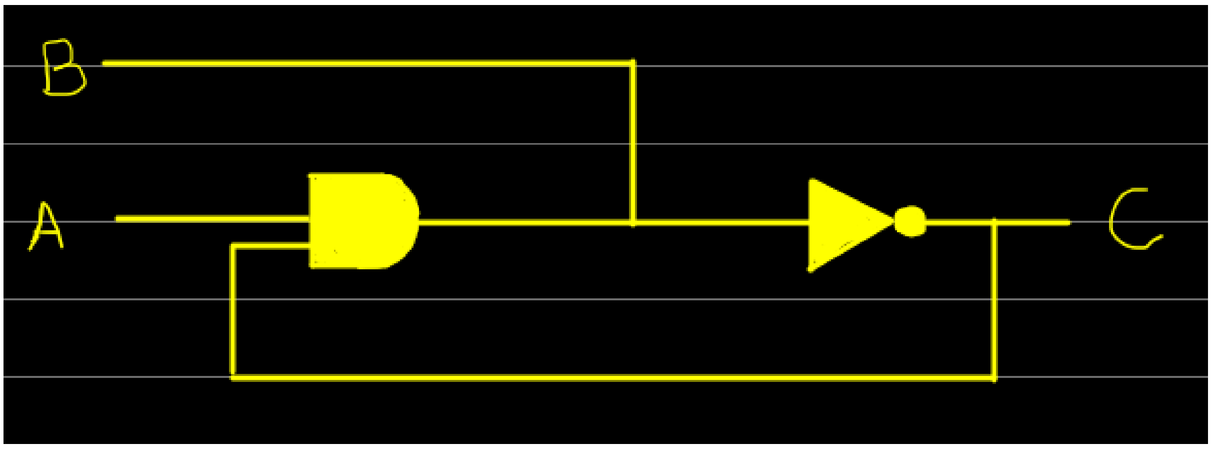
PROJETO I - Sistemas Digitais

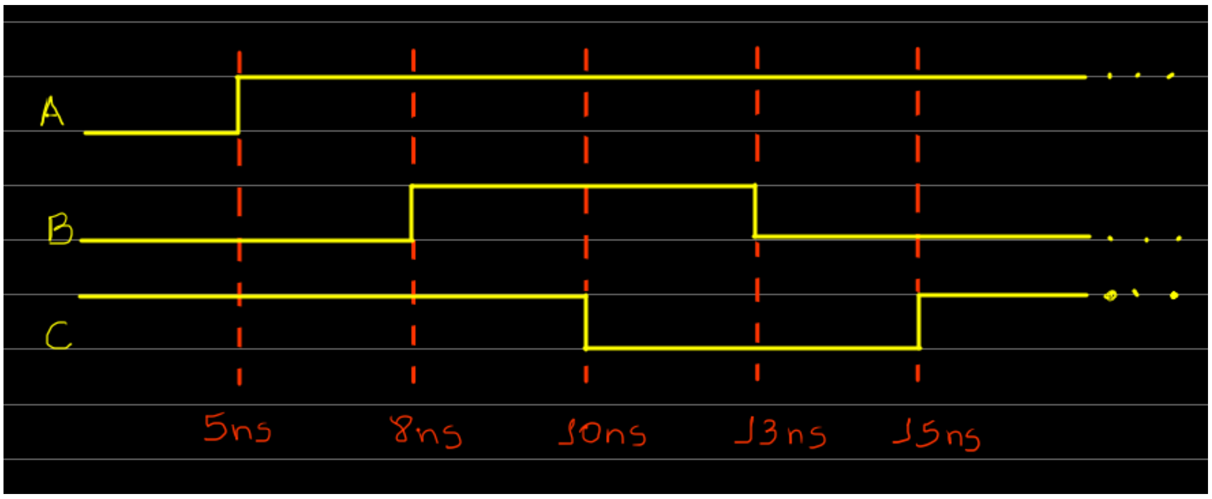
Alunos:

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

1.a)



b)



2)

```
`timescale 1ns/1ps
module circuito(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
```

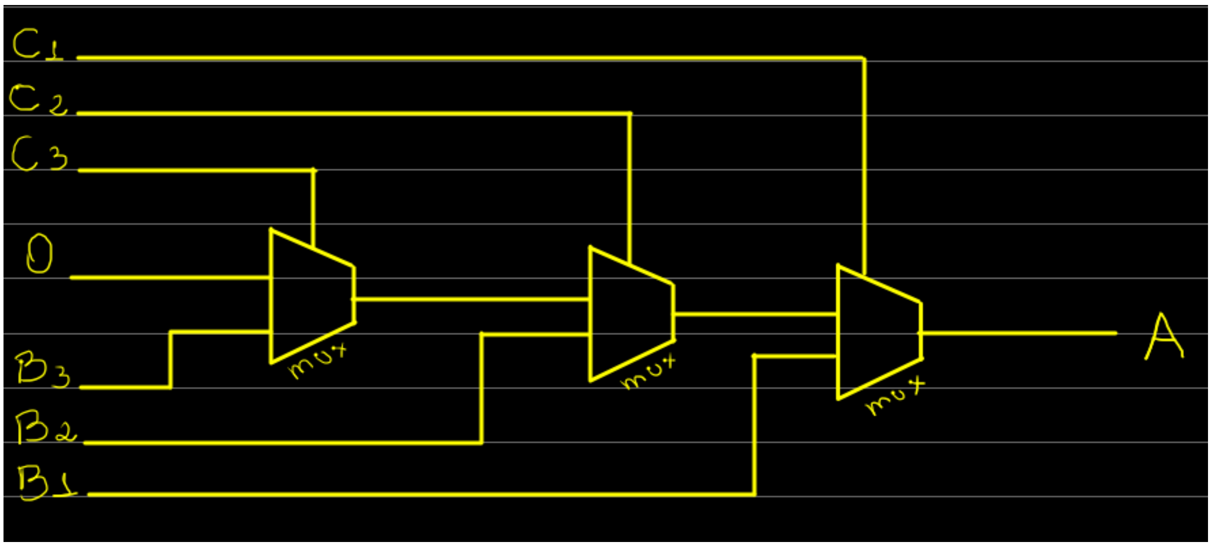
3.a) Primeiro Método:

```
module primeiro_metodo(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
```

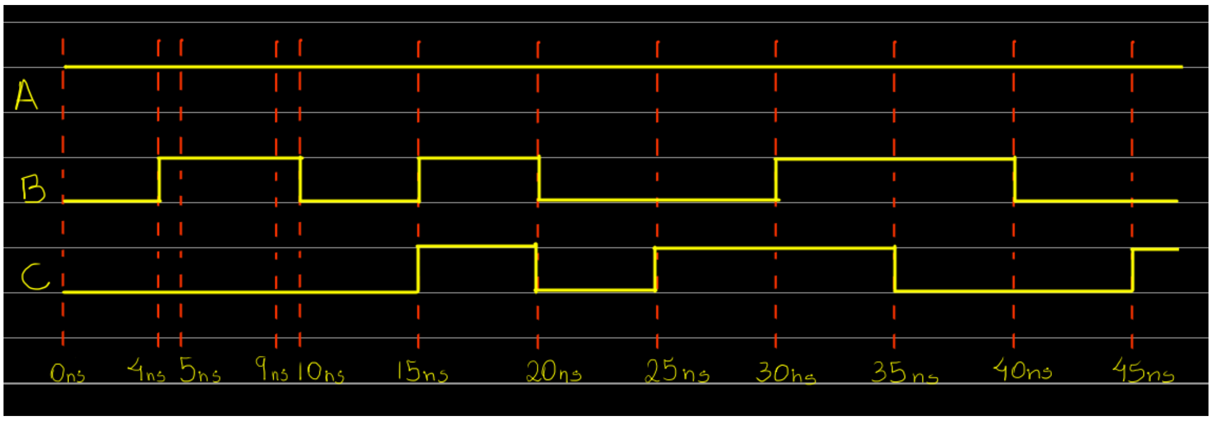
Segundo Método:

```
module segundo_metodo(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
```

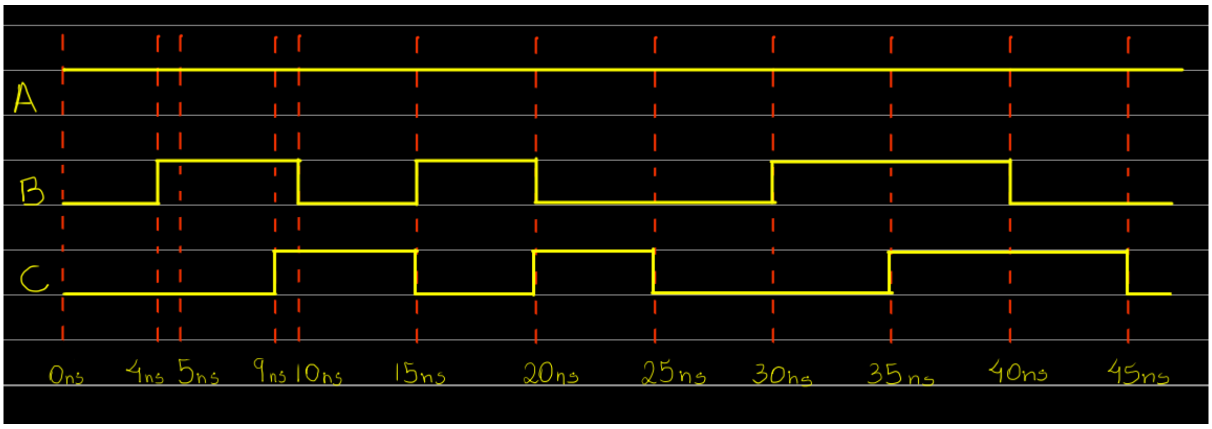
b)



4.a)



b)



5.a)

```
module systemx
  (output reg [1:0] Out,
  input wire A, B, C);

  reg [1:0] ROM8x2 [7:0];
  reg [2:0] I;

  always @(A, B, C) begin
    I =  { A , B , C };

    Out <= ROM8x2[I];
  end

  initial begin
    ROM8x2[0] <= 2'b11;
    ROM8x2[1] <= 2'b11;
    ROM8x2[2] <= 2'b10;
    ROM8x2[3] <= 2'b00;
    ROM8x2[4] <= 2'b01;
    ROM8x2[5] <= 2'b01;
    ROM8x2[6] <= 2'b10;
    ROM8x2[7] <= 2'b01;
  end
endmodule
```

b)

```
module counter4bits
  (output reg [2:0] Out,
  input wire A, B, C, D);

  reg [2:0] ROM16x3 [15:0];
  reg [3:0] I;

  always @(A, B, C, D) begin
    I =  { A , B , C , D};

    Out <= ROM16x3[I];
  end

  initial begin
    ROM16x3[0] <= 3'b000; //0000 = 0
    ROM16x3[1] <= 3'b001; //0001 = 1
    ROM16x3[2] <= 3'b001; //0010 = 1
    ROM16x3[3] <= 3'b010; //0011 = 2
    ROM16x3[4] <= 3'b001; //0100 = 1
    ROM16x3[5] <= 3'b010; //0101 = 2
    ROM16x3[6] <= 3'b010; //0110 = 2
    ROM16x3[7] <= 3'b011; //0111 = 3
    ROM16x3[8] <= 3'b001; //1000 = 1
    ROM16x3[9] <= 3'b010; //1001 = 2
    ROM16x3[10] <= 3'b010;//1010 = 2
    ROM16x3[11] <= 3'b011;//1011 = 3
    ROM16x3[12] <= 3'b010;//1100 = 2
    ROM16x3[13] <= 3'b011;//1101 = 3
    ROM16x3[14] <= 3'b011;//1110 = 3
    ROM16x3[15] <= 3'b100;//1111 = 4
  end
endmodule
```

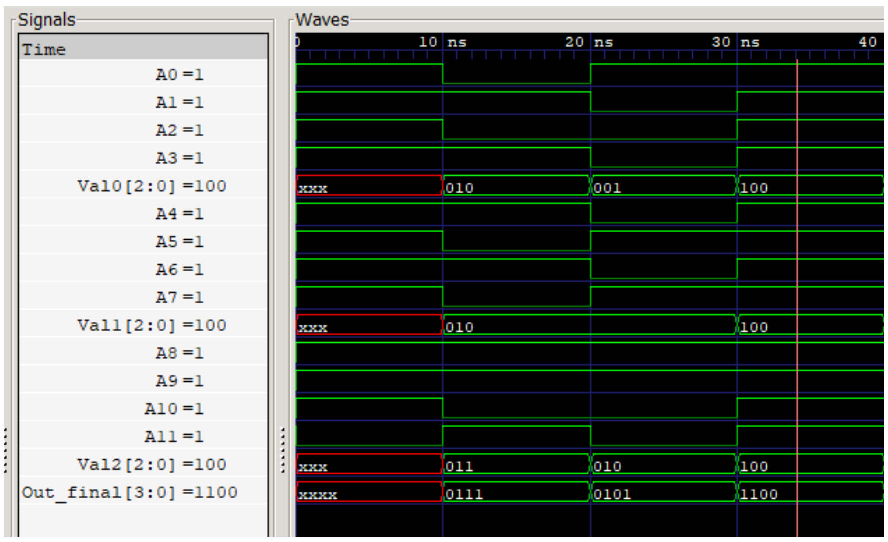
c)

```
module counter12bits(
  output reg [3:0] Out_final,
  input wire A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10,
A11);
```

```
  wire [2:0] Val0;
  wire [2:0] Val1;
  wire [2:0] Val2;

  counter4bits U0(.A(A3), .B(A2), .C(A1), .D(A0),
.Out(Val0));
  counter4bits U1(.A(A7), .B(A6), .C(A5), .D(A4),
.Out(Val1));
  counter4bits U2(.A(A11), .B(A10), .C(A9), .D(A8),
.Out(Val2));

  always @(Val0, Val1, Val2) begin
    Out_final <= Val0 + Val1 + Val2;
  end
endmodule

module counter4bits
  (output reg [2:0] Out,
  input wire A, B, C, D);

  reg [2:0] ROM16x3 [15:0];
  reg [3:0] I;

  always @(A, B, C, D) begin
    I =  { A , B , C , D};

    Out <= ROM16x3[I];
  end

  initial begin
    ROM16x3[0] <= 3'b000; //0000 = 0
    ROM16x3[1] <= 3'b001; //0001 = 1
    ROM16x3[2] <= 3'b001; //0010 = 1
    ROM16x3[3] <= 3'b010; //0011 = 2
    ROM16x3[4] <= 3'b001; //0100 = 1
    ROM16x3[5] <= 3'b010; //0101 = 2
    ROM16x3[6] <= 3'b010; //0110 = 2
    ROM16x3[7] <= 3'b011; //0111 = 3
    ROM16x3[8] <= 3'b001; //1000 = 1
    ROM16x3[9] <= 3'b010; //1001 = 2
    ROM16x3[10] <= 3'b010;//1010 = 2
    ROM16x3[11] <= 3'b011;//1011 = 3
    ROM16x3[12] <= 3'b010;//1100 = 2
    ROM16x3[13] <= 3'b011;//1101 = 3
    ROM16x3[14] <= 3'b011;//1110 = 3
    ROM16x3[15] <= 3'b100;//1111 = 4
  end
endmodule
```

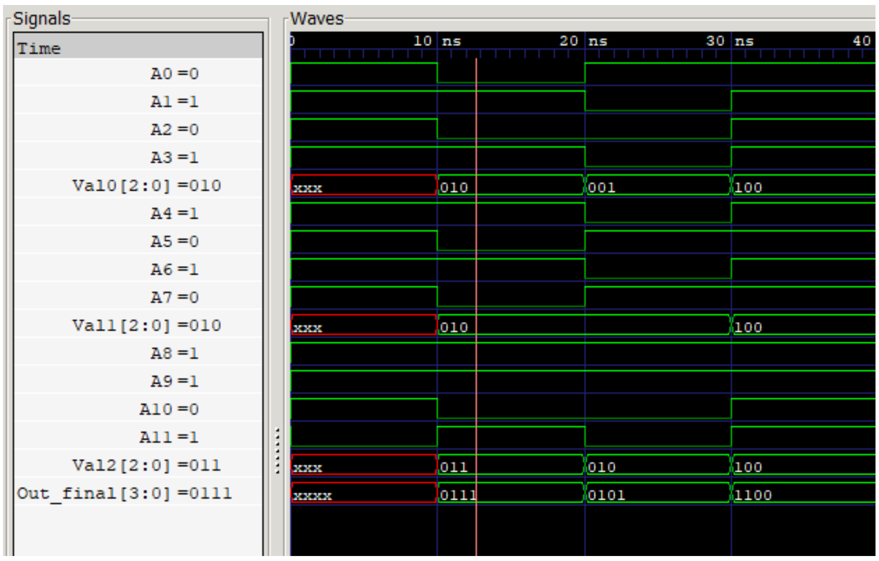
d) ("Out_final" mostra o valor em binário da quantidade de 1s de cada cadeia)

[111111111111]:



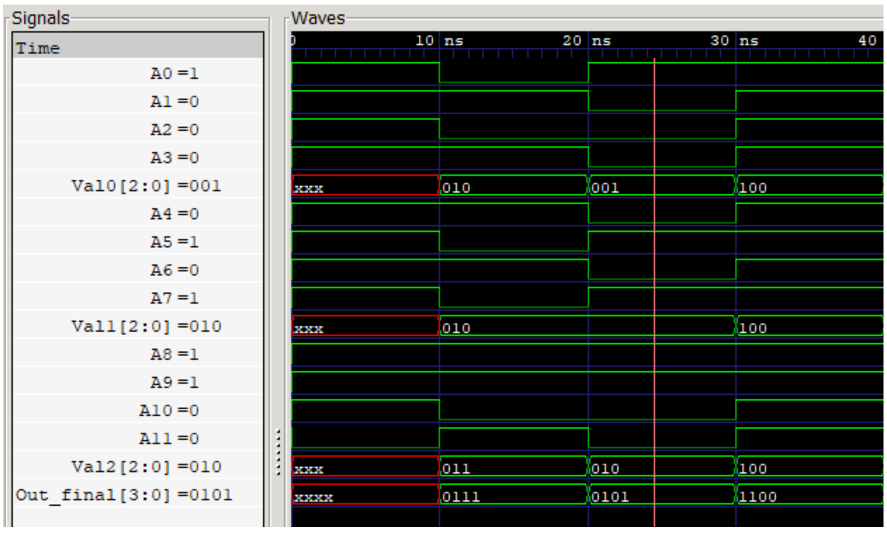
```
Time
A0 =1
A1 =1
A2 =1
A3 =1
Val0[2:0] =100
A4 =1
A5 =1
A6 =1
A7 =1
Val1[2:0] =100
A8 =1
A9 =1
A10 =1
A11 =1
Val2[2:0] =100
Out_final[3:0] =1100
```

[010110101101]:



```
Time
A0 =0
A1 =1
A2 =0
A3 =1
Val0[2:0] =010
A4 =1
A5 =0
A6 =1
A7 =0
Val1[2:0] =010
A8 =1
A9 =1
A10 =0
A11 =1
Val2[2:0] =011
Out_final[3:0] =0111
```

[100001011100]:



```
Time
A0 =1
A1 =0
A2 =0
A3 =0
Val0[2:0] =001
A4 =0
A5 =1
A6 =0
A7 =1
Val1[2:0] =010
A8 =1
A9 =1
A10 =0
A11 =0
Val2[2:0] =010
Out_final[3:0] =0101
```