```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 23.02.2016 13:19:49
7  // Design Name:
8  // Module Name: CORDIC_FSM
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 CORDIC_FSM
24 (
25 //Input Signals
26 input wire clk,                                    //  Reloj del sitema.
27 input wire reset,                                  //  Reset del sitema.
28 input wire beg_FSM_CORDIC,                         //  Señal de inicio de la     ↵
   maquina de estados.
29 input wire ACK_FSM_CORDIC,                        //  Señal proveniente del     ↵
   modulo que recibe el resultado, indicado que el dato ha sido recibido.
30 input wire operation,                            //  Señal que determina si    ↵
   lo que se requiere es realizar un coseno(1´b0) o seno (1'b1).
31 input wire [1:0] shift_region_flag,              //  Señal que indica si el    ↵
   angulo a calcular se encuentra fuera del rango de calculo del algoritmo CORDIC.
32 input wire [1:0] cont_var,                       //  Señal que indica cual     ↵
   varible se va a calcular. Proveniente del contador de variables.
33 input wire ready_add_subt,                       //  Señal proveniente del     ↵
   módulo de suma/resta, indica que se ha terminado la operacion y que se puede  ↵
   disponer del resultado de dicho modulo.
34 input wire max_tick_iter, min_tick_iter,         //  Señales que indican la    ↵
   maxima y minima cuenta, respectivamente, en el contador de iteraciones.
35 input wire max_tick_var, min_tick_var,           //  Señales que indican la    ↵
   maxima y minima cuenta, respectivamente, en el contador de variables.
36
37 //Output Signals
38 output reg ready_CORDIC,                         //  Señal que indica que el   ↵
   calculo CORDIC se ha terminado.
39 output reg beg_add_subt,                         //  Señal que indica al       ↵
   modulo de suma/resta que inicie su operacion.
40 output reg ack_add_subt,                        //  Señal que le indica al    ↵
   modulo de suma/resta que se ha recibido exitosamente el resultado que este entrega.
41 output reg sel_mux_1, sel_mux_3,                //  Señales de seleccion de   ↵
```

```
    mux, la primera escoge el canal 0 si es la primera iteracion, en otro caso escoge el canal 1, y la segunda escoge cual variable (X o Y) debe aparecer a la salida.
42  output reg [1:0] sel_mux_2,                          //  Señal de seleccion de mux, que escoge entre X, Y o Z dependiendo de cual variable se deba calcular en ese momento.
43  output reg mode,                                     //  1'b0 si el modo es rotacion(signo de Y), 1'b1 si el modo es vectorizacion(signo de Z).
44  output reg enab_cont_iter, load_cont_iter,           //  Señales de habilitacion y carga, respectivamente, en el contador de iteraciones.
45  output reg enab_cont_var,  load_cont_var,            //  Señales de habilitacion y carga, respectivamente, en el contador de variables.
46  output reg enab_RB1, enab_RB2,                       //  Señales de habilitacion para los registros de variables de entrada y para los valores de las variables despues de los primeros mux, respectivamente.
47  output reg enab_d_ff_Xn, enab_d_ff_Yn, enab_d_ff_Zn,    //  Señales de habilitacion para los registros que guardan los resultados de cada variable en cada iteracion provenientes del modulo de suma/resta.
48  output reg enab_dff5, enab_d_ff_out,                 //  Señales de habilitacion para los registros en la salida, el primero antes del cambio de signo y el segundo es el que se encuentra en la salida.
49  output reg enab_dff_shifted_x, enab_dff_shifted_y,   //  Señales de habilitacion para los registros que guardan el valor de las variables X y Y luego de realizarles los desplazamientos.
50  output reg enab_dff_LUT, enab_dff_sign               //  Señales de habilitacion para los registros  que guardan los valores provenientes de la look-up table y del signo, respectivamente.
51  );
52
53  //symbolic state declaration
54  localparam [3:0]    est0 = 4'b0000,
55                      est1 = 4'b0001,
56                      est2 = 4'b0010,
57                      est3 = 4'b0011,
58                      est4 = 4'b0100,
59                      est5 = 4'b0101,
60                      est6 = 4'b0110,
61                      est7 = 4'b0111,
62                      est8 = 4'b1000,
63                      est9 = 4'b1001,
64                      est10 = 4'b1010,
65                      est11 = 4'b1011;
66
67  //signal declaration
68  reg state_reg, state_next;  //  Guardan el estado actual y el estado futuro, respectivamente.
69
70  //state register
71
72  always @(clk, reset)
73      begin
74          if(reset)   // Si hay reset, el estado actual es el estado inicial.
75              state_reg <= est0;
76          else        //Si no hay reset el estado actual es igual al estado siguiente.
```

```
77              state_reg <= state_next;
78      end
79
80  //next-state logic and output logic
81
82  always@*
83      begin
84      state_next = state_reg; // default state : the same
85
86      //declaration of default outputs.
87      ready_CORDIC = 1'b0;
88      beg_add_subt = 1'b0;
89      ack_add_subt = 1'b0;
90      sel_mux_1 = 1'b0;
91      sel_mux_2 = 2'b10;
92      sel_mux_3 = 1'b0;
93      mode = 1'b0;
94      enab_cont_iter = 1'b0;
95      load_cont_iter = 1'b0;
96      enab_cont_var = 1'b0;
97      load_cont_var = 1'b0;
98      enab_RB1 = 1'b0;
99      enab_RB2 = 1'b0;
100     enab_d_ff_Xn = 1'b0;
101     enab_d_ff_Yn = 1'b0;
102     enab_d_ff_Zn = 1'b0;
103     enab_d_ff_out = 1'b0;
104     enab_dff_shifted_x = 1'b0;
105     enab_dff_shifted_y = 1'b0;
106     enab_dff_LUT = 1'b0;
107     enab_dff_sign = 1'b0;
108     enab_dff5 = 1'b0;
109
110         case(state_reg)
111         est0:
112         begin
113             state_next = est1;
114         end
115
116         est1:
117         begin
118             if(beg_FSM_CORDIC)
119             begin
120                 state_next = est2;
121                 enab_RB1 = 1'b1;
122                 load_cont_iter = 1'b1;
123                 load_cont_iter = 1'b1;
124             end
125             else
126                 state_next = est1;
127         end
128
129         est2:
```

```
130             begin
131                 enab_RB1 = 1'b1;
132                 state_next = est3;
133             end
134 
135             est3:
136             begin
137                 enab_RB1 = 1'b0;
138                 enab_RB2 = 1'b1;
139                 if(max_tick_iter)
140                     sel_mux_1 = 1'b0;
141                 else
142                     sel_mux_1 = 1'b1;
143                 state_next = est4;
144             end
145 
146             est4:
147             begin
148                 enab_RB2 = 1'b1;
149                 mode = 1'b0;
150                 enab_RB2 = 1'b0;
151                 enab_dff_shifted_x = 1'b1;
152                 enab_dff_shifted_y = 1'b1;
153                 enab_dff_LUT = 1'b1;
154                 enab_dff_sign = 1'b1;
155                 state_next = est5;
156             end
157 
158             est5:
159             begin
160                 enab_dff_shifted_x = 1'b1;
161                 enab_dff_shifted_y = 1'b1;
162                 enab_dff_LUT = 1'b1;
163                 enab_dff_sign = 1'b1;
164 
165                 if(min_tick_iter)
166                 begin
167                     if(operation == 1'b0)
168                     begin
169                         if(shift_region_flag == (2'b00 || 2'b11))
170                             sel_mux_2 = 2'b10;
171                         else
172                             sel_mux_2 = 2'b01;
173                     end
174 
175                     else
176                     begin
177                         if(shift_region_flag == (2'b00 || 2'b11))
178                             sel_mux_2 = 2'b01;
179                         else
180                             sel_mux_2 = 2'b10;
181                     end
182                     state_next = est7;
```

```
183                 end
184                 else
185                     state_next = est6;
186             end
187 
188             est6:
189             begin
190                 if(min_tick_var)
191                 begin
192                     enab_cont_iter = 1'b1;
193                     state_next = est3;
194                 end
195 
196                 else
197                 begin
198                     sel_mux_2 = cont_var;
199                     state_next = est7;
200                 end
201             end
202 
203             est7:
204             begin
205                 beg_add_subt = 1'b1;
206                 state_next = est8;
207             end
208 
209             est8:
210             begin
211                 if(ready_add_subt)
212                 begin
213                     if(min_tick_iter)
214                     begin
215                         if(operation == 1'b0)
216                             enab_d_ff_Xn = 1'b1;
217                         else
218                             enab_d_ff_Yn = 1'b1;
219                     end
220                     else
221                     begin
222                         if(max_tick_var)
223                             enab_d_ff_Xn = 1'b1;
224                         else if(min_tick_var)
225                             enab_d_ff_Zn = 1'b1;
226                         else
227                             enab_d_ff_Yn = 1'b1;
228                     end
229                     state_next = est9;
230                 end
231                 else
232                     state_next = est8;
233             end
234             est9:
235             begin
```

```
236             if(min_tick_iter)
237             begin
238                 if(operation == 1'b0)
239                 begin
240                     if(shift_region_flag == (2'b01 || 2'b10))
241                         sel_mux_3 = 1'b1;
242                     else
243                         sel_mux_3 = 1'b0;
244                 end
245                 else
246                 begin
247                     if(shift_region_flag == (2'b01 || 2'b10))
248                         sel_mux_3 = 1'b0;
249                     else
250                         sel_mux_3 = 1'b1;
251                 end
252                 state_next = est10;
253                 enab_dff5 = 1'b1;
254             end
255             else
256             begin
257                 enab_cont_var = 1'b1;
258                 state_next = est6;
259             end
260         end
261 
262         est10:
263         begin
264             enab_d_ff_out = 1'b1;
265             state_next = est11;
266         end
267 
268         est11:
269         begin
270             ready_CORDIC = 1'b1;
271             if(ACK_FSM_CORDIC)
272                 state_next = est0;
273             else
274                 state_next = est11;
275         end
276 
277         default : state_next = est0;
278         endcase
279     end
280 endmodule
281 
```