**移位相加乘法器（Debug后）**

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;**//需置零，否则是XXX0**

// result = 0;

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

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

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

begin

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

F\_mult = F\_mult + (temp << index);**//移位运算优先级比+-\*/低，要加括号**

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

$monitor("F\_mult = %d", F\_mult);

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

$monitor("2 \* WIDTH{S\_data1[index]} %d", {2 \* WIDTH{S\_data1[0]}});

$monitor("1111 & 1000 = %d", 4'b1111 & 4'b1000);

end

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

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

end

endmodule

Testbench：

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

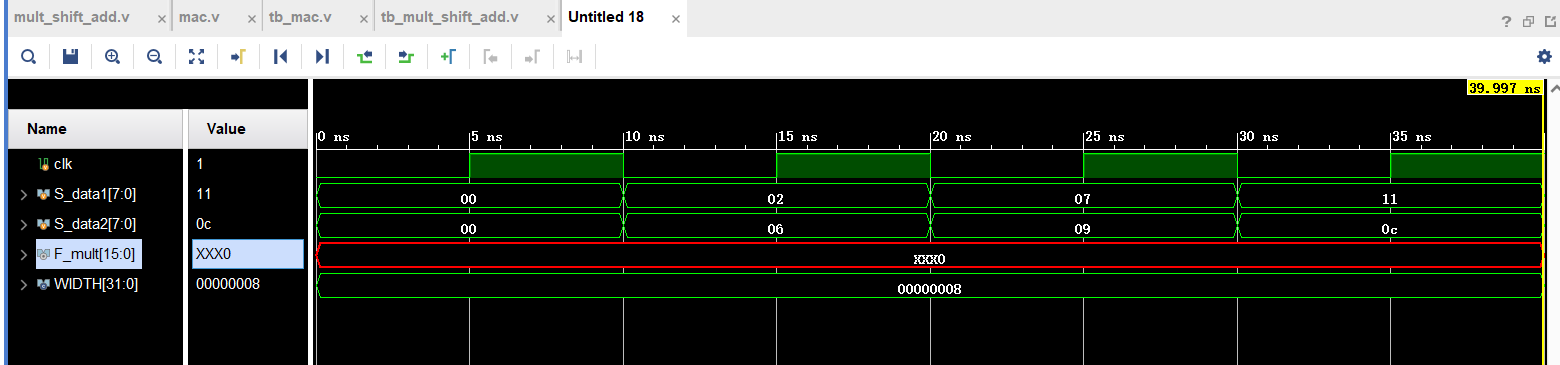
end

always #5 clk = ~clk;

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

endmodule

仿真：



Debug后的仿真：

