乘累加器

原理：

一位乘法相加，若乘数第i位为1，则加上被乘数的2的i次幂，若乘数第i位为0，则被乘数加上0.

代码：

module mac(

input clk,

input clr,

input [7:0] opa,

input [7:0] opb,

output [15:0] out

);

reg [15:0] out;

wire [15:0] sum;

function [15:0] mult;

input [7:0] opa, opb;

reg [15:0] result;

integer index;

begin

result = opa[0] ? {{8'b0}, opb} : 16'b0;

for (index = 1; index < 8; index = index + 1)

begin

if (opa[index] == 1) result = result + (opb<<index);

else result = result + {16'b0};

end;

mult = result;

end

endfunction

assign sum = mult(opa, opb) + out;

always @(posedge clk or posedge clr)

begin

if (clr)

out <= 0;

else

out <= sum;

end

endmodule

测试代码：

module tb\_mac();

reg [7:0] opa, opb;

reg clk, clr;

wire [15:0] out;

parameter DELAY = 100;

mac u1(.clk(clk), .clr(clr), .opa(opa), .opb(opb), .out(out));

initial begin

clk = 1;

clr = 1;

opa = 8'd0;

opb = 8'd0;

#DELAY clr = 0; opa = 8'd1; opb = 8'd5;

#DELAY clr = 0; opa = 8'd4; opb = 8'd7;

#DELAY clr = 0; opa = 8'd12; opb = 8'd12;

#DELAY clr = 0; opa = 8'd9; opb = 8'd13;

#DELAY $stop;

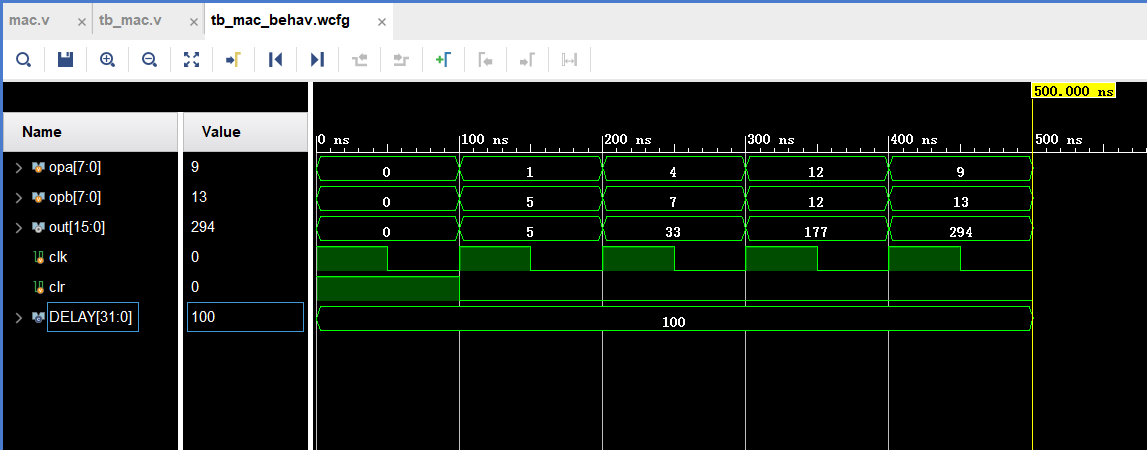
end

always #(50) clk = ~clk;

initial $monitor($time,,, "clr = %d opa = %d opb = %d out = %d", clr, opa, opb, out);

endmodule

仿真结果：



**移位相加乘法器**

代码1：（有问题）

module mult\_shift\_add #(parameter WIDTH = 8) (

input [WIDTH - 1 : 0] S\_data1,

input [WIDTH - 1 : 0] S\_data2,

output reg [2 \* WIDTH - 1 : 0] F\_mult

);

reg [2 \* WIDTH - 1 : 0] temp;

//wire [WIDTH - 1 : 0] S\_data1, S\_data2;

reg [2 \* WIDTH - 1 : 0] S\_data2\_temp;

function [15:0] mult;

input [WIDTH - 1 : 0] S\_data1, S\_data2;

integer index;

reg [2 \* WIDTH - 1 : 0] result;

begin

// F\_mult = 0;

S\_data2\_temp = {{WIDTH{1'b0}}, S\_data2}; //Expand

for (index = 0; index < WIDTH; index = index + 1)

begin

temp = {2 \* WIDTH{S\_data1[index]}} & {S\_data2\_temp};

result = result + temp << index;

$monitor("index = %d", index);

end

// $monitor("S\_data1 = %d", S\_data1);

// $monitor("S\_data2 = %d", S\_data2\_temp);

mult = result;

end

endfunction

always @(\*) F\_mult = mult(S\_data1, S\_data2);

endmodule

代码2：（依然有问题）

module mult\_shift\_add #(parameter WIDTH = 8) (

input [WIDTH - 1 : 0] S\_data1,

input [WIDTH - 1 : 0] S\_data2,

output reg [2 \* WIDTH - 1 : 0] F\_mult

);

reg [2 \* WIDTH - 1 : 0] temp;

//wire [WIDTH - 1 : 0] S\_data1, S\_data2;

reg [2 \* WIDTH - 1 : 0] S\_data2\_temp;

integer index;

reg [2 \* WIDTH - 1 : 0] result;

always @(\*)

begin

F\_mult = 0;

S\_data2\_temp = {{WIDTH{1'b0}}, S\_data2}; //Expand

for (index = 0; index < WIDTH; index = index + 1)

begin

temp = {2 \* WIDTH{S\_data1[index]}} & {S\_data2\_temp};

result = result + temp << index;

$monitor("index = %d", index);

end

// $monitor("S\_data1 = %d", S\_data1);

// $monitor("S\_data2 = %d", S\_data2\_temp);

end

endmodule

测试代码：（应该没有问题）

module tb\_mult\_shift\_add();

parameter WIDTH = 8;

reg clk;

reg [WIDTH - 1 : 0] S\_data1,S\_data2;

wire [2 \* WIDTH - 1 : 0] F\_mult;

initial begin

clk = 0;

S\_data1 = 0;

S\_data2 = 0;

#10

S\_data1 = 2;

S\_data2 = 6;

#10

S\_data1 = 7;

S\_data2 = 9;

#10

S\_data1 = 17;

S\_data2 = 12;

#10 $stop;

// #5 repeat (5) @(posedge clk)

// begin

// S\_data1 <= ($random);

// S\_data2 <= ($random);

// end

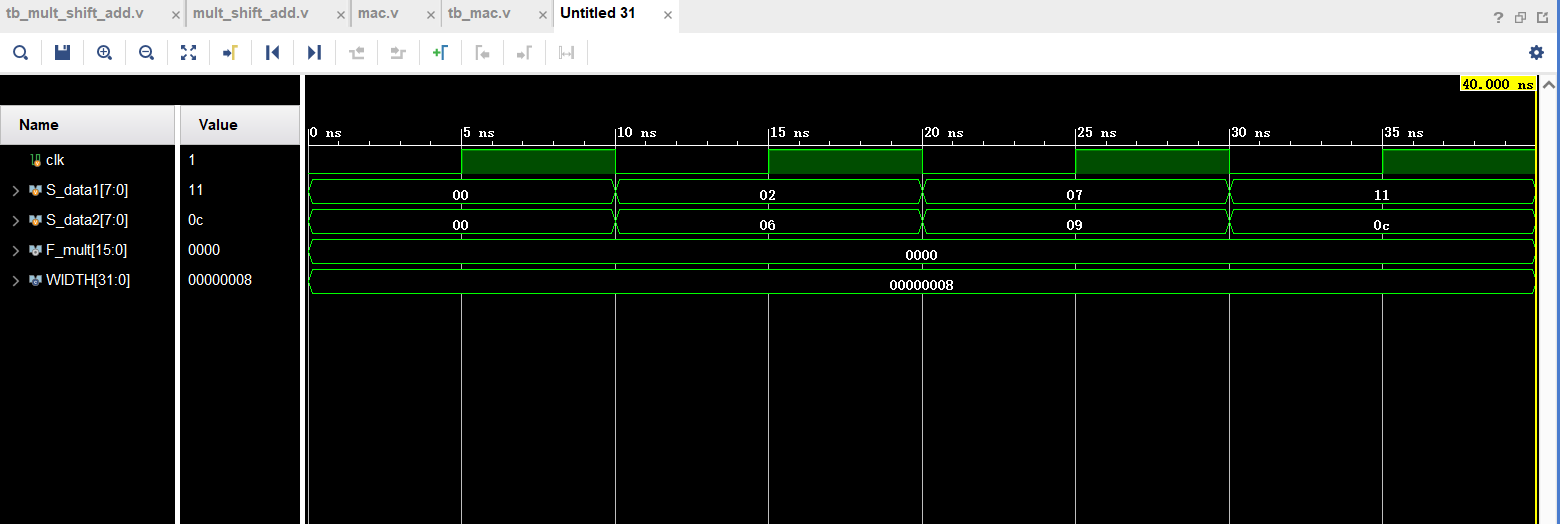
end

always #5 clk = ~clk;

mult\_shift\_add #(WIDTH) mult\_shift\_adder(S\_data1, S\_data2, F\_mult);

endmodule

仿真结果：（显然也是不对的）



Booth乘法器（一位补码乘法）

代码：（原理理解，代码实现不了，这个是参考网上的一份代码）

module mult\_comp(X, Y, out);

input [5:0] X, Y;

output reg [9:0] out;

reg [5:0] a, b, c;

reg [3:0] n;

reg p, q;

always @(X, Y) begin

if (X == 0 || Y == 0) out <= 0;

else begin

a = 6'b0;

n = 4'b1111;

p = 1'b1;

q = 1'b0;

b = X;

c = Y;

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 / 2;

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;

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

end

end

endmodule