**Booth乘法器**

# 一、Booth乘法器原理：

在计算两个补码相乘时，可以通过Booth算法来实现定点补码一位乘的功能。布斯(Booth)算法采用相加和相减的操作计算补码数据的乘积，Booth算法对乘数从低位开始判断，根据后两个数据位的情况决定进行加法、减法还是仅仅进行移位操作。讨论当相乘的两个数中有一个或二个为负数的情况，在讨论补码乘法运算时，对被乘数或部分积的处理上与原码乘法有某些类似，差别仅表现在被乘数和部分积的符号位要和数值一起参加运算。

Booth乘法规则如下：

假设X、Y都是用补码形式表示的机器数，[X]补和[Y]补=Ys.Y1Y2…Yn，都是任意符号表示的数。比较法求新的部分积，取决于两个比较位的数位，即Yi+1Yi的状态。

首先设置附加位Yn+1=0，部分积初值[Z0]补=0。

当n≠0时，判断YnYn+1，

若YnYn+1=00或11，即相邻位相同时，上次部分积右移一位，直接得部分积。

若YnYn+1=01，上次部分积加[X]补，然后右移一位得新部分积。

若YnYn+1=10，上次部分积加[-X]补，然后右移一位得新部分积。

当n=0时，判YnYn+1(对应于Y0Y1)，运算规则同(1)只是不移位。即在运算的最

后一步，乘积不再右移。

# 二、设计思路

程序首先进行判0操作，如果乘数中有一个或两个为0，则直接输出结果0，否则进入程序主体。

程序主体分成三个判断模块进行，当乘数最低位与次低位值相等时，先将乘数右移一位，再将原部分累加和右移一位至乘数最高位，同时部分积累加和的最高位根据次高位正负补0或1；如果乘数最低位与次低位分别为1，0时，将原部分累加和加上被乘数X补后，再右移一位至乘数最高位，同时部分积累加和的最高位根据次高位正负补0或1；如果乘数最低位与次低位分别为0，1时，将原部分累加和减去被乘数X补后，再右移一位至乘数最高位，同时部分积累加和的最高位根据次高位正负补0或1。

每次比较一次乘数的最末两位，进行相应运算后，共循环4次。循环结束后，再进行一次判断，如果乘数最低位与次低位分别为1、0，将原部分累加和加上被乘数X补。如果乘数最低位与次低位分别为0、1，将原部分累加和减去被乘数X补。此时，最终累加和就是乘积的高位结果，取乘数的高四位作为低位结果，拼接即为最终乘法结果。



图1

一个具体的例子（以7（0111）\*2（0010）为例）：



图2

# 三、代码实现

module mul\_Booth(Mx, My, Mout);

input [5:0] Mx, My;

output reg [9:0] Mout;

reg [5:0] a;

reg [5:0] b, c;

reg [3:0] n;

reg p ,q;

always @(Mx, My) begin

if (Mx == 0 || My == 0) Mout <= 0;

else begin

a = 6'b0;

n = 4'b1111;

p = 1'b1;

q = 1'b0;

b = Mx;

c = My;

c = {c[4:0], q};

while (n) begin

n = n >> 1;

if ((c[0] == 0 && c[1] == 0) || (c[0] == 1 && c[1] == 1)) begin

c = c >> 1;

c[5] = a[0];

a = a >> 1;

if (a[4] == 1) a = {p, a[4:0]};

else a = a;

end

else if (c[0] == 1 && c[1] == 0) begin

a = a + b;

c = c >> 1;

c[5] = a[0];

a= a >> 1;

if (a[4] == 1) a = {p, a[4:0]};

else a = a;

end

else if (c[0] == 0 && c[1] == 1) begin

a = a - b;

c = c >> 1;

c[5] = a[0];

a = a >> 1;

if (a[4] == 1) a = {p, a[4:0]};

else a = a;

end

end

if (c[0] == 1 && c[1] == 0) a = a + b;

else if (c[0] == 0 && c[1] == 1) a = a - b;

Mout = {a, c[5:2]};

end

end

endmodule

# 四、仿真测试

module tb\_mul\_Booth();

reg clk;

reg [5:0] Mx, My;

wire [9:0] Mout;

initial begin

clk = 0;

Mx = 0;

My = 0;

#10

Mx = 9;

My = 6;

#10

Mx = 12;

My = 12;

#10 repeat(10) @(posedge clk) begin

Mx <= {$random} % 16;

My <= {$random} % 16;

end

$stop;

end

always #5 clk = ~clk;

mul\_Booth Booth(Mx, My, Mout);

endmodule

# 五、仿真结果

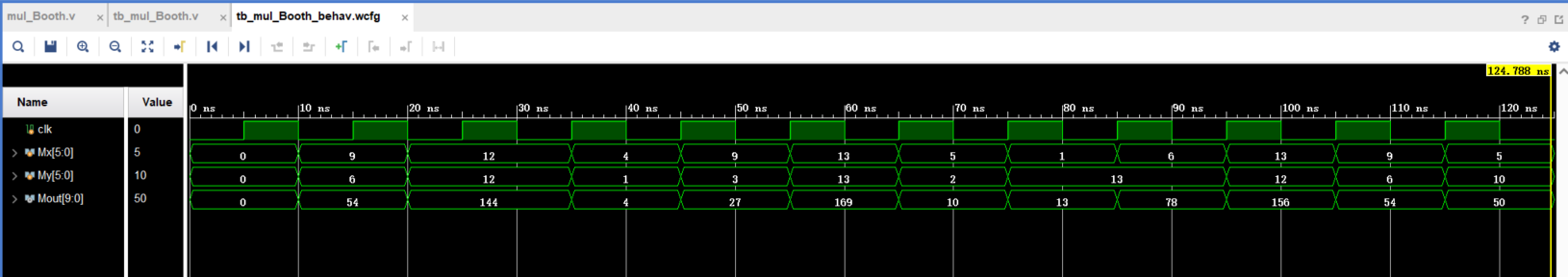


图1