求在实验前完成。进实验室时会检查。验收时会围绕核心代码进行提问。

endmodule

**2、在Vivado中进行仿真测试主模块。**

建立了乘法模块后，如果需要对模块进行测试的话，就需要给电路添加相应的激励输入，观察电路输出所具有的特点。例如：对两个乘数，分别赋值为21H和36H，然后观察乘积输出结果。观察输出波形是否正确。

**激励输入模块：**

module MULTUtest(…… )

请在此处补充对应的verilog代码，要求在实验前完成。进实验室时会检查。验收时会围绕核心代码进行提问。

endmodule

进行仿真，验证仿真后的波形图是否正确。

在下面粘贴你的仿真截图，并分析结果的正确性。

**3、在Vivado中设计输入控制电路，将按键sw0~sw9分别对应于十进制数中的0~9，消除抖动的影响。例如：输入1234H和5678H。**

（1）输入控制电路，将按键sw0~sw9分别对应于十进制数中的0~9。例如：sw6推向上方，表示输入一个6，sw8推向上方，表示输入一个8。为了消除按钮的抖动对输入的影响，可以设计一个定时时间为10ms的定时器。第一次输入读数后，延时10ms后再读第二次，如果第二次和第一次读到的对应按键的状态相同，就认为没有抖动影响。

**定时10ms模块：**

module counter10(…… )

感兴趣的同学请在此处补充对应的verilog代码。进实验室时不会检查。不作验收要求。

endmodule

**输入模块：**

module inputTWO(…… )

感兴趣的同学请在此处补充对应的verilog代码。进实验室时不会检查。不作验收要求。

endmodule

**4、在Vivado中进行引脚绑定，并且将乘积的结果通过七段显示器进行动态显示。**

（1）要想在多个七段显示器上进行动态刷新显示乘积结果，需要设计定时50ms的时钟，每隔50ms就在一个七段显示器上显示乘积结果中的一位。由于视觉暂停效应，人眼是看不清多个七段显示器的值是分时显示的。人眼感觉到的是多个七段显示器上值是同时显示的。

**定时50ms的模块：**

module counter50(…… )

感兴趣的同学请在此处补充对应的verilog代码。进实验室时不会检查。不作验收要求。

endmodule

**每隔50ms显示七段显示器一位的控制模块**

module showMUL(…… )

感兴趣的同学请在此处补充对应的verilog代码。进实验室时不会检查。不作验收要求。

endmodule

**5、在vivado中生成bit流文件和下载，在EGO1开发板上进行乘数输入(**不作验收要求。**),观察七段显示器上的乘积输出结果。验证结果是否正确。**

**六、实验思考：**

1、按照实验原理里的第4条思路（2）进行设计，所对应的电路结构是怎样的。

2、按照实验原理里的第4条思路（3）进行设计，所对应的电路结构是怎样的。