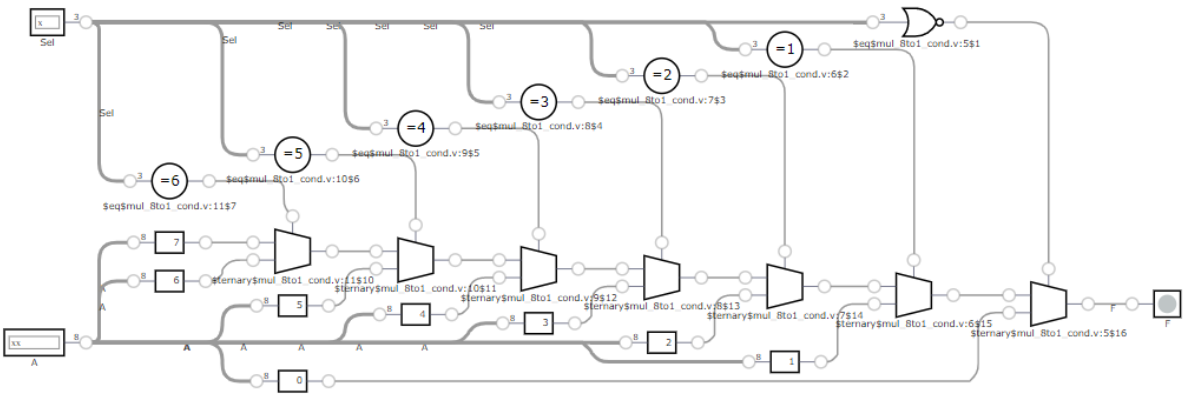
PROJETO PII - Sistemas Digitais

Alunos:

- Brenda Vanessa Guerra Alves (bvga)
- Rubens Nascimento de Lima (rnl2)
- Pedro Barros de Souza Lima (pbsl)
- Matheus Barney Mara Galindo (mbmg)

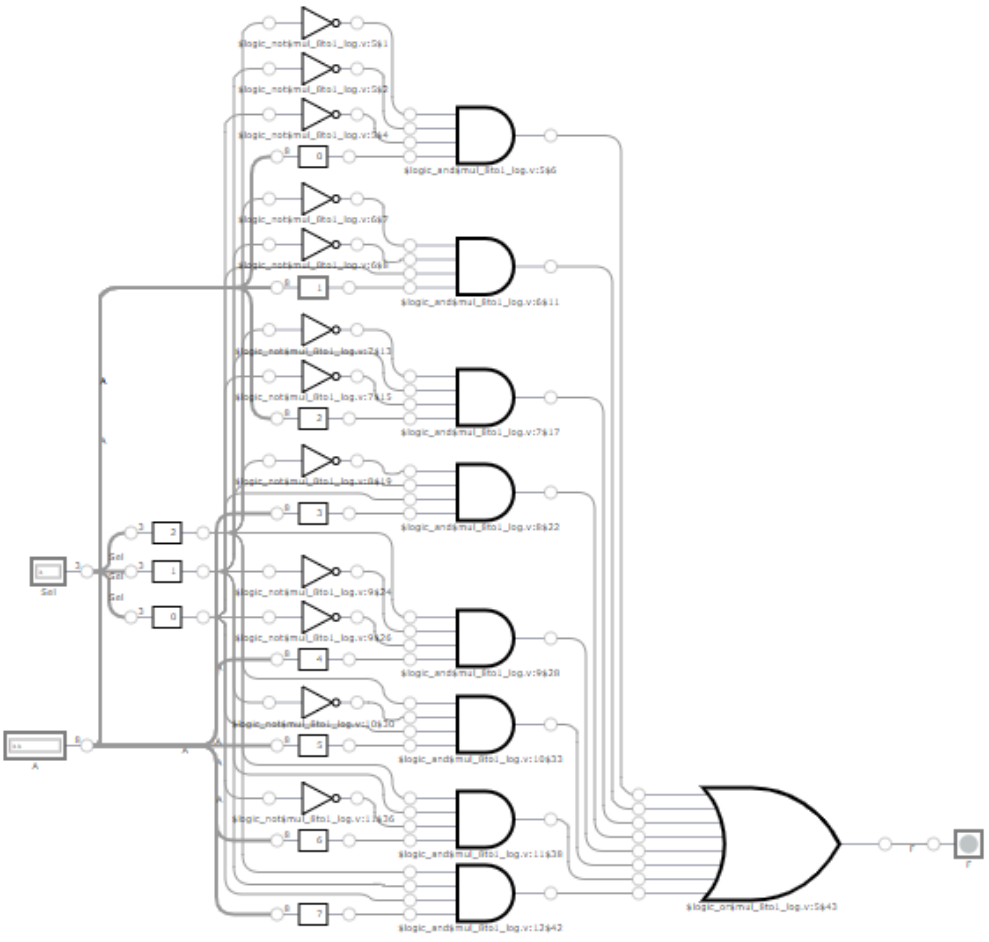
1.a) Use atribuição contínua e operadores condicionais.

```
module mul_8to1(output wire F,
                input wire [7:0] A,
                input wire [2:0] Sel);

  assign F = (Sel == 3'b000) ? A[0]:
             (Sel == 3'b001) ? A[1]:
             (Sel == 3'b010) ? A[2]:
             (Sel == 3'b011) ? A[3]:
             (Sel == 3'b100) ? A[4]:
             (Sel == 3'b101) ? A[5]:
             (Sel == 3'b110) ? A[6]:
             (Sel == 3'b111) ? A[7]:
             1'bX;

endmodule
```



1.b) Use atribuição contínua e operadores lógicos.

```
module mul_8to1(output wire F,
                        input wire [7:0] A,
                        input wire [2:0] Sel);

  assign F = (!Sel[2] && !Sel[1] && !Sel[0] && A[0] ||
                  !Sel[2] && !Sel[1] && Sel[0] && A[1] ||
                  !Sel[2] && Sel[1] && !Sel[0] && A[2] ||
                  !Sel[2] && Sel[1] && Sel[0] && A[3] ||
                  Sel[2] && !Sel[1] && !Sel[0] && A[4] ||
                  Sel[2] && !Sel[1] && Sel[0] && A[5] ||
                  Sel[2] && Sel[1] && !Sel[0] && A[6] ||
                  Sel[2] && Sel[1] && Sel[0] && A[7]);

endmodule
```

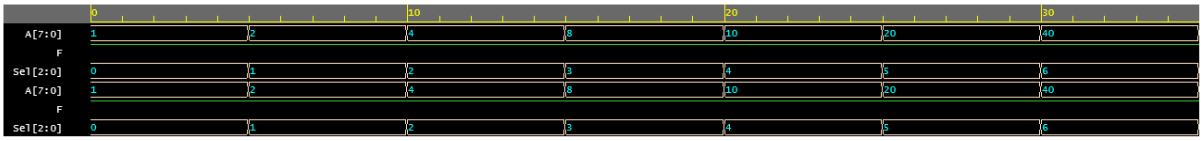


1.c) Verifique e simule seus módulos com a ajuda de uma bancada de teste.

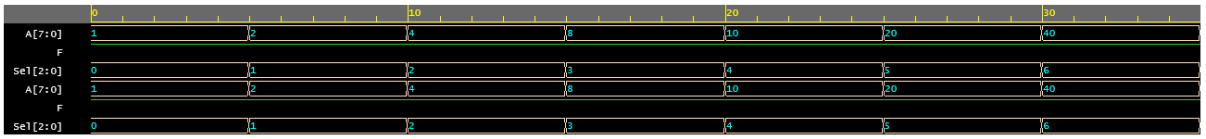
- Código da bancada de testes:

```
module systemx_tb();

  wire F;
  reg [7:0] A;
  reg [2:0] Sel;

  mul_8to1 dut(F, A, Sel);

  initial
        begin

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

        A = 8'b00000001;
        Sel = 3'b000;

        #5 A = 8'b00000010;
        Sel = 3'b001;

        #5 A = 8'b00000100;
        Sel = 3'b010;

        #5 A = 8'b00001000;
        Sel = 3'b011;

        #5 A = 8'b00010000;
        Sel = 3'b100;

        #5 A = 8'b00100000;
        Sel = 3'b101;

        #5 A = 8'b01000000;
        Sel = 3'b110;

        #5 A = 8'b10000000;
        Sel = 3'b111;

        end

endmodule
```

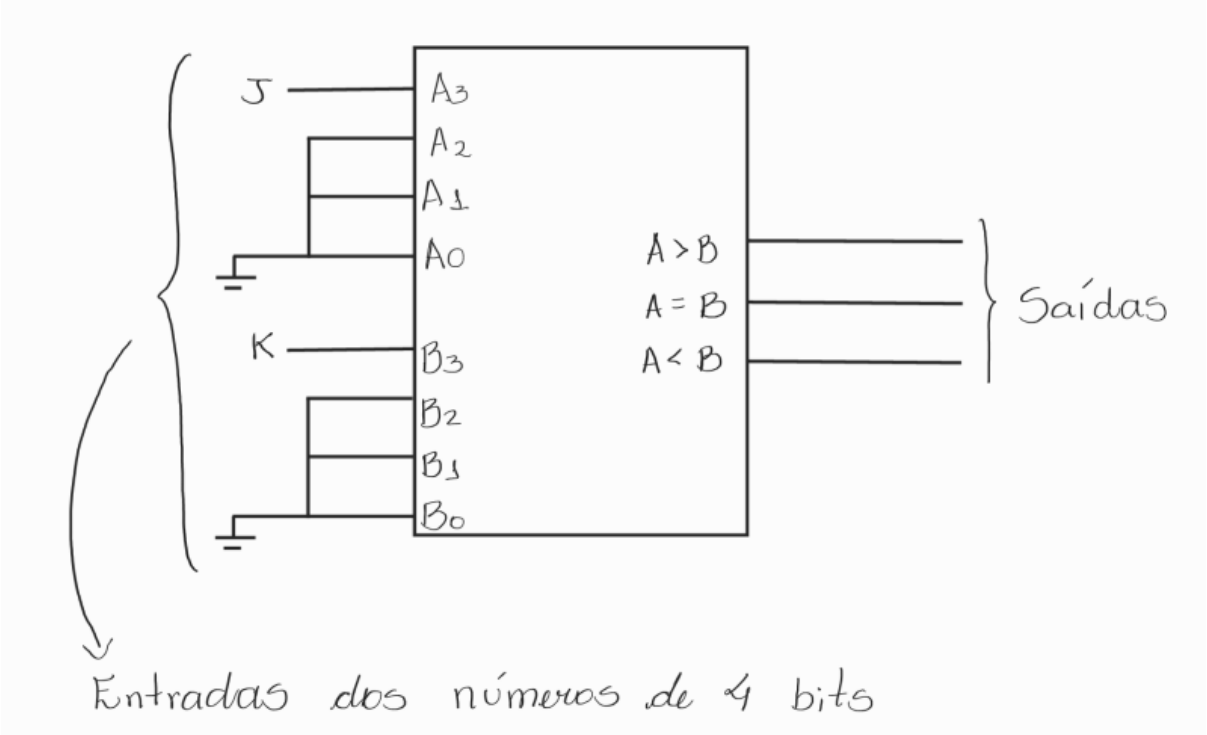
- Verificação do código de comparadores lógicos:



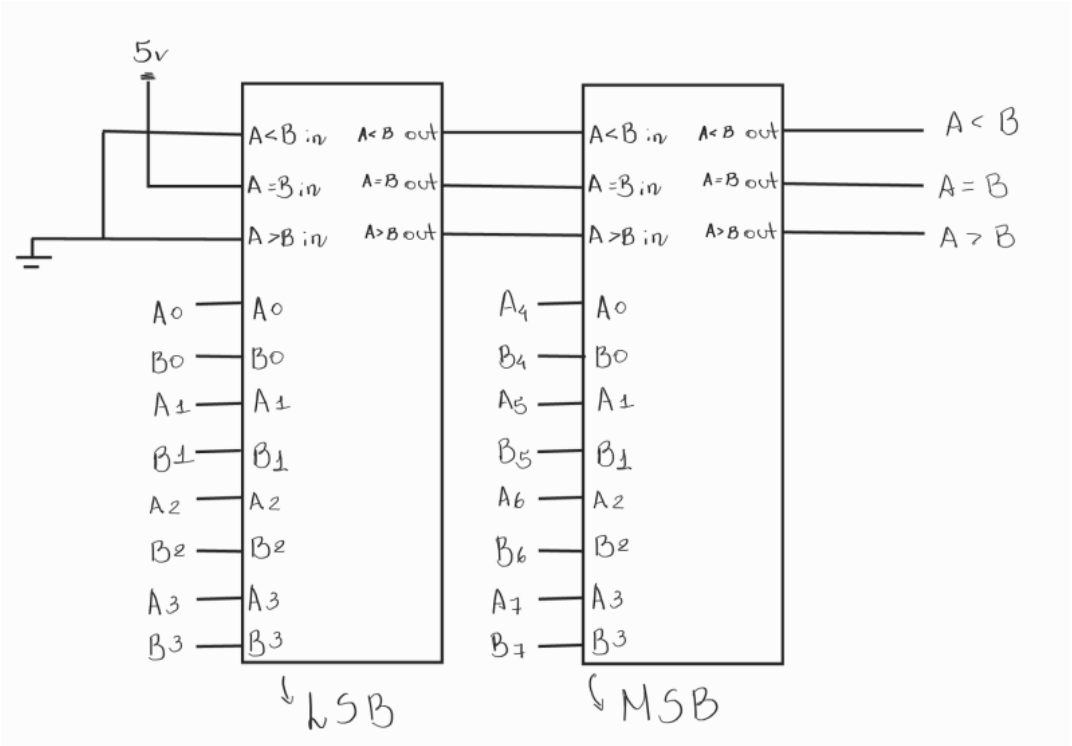
- Verificação do código de comparadores condicionais::



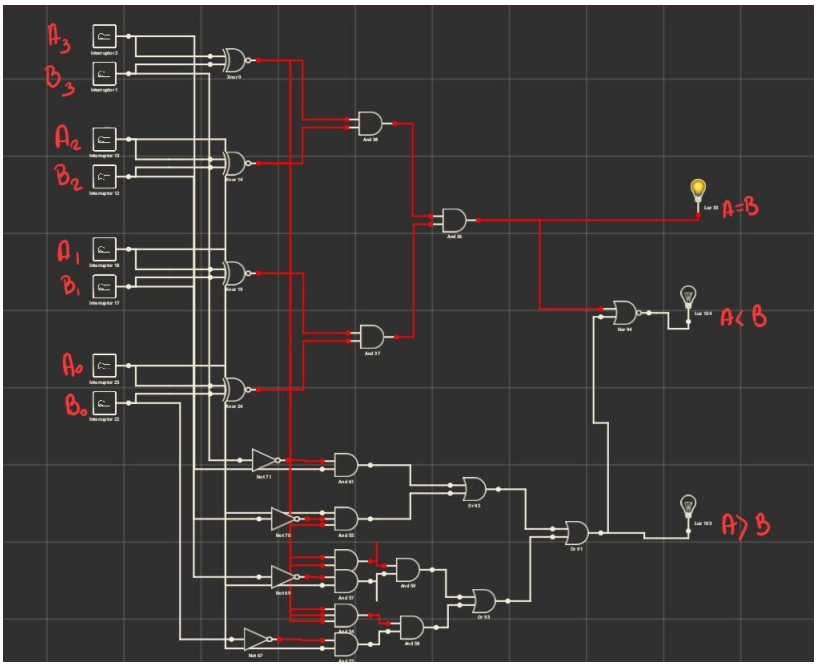
2.a) Desenhe um diagrama de blocos de um comparador de magnitude de 4 bits.



2.b) Desenhe um diagrama de blocos para indicar como você pode construir um comparador de magnitude de 8 bits usando dois comparadores de magnitude de 4 bits.



2.c) Escreva uma descrição comportamental do Verilog para o comparador de 4 bits.



Obs: ao escrever o texto estava me baseando num modelo onde as portas podiam ter mais de uma entrada, porém, no app podiam até 3

Comparando se A = B:

Se baseando no circuito mostrado acima, podemos usar 4 portas NXOR que são ligadas a uma porta AND, as portas NXOR, ou não ou-exclusivo tem um comportamento de apresentar como saída o valor 1 se os seus conectantes tem valores iguais, tanto para 0 como para 1. suas entradas, para cada uma são os valores de An com Bn. E então unimos ela a porta AND que retorna 1 se todos os itens forem 1 também, ou seja, se todos os itens forem iguais. E se apenas um dos itens for diferente não será mais.

Comparando se A > B:

Para verificar se A é maior que B, vamos precisar de 4 portas AND e uma OR. Por ser um pouco mais complexo que a verificação, cada porta AND será detalhada. A ordem do nome será dado a partir de cima para baixo da imagem. Como a porta seguinte após os ANDs é uma OR, basta que apenas uma das portas AND tenha valor 1 para A ser maior que B.

1. Porta 1 - A primeira porta AND recebe a negação de B3(bit mais significativo de B), e o A3, o bit mais significativo de A3, ou seja o bit mais significativo de ambos, a negação do B3 é para que se ele for 0 e a A3 for 1, a saída da porta será 1, isso significa que A3 é maior que B, pois pode obter valores até 15, enquanto B só até 7 por iniciar em 0. Nos demais casos a saída será 0. Todas as demais portas seguirão a lógica de verificar os bits de An e Bn, negando o de Bn para que se for O o valor de Bn e 1 o valor de An, o valor resultante do AND será 1, dependente dos outros.

2. Porta 2 - A segunda porta além de verificar se os valores de A2 e B2, A2 tem um valor "maior" que o de 1, também recebe o valor da saída do primeiro XNOR, ou seja se o resultado dos bits mais significativos forem iguais, caso sejam iguais, e A2 seja 1 e B2 seja 0, a porta terá valor 1, caso não, a porta terá valor 0.

3. Porta 3 e 4 - Seguem a mesma lógica da Porta 2 e Porta 1, tem uma entrada que verifica se o conjunto de An- 1, ..., e Bn-1, ... são iguais, e a mesma lógica da Porta 1, onde vai verificar se o Bit de An tem valor "maior" que o de Bn.

Feito isso, unimos todos numa porta or, para que se em um dos casos for 1, isso significa que A > B.

Comparando se A < B:

Para isso, usaremos uma porta NOR, e iremos colocar nela os valores resultantes de A = B e A > B, se forem 0 nos dois casos, ou seja se não for igual ou menor, será então 1.

2.d)

```
module comp_4bit
  (output wire G_out, L_out, E_out,
   input wire G_in, L_in, E_in,
   input wire [3:0] A, B);

  // G = maior
  assign G_out = (A > B) ? 1'b1 :
                 (G_in == 1 && A==B) ? 1'b1 :
                 1'b0;
  // L = menor
  assign L_out = (A < B) ? 1'b1 :
                 (L_in == 1 && A==B) ? 1'b1 :
                 1'b0;
  // E = igual
  assign E_out = (A == B && E_in==1) ? 1'b1 : 1'b0;

endmodule

module comp_8bit
  (output wire Ma_out, Me_out, I_out,
   input wire Ma_in, Me_in, I_in,
   input wire [7:0] X, Y);

  wire G;
  wire L;
  wire E;

  comp_4bit U0(.A(X[3:0]), .B(Y[3:0]), .G_in(Ma_in), .L_in(Me_in), .E_in(I_in), .G_out(G),
.L_out(L), .E_out(E));

  comp_4bit U1(.A(X[7:4]), .B(Y[7:4]), .G_in(G), .L_in(L), .E_in(E), .G_out(Ma_out),
.L_out(Me_out), .E_out(I_out));

endmodule
```

2.e)

```
`timescale 1ns/1ps
`include "comparador.v"

module bancada();

  reg [7:0] A_TB, B_TB; //inputs
  reg G = 1'b0;
  reg L = 1'b0;
  reg E = 1'b1;
  wire MAIOR_TB, MENOR_TB, IGUAL_TB; //outputs

  comp_8bit OUT(.X(A_TB), .Y(B_TB), .Ma_in(G), .Me_in(L), .I_in(E), .Ma_out(MAIOR_TB),
.Me_out(MENOR_TB), .I_out(IGUAL_TB));

  initial
    begin

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

      A_TB=8'b11110000; B_TB=8'b00001111;
    #10 A_TB=8'b00010000; B_TB=8'b00010001;
    #10 A_TB=8'b11111111; B_TB=8'b11111111;

  end
endmodule
```