# Logic Design using FPGAs
# Structural Design - Comparator

**Objective:** Structural Design using Verilog modules and using the RTL viewer to verify the nested structure of the design.

## Design:

(a) Write a gate-level model describing the operation of a TwoBit comparator. The module will have 2 Two-bit inputs: **a**, and **b**. Three outputs indicate whether a equals b, a is less than b, or a is greater than b. Create the Karnaugh map and find the expression for the switching function for **the three outputs**.

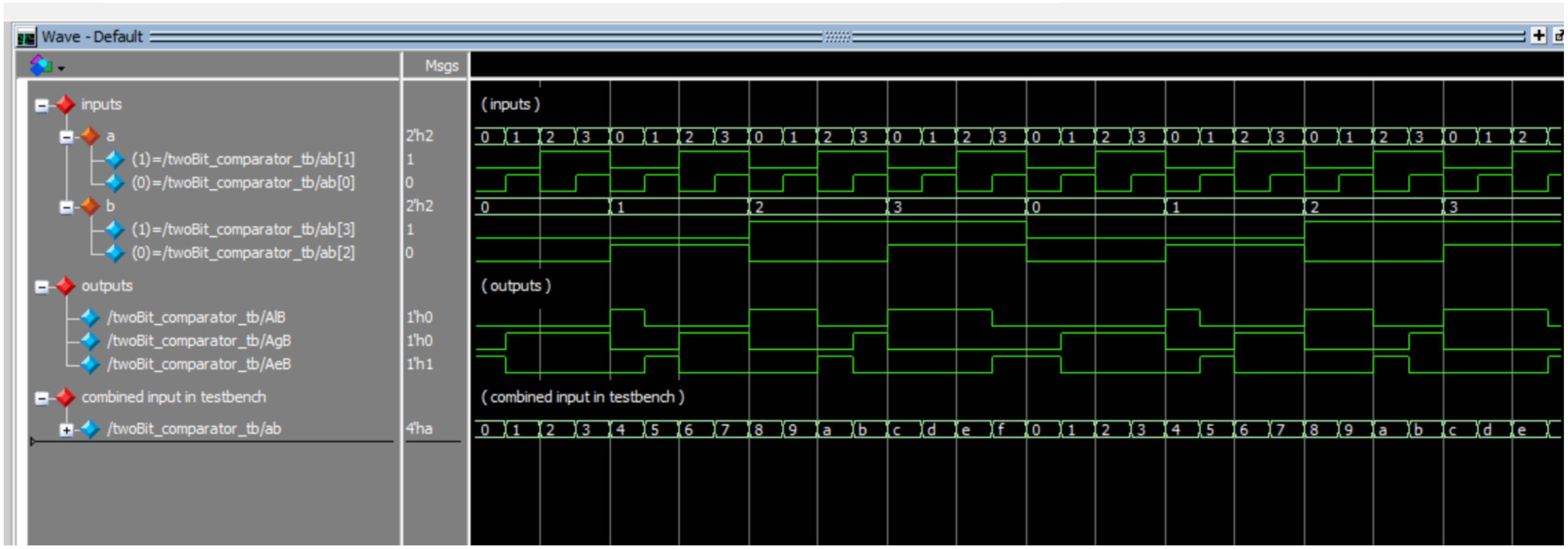
(b) Use the TwoBit comparator to design a FourBit comparator. Add any additional logic needed.

(c) Verify the connection using the RTL Viewer, and verify the interconnections of the instantiated module.

(d) The Verilog Code should be well documented.

```
module twoBit_comparator (a, b, a_less_b, a_greater_b, a_equal_b);

input [1:0] a,b;
output      a_less_b, a_greater_b, a_equal_b;

wire a0,    a1,    b0,    b1;
wire a0bar, a1bar, b0bar, b1bar;

assign a0 = a[0];       assign a0bar = ~a0;
assign a1 = a[1];       assign a1bar = ~a1;
assign b0 = b[0];       assign b0bar = ~b0;
assign b1 = b[1];       assign b1bar = ~b1;

assign a_less_b    = (b1 & a1bar) | (a0bar & a1bar & b0) | (b0 & b1 & a0bar);

assign a_greater_b = (a1 & b1bar) | (a0 & b1bar & b0bar) | (a0 & a1 & b0bar);

assign a_equal_b   = (a0bar & a1bar & b0bar & b1bar) |
                     (a0 & a1bar & b0 & b1bar) |
                     (a0bar & a1 & b0bar & b1) |
                     (a0 & a1 & b0 & b1);

endmodule
```

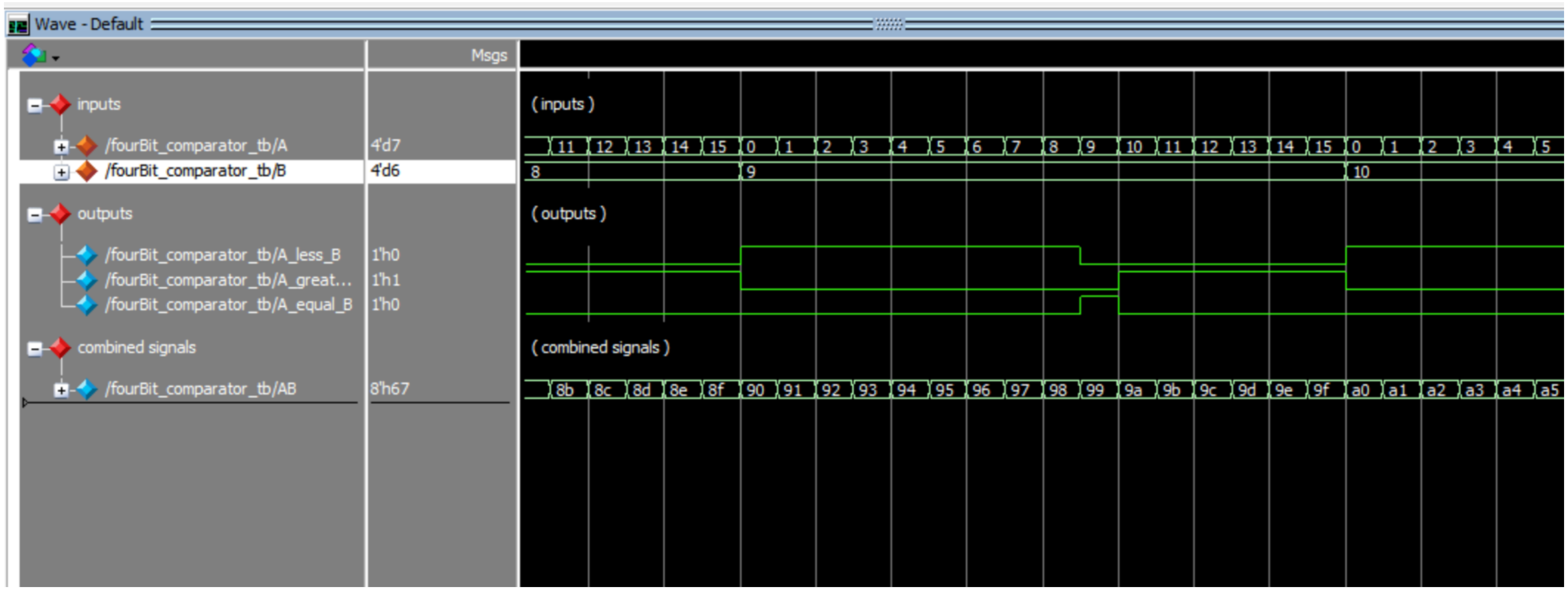
```
module twoBit_comparator_tb ();

reg [3:0] ab;
wire AlB, AgB, AeB;


twoBit_comparator uut (.a (ab[1:0]),
                       .b (ab[3:2]) ,
                       .a_less_b    (AlB),
                       .a_greater_b (AgB),
                       .a_equal_b   (AeB)   );

initial
begin
ab = 4'b0000;
forever
#10 ab = ab + 4'b0001;
end

endmodule
```



```
inputs
  a                                 2'h2
    (1)=/twoBit_comparator_tb/ab[1]  1
    (0)=/twoBit_comparator_tb/ab[0]  0
  b                                 2'h2
    (1)=/twoBit_comparator_tb/ab[3]  1
    (0)=/twoBit_comparator_tb/ab[2]  0
outputs
  /twoBit_comparator_tb/AlB          1'h0
  /twoBit_comparator_tb/AgB          1'h0
  /twoBit_comparator_tb/AeB          1'h1
combined input in testbench
  /twoBit_comparator_tb/ab           4'ha
```

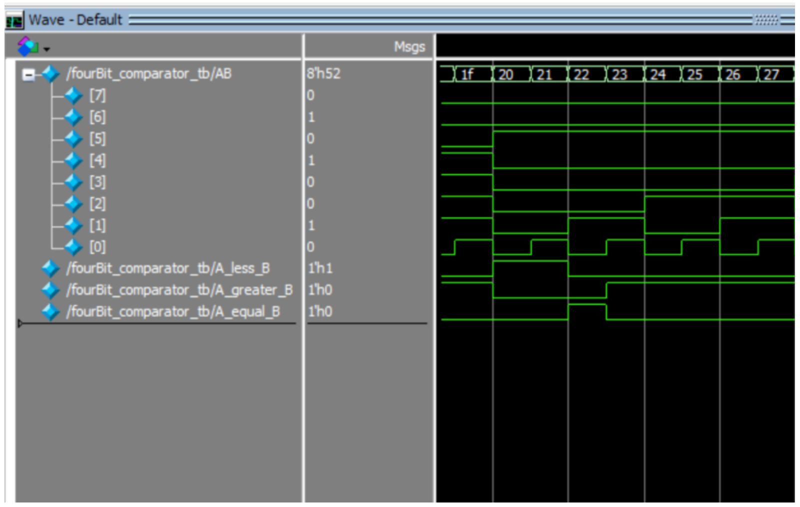
```
module fourBit_comparator (A,B,A_less_B, A_greater_B, A_equal_B);

input [3:0] A,B;
output      A_less_B, A_greater_B, A_equal_B;

wire MSB_A_less_B, MSB_A_greater_B, MSB_A_equal_B;
wire LSB_A_less_B, LSB_A_greater_B, LSB_A_equal_B;


twoBit_comparator MSB (A[3:2], B[3:2], MSB_A_less_B, MSB_A_greater_B, MSB_A_equal_B);
twoBit_comparator LSB (A[1:0], B[1:0], LSB_A_less_B, LSB_A_greater_B, LSB_A_equal_B);

assign A_equal_B    = MSB_A_equal_B & LSB_A_equal_B;
assign A_less_B     = MSB_A_less_B    | (MSB_A_equal_B & LSB_A_less_B);
assign A_greater_B  = MSB_A_greater_B | (MSB_A_equal_B & LSB_A_greater_B);

endmodule
```

```
module fourBit_comparator_tb ();


reg   [7:0] AB;
wire        A_less_B, A_greater_B, A_equal_B;

//fourBit_comparator (A,B,A_less_B, A_greater_B, A_equal_B);

fourBit_comparator UUT (.A (AB[3:0]),
                        .B (AB[7:4]),
                        .A_less_B     (A_less_B),
                        .A_greater_B  (A_greater_B),
                        .A_equal_B    (A_equal_B)       );

initial
begin
    AB = 8'b0000_0000;
    forever
    #10 AB = AB + 1;
end

endmodule
```

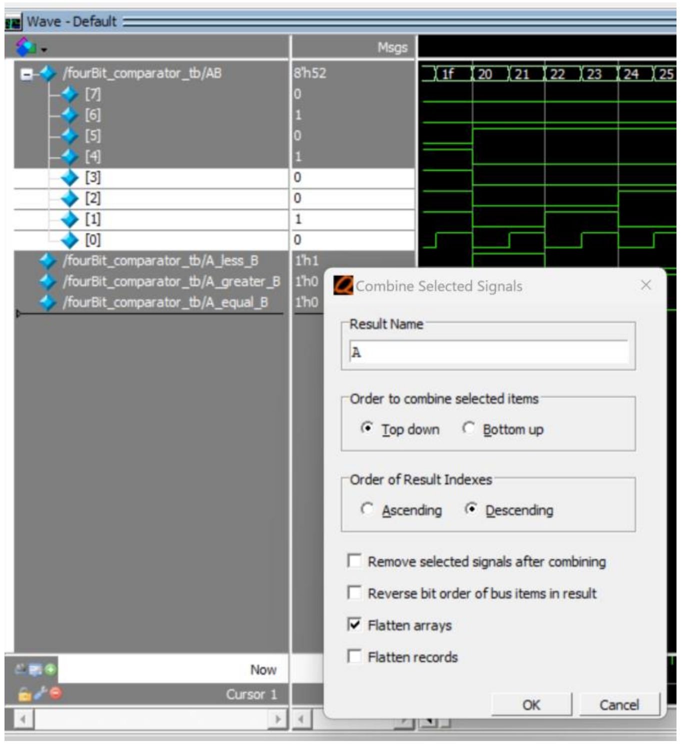


```
Wave - Default
                                              Msgs
inputs                                                 ( inputs )
  /fourBit_comparator_tb/A          4'd7      11 12 13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5
  /fourBit_comparator_tb/B          4'd6      8 9 10
outputs                                                ( outputs )
  /fourBit_comparator_tb/A_less_B   1'h0
  /fourBit_comparator_tb/A_great... 1'h1
  /fourBit_comparator_tb/A_equal_B  1'h0
combined signals                                       ( combined signals )
  /fourBit_comparator_tb/AB         8'h67     8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5
```

1. Expand the created signal AB into single bits in the simulation waveform window.
2. Select the bits [3:0] and **combine signals** them into one bus and call it **A**.
3. Select the bits [7:4] and **combine signals** them into one bus and call it **B**.
4. Select A and B buses and choose **group**.
5. Call the group inputs adjust its height as required.



| | Msgs |
|---|---|
| /fourBit_comparator_tb/AB | 8'h52 |
| [7] | 0 |
| [6] | 1 |
| [5] | 0 |
| [4] | 1 |
| [3] | 0 |
| [2] | 0 |
| [1] | 1 |
| [0] | 0 |
| /fourBit_comparator_tb/A_less_B | 1'h1 |
| /fourBit_comparator_tb/A_greater_B | 1'h0 |
| /fourBit_comparator_tb/A_equal_B | 1'h0 |



| | Msgs |
|---|---|
| /fourBit_comparator_tb/AB | 8'h52 |
| [7] | 0 |
| [6] | 1 |
| [5] | 0 |
| [4] | 1 |
| [3] | 0 |
| [2] | 0 |
| [1] | 1 |
| [0] | 0 |
| /fourBit_comparator_tb/A_less_B | 1'h1 |
| /fourBit_comparator_tb/A_greater_B | 1'h0 |
| /fourBit_comparator_tb/A_equal_B | 1'h0 |

Combine Selected Signals

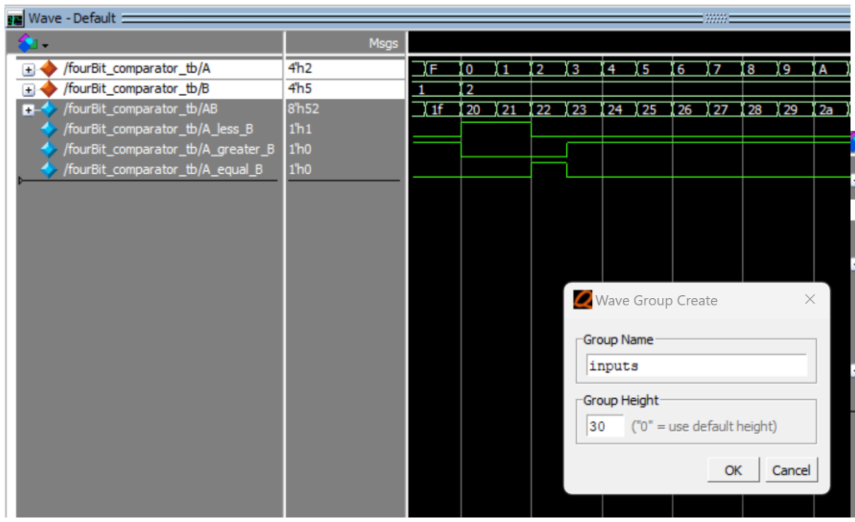
Result Name: A

Order to combine selected items: [✓ Top down] Bottom up

Order of Result Indexes: Ascending [✓ Descending]

- [ ] Remove selected signals after combining
- [ ] Reverse bit order of bus items in result
- [x] Flatten arrays
- [ ] Flatten records

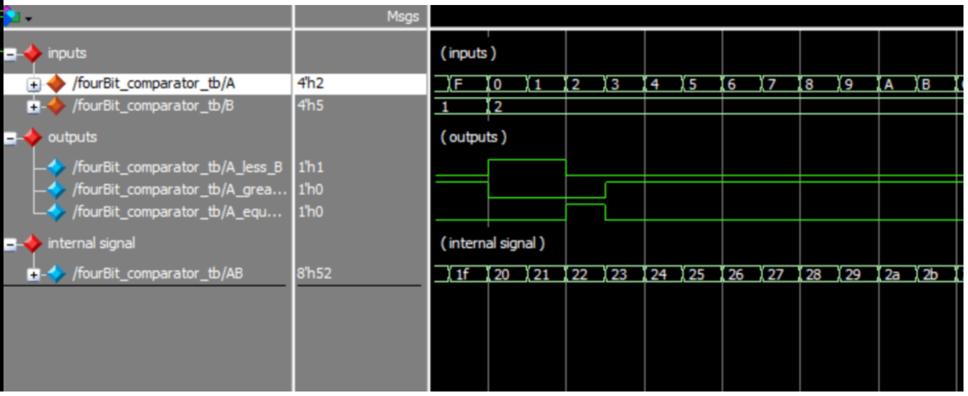
OK Cancel



```
/fourBit_comparator_tb/A           4'h2
/fourBit_comparator_tb/B           4'h5
/fourBit_comparator_tb/AB          8'h52
/fourBit_comparator_tb/A_less_B    1'h1
/fourBit_comparator_tb/A_greater_B 1'h0
/fourBit_comparator_tb/A_equal_B   1'h0
```

Group Name: inputs

Group Height: 30 ("0" = use default height)



```
inputs
    /fourBit_comparator_tb/A        4'h2
    /fourBit_comparator_tb/B        4'h5
outputs
    /fourBit_comparator_tb/A_less_B 1'h1
    /fourBit_comparator_tb/A_grea... 1'h0
    /fourBit_comparator_tb/A_equ... 1'h0
internal signal
    /fourBit_comparator_tb/AB       8'h52
```

You can change the order of the signals by dragging them up or down.

You can also right click on any of the buses and change the **radix** to **unsigned** to display unsigned decimal or choose from the different radix options such as: binary, hexadecimal or octal.