```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 08/27/2016 10:10:41 PM
// Design Name:
// Module Name: KOA
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module KOA_1
#(parameter SW = 24)
  // #(parameter SW = 54)
   (
   input wire [SW-1:0] Data_A_i, //lenght = SW
   input wire [SW-1:0] Data_B_i, //lenght = SW
   output wire [2*SW-1:0] sgf_result_o  //lenght = 2*SW
   );



   //wire [SW-1:0] result_left_mult;
   //wire [2*(SW/2+1)-1:0] result_right_mult;
   wire [SW/2+1:0] result_A_adder;
   //wire [SW/2+1:0] Q_result_A_adder;
   wire [SW/2+1:0] result_B_adder;
   //wire [SW/2+1:0] Q_result_B_adder;
   //wire [2*(SW/2+2)-1:0] result_middle_mult;

   wire [SW-1:0] Q_left;
   wire [2*(SW/2+1)-1:0] Q_right;
   wire [2*(SW/2+2)-1:0] Q_middle;

   wire [2*(SW/2+2)-1:0] S_A;
   wire [2*(SW/2+2)-1:0] S_B;

   wire [4*(SW/2)+2:0] Result;
   ///////////////////////////////////////////////////////////
   wire [1:0] zero1;
   wire [3:0] zero2;
   assign zero1 =2'b00;
```

```
    assign zero2 =4'b0000;
    ///////////////////////////////////////////////////////////
    wire [SW/2-1:0] rightside1;
    wire [SW/2-1:0] leftside1;

    wire [SW/2:0] rightside2;

    wire [4*(SW/2)-1:0] sgf_r;

    assign rightside1 = (SW/2) *1'b0;
    assign leftside1 = (SW/2) *1'b0;

    assign rightside2 = (SW/2+1)*1'b0;

    localparam half = SW/2;
    localparam full_port = SW - 1;
    //localparam level1=4;
    //localparam level2=5;

    ////////////////////////////////////
 generate
    case (SW%2)
        0:begin
        //////////////////////////////////even////////////////////////////////
       //Multiplier for left side and right side


 //       wire [SW-1:0] Q_left;


           multiplier_C #(.W(half)/*,.level(level1)*/) left(
               .Data_A_i(Data_A_i[full_port:half]), //Port width is SW/2
               .Data_B_i(Data_B_i[full_port:half]), //Port width is SW/2
               .Data_S_o(Q_left)                /*result_left_mult[SW-1:0]*/
           );


 //       wire [2*(SW/2+1)-1:0] Q_right;

           multiplier_C #(.W(half)/*,.level(level1)*/) right(
               .Data_A_i(Data_A_i[half-1:0]),   //Port width is SW/2
               .Data_B_i(Data_B_i[half-1:0]),    //Port width is SW/2
               .Data_S_o(/*result_right_mult[SW-1:0]*/Q_right[SW-1:0])   //Port width is SW
           );

          //Adders for middle

           adder #(.W(SW/2)) A_operation (
               .Data_A_i(Data_A_i[SW-1:SW/2]),  //Port width is SW/2
               .Data_B_i(Data_A_i[SW/2-1:0]),   //Port width is SW/2
               .Data_S_o(result_A_adder[SW/2:0])   //Port width is SW
           );

           adder #(.W(SW/2)) B_operation (
               .Data_A_i(Data_B_i[SW-1:SW/2]),   //Port width is SW/2
               .Data_B_i(Data_B_i[SW/2-1:0]),  //Port width is SW/2
               .Data_S_o(result_B_adder[SW/2:0])   //Port width is SW+1
           );


//       wire [2*(SW/2+2)-1:0] Q_middle;

           multiplier_C #(.W(SW/2+1)/*,.level(level1)*/) middle (
               .Data_A_i(/*Q_result_A_adder[SW/2:0]*/result_A_adder[SW/2:0]), //Port width is SW/2+1
               .Data_B_i(/*Q_result_B_adder[SW/2:0]*/result_B_adder[SW/2:0]), //Port width is SW/2+1
               .Data_S_o(/*result_middle_mult[SW+1:0]*/Q_middle[SW+1:0]) //Port width is SW+2
           );

           //Recordar que:
//             ///////////////////////////////////////////////////////////
//            wire [1:0] zero1;
//            wire [3:0] zero2;
//            assign zero1 =2'b00;
//            assign zero2 =4'b0000;
//            ///////////////////////////////////////////////////////////

//       wire [SW-1:0] Q_left;
//       wire [2*(SW/2+1)-1:0] Q_right;
//       wire [2*(SW/2+2)-1:0] Q_middle;

           substractor #(.W(SW+2)) Subtr_1 (
               .Data_A_i(Q_middle[SW+1:0]/*P=SW+2*/),  //Port width is SW+2 result_middle_mult//*
               .Data_B_i({zero1/*P=2*/, Q_left/*P=SW*/}),    //Port width is SW+2 result_left_mult
               .Data_S_o(S_A[SW+1:0])
           );

//        wire [2*(SW/2+1)-1:0] Q_right;
    //    wire [1:0] zero1;
    //    wire [3:0] zero2;
    //    assign zero1 =2'b00;
    //    assign zero2 =4'b0000;

           substractor #(.W(SW+2)) Subtr_2 (
               .Data_A_i(S_A[SW+1:0]/*P=SW+2*/),
               .Data_B_i({zero1/*P=2*/,Q_right[SW-1:0]/*P=SW*/}), //result_right_mult
               .Data_S_o(S_B[SW+1:0]/*P=2*/)
           );

//
//       tambien tomar en cuenta que:
```

```
//         assign rightside1 = (SW/2) *1'b0;
//        assign rightside2 = (SW/2+1)*1'b0;
//        assign leftside1 = (SW/2-2) *1'b0;


          //Final adder of lenght 2*SW
           adder #(.W(2*SW)) Final(
               .Data_A_i({Q_left,Q_right[SW-1:0]}
               /*result_left_mult[SW-1:0],result_right_mult[SW-1:0]*/),
               .Data_B_i({leftside1,S_B[SW+1:0],rightside1}),
               //Rellenamos con ceros el resto de el bus.
               .Data_S_o(Result[2*SW:0])  //Output Port lenght 2*SW+1
           );

       assign sgf_result_o = Result[(2*SW):0];

       end
    1:begin
       //////////////////////////////////odd//////////////////////////////////
             //Multiplier for left side and right side
       //Multiplier for left side and right side

          multiplier_C #(.W(SW/2)/*,.level(level2)*/) left(
                       .Data_A_i(Data_A_i[SW-1:SW/2]),
                       .Data_B_i(Data_B_i[SW-1:SW/2]),
                       .Data_S_o(/*result_left_mult*/Q_left)
           );

           multiplier_C #(.W((SW/2)+1)/*,.level(level2)*/) right(
               .Data_A_i(Data_A_i[SW/2-1:0]),
               .Data_B_i(Data_B_i[SW/2-1:0]),
               .Data_S_o(/*result_right_mult*/Q_right)
           );


          //Adders for middle

           adder #(.W(SW/2+1)) A_operation (
               .Data_A_i({1'b0,Data_A_i[SW-1:SW-SW/2]}),
               .Data_B_i(Data_A_i[SW/2-1:0]),
               .Data_S_o(result_A_adder)
           );

           adder #(.W(SW/2+1)) B_operation (
               .Data_A_i({1'b0,Data_B_i[SW-1:SW-SW/2]}),
               .Data_B_i(Data_B_i[SW/2-1:0]),
               .Data_S_o(result_B_adder)
           );


           multiplier_C #(.W(SW/2+2)/*,.level(level2)*/) middle (

               .Data_A_i(/*Q_result_A_adder*/result_A_adder),
               .Data_B_i(/*Q_result_B_adder*/result_B_adder),
               .Data_S_o(/*result_middle_mult*/Q_middle)
           );


          ///Subtractors for middle

           substractor #(.W(2*(SW/2+2))) Subtr_1 (
               .Data_A_i(/*result_middle_mult//*/Q_middle),
               .Data_B_i({zero2, /*result_left_mult//*/Q_left}),
               .Data_S_o(S_A)
           );

           substractor #(.W(2*(SW/2+2))) Subtr_2 (
               .Data_A_i(S_A),
               .Data_B_i({zero1, /*result_right_mult//*/Q_right}),
               .Data_S_o(S_B)
           );

          //Final adder

           adder #(.W(4*(SW/2)+2)) Final(
               .Data_A_i({/*result_left_mult,result_right_mult*/Q_left,Q_right}),
               .Data_B_i({S_B,rightside2}),
               .Data_S_o(Result[4*(SW/2)+2:0])
           );

          //Final assignation
          assign sgf_result_o = Result[2*SW-1:0];

       end
   endcase
endgenerate

endmodule


```