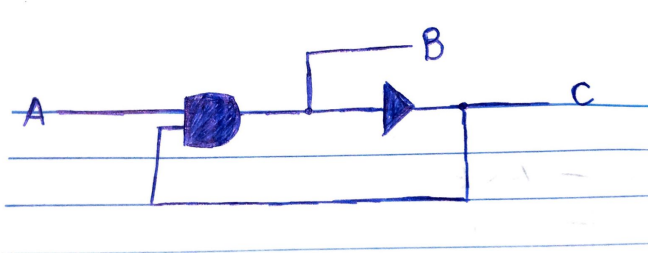
Projeto I

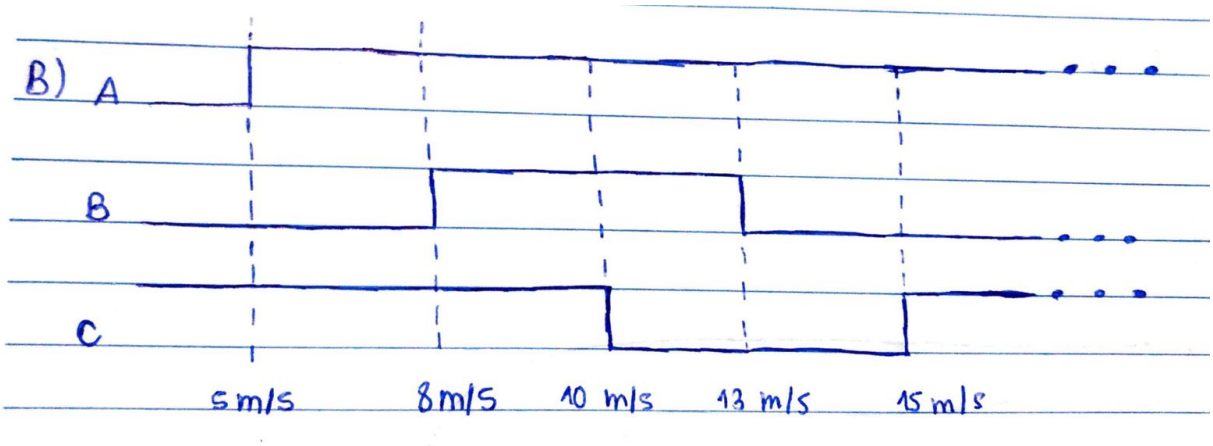
Estudantes:

- Amanda Quirino Rodrigues dos Santos - aqrs
- Erlan Lira Soares Júnior - elsj

1. a)



b)



2.

```
`timescale 1ns/1ps
module circuit_segunda(input wire A, B, C, D,
              output wire Z);

      wire F, E;
      wire F1, F2, F3, F4, E1;

      assign #5 E1 = A && B && C;
      assign #5 F1 = (B || C);
      assign #2 F2 = ~F1;

      assign #5 E = E1 || D;
      assign #5 F3 = F2 && A;
      assign #2 F4 = ~F3;
      assign #2 F = ~F4;

      assign #5 Z = (F^E);

endmodule
```

3. a)

```
`timescale 1ns/1ps
module mux_4_to_1_metodo_1(input wire A, B, C, D,
                           output wire F);
 wire An = ~A;
 wire Bn = ~B;
 assign #10 F = (C == 0 && D == 0) ? An :
      (C == 0 && D == 1) ? B :
      (C == 1 && D == 0) ? Bn :
      (C == 1 && D == 1) ? 0 :
       00;

endmodule
```

b)

```
`timescale 1ns/1ps
module mux4to1_terceiro_metodo(input wire A, B, C, D,
                               output wire F);


  wire An = ~A;
  wire Bn = ~B;

always @*

    begin
      if (C ==0 && D==0)
        #10 F= An;
      else if (C==0 && D ==1)
        #10 F= B;
      else if (C==1 && D==0)
        #10 F= Bn;
      else
        #10 F=0;
    end

endmodule
```

c)

```
`timescale 1ns/1ps
module mux4to1_terceiro_metodo(input wire A, B, C, D,
                               output wire F);


  wire An = ~A;
  wire Bn = ~B;
  wire [1:0] Sel = {C, D};

always @*

    begin
      casez(Sel)
          2'b00: #10 F = An;
          2'b01: #10 F = B;
          2'b10: #10 F = Bn;
          2'b11: #10 F = 0;
          2'bxx: #10 F = x;
        endcase
    end

endmodule
```

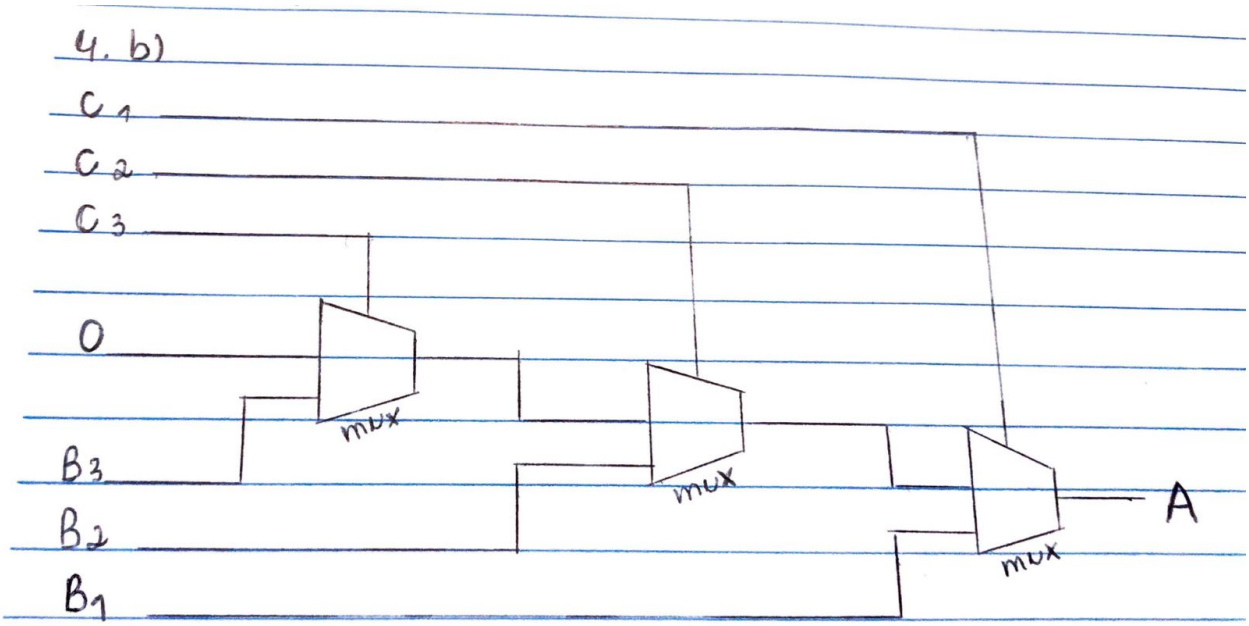
4. a) Método I

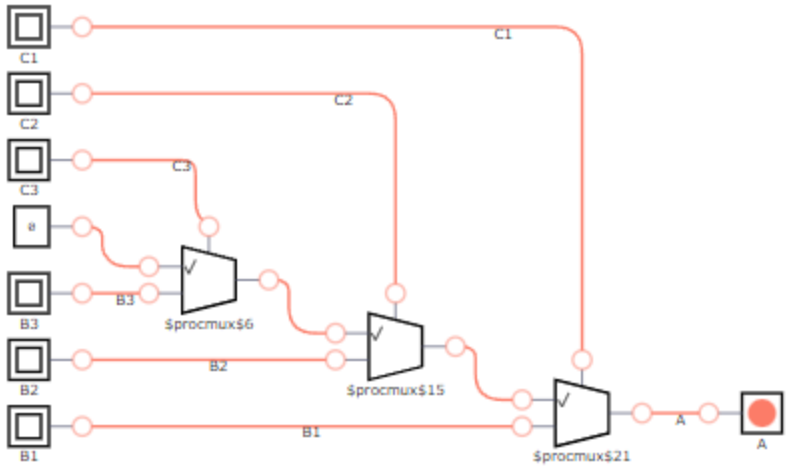
```
module metodo_um(input wire [1:0] C,
                     input wire B1, B2, B3,
                     output wire A);


always @*
    begin
      casez(C)
          2'b01: A = B1;
          2'b10: A = B2;
          2'b11: A = B3;
          2'bxx: A = 00;
        endcase
    end

endmodule
```

Método II

```
module metodo_dois(input wire [1:0]C,
                     input wire B1, B2, B3,
                     output wire A);

  assign A = (C == 2'b01) ? B1 :
             (C == 2'b10) ? B2 :
             (C == 2'b11) ? B3 :
             1'b0;

endmodule
```

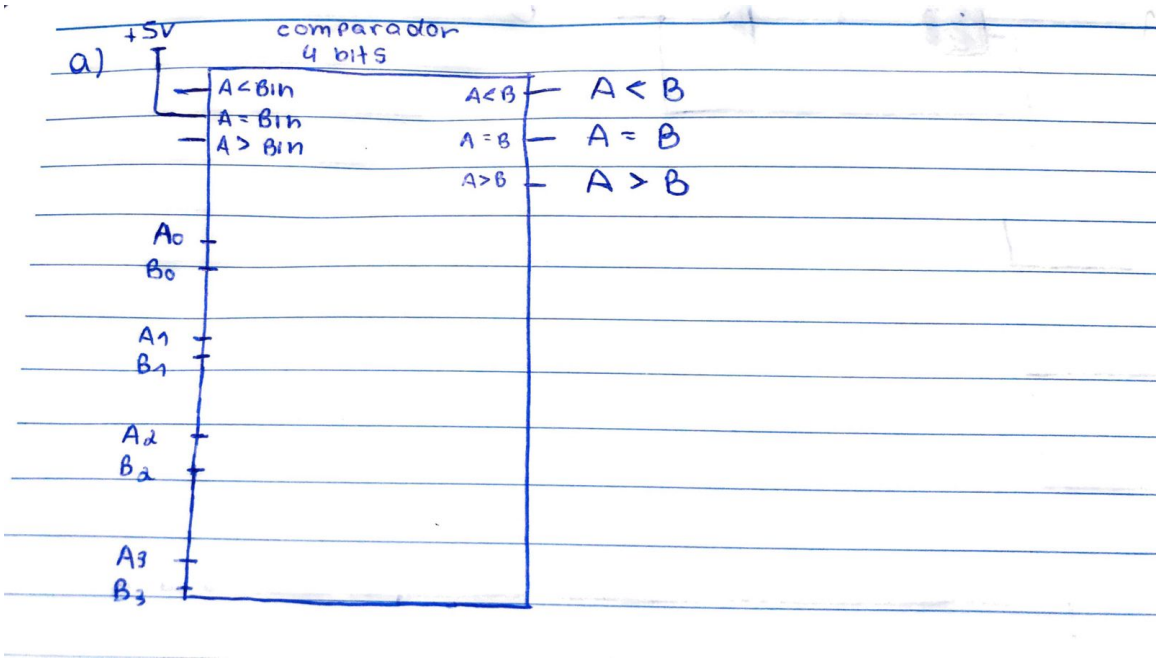
b)



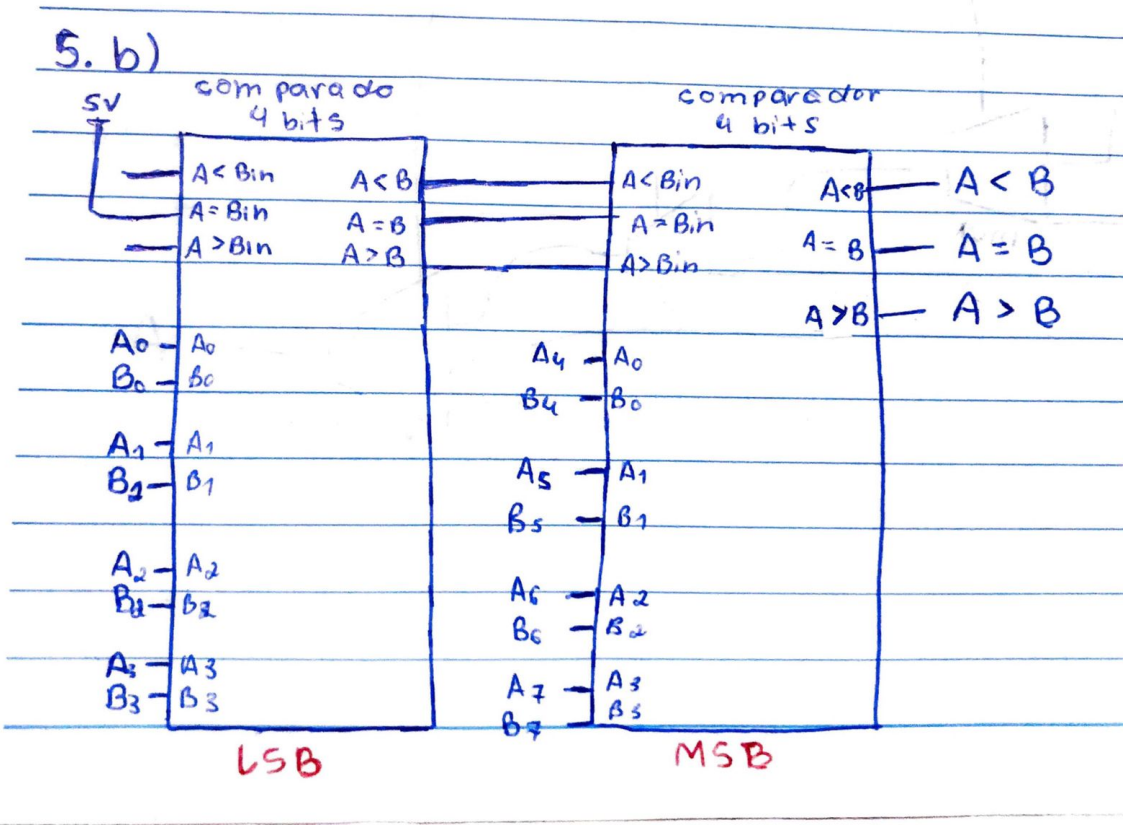


Print verilog

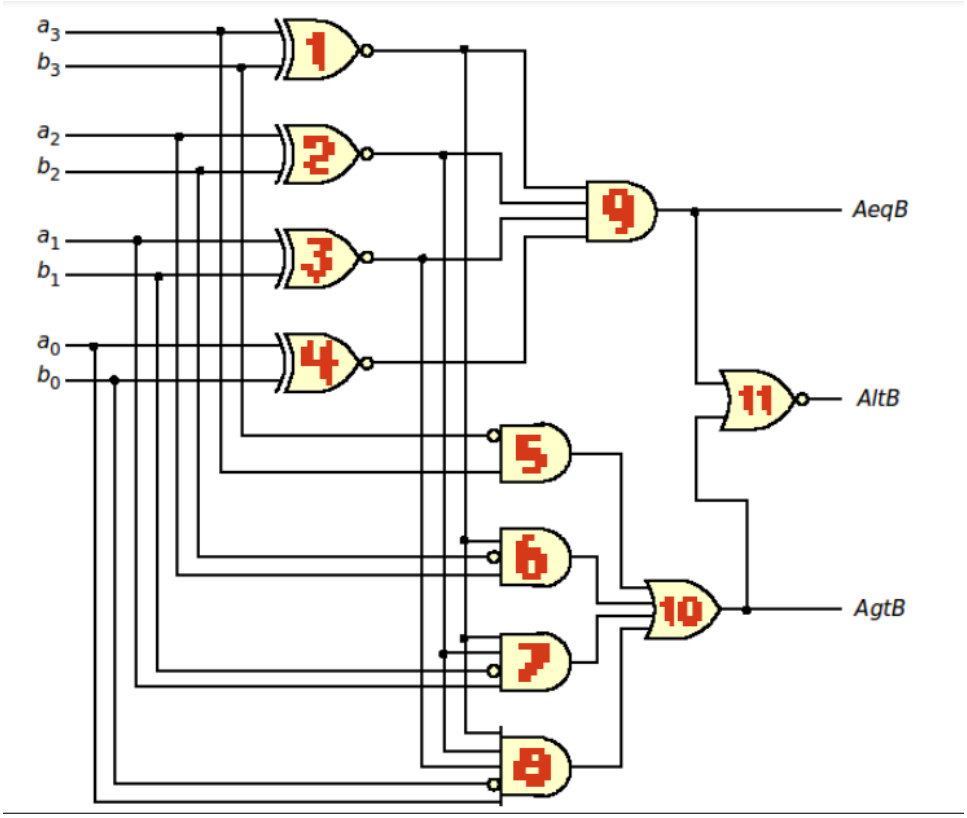
5. a)


a)
+5V
comparador
4 bits
A<Bin
A=Bin
A>Bin
A<B
A=B
A>B
A < B
A = B
A > B
A0
B0
A1
B1
A2
B2
A3
B3

b)


5. b)
5V
comparador
4 bits
comparador
4 bits
A<Bin
A=Bin
A>Bin
A<B
A=B
A>B
A < B
A = B
A > B
A0
B0
A1
B1
A2
B2
A3
B3
A4
B4
A5
B5
A6
B6
A7
B7
LSB
MSB

c)



Usaremos o quarto bit de A e de B como os mais significativos de seus respectivos número.

Como o comparador compara se A = B:

Usa as portas 1, 2, 3 e 4 que são portas NXOR e a porta 9, que é uma porta AND, pois NXOR só retorna 1 quando as entradas são iguais, portanto se a saída das portas 1, 2, 3 e 4 são 1, então A == B, e para verificar se a saída dessas portas são iguais a 1, usamos uma porta AND (Porta 9) cujas entradas são as saídas dessas portas XNOR, pois a porta AND só terá saída 1 quando todas as entradas forem 1, portanto caso A == B, a porta 9 terá saída 1.

Como o comparador compara se A > B:

Usa as portas 1, 2, 3, que são portas XNOR, 5, 6, 7, 8, que são portas AND, e a porta 10, que é uma porta OR, em que o quarto bit de A e o quarto bit de B serão as entradas da porta 1, o terceiro bit de A e o terceiro bit de B serão as entradas da porta 2 e o segundo bit de A e o segundo bit de B serão as entradas da porta 3. Já a porta 5 terá o quarto bit de A e a negação do quarto bit de B como entradas, com o terceiro bit de A, a saída da porta 1 e a negação do terceiro bit de B como entradas da porta 6, o segundo bit de A, as saídas das portas 1 e 2, e a negação do segundo bit de B como entradas da porta 7 e o primeiro bit de A, as saídas das portas 1, 2 e 3, e a negação do primeiro bit de B como entradas da porta 8 e as saídas das portas 5, 6, 7 e 8 como entradas da porta 10, assim a saída da porta 10 será 1 se A > B e 0, caso contrário.

Isso acontece, pois caso A == B, então como todas as portas AND usadas aqui tem como entrada um bit de A e a negação de um bit de B, então caso o bit de A seja 1, o bit de B será 1 e a sua negação será 0, logo a porta AND que tem esses bits como entrada retornará 0, e caso o bit de A seja 0, a mesma lógica se aplica. Portanto caso A == B, as portas AND terão (1, 1, 0) como entradas, logo a saída de todas as portas AND será 0 e como essas saídas são as entradas da porta OR que determina se A > B, então a saída da porta OR será 0, indicando que A não é maior que B.

Além disso, caso A < B, então o primeiro bit, que chamaremos de ‘a’, de A que for diferente de B tem que ser 0, então como os bits que foram comparados antes de ‘a’ são iguais aos bits de B, então as portas AND que tem esses bits como entrada terão saída 0, segundo a lógica explicada no parágrafo acima, e como ‘a’ = 0 e o bit de B será igual a 1, para que A < B, então a porta XNOR que tem esses bits como entrada terá saída 0 e como as portas AND que usa os próximos bits como entrada também usa todos os XNOR que usam os bits anteriores como entrada, então como terá ao menor um XNOR com saída 0, a saída desses AND também será 0, assim as saídas de todas as portas AND usadas para comparar se A > B terão saída 0, então a saída da porta 10 também será 0, pois tem como entrada as saídas das portas AND, logo A não é maior que B.

Já no caso em que A > B, nota-se que o primeiro bit, que chamaremos de ‘a’, de A que for diferente de B tem que ser 1 e o bit de B que está na mesma posição que

‘a’ tem que ser 0, logo a negação desse bit será 1 e a porta XNOR que tem como entradas o bit anterior a ‘a’ terá saída 1 pois esse bit anterior de A será igual ao bit de B que está na mesma posição, logo as entradas da porta AND que receberá ‘a’, a negação do bit de B na mesma posição que ‘a’ e a porta XNOR terá a entrada (1, 1, 1), logo a sua saída será 1, então, como no mínimo uma das portas AND cujas saídas serão as entradas da porta 10 terá saída 1, então a saída da porta 10 será 1, indicando que A > B.

Como o comparador compara se B < A:

Usa a porta 10, que é uma porta OR, a porta 9, que é uma porta AND, e a porta 11, que é uma porta NOR, pois a porta 10 diz se A > B e a porta 9 diz se A == B, então caso a porta 10 e a porta 9 tiverem saída 0, então A não é maior a B, nem é maior, portanto A será menor que B. Dessa forma, como a porta NOR tem saída 1 apenas quando todas as suas entradas são 0, então como as saídas da porta 9 e 10 serão as entradas da porta 11, então caso a saída de no mínimo uma delas for 1, então a saída da porta 11 será 0, visto que A não será menor que B, e caso as saídas das portas 10 e 9 sejam 0, então a saída da porta 11 será 1, pois A < B nesse cenário.

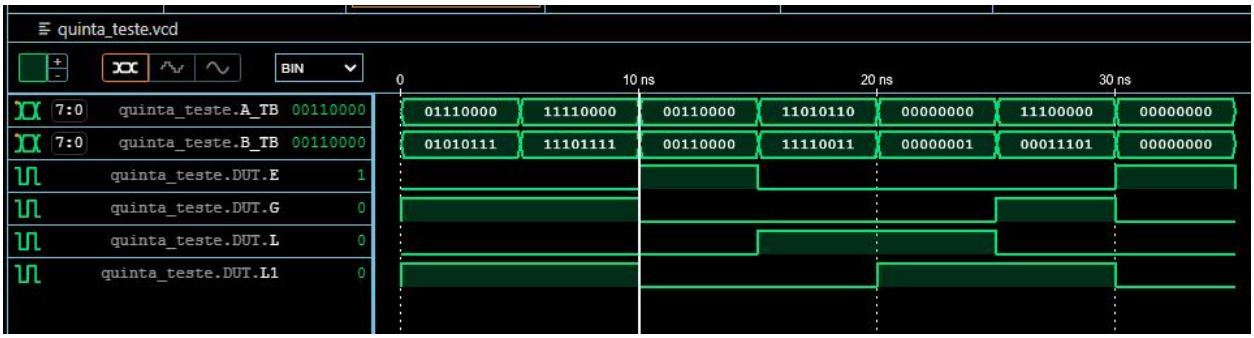
d)

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

  wire G1, L1, E1;

  wire [3:0]A1;
  wire [3:0]A2;
  wire [3:0]B1;
  wire [3:0]B2;

  assign A1 = {A[3:0]};
  assign A2 = {A[7:4]};
  assign B1 = {B[3:0]};
  assign B2 = {B[7:4]};


  assign G1 = (A1>B1);
  assign L1 = (A1<B1);
```

```
  assign E1 = (A1==B1);

  assign G = (A2>B2) || (A2 == B2) && G1;
  assign L = (A2<B2) || (A2 == B2) && L1;
  assign E = (A2==B2) && E1;


endmodule
```

e)

```
`timescale 1ns/1ps

module quinta_teste();

    wire G_TB, L_TB, E_TB;
    reg [7:0]A_TB, B_TB;

    quinta DUT(.A(A_TB), .B(B_TB));

    initial
        begin

            $dumpfile("quinta_teste.vcd");
            $dumpvars(0,quinta_teste);

                A_TB=8'b11110000; B_TB=8'b00001111;
            #5  A_TB=8'b11110000; B_TB=8'b11101111;
            #5  A_TB=8'b00110000; B_TB=8'b00110000;
            #5  A_TB=8'b11010110; B_TB=8'b11110011;
            #5  A_TB=8'b00000000; B_TB=8'b00000001;
            #5  A_TB=8'b11100000; B_TB=8'b00011101;
            #5  A_TB=8'b00000000; B_TB=8'b00000000;
            #5  A_TB=8'b11111111; B_TB=8'b11111111;
        end

endmodule
```



Resultado da bancada de testes