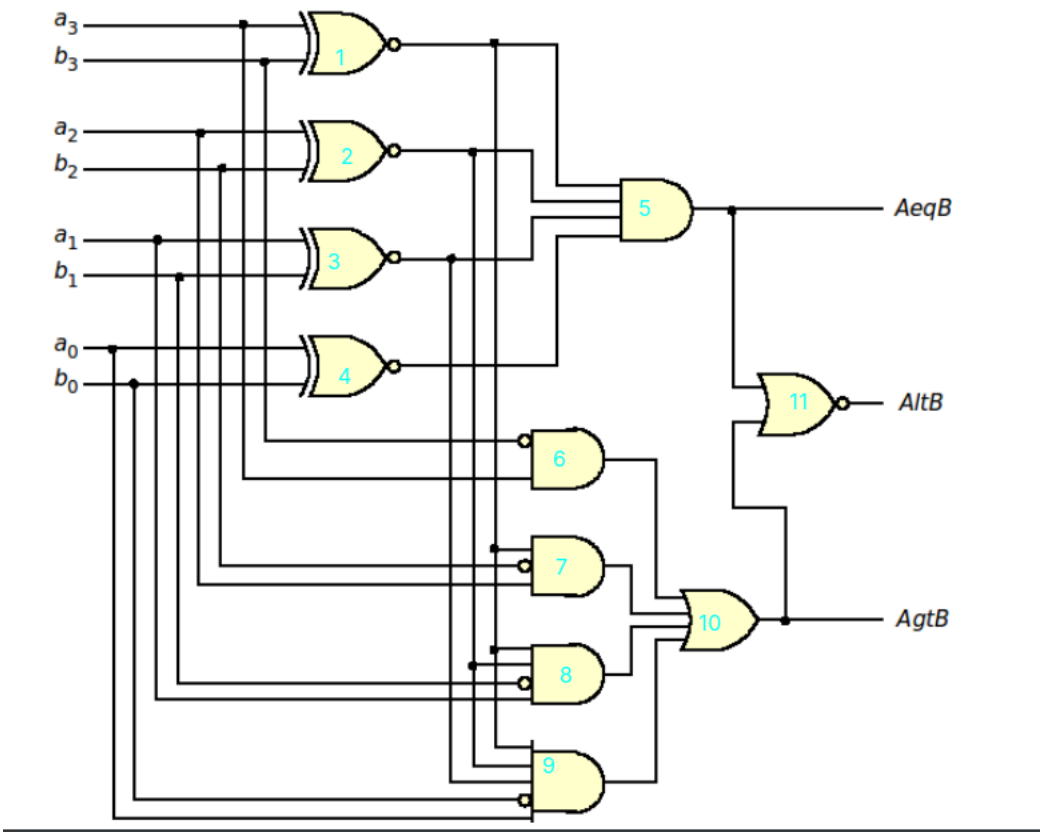


O comparador de 4 bits faz a comparação bit a bit, começando do mais significativo para o menos significativo. Então se o bit An = Bn, ele simplismente passa para os próximos bits até An-1, Bn-1, realizando a comparação bit a bit.

Para o caso em que A = B, o sistema passará pelas portas nXOR 1,2, 3, 4. Essa função lógica retorna 1 caso os valores inseridos nela sejam iguais. Dessa forma as saídas dessas portas ao serem inseridas na porta and(5), serão iguais a 1 caso todos os bits sejam iguais, e 0 caso haja alguma diferença diferença entre esses bits, visto que a porta and só irá retornar 1 se todas as suas entradas forem 1.

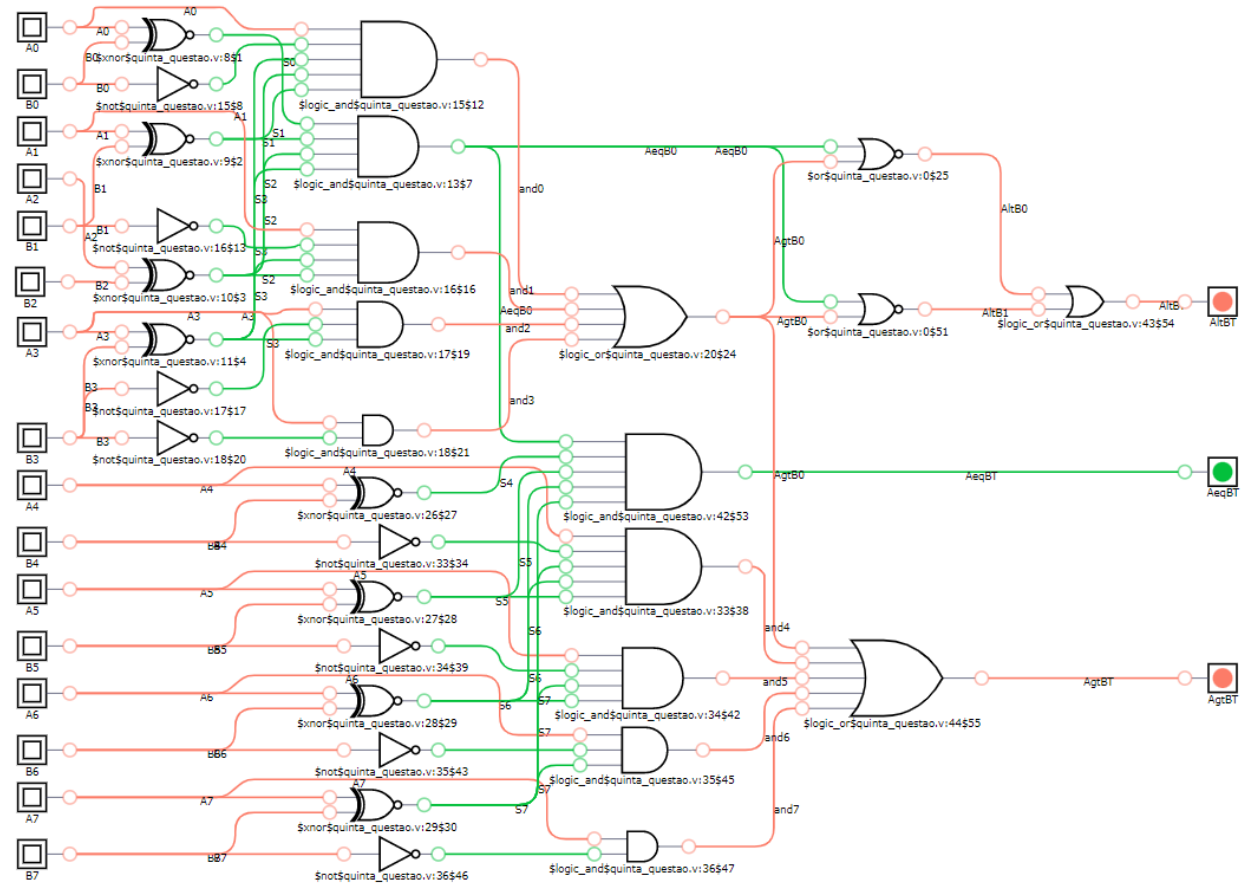
Para o caso A > B, utilizamos as nxor do comparador de igualdade para saber do bit mais significativo ao menos significativo se são iguais, porque, caso haja bit's mais significativos iguais, quem vai definir se A > B, serão os bits menos significativos subsequentes. Desse modo, as portas and 6 - 9, sempre irão analisar se o bit An > Bn, caso eles sejam iguais, passa-se para o próximo and. Como ele está sempre verificando as portas, ele irá sempre barrar o B, visto que ele espera para identificar se A > B, ele espera que as entradas sejam 1 do A e 0 do B, caso o A venha 0, o B venha 1 ou o resultado do anterior a essa comparação de Bit's nXOR seja 0, a porta and irá retornar 0, indicando que A < B. Caso todas as portas retornem 0, a será menor que B, entretanto se ao menos uma retornar 1, saberemos pelo bit mais significativo que A é maior que B.

Para o caso A < B, ele será dado caso o resultado do and(5) e do and(6) retornarem 0, visto que elas não irão ativar suas respectivas saídas, fazendo com que ele ative a porta ou(11), pois sua saída é invertida, indicando que A < B, dado que ela não retornou 1 em nenhum dos dois outros resultados.

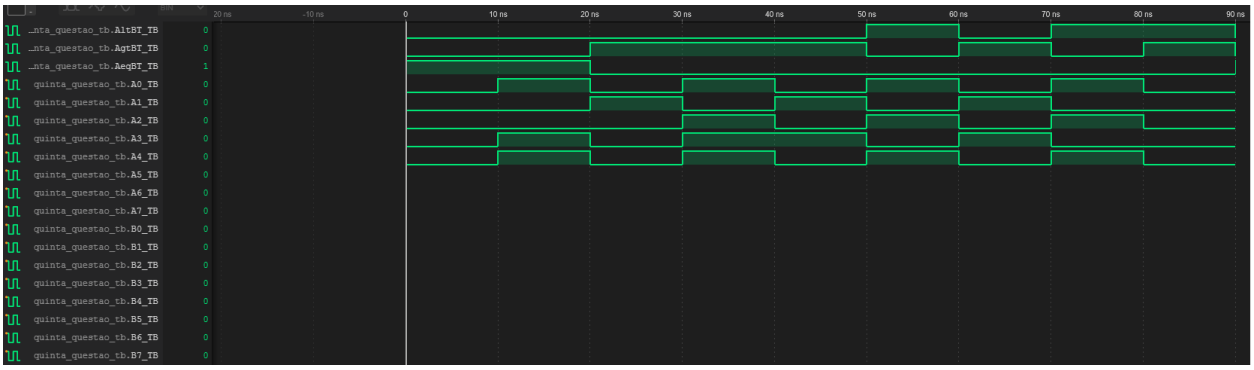
d)

Segue-se a mesma lógica aplicada na letra C, a diferença é que agora estamos aplicando dois comparadores 4 bits, onde o primeiro comparador de 4 bits os resultados da nxor vão de S0 à S3 os resultado das ands vão de and0 a and 3, os resultado da Aeqb, Agtb e altb são AeqB0, AgtB0, AltB0. Ao final, iremos unir os resultados das duas comparadoras de 4 bits para chegar a um resultado final único, que indique se A > B, A < B ou A == B.

```
module quinta_questao(output wire AeqBT, AltBT, AgtBT, input wire A0, A1, A2, A3, A4,
A5, A6, A7, B0, B1, B2, B3, B4, B5, B6, B7);

    wire S0, S1, S2, S3, S4, S5, S6, S7;
    wire AeqB0, AltB0, AgtB0, AeqB1, AltB1, AgtB1;
    wire and0, and1, and2, and3, and4, and5, and6, and7;

    assign S0 = A0 ~^ B0;
    assign S1 = A1 ~^ B1;
    assign S2 = A2 ~^ B2;
    assign S3 = A3 ~^ B3;

    assign AeqB0 = S0 && S1 && S2 && S3;

    assign and0 = A0 && ~B0 && S3 && S2 && S1;
    assign and1 = A1 && ~B1 && S3 && S2;
    assign and2 = A2 && ~B2 && S3;
    assign and3 = A3 && ~B3;

    assign AgtB0 = and0 || and1 || and2 || and3;

    assign AltB0 = AeqB0 ~| AgtB0;



    assign S4 = A4 ~^ B4;
    assign S5 = A5 ~^ B5;
    assign S6 = A6 ~^ B6;
    assign S7 = A7 ~^ B7;

    assign AeqB1 = S4 && S5 && S6 && S7;

    assign and4 = A4 && ~B4 && S7 && S6 && S5;
    assign and5 = A5 && ~B5 && S7 && S6;
    assign and6 = A6 && ~B6 && S7;
    assign and7 = A7 && ~B7;

    assign AgtB1 = and4 || and5 || and6 || and7;

    assign AltB1 = AeqB0 ~| AgtB0;

    assign AeqBT = AeqB0 && AeqB1;
    assign AltBT = AltB0 || AltB1;
    assign AgtBT = AgtB0 || AgtB1;

endmodule
```



e)




```
`timescale 1ns/1ps
module quinta_questao_tb();

    reg A0_TB, A1_TB, A2_TB, A3_TB, A4_TB, A5_TB, A6_TB, A7_TB, B0_TB, B1_TB, B2_TB, B3_TB, B4_TB, B5_TB, B6_TB, B7_TB;
    wire AeqBT_TB, AltBT_TB, AgtBT_TB;

    quinta_questao DUT(.AeqBT(AeqBT_TB), .AltBT(AltBT_TB), .AgtBT(AgtBT_TB), .A0(A0_TB), .A1(A1_TB), .A2(A2_TB), .A3(A3_TB), .A4(A4_TB), .A

    initial
        begin
            $dumpfile("quinta_questao_tb.vcd");
            $dumpvars(0, quinta_questao_tb);
```

```
                A0_TB=0; A1_TB=0; A2_TB=0; A3_TB=0; A4_TB=0; A5_TB=0; A6_TB=0; A7_TB=0; B0_TB=0; B1_TB=0; B2_TB=0; B3_TB=0; B4_TB=0; B5_TB
            #10  A0_TB=1; A1_TB=0; A2_TB=0; A3_TB=1; A4_TB=1; A5_TB=0; A6_TB=0; A7_TB=1; B0_TB=1; B1_TB=0; B2_TB=0; B3_TB=1; B4_TB=1; B5_TB
            #10  A0_TB=0; A1_TB=1; A2_TB=0; A3_TB=0; A4_TB=0; A5_TB=0; A6_TB=0; A7_TB=0; B0_TB=0; B1_TB=0; B2_TB=0; B3_TB=0; B4_TB=0; B5_TB
            #10  A0_TB=1; A1_TB=0; A2_TB=1; A3_TB=1; A4_TB=1; A5_TB=0; A6_TB=0; A7_TB=1; B0_TB=1; B1_TB=0; B2_TB=0; B3_TB=1; B4_TB=1; B5_TB
            #10  A0_TB=0; A1_TB=1; A2_TB=0; A3_TB=1; A4_TB=0; A5_TB=0; A6_TB=1; A7_TB=0; B0_TB=0; B1_TB=0; B2_TB=1; B3_TB=0; B4_TB=0; B5_TB
            #10  A0_TB=1; A1_TB=0; A2_TB=1; A3_TB=0; A4_TB=1; A5_TB=0; A6_TB=0; A7_TB=1; B0_TB=1; B1_TB=1; B2_TB=0; B3_TB=1; B4_TB=1; B5_TB
            #10  A0_TB=0; A1_TB=1; A2_TB=0; A3_TB=1; A4_TB=0; A5_TB=1; A6_TB=0; A7_TB=0; B0_TB=0; B1_TB=0; B2_TB=0; B3_TB=0; B4_TB=0; B5_TB
            #10  A0_TB=1; A1_TB=0; A2_TB=1; A3_TB=0; A4_TB=1; A5_TB=0; A6_TB=1; A7_TB=0; B0_TB=1; B1_TB=1; B2_TB=1; B3_TB=1; B4_TB=1; B5_TB
            #10  A0_TB=0; A1_TB=0; A2_TB=0; A3_TB=0; A4_TB=0; A5_TB=0; A6_TB=0; A7_TB=1; B0_TB=1; B1_TB=0; B2_TB=0; B3_TB=0; B4_TB=0; B5_TB
            #10  A0_TB=0; A1_TB=0; A2_TB=0; A3_TB=0; A4_TB=0; A5_TB=0; A6_TB=0; A7_TB=0; B0_TB=0; B1_TB=0; B2_TB=0; B3_TB=0; B4_TB=0; B5_TB

        end

endmodule
```