2021～2022 学年秋季学期

A3705060050《计算机系统》必修课

课设实验报告

——32 周期移位乘法器

班级：人工智能 1901

组长：　　于之晟

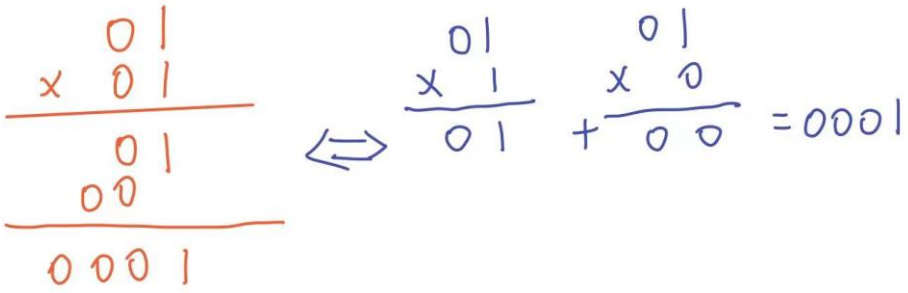
组员：　　黄元通

报告日期：2021.12.19

1、实现思路

要将原本一周期实现的乘法转换为 32 周期的乘法，实际上是将 32 位的乘法等价替代为 32 次 32 位加法。因为在竖式乘法算式中，乘数分别与被乘数的每一位相乘，得到的结果一次前移一位后进行加法从而得到乘法结果。

如下图两位乘法所示：



2、具体实现

mul.v:

(1)端口

名称	输入/输出	宽度	作用
rst	input wire	1	复位
clk	input wire	1	时钟
signed_mul_i	input wire	1	是否为有符号乘法运算，1 位有符号
opdata1_i	input wire	32	乘数
opdata2_i	input wire	32	被乘数
start_i	input wire	1	是否开始乘法运算
annul_i	input wire	1	是否取消乘法运算，1 位取消
result_o	output reg	64	乘法运算结果
ready_o	output reg	1	乘法运算是否结束

(2)信号

名称	类型	宽度	作用
cnt	reg	5	记录乘加法进行了几轮
multdend	reg	64	中间计算结果
state	reg	1	乘法器处于的状态

temp_op1	reg	32	乘数绝对值
temp_op2	reg	32	被乘数绝对值
mul_add	wire	64	中间乘数分解得到的加法数
mul_y	reg	32	乘数，运算时每次右移一位
mul_x	reg	64	加载被乘数，运算时每次左移一位

其中 mul_add = mul_y[0] ? mul_x : {`ZeroWord, `ZeroWord} ;

(3)乘法器状态：

1. 空闲状态(信息初始化)：

```
if (rst) begin
        state <= `MulFree;
        result_o <= {`ZeroWord,`ZeroWord};
        ready_o <= `MulResultNotReady;
```

2. 乘法有零参与：

对于有零参与的乘法，计算结果直接赋值为零

状态判断：

```
if(opdata2_i == `ZeroWord||opdata1_i== `ZeroWord) begin//如果乘法中有数是 0
    state <= `MulByZero;
```

状态赋值：

```
    multdend <= {`ZeroWord, `ZeroWord};
    state <= `MulEnd;
```

3．乘法正常计算：

由于存在有符号运算，所以需要先将运算数进行取绝对值操作：

```
    if(signed_mul_i == 1'b1 && opdata1_i[31] == 1'b1) begin    //被乘数为负数
        temp_op1 = ~opdata1_i + 1;
  end else begin
        temp_op1 = opdata1_i;
  end
  if (signed_mul_i == 1'b1 && opdata2_i[31] == 1'b1 ) begin//乘数为负数
        temp_op2 = ~opdata2_i + 1;
  end else begin
        temp_op2 = opdata2_i;
  end
  multdend <= {`ZeroWord, `ZeroWord};
  mul_x <= {`ZeroWord, temp_op1};
  mul_y <= temp_op2;
```

将乘法运算转换为加法运算，乘数运算时右移一位，被乘数运算时左移一位；中间计算结果需要增添加法数：

```
mul_x <= {mul_x[62:0],1'b0};   //被乘数 x 每次左移一位
mul_y <= {1'b0,mul_y[31:1]}; //相当于乘数 y 右移一位
multdend <= multdend + mul_add;//中间结果加上加法数
cnt <= cnt +1;        //乘加法运算次数
```

特殊：乘法计算过程中需要判断是否是有符号乘法且乘数和被乘数中有负数，需要对运算结果进行取补码；

```
if ((signed_mul_i == 1'b1) && ((opdata1_i[31] ^ opdata2_i[31]) == 1'b1)) begin
//乘法中有负数
    multdend <= (~multdend + 1);
```

4. 乘法结束：

当 cnt= 6'b100000 (即 32)时，乘法转换为的 32 次加法结束，取出计算结果并输出

```
result_o <= multdend;
  ready_o <= `MulResultReady;
  if (start_i == `MulStop) begin
        state <= `MulFree;
        ready_o <= `MulResultNotReady;
        result_o <= {`ZeroWord, `ZeroWord};
      end
```

(4)EX 段调用：

乘法指令为有符号乘法 mult 和无符号乘法 multu;

mult:

当乘法运算完成时：(mul_ready_i == `MulResultReady)

```
mul_opdata1_o = rf_rdata1;
mul_opdata2_o = rf_rdata2;
mul_start_o = `MulStop;
signed_mul_o = 1'b1;              //有符号乘法
stallreq_for_mul = `NoStop;       //流水线停止暂停
```

当乘法运算未完成时：(mul_ready_i == `MulResultNotReady)

```
mul_opdata1_o = rf_rdata1;
mul_opdata2_o = rf_rdata2;
mul_start_o = `MulStart;
signed_mul_o = 1'b1;              //有符号乘法
stallreq_for_mul = `Stop;         //流水线暂停
```

multu:

当乘法运算完成时：(mul_ready_i == `MulResultReady)

```
mul_opdata1_o = rf_rdata1;
mul_opdata2_o = rf_rdata2;
mul_start_o = `MulStop;
signed_mul_o = 1'b0;                    //无符号乘法
stallreq_for_mul = `NoStop;             //流水线停止暂停
```

当乘法运算未完成时：(mul_ready_i == `MulResultNotReady)

```
mul_opdata1_o = rf_rdata1;
mul_opdata2_o = rf_rdata2;
mul_start_o = `MulStart;
signed_mul_o = 1'b0;                    //无符号乘法
stallreq_for_mul = `Stop;               //流水线暂停
```

其他情况：

```
mul_opdata1_o = `ZeroWord;
mul_opdata2_o = `ZeroWord;
mul_start_o = `MulStop;
signed_mul_o = 1'b0/1'b1;
stallreq_for_mul = `NoStop;
```

3、参考资料

https://blog.csdn.net/weixin_43074474/article/details/90473709