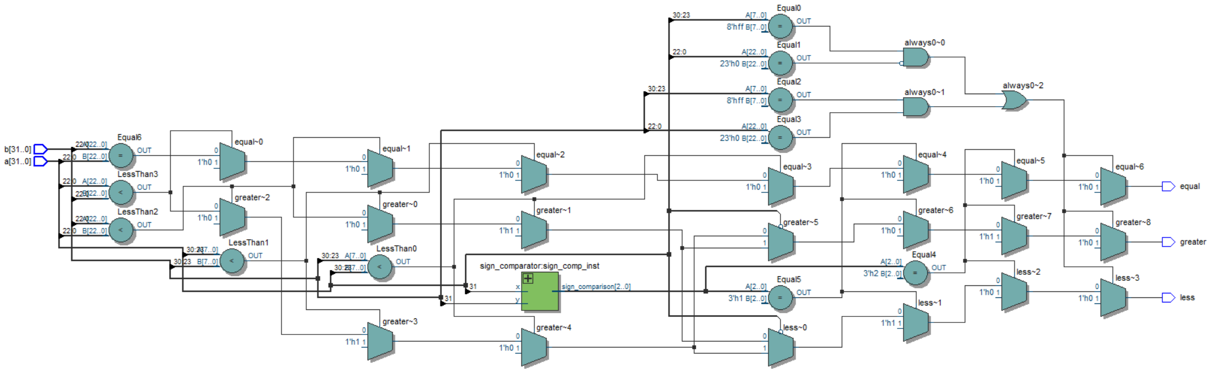
# DISEÑO EN VERILOG DE UN COMPARADOR DE NÚMEROS CON PUNTO FLOTANTE

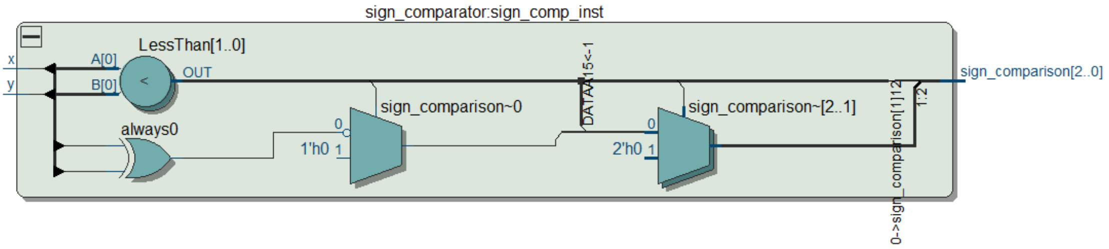
## Diagrama de bloques:

Circuito general del comparador en coma flotante:

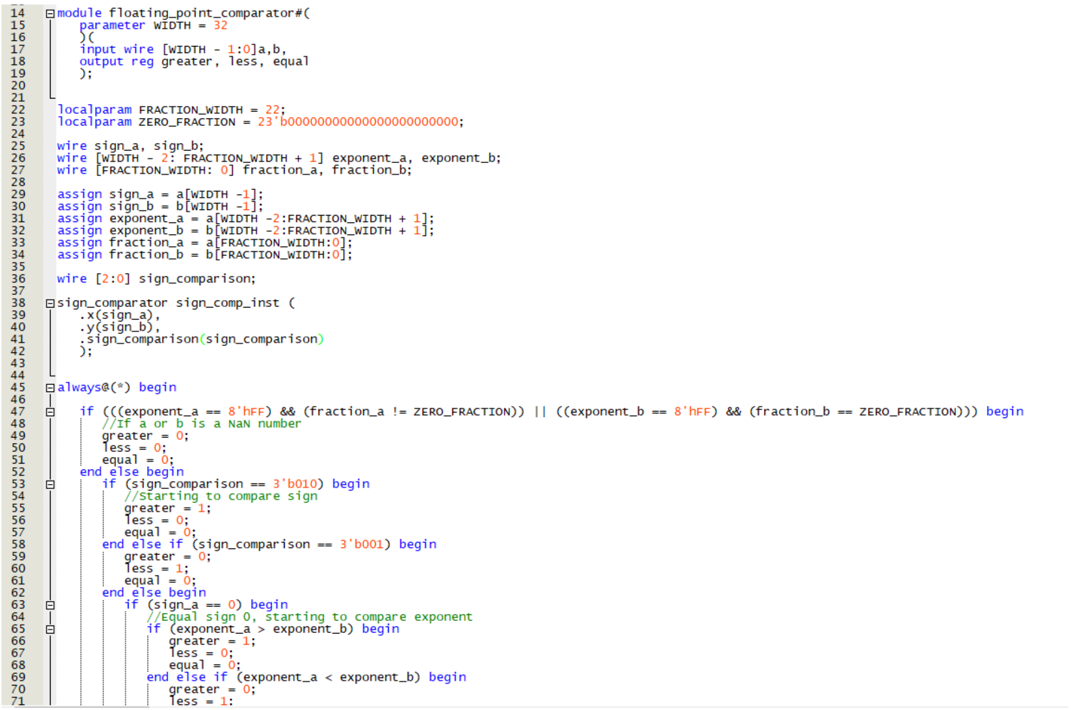
Este circuito compara el signo de dos números en el estándar IEEE 754. Primero se analiza si el circuito no corresponde a un NaN o número no existente. Los NaN no representan números en este estándar, solo sirven para representar errores u otros, por lo que no cuentan con un valor numérico para la evaluación. Luego se compara el bit más significativo para determinar si los números son positivos o negativos, posteriormente se analiza el exponente y la fracción o también llamada matinsa. Finalmente, se determina si el primer número es mayor, menor o igual al segundo número ingresado. Si la trama de bits no es un número o corresponde a un NaN se obtiene el valor de cero para todas las salidas.



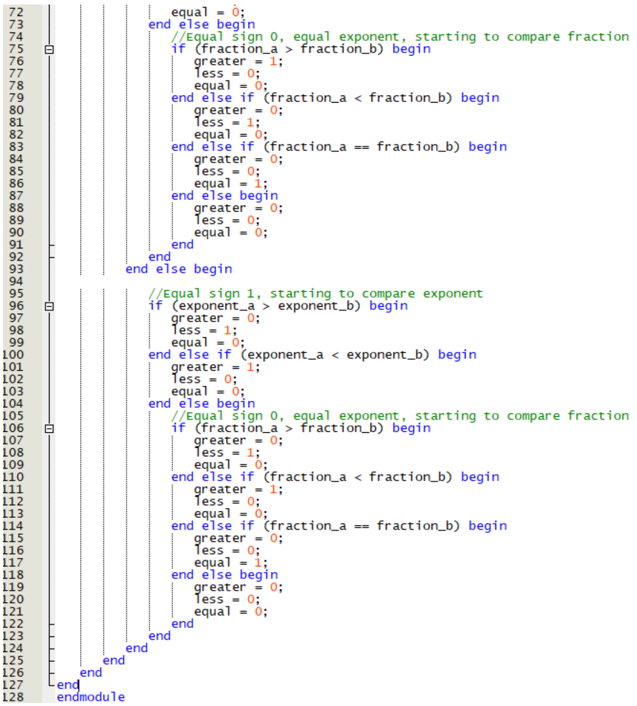
Circuito comparador de signo:



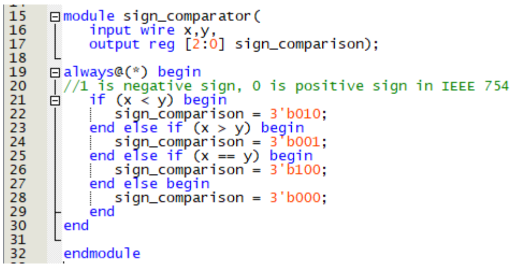
# RTL del circuito:



```
module floating_point_comparator#(
   parameter WIDTH = 32
   )(
   input wire [WIDTH - 1:0]a,b,
   output reg greater, less, equal
   );


localparam FRACTION_WIDTH = 22;
localparam ZERO_FRACTION = 23'b00000000000000000000000;

wire sign_a, sign_b;
wire [WIDTH - 2: FRACTION_WIDTH + 1] exponent_a, exponent_b;
wire [FRACTION_WIDTH: 0] fraction_a, fraction_b;

assign sign_a = a[WIDTH -1];
assign sign_b = b[WIDTH -1];
assign exponent_a = a[WIDTH -2:FRACTION_WIDTH + 1];
assign exponent_b = b[WIDTH -2:FRACTION_WIDTH + 1];
assign fraction_a = a[FRACTION_WIDTH:0];
assign fraction_b = b[FRACTION_WIDTH:0];

wire [2:0] sign_comparison;

sign_comparator sign_comp_inst (
   .x(sign_a),
   .y(sign_b),
   .sign_comparison(sign_comparison)
   );


always@(*) begin

   if (((exponent_a == 8'hFF) && (fraction_a != ZERO_FRACTION)) || ((exponent_b == 8'hFF) && (fraction_b == ZERO_FRACTION))) begin
      //If a or b is a NaN number
      greater = 0;
      less = 0;
      equal = 0;
   end else begin
      if (sign_comparison == 3'b010) begin
         //Starting to compare sign
         greater = 1;
         less = 0;
         equal = 0;
      end else if (sign_comparison == 3'b001) begin
         greater = 0;
         less = 1;
         equal = 0;
      end else begin
         if (sign_a == 0) begin
            //Equal sign 0, starting to compare exponent
            if (exponent_a > exponent_b) begin
               greater = 1;
               less = 0;
               equal = 0;
            end else if (exponent_a < exponent_b) begin
               greater = 0;
               less = 1;
```

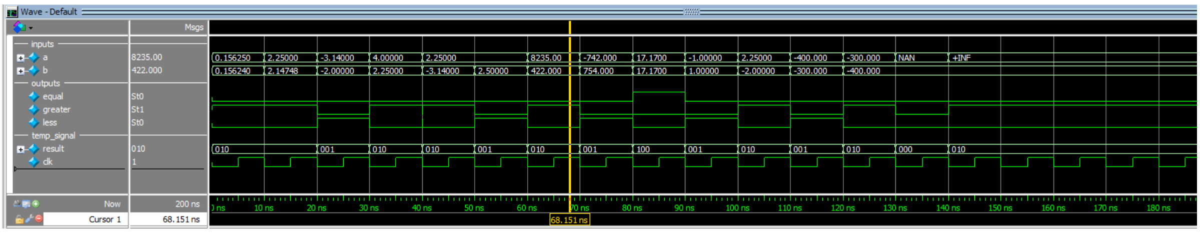


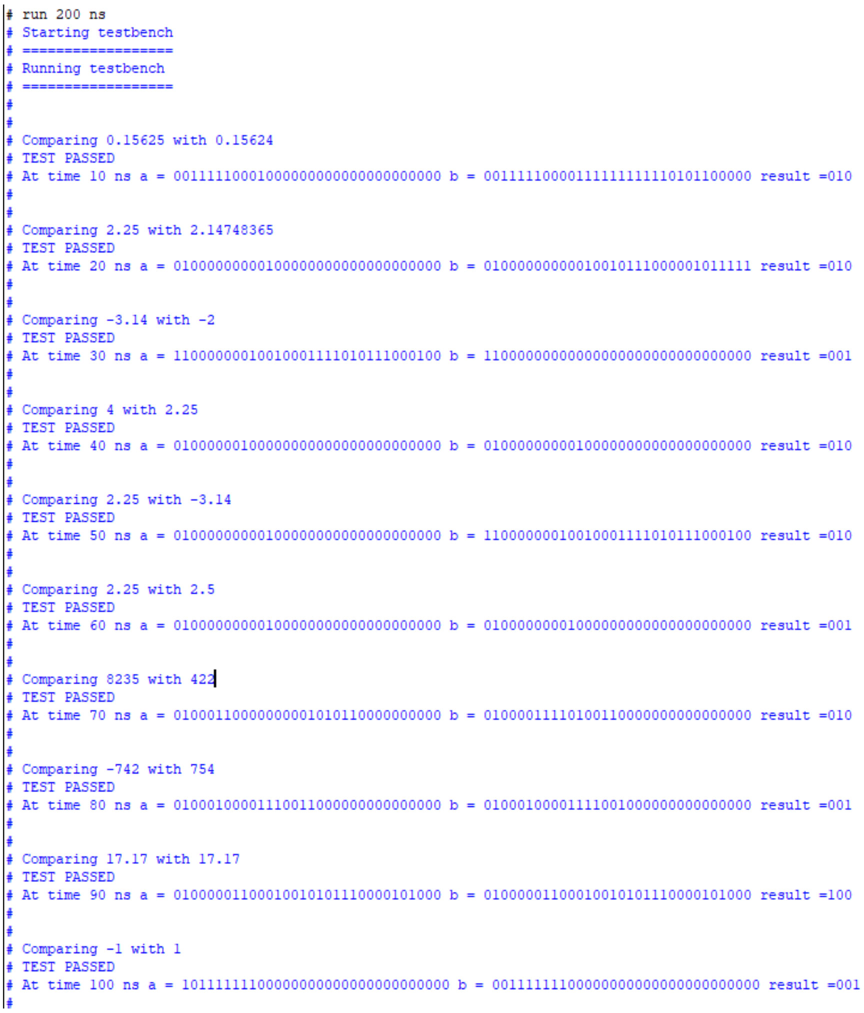
```
               equal = 0;
            end else begin
               //Equal sign 0, equal exponent, starting to compare fraction
               if (fraction_a > fraction_b) begin
                  greater = 1;
                  less = 0;
                  equal = 0;
               end else if (fraction_a < fraction_b) begin
                  greater = 0;
                  less = 1;
                  equal = 0;
               end else if (fraction_a == fraction_b) begin
                  greater = 0;
                  less = 0;
                  equal = 1;
               end else begin
                  greater = 0;
                  less = 0;
                  equal = 0;
               end
            end
         end else begin

            //Equal sign 1, starting to compare exponent
            if (exponent_a > exponent_b) begin
               greater = 0;
               less = 1;
               equal = 0;
            end else if (exponent_a < exponent_b) begin
               greater = 1;
               less = 0;
               equal = 0;
            end else begin
               //Equal sign 0, equal exponent, starting to compare fraction
               if (fraction_a > fraction_b) begin
                  greater = 0;
                  less = 1;
                  equal = 0;
               end else if (fraction_a < fraction_b) begin
                  greater = 1;
                  less = 0;
                  equal = 0;
               end else if (fraction_a == fraction_b) begin
                  greater = 0;
                  less = 0;
                  equal = 1;
               end else begin
                  greater = 0;
                  less = 0;
                  equal = 0;
               end
            end
         end
      end
   end
end
endmodule
```



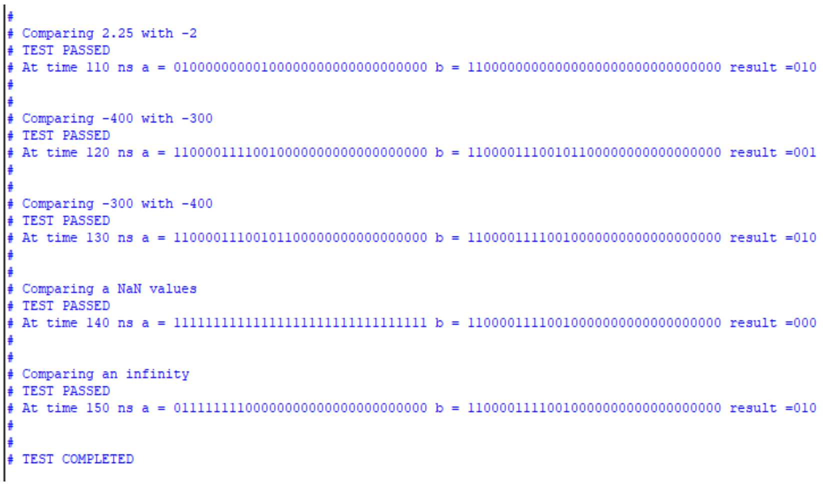
```
module sign_comparator(
   input wire x,y,
   output reg [2:0] sign_comparison);

always@(*) begin
//1 is negative sign, 0 is positive sign in IEEE 754
   if (x < y) begin
      sign_comparison = 3'b010;
   end else if (x > y) begin
      sign_comparison = 3'b001;
   end else if (x == y) begin
      sign_comparison = 3'b100;
   end else begin
      sign_comparison = 3'b000;
   end
end

endmodule
```

Simulación del circuito:





```
# run 200 ns
# Starting testbench
# ==================
# Running testbench
# ==================
#
#
# Comparing 0.15625 with 0.15624
# TEST PASSED
# At time 10 ns a = 00111110001000000000000000000000 b = 00111110000111111111110101100000 result =010
#
#
# Comparing 2.25 with 2.14748365
# TEST PASSED
# At time 20 ns a = 01000000000100000000000000000000 b = 01000000000010010111000001011111 result =010
#
#
# Comparing -3.14 with -2
# TEST PASSED
# At time 30 ns a = 11000000010010001111010111000100 b = 11000000000000000000000000000000 result =001
#
#
# Comparing 4 with 2.25
# TEST PASSED
# At time 40 ns a = 01000000100000000000000000000000 b = 01000000000100000000000000000000 result =010
#
#
# Comparing 2.25 with -3.14
# TEST PASSED
# At time 50 ns a = 01000000000100000000000000000000 b = 11000000010010001111010111000100 result =010
#
#
# Comparing 2.25 with 2.5
# TEST PASSED
# At time 60 ns a = 01000000000100000000000000000000 b = 01000000001000000000000000000000 result =001
#
#
# Comparing 8235 with 422
# TEST PASSED
# At time 70 ns a = 01000110000000001010110000000000 b = 01000011110100110000000000000000 result =010
#
#
# Comparing -742 with 754
# TEST PASSED
# At time 80 ns a = 01000100001110011000000000000000 b = 01000100001111001000000000000000 result =001
#
#
# Comparing 17.17 with 17.17
# TEST PASSED
# At time 90 ns a = 01000001100010010101110000101000 b = 01000001100010010101110000101000 result =100
#
#
# Comparing -1 with 1
# TEST PASSED
# At time 100 ns a = 10111111100000000000000000000000 b = 00111111100000000000000000000000 result =001
#
```



```
#
# Comparing 2.25 with -2
# TEST PASSED
# At time 110 ns a = 01000000000100000000000000000000 b = 11000000000000000000000000000000 result =010
#
#
# Comparing -400 with -300
# TEST PASSED
# At time 120 ns a = 11000011110010000000000000000000 b = 11000011100101100000000000000000 result =001
#
#
# Comparing -300 with -400
# TEST PASSED
# At time 130 ns a = 11000011100101100000000000000000 b = 11000011110010000000000000000000 result =010
#
#
# Comparing a NaN values
# TEST PASSED
# At time 140 ns a = 11111111111111111111111111111111 b = 11000011110010000000000000000000 result =000
#
#
# Comparing an infinity
# TEST PASSED
# At time 150 ns a = 01111111100000000000000000000000 b = 11000011110010000000000000000000 result =010
#
#
# TEST COMPLETED
```

Bibliografía:

https://www.youtube.com/watch?v=D7iVR7_PGSc

https://informatica.uv.es/seguia/FAC/Teoria/IEEE-754.htm

https://www.youtube.com/watch?v=RcRhqg4jKHo

https://medium.com/@matematicasdiscretaslibro/cap%C3%ADtulo-3-punto-flotante-c689043db98b

Material del curso Verilog – Módulo 1- Maelpro.