```
 1 %
 2 Diseño en AHDL para el cálculo del logarítmo en base 2.
 3 Recibe un número in_X de N bits de parte entera y M bits de parte
 4 decimal. Arroja como salida a out_Y que es un número en formato
 5 signo-magnitud con P+Q bits, donde P es el tamaño de la característica
 6 con signo y Q es el tamaño de la mantisa.
 7
 8 Consta de cinco estados.
 9 0)  Inicial. Aquí se arranca y aquí se termina. La señal ok indica que
10     se puede leer el dato. La señal _inf indica X=0 y log2 0 = -inf.
11     Sólo se pasa al estado uno si la señal start=1.
12 1)  Cargar. Recarga los contadores C y Q, también carga el dato X.
13 2)  Calculando característica. Decrementa C y desplaza X. Esto se hace
14     hasta que C = -(M+1) o hasta que msb(X) sea 1.
15 3)  Cuadrado de X. Guarda parte alta de X^2 en X.
16 4)  Calculando mantisa. Desplaza mantisa con el msb(X) y desplaza X si
17     msb(X)=0, también decrementa Q. Vuelve al estado tres si Q no es
18     cero, de lo contrario va al estado cero.
19     El desplazamiento de X se hace con el msb de la parte baja de X^2
20     para mejorar la precisión.
21
22 Bajo operación normal, en el mejor de los casos el algoritmo demora
23 2(Q+1) ciclos en volver al estado cero. En el peor de los casos tarda
24 2(Q+1)+(N+M) ciclos en terminar la operación.
25
26 Diseñado por:
27     Alvaro José Caicedo Beltrán
28     Ingeniería en Electrónica
29     Escuela de Ingeniería Eléctrica y Electrónica EIEE
30     Universidad del Valle. Cali - Colombia
31     Mayo de 2008
32 %
33
34 % Adaptación para N=8, M=8, Q=8;                                        %
35
36 TITLE "Logaritmo en base 2";
37
38 % Librerías de altera utilizadas - Megafunciones parametrizadas         %
39 INCLUDE "lpm_counter.inc";
40 INCLUDE "lpm_mux";
41 INCLUDE "lpm_shiftreg";
42 INCLUDE "lpm_mult";
43
44 CONSTANT N = 8;          % #bits de la parte entera de X               %
45 CONSTANT M = 8;          % #bits de la parte decimal de X              %
46 CONSTANT Q = 8;          % #bits de la mantisa                         %
47 CONSTANT P = 5;          % #bits de la característica con signo        %
48
49 % Definición de puertos de i/o                                          %
50 SUBDESIGN fsm_log
51 (
52     % Señal de reloj, reinicio, y arranque de la fsm                   %
53     clk, rst, start                    :INPUT = GND;
54     % Dato de entrada para calcularsele el logarítmo                   %
55     in_X[(N+M-1)..0]                   :INPUT = GND;
56     % Datos de salida: Característica (C2) y 0.Mantisa                 %
57     out_C[(P-1)..0],out_M[(Q-1)..0]    :OUTPUT;
```

```
    % Salida Y=log2 X (SM) y señal para -inf                        %
    out_Y[(P+Q-1)..0], _inf             :OUTPUT;
    % Estados de fsm y señal que indica listo para recalcular       %
    estados[2..0], ready                :OUTPUT;
)

VARIABLE
    % Contador de P bits descendente para la característica         %
    contador_C: lpm_counter WITH (
        LPM_WIDTH = P,
        LPM_DIRECTION = "DOWN"
    );
    % Contador modulo Q descendente para bits de mantisa            %
    contador_Q: lpm_counter WITH (
        LPM_WIDTH = LOG2(Q)+1,
        LPM_DIRECTION = "DOWN"
    );
    % Registro de desplazamiento para la mantisa                    %
    shifter_M: lpm_shiftreg WITH (
        LPM_WIDTH = Q,
        LPM_DIRECTION = "LEFT"
    );
    % Registro de desplazamiento para la variable X                 %
    shifter_X: lpm_shiftreg WITH (
        LPM_WIDTH = N+M,
        LPM_DIRECTION = "LEFT"
    );
    % Multiplexor para cargar X con valor inicial o con X^2         %
    mux_X: lpm_mux WITH (
        LPM_WIDTH = N+M,
        LPM_SIZE = 2,
        LPM_WIDTHS = 1
    );
    % Multiplicador para calcular X^2. Salida truncada para X       %
    cuadrado_X: lpm_mult WITH (
        LPM_WIDTHA = N+M,
        LPM_WIDTHB = N+M,
        LPM_WIDTHS = 1,
        LPM_WIDTHP = (N+M)+1
        % Bits más significativos de a*b + s                        %
        % Mostrando sólo los bits para X más el próximo LSB        %
    );
    msb         : NODE;         % Bit de salida del registro X      %
    q_min       : NODE;         % Q alcanza valor 0                 %
    c_min       : NODE;         % C alcanza valor -M-1              %

    ss: MACHINE
        % Señales que dependen del estado presente                  %
        OF BITS(
        dec_C,          % Controla el decremento de C               %
        load_C,         % Recargar contador C                       %
        dec_Q,          % Decrementar el contador Q                 %
        load_Q,         % Recarga del contador Q                    %
        shift_regM,     % Desplazar mantisa a la izquierda          %
        sel_sourceX,    % Cargar X con in_X o con X^2               %
        en_regX,        % Habilitar carga/desplazamiento de X       %
        loadshift_regX, % Cargar/desplazar X si está habilitado     %
```

```
115         inbit_regX, % Habilitar carga del msb de parte baja de X^2  %
116         estados[2..0],  % Codificación de estados                    %
117         ready           % Indicador de dato listo                    %
118         )
119         WITH STATES(
120             s0=B"00000X0XX0001",    % Entrega resultado              %
121             s1=B"01010011X0010",    % Recarga C, Q y X=in_X          %
122             s2=B"10000X1000100",    % Desplaza X,0 y decrementa C    %
123             s3=B"00000111X0110",    % Hacer X = X^2 (parte alta+1)   %
124             s4=B"00101X0011000"     % Desplazar regM, decrementar Q  %
125                                     % y normalizar X si MSB=0        %
126         );
127 BEGIN
128     DEFAULTS
129         ss.reset = VCC;
130     END DEFAULTS;
131
132     % Selección de fuente de carga para X                            %
133     mux_X.data[0][] = in_X[];
134     mux_X.data[1][] = cuadrado_X.result[(N+M)..1];
135     mux_X.sel[0] = sel_sourceX;
136
137     % Manejo del contador de característica                          %
138     contador_C.clock  = !clk;
139     contador_C.clk_en = dec_C;
140     contador_C.aload  = load_C;
141     contador_C.data[] = N-1;
142     out_C[] = contador_C.q[];
143
144     % Manejo del contador Q                                          %
145     contador_Q.clock  = !clk;
146     contador_Q.clk_en = dec_Q;
147     contador_Q.aload  = load_Q;
148     contador_Q.data[] = Q;
149
150     % Manejo del registro de desplazamiento X                        %
151     shifter_X.clock = !clk;
152     shifter_X.enable = en_regX # (inbit_regX & !msb);
153     shifter_X.shiftin = inbit_regX & cuadrado_X.result[0];
154     shifter_X.load = loadshift_regX;
155     shifter_X.data[] = mux_X.result[];
156     cuadrado_X.dataa[] = shifter_X.q[];
157     cuadrado_X.datab[] = shifter_X.q[];
158     msb = shifter_X.shiftout;
159
160     % Manejo del registro de desplazamiento de mantisa               %
161     shifter_M.clock = !clk;
162     shifter_M.enable = shift_regM;
163     shifter_M.shiftin = msb;
164     out_M[] = shifter_M.q[];
165
166     % Señales de control de la máquina de estados                    %
167     ss.clk = clk;
168     ss.reset = rst;
169
170     % Expresiones de las señales de control o indicadores            %
171     q_min = (contador_Q.q[] == 0);
```

```
172     c_min = (contador_C.q[] == -M-1);
173
174     % Formateando la salida (signo, característica, mantisa)        %
175     IF out_C[P-1] THEN
176         out_Y[] = !(out_C[], out_M[])+1;        % -out_C - out_M    %
177         out_Y[P+Q-1] = VCC;
178     ELSE
179         out_Y[] = (out_C[], out_M[]);
180     END IF;
181
182     _inf = c_min;
183
184     % Tabla de transición de estados                                  %
185     TABLE
186         ss, start,  q_min,  c_min,  msb =>  ss;
187         s0, 0,      x,      x,      x   =>  s0;
188         s0, 1,      x,      x,      x   =>  s1;
189         s1, x,      x,      0,      0   =>  s2;
190         s1, x,      x,      0,      1   =>  s3;
191         s1, x,      x,      1,      x   =>  s0;
192         s2, x,      x,      0,      0   =>  s2;
193         s2, x,      x,      0,      1   =>  s3;
194         s2, x,      x,      1,      x   =>  s0;
195         s3, x,      x,      x,      x   =>  s4;
196         s4, x,      0,      x,      x   =>  s3;
197         s4, x,      1,      x,      x   =>  s0;
198     END TABLE;
199 END;
```