Combinational Logic Design: Part 6: 8-bit Hierarchical Comparator

Author Adil R.

May 2025

1 Introduction

This report presents the design and implementation of an 8-bit hierarchical comparator using Verilog HDL. The comparator extends the basic 4-bit design to handle larger numbers while maintaining efficient logic utilization. The circuit compares two 8-bit binary numbers and produces three outputs indicating their relationship (greater than, equal to, or less than).

2 Design Description

The 8-bit comparator is implemented using a hierarchical approach with two 4-bit comparator modules and additional logic for combining results.

2.1 Architecture

- Two 4-bit comparator modules compare the upper and lower nibbles
- Combination logic resolves final output based on nibble comparisons
- Three output signals: G (Greater), E (Equal), L (Less)

2.2 Truth Table

Upper Nibble (A[7:4], B[7:4])		Lower Nibble (A[3:0], B[3:0])		Output		
G	E	G	E	G	E	L
1	0	X	X	1	0	0
0	1	1	0	1	0	0
0	1	0	1	0	1	0
0	1	0	0	0	0	1
0	0	X	X	0	0	1

Table 1: 8-bit Comparator Truth Table (X = Don't Care)

2.3 Boolean Equations

The output logic can be expressed as:

$$G = G_{upper} + (E_{upper} \cdot G_{lower})$$
$$E = E_{upper} \cdot E_{lower}$$
$$L = L_{upper} + (E_{upper} \cdot L_{lower})$$

3 Verilog Implementation

3.1 4-bit Comparator Module

```verilog
`timescale 1ns / 1ps

module comp_4bit(
    input [3:0] A, B,
    output reg G, E, L
);

always @(*) begin
    G = 1'b0;
    E = 1'b0;
    L = 1'b0;

    if (A > B)
        G = 1'b1;
    else if (A == B)
        E = 1'b1;
    else
        L = 1'b1;
end

endmodule
```

3.2 8-bit Hierarchical Comparator

```verilog
`timescale 1ns / 1ps

module comp_8bit(
    input [7:0] A, B,
    output G, E, L
);

wire G_upper, E_upper, L_upper;
wire G_lower, E_lower, L_lower;

// Upper nibble comparison
comp_4bit upper_comp(
    .A(A[7:4]),
    .B(B[7:4]),
    .G(G_upper),
    .E(E_upper),
    .L(L_upper)
);

// Lower nibble comparison
compr_4bit lower_comp(
    .A(A[3:0]),
    .B(B[3:0]),
    .G(G_lower),
    .E(E_lower),
    .L(L_lower)
);

// Output logic
assign G = G_upper | (E_upper & G_lower);
assign E = E_upper & E_lower;
assign L = L_upper | (E_upper & L_lower);

endmodule
```

4 Test Bench

```
`timescale 1ns / 1ps

module tb_comp_8bit;
    reg [7:0] A, B;
    wire G, E, L;

    comp_8bit uut(
        .A(A),
        .B(B),
        .G(G),
        .E(E),
        .L(L)
    );

    initial begin
        // Test case groups
        // 1. Equal numbers
        A = 8'h00; B = 8'h00;
        #10 A = 8'hFF; B = 8'hFF;
        #10 A = 8'hA5; B = 8'hA5;

        // 2. A > B cases
        #10 A = 8'h01; B = 8'h00;  // Lower difference
        #10 A = 8'h10; B = 8'h0F;  // Upper difference
        #10 A = 8'hF1; B = 8'hF0;  // Both  different

        // 3. A < B cases
        #10 A = 8'h00; B = 8'h01;
        #10 A = 8'h0F; B = 8'h10;
        #10 A = 8'hF0; B = 8'hF1;

        // 4. Boundary cases
        #10 A = 8'h80; B = 8'h7F;  // MSB difference
        #10 A = 8'h7F; B = 8'h80;
        #10 A = 8'hFF; B = 8'h00;
        #10 A = 8'h00; B = 8'hFF;

        #20 $finish;
    end

    initial begin
        $monitor("Time = %t, A = %h, B = %h, G = %b, E = %b, L = %b",
                  $time, A, B, G, E, L);
    end
endmodule
```

5 Conclusion

- Successfully designed and implemented an 8-bit hierarchical comparator
- Testbench verified all comparison cases

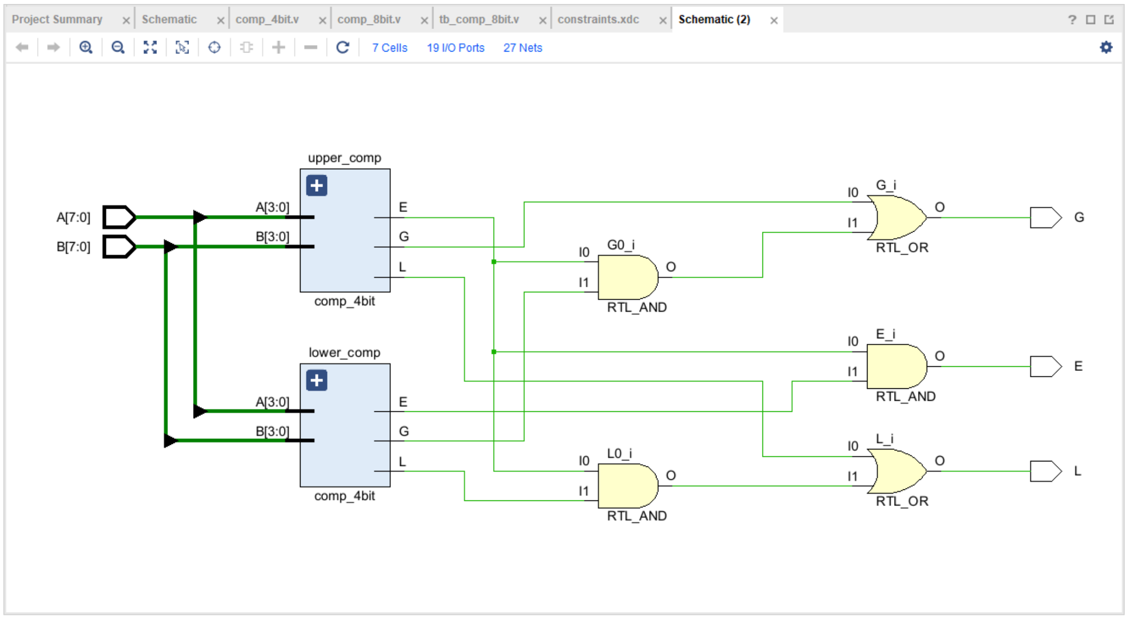

Figure 1: Hierarchical 8-bit Comparator Schematic

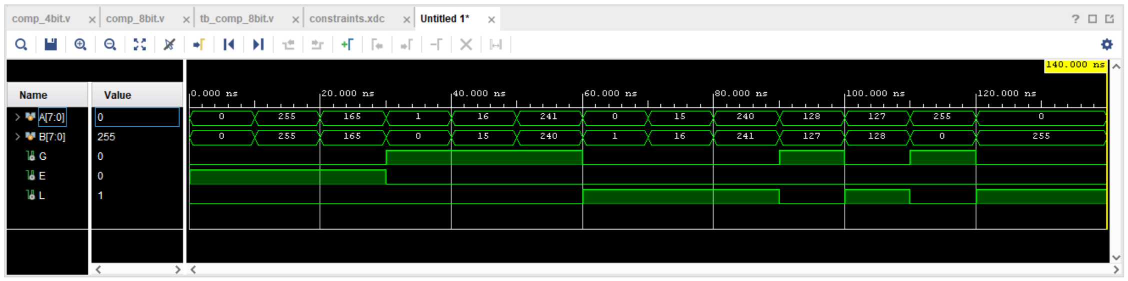

Figure 2: Simulation Waveform Showing All Comparison Cases