

```
1  `timescale 1ns / 1ps
2  ///////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 08/27/2016 10:10:41 PM
7  // Design Name:
8  // Module Name: K0A
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module K0A_1
24 # (parameter SW = 24)
25   // #(parameter SW = 54)
26   (
27     input wire [SW-1:0] Data_A_i, //lenght = SW
28     input wire [SW-1:0] Data_B_i, //lenght = SW
29     output wire [2*SW-1:0] sgf_result_o //lenght = 2*SW
30   );
31
32
33
34   //wire [SW-1:0] result_left_mult;
35   //wire [2*(SW/2+1)-1:0] result_right_mult;
36   wire [SW/2+1:0] result_A_adder;
37   //wire [SW/2+1:0] Q_result_A_adder;
38   wire [SW/2+1:0] result_B_adder;
39   //wire [SW/2+1:0] Q_result_B_adder;
40   //wire [2*(SW/2+2)-1:0] result_middle_mult;
41
42   wire [SW-1:0] Q_left;
43   wire [2*(SW/2+1)-1:0] Q_right;
44   wire [2*(SW/2+2)-1:0] Q_middle;
45
46   wire [2*(SW/2+2)-1:0] S_A;
47   wire [2*(SW/2+2)-1:0] S_B;
48
49   wire [4*(SW/2)+2:0] Result;
50   ///////////////////////////////////////////////////////////////////
51   wire [1:0] zero1;
52   wire [3:0] zero2;
53   assign zero1 = 2'b00;
```

```

54    assign zero2 =4'b0000;
55    ///////////////////////////////////////////////////
56    wire [SW/2-1:0] rightside1;
57    wire [SW/2-1:0] leftside1;
58
59    wire [SW/2:0] rightside2;
60
61    wire [4*(SW/2)-1:0] sgf_r;
62
63    assign rightside1 = (SW/2) *1'b0;
64    assign leftside1 = (SW/2) *1'b0;
65
66    assign rightside2 = (SW/2+1)*1'b0;
67
68    localparam half = SW/2;
69    localparam full_port = SW - 1;
70    //localparam level1=4;
71    //localparam level2=5;
72
73    ///////////////////////////////////
74    generate
75        case (SW%2)
76            0:begin
77                //////////////////////////////////////////////////even/////////////////////////////////
78                //Multiplier for left side and right side
79
80
81            //    wire [SW-1:0] Q_left;
82
83
84                multiplier_C #(.W(half)*,.level(level1)*/) left(
85                    .Data_A_i(Data_A_i[full_port:half]), //Port width is SW/2
86                    .Data_B_i(Data_B_i[full_port:half]), //Port width is SW/2
87                    .Data_S_o(Q_left)                      /*result_left_mult[SW-1:0]*/
88                );
89
90
91            //    wire [2*(SW/2+1)-1:0] Q_right;
92
93                multiplier_C #(.W(half)*,.level(level1)*/) right(
94                    .Data_A_i(Data_A_i[half-1:0]), //Port width is SW/2
95                    .Data_B_i(Data_B_i[half-1:0]), //Port width is SW/2
96                    .Data_S_o(/*result_right_mult[SW-1:0]*/Q_right[SW-1:0]) //Port
97                        width is SW
98                );
99
100            //Adders for middle
101
102            adder #(.W(SW/2)) A_operation (
103                .Data_A_i(Data_A_i[SW-1:SW/2]), //Port width is SW/2
104                .Data_B_i(Data_A_i[SW/2-1:0]), //Port width is SW/2
105                .Data_S_o(result_A_adder[SW/2:0]) //Port width is SW

```

```

106
107         adder #(.W(SW/2)) B_operation (
108             .Data_A_i(Data_B_i[SW-1:SW/2]), //Port width is SW/2
109             .Data_B_i(Data_B_i[SW/2-1:0]), //Port width is SW/2
110             .Data_S_o(result_B_adder[SW/2:0]) //Port width is SW+1
111         );
112
113
114 //      wire [2*(SW/2+2)-1:0] Q_middle;
115
116         multiplier_C #(.W(SW/2+1)*, .level(level1)* ) middle (
117             .Data_A_i(*Q_result_A_adder[SW/2:0]*/result_A_adder[SW/2:0]), //Port width is SW/2+1
118             .Data_B_i(*Q_result_B_adder[SW/2:0]*/result_B_adder[SW/2:0]), //Port width is SW/2+1
119             .Data_S_o(*result_middle_mult[SW+1:0]*/Q_middle[SW+1:0]) //Port width is SW+2
120         );
121
122         //Recordar que:
123 //      //////////////////////////////////////////
124 //      wire [1:0] zero1;
125 //      wire [3:0] zero2;
126 //      assign zero1 =2'b00;
127 //      assign zero2 =4'b0000;
128 //      //////////////////////////////////////////
129
130 //      wire [SW-1:0] Q_left;
131 //      wire [2*(SW/2+1)-1:0] Q_right;
132 //      wire [2*(SW/2+2)-1:0] Q_middle;
133
134         subtractor #(.W(SW+2)) Subtr_1 (
135             .Data_A_i(Q_middle[SW+1:0]*P=SW+2*), //Port width is SW+2
136             .Data_B_i({zero1/*P=2*/, Q_left/*P=SW*/}), //Port width is SW+2
137             .Data_S_o(S_A[SW+1:0])
138         );
139
140 //      wire [2*(SW/2+1)-1:0] Q_right;
141 //      wire [1:0] zero1;
142 //      wire [3:0] zero2;
143 //      assign zero1 =2'b00;
144 //      assign zero2 =4'b0000;
145
146         subtractor #(.W(SW+2)) Subtr_2 (
147             .Data_A_i(S_A[SW+1:0]*P=SW+2*),
148             .Data_B_i({zero1/*P=2*/, Q_right[SW-1:0]*P=SW*}), //result_right_mult
149             .Data_S_o(S_B[SW+1:0]*P=2*)
150         );
151
152 //
153 //      tambien tomar en cuenta que:

```

```

154 //      assign rightside1 = (SW/2) *1'b0;
155 //      assign rightside2 = (SW/2+1)*1'b0;
156 //      assign leftside1 = (SW/2-2) *1'b0;
157
158
159 //Final adder of lenght 2*SW
160 adder #(.W(2*SW)) Final(
161     .Data_A_i({Q_left,Q_right[SW-1:0]}
162     /*result_left_mult[SW-1:0],result_right_mult[SW-1:0]*/),
163     .Data_B_i({leftside1,S_B[SW+1:0],rightside1}),
164     //Rellenamos con ceros el resto de el bus.
165     .Data_S_o(Result[2*SW:0]) //Output Port lenght 2*SW+1
166 );
167
168 assign sgf_result_o = Result[(2*SW):0];
169
170 end
171 1:begin
172 //////////////////////////////////////////////////odd////////////////////////////////////
173 //Multiplier for left side and right side
174 //Multiplier for left side and right side
175 multiplier_C #(.W(SW/2)/*,.level(level2)*/) left(
176     .Data_A_i(Data_A_i[SW-1:SW/2]),
177     .Data_B_i(Data_B_i[SW-1:SW/2]),
178     .Data_S_o(/*result_left_mult*/Q_left)
179 );
180
181 multiplier_C #(.W((SW/2)+1)/*,.level(level2)*/) right(
182     .Data_A_i(Data_A_i[SW/2-1:0]),
183     .Data_B_i(Data_B_i[SW/2-1:0]),
184     .Data_S_o(/*result_right_mult*/Q_right)
185 );
186
187
188 //Adders for middle
189
190 adder #(.W(SW/2+1)) A_operation (
191     .Data_A_i({1'b0,Data_A_i[SW-1:SW-SW/2]}),
192     .Data_B_i(Data_A_i[SW/2-1:0]),
193     .Data_S_o(result_A_adder)
194 );
195
196 adder #(.W(SW/2+1)) B_operation (
197     .Data_A_i({1'b0,Data_B_i[SW-1:SW-SW/2]}),
198     .Data_B_i(Data_B_i[SW/2-1:0]),
199     .Data_S_o(result_B_adder)
200 );
201
202
203 multiplier_C #(.W(SW/2+2)/*,.level(level2)*/) middle (
204
205     .Data_A_i(/*Q_result_A_adder*/result_A_adder),

```

```
206         .Data_B_i(/*Q_result_B_adder*/result_B_adder),
207         .Data_S_o(/*result_middle_mult*/Q_middle)
208     );
209
210
211     ///Subtractors for middle
212
213     subtractor #(.W(2*(SW/2+2))) Subtr_1 (
214         .Data_A_i(/*result_middle_mult*/Q_middle),
215         .Data_B_i({zero2, /*result_left_mult*/Q_left}),
216         .Data_S_o(S_A)
217     );
218
219     subtractor #(.W(2*(SW/2+2))) Subtr_2 (
220         .Data_A_i(S_A),
221         .Data_B_i({zero1, /*result_right_mult*/Q_right}),
222         .Data_S_o(S_B)
223     );
224
225     //Final adder
226
227     adder #(.W(4*(SW/2)+2)) Final(
228         .Data_A_i({/*result_left_mult,result_right_mult*/Q_left,Q_right}),
229         .Data_B_i({S_B,rightside2}),
230         .Data_S_o(Result[4*(SW/2)+2:0])
231     );
232
233     //Final assignation
234     assign sgf_result_o = Result[2*SW-1:0];
235
236     end
237     endcase
238 endgenerate
239
240 endmodule
241
242
```