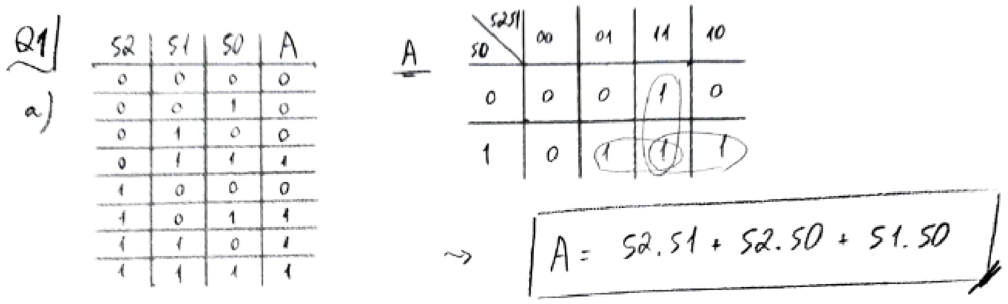
Instituto Tecnológico de Aeronáutica
Departamento de Eletrônica

EEA25 - Prova 1

Daniel Araujo Cavassani

Prof. André da Fontoura Ponchet
São José dos Campos, São Paulo, 22/09/2023

1. **Alarm System**
   a. **Tabela Verdade + Equação Lógica**



Q1

a)

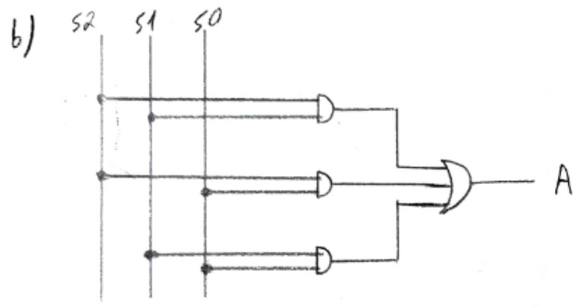
S2	S1	S0	A
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A

S0 \ S2S1	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$\rightsquigarrow$ $\boxed{A = S2.S1 + S2.S0 + S1.S0}$

   b. **Circuito**



   c. **Código Verilog**

CÓDIGO DA LÓGICA:

```
module AlarmSystem(
    input S0,
    input S1,
    input S2,
    output A
);

  wire and1, and2, and3;

  // AND gates
  and(and1, S0, S1);
  and(and2, S1, S2);
  and(and3, S0, S2);

  // OR gate
  or(A, and1, and2, and3);

endmodule
```

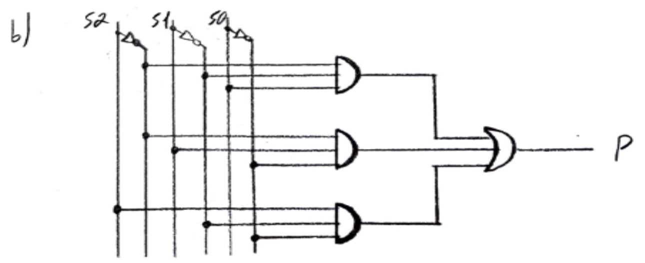
2. **Patrol System**
    a. **Tabela Verdade + Equação Lógica**

Q2
a)

S2	S1	S0	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$\rightsquigarrow$ $\boxed{P = \overline{S2}.\overline{S1}.S0 + \overline{S2}.S1.\overline{S0} + S2.\overline{S1}.\overline{S0}}$

    b. **Circuito**



    c. **Código Verilog**

CÓDIGO DA LÓGICA:

```
module PatrolSystem(
    input S0,
    input S1,
    input S2,
    output P
);
  wire and1, and2, and3;
  wire notS0, notS1, notS2;

  // NOT
  not(notS0, S0);
  not(notS1, S1);
  not(notS2, S2);

  // AND
  and(and1, notS0, notS1, S2);
  and(and2, notS0, S1, notS2);
  and(and3, S0, notS1, notS2);

  // OR
  or(P, and1, and2, and3);
endmodule
```

3. **Simplified Circuit**

   a. **Equação Simplificada**

Q3

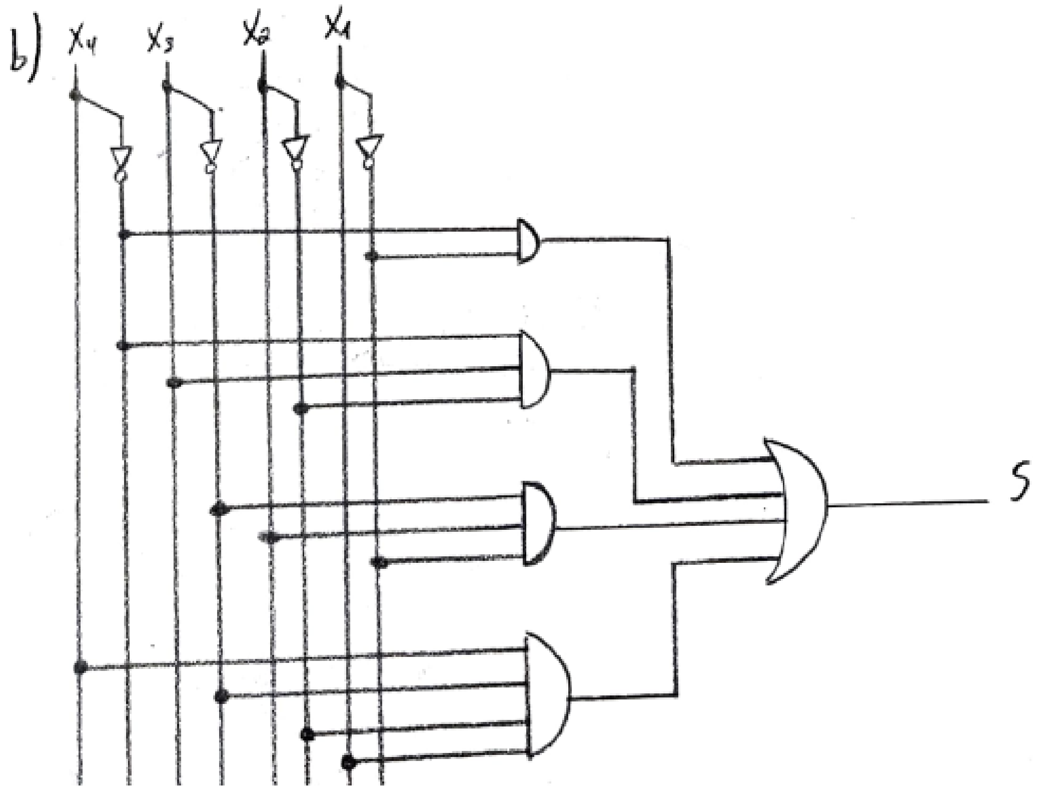
a) Mapa de Karnaugh :

S

$X_2X_1 \backslash X_4X_3$	00	01	11	10
00	1	1	0	0
01	0	1	0	1
11	0	0	0	0
10	1	1	0	1

$$\rightsquigarrow \boxed{S = \overline{X_4}.\overline{X_1} + \overline{X_4}.X_3.\overline{X_2} + \overline{X_3}.X_2.\overline{X_1} + X_4\overline{X_3}\,\overline{X_2}\,X_1}$$

   b. **Circuito**



c. **Código Verilog**

CÓDIGO DA LÓGICA:

```
module SimplifiedCircuit(
    input X1,
    input X2,
    input X3,
    input X4,
    output S
);

  wire and1, and2, and3, and4;
  wire notX1, notX2, notX3, notX4;

  // NOT
  not(notX1, X1);
  not(notX2, X2);
  not(notX3, X3);
  not(notX4, X4);

  // AND
  and(and1, notX4, notX1);
  and(and2, notX4, X3, notX2);
  and(and3, notX3, X2, notX1);
  and(and4, X4, notX3, notX2, X1);

  // OR
  or(S, and1, and2, and3, and4);

endmodule
```

d. **Código Test Verilog + Sinais no GtkWave**

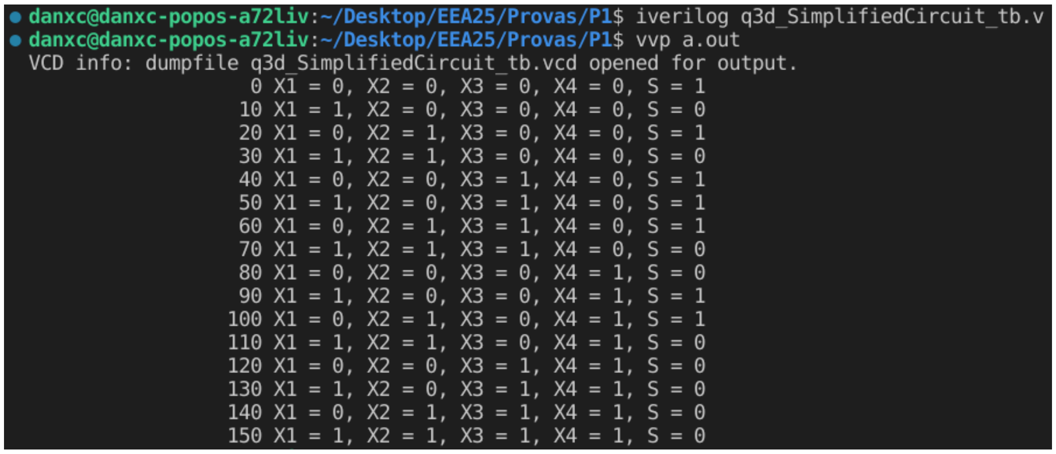
CÓDIGO DO TESTBENCH:

```
`include "q3c_SimplifiedCircuit.v"
`timescale 1ns/1ps

module SimplifiedCircuit_tb;

```

```verilog
// Inputs
reg X1, X2, X3, X4;

// Outputs
wire S;

SimplifiedCircuit uut (
    .X1(X1),
    .X2(X2),
    .X3(X3),
    .X4(X4),
    .S(S)
);

initial begin
  $dumpfile("q3d_SimplifiedCircuit_tb.vcd");
  $dumpvars(0, SimplifiedCircuit_tb);
end

// Monitor the variables
initial $monitor($time,
                 " X1 = %b, X2 = %b, X3 = %b, X4 = %b, S = %b",
                 X1, X2, X3, X4, S);

initial begin
    // Testa todas as combinações
    {X4, X3, X2, X1} = 4'b0000; #10;
    {X4, X3, X2, X1} = 4'b0001; #10;
    {X4, X3, X2, X1} = 4'b0010; #10;
    {X4, X3, X2, X1} = 4'b0011; #10;
    {X4, X3, X2, X1} = 4'b0100; #10;
    {X4, X3, X2, X1} = 4'b0101; #10;
    {X4, X3, X2, X1} = 4'b0110; #10;
    {X4, X3, X2, X1} = 4'b0111; #10;
    {X4, X3, X2, X1} = 4'b1000; #10;
    {X4, X3, X2, X1} = 4'b1001; #10;
    {X4, X3, X2, X1} = 4'b1010; #10;
    {X4, X3, X2, X1} = 4'b1011; #10;
    {X4, X3, X2, X1} = 4'b1100; #10;
```

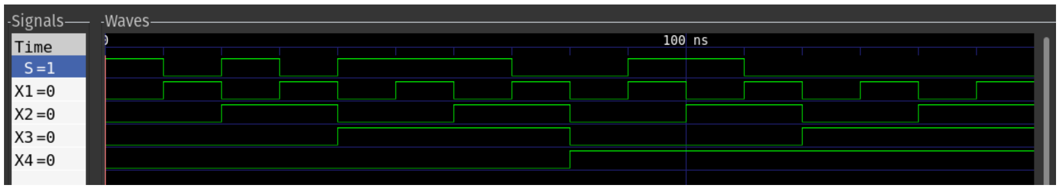
```
      {X4, X3, X2, X1} = 4'b1101; #10;
      {X4, X3, X2, X1} = 4'b1110; #10;
      {X4, X3, X2, X1} = 4'b1111; #10;
    $finish;
  end

endmodule
```

OUTPUT:



```
● danxc@danxc-popos-a72liv:~/Desktop/EEA25/Provas/P1$ iverilog q3d_SimplifiedCircuit_tb.v
● danxc@danxc-popos-a72liv:~/Desktop/EEA25/Provas/P1$ vvp a.out
  VCD info: dumpfile q3d_SimplifiedCircuit_tb.vcd opened for output.
                     0 X1 = 0, X2 = 0, X3 = 0, X4 = 0, S = 1
                    10 X1 = 1, X2 = 0, X3 = 0, X4 = 0, S = 0
                    20 X1 = 0, X2 = 1, X3 = 0, X4 = 0, S = 1
                    30 X1 = 1, X2 = 1, X3 = 0, X4 = 0, S = 0
                    40 X1 = 0, X2 = 0, X3 = 1, X4 = 0, S = 1
                    50 X1 = 1, X2 = 0, X3 = 1, X4 = 0, S = 1
                    60 X1 = 0, X2 = 1, X3 = 1, X4 = 0, S = 1
                    70 X1 = 1, X2 = 1, X3 = 1, X4 = 0, S = 0
                    80 X1 = 0, X2 = 0, X3 = 0, X4 = 1, S = 0
                    90 X1 = 1, X2 = 0, X3 = 0, X4 = 1, S = 1
                   100 X1 = 0, X2 = 1, X3 = 0, X4 = 1, S = 1
                   110 X1 = 1, X2 = 1, X3 = 0, X4 = 1, S = 0
                   120 X1 = 0, X2 = 0, X3 = 1, X4 = 1, S = 0
                   130 X1 = 1, X2 = 0, X3 = 1, X4 = 1, S = 0
                   140 X1 = 0, X2 = 1, X3 = 1, X4 = 1, S = 0
                   150 X1 = 1, X2 = 1, X3 = 1, X4 = 1, S = 0
```

SINAIS NO GTKWAVE:



4. **Full Adder**

a. **Equações Lógicas**

Q4

a) De acordo com a Tabela Verdade do Full Adder:

$\rightsquigarrow$ $\boxed{Cout_i = a_i . b_i + Cin_{i-1} . (a_i \oplus b_i)}$ //

$\rightsquigarrow$ $\boxed{Sum_i = a_i \oplus b_i \oplus Cin_{i-1}}$ //

b. **Código Verilog (estrutural)**

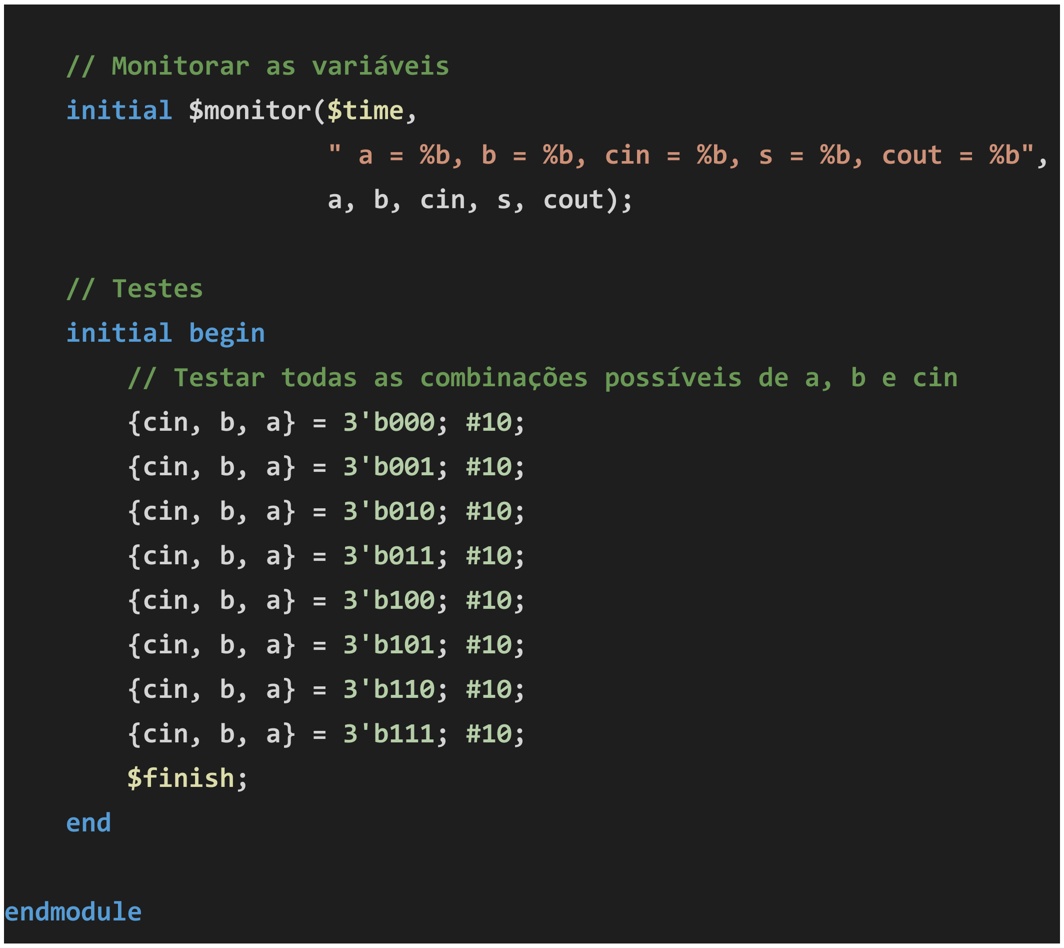
CÓDIGO DA LÓGICA:

```
module FullAdder(
    input a,      // Entrada a
    input b,      // Entrada b
    input cin,    // Carry in

    output s,     // Soma
    output cout   // Carry out
);


    // Fios intermediários
    wire w1, w2, w3;

    // XOR para calcular a XOR entre a e b
    xor u0 (w1, a, b);

    // XOR para calcular a soma final (sum)
    xor u1 (s, w1, cin);

    // AND para calcular a parte do carry out com cin e XOR de a e b
    and u2 (w2, cin, w1);
    // AND para calcular a parte do carry out com a e b
    and u3 (w3, a, b);

    // OR para calcular o carry out final
    or u4 (cout, w2, w3);

endmodule
```

c. **Código Test Verilog + Sinais no GtkWave**

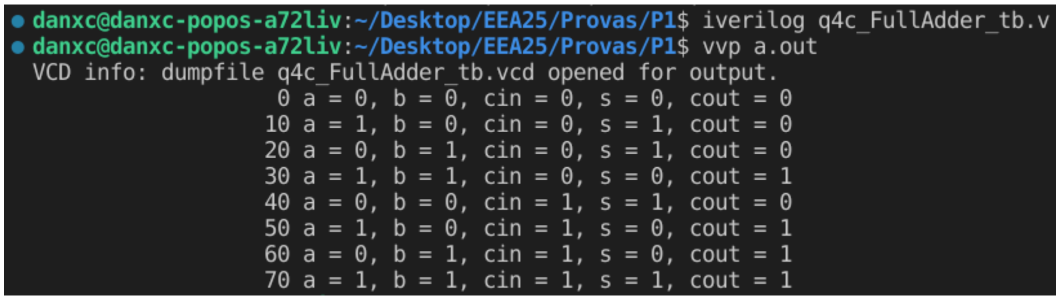
CÓDIGO DO TESTBENCH:

```
`include "q4b_FullAdder.v"
`timescale 1ns/1ps

module FullAdder_tb;

    // Entradas
    reg a;
    reg b;
    reg cin;

    // Saídas
    wire s;
    wire cout;

    // Instanciar o módulo FullAdder
    FullAdder uut (
        .a(a),
        .b(b),
        .cin(cin),
        .s(s),
        .cout(cout)
    );

    initial begin
        $dumpfile("q4c_FullAdder_tb.vcd");
        $dumpvars(0, FullAdder_tb);
    end
```



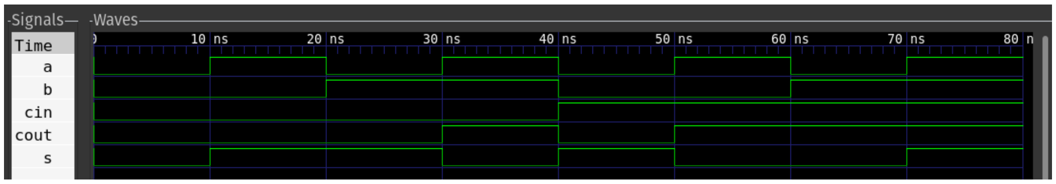
```
    // Monitorar as variáveis
    initial $monitor($time,
                     " a = %b, b = %b, cin = %b, s = %b, cout = %b",
                     a, b, cin, s, cout);

    // Testes
    initial begin
        // Testar todas as combinações possíveis de a, b e cin
        {cin, b, a} = 3'b000; #10;
        {cin, b, a} = 3'b001; #10;
        {cin, b, a} = 3'b010; #10;
        {cin, b, a} = 3'b011; #10;
        {cin, b, a} = 3'b100; #10;
        {cin, b, a} = 3'b101; #10;
        {cin, b, a} = 3'b110; #10;
        {cin, b, a} = 3'b111; #10;
        $finish;
    end

endmodule
```

OUTPUT:



```
● danxc@danxc-popos-a72liv:~/Desktop/EEA25/Provas/P1$ iverilog q4c_FullAdder_tb.v
● danxc@danxc-popos-a72liv:~/Desktop/EEA25/Provas/P1$ vvp a.out
  VCD info: dumpfile q4c_FullAdder_tb.vcd opened for output.
                     0 a = 0, b = 0, cin = 0, s = 0, cout = 0
                    10 a = 1, b = 0, cin = 0, s = 1, cout = 0
                    20 a = 0, b = 1, cin = 0, s = 1, cout = 0
                    30 a = 1, b = 1, cin = 0, s = 0, cout = 1
                    40 a = 0, b = 0, cin = 1, s = 1, cout = 0
                    50 a = 1, b = 0, cin = 1, s = 0, cout = 1
                    60 a = 0, b = 1, cin = 1, s = 0, cout = 1
                    70 a = 1, b = 1, cin = 1, s = 1, cout = 1
```

SINAIS NO GTKWAVE:



5. **Clock Divider**

CÓDIGO DA LÓGICA:

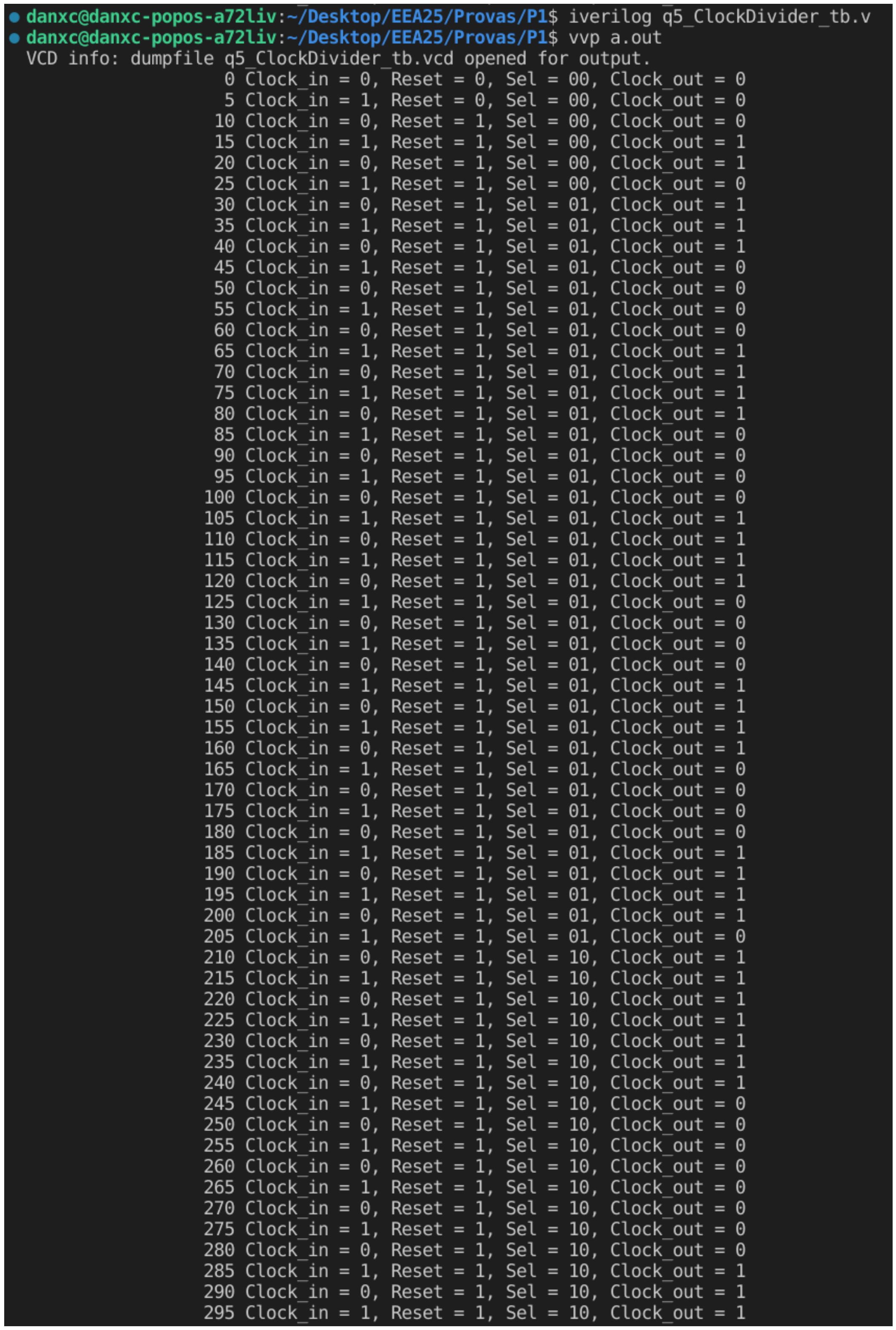
```
// Módulo do D-flip-flop
module d_flip_flop (
    output reg Q, Qn,
    input wire Clock, Reset, D
);

    // Inicialização dos estados dos flip-flops
    initial begin
        Q = 1'b0;
        Qn = 1'b1;
    end

    always @(posedge Clock or negedge Reset) begin
        if (!Reset) begin
            Q <= 1'b0;
            Qn <= 1'b1;
        end else begin
            Q <= D;
            Qn <= ~D;
        end
    end
endmodule

// Módulo do Divisor de Clock Programável
module ClockDivider (
    output wire Clock_out,
    input wire Clock_in,
    input wire Reset,
    input wire [1:0] Sel
);

    wire d2, d4, d8, d16;       // Saídas dos flip-flops
    wire d2n, d4n, d8n, d16n;   // Saídas Inversas dos flip-flops

    // Instanciando D-flip-flops para o Ripple Counter
    d_flip_flop ff1 (.Q(d2), .Qn(d2n), .Clock(Clock_in),
```

```verilog
                     .Reset(Reset), .D(d2n));
    d_flip_flop ff2 (.Q(d4), .Qn(d4n), .Clock(d2n), .Reset(Reset),
                     .D(d4n));
    d_flip_flop ff3 (.Q(d8), .Qn(d8n), .Clock(d4n), .Reset(Reset),
                     .D(d8n));
    d_flip_flop ff4 (.Q(d16), .Qn(d16n), .Clock(d8n), .Reset(Reset),
                     .D(d16n));

    // Instanciando o Multiplexador
    assign Clock_out = (Sel == 2'b00) ? d2 :
                       (Sel == 2'b01) ? d4 :
                       (Sel == 2'b10) ? d8 : d16;
endmodule
```

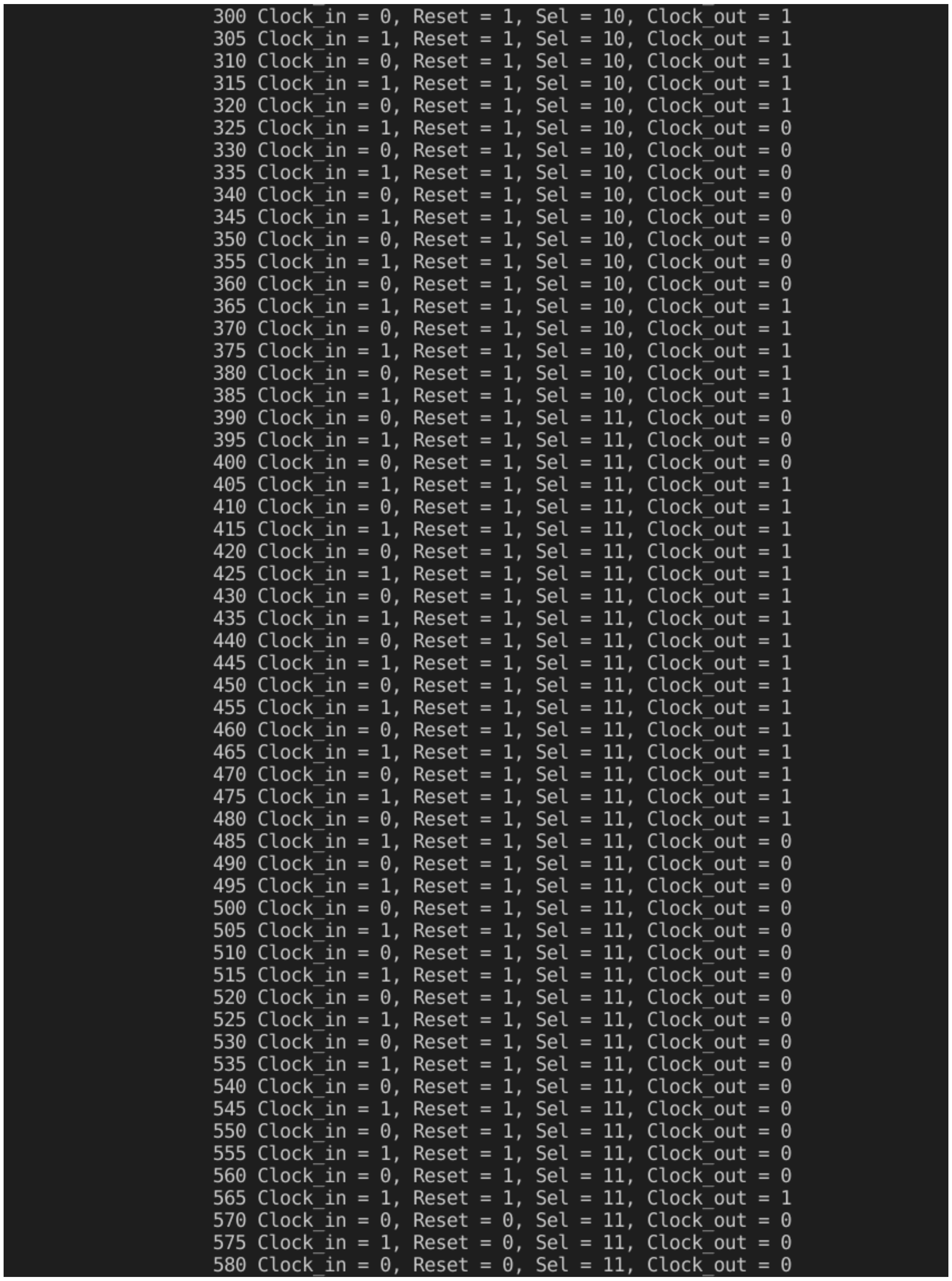
CÓDIGO DO TESTBENCH:

```verilog
`include "q5_ClockDivider.v"
`timescale 1ns/1ps

module ClockDivider_tb;

    // Entradas
    reg Clock_in;
    reg Reset;
    reg [1:0] Sel;

    // Saídas
    wire Clock_out;

    // Instanciar o módulo ClockDivider
    ClockDivider uut (
        .Clock_in(Clock_in),
        .Reset(Reset),
        .Sel(Sel),
        .Clock_out(Clock_out)
    );

    initial begin
        $dumpfile("q5_ClockDivider_tb.vcd");
        $dumpvars(0, ClockDivider_tb);
    end

    // Monitorar as variáveis
    initial $monitor($time,
                     " Clock_in = %b, Reset = %b, Sel = %b,
                     Clock_out = %b",
                     Clock_in, Reset, Sel, Clock_out);

    // Gerar Clock
    initial forever #5 Clock_in = ~Clock_in;

    // Testes
    initial begin
        Reset = 1'b0;  // Ativar o Reset (nível baixo)
        Sel = 2'b00;
        Clock_in = 1'b0;  // Inicializar o Clock

        #10 Reset = 1'b1;  // Liberar o Reset (nível alto)

        // Testar diferentes seleções e esperar 16 ciclos de clock
para cada um
        #20 Sel = 2'b01; #160;
        #20 Sel = 2'b10; #160;
        #20 Sel = 2'b11; #160;

        // Reativar o Reset
        #20 Reset = 1'b0;  // Ativar o Reset novamente (nível baixo)

        // Finalizar a simulação
        #10 $finish;
    end

endmodule
```

OUTPUT:

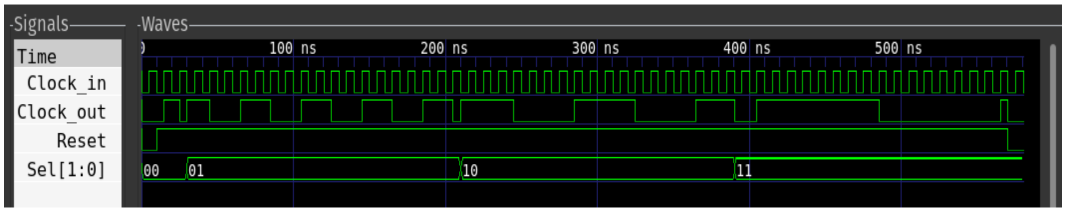


```
● danxc@danxc-popos-a72liv:~/Desktop/EEA25/Provas/P1$ iverilog q5_ClockDivider_tb.v
● danxc@danxc-popos-a72liv:~/Desktop/EEA25/Provas/P1$ vvp a.out
  VCD info: dumpfile q5_ClockDivider_tb.vcd opened for output.
                     0 Clock_in = 0, Reset = 0, Sel = 00, Clock_out = 0
                     5 Clock_in = 1, Reset = 0, Sel = 00, Clock_out = 0
                    10 Clock_in = 0, Reset = 1, Sel = 00, Clock_out = 0
                    15 Clock_in = 1, Reset = 1, Sel = 00, Clock_out = 1
                    20 Clock_in = 0, Reset = 1, Sel = 00, Clock_out = 1
                    25 Clock_in = 1, Reset = 1, Sel = 00, Clock_out = 0
                    30 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                    35 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                    40 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                    45 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                    50 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                    55 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                    60 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                    65 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                    70 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                    75 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                    80 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                    85 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                    90 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                    95 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                   100 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                   105 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                   110 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                   115 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                   120 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                   125 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                   130 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                   135 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                   140 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                   145 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                   150 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                   155 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                   160 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                   165 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                   170 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                   175 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                   180 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 0
                   185 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                   190 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                   195 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 1
                   200 Clock_in = 0, Reset = 1, Sel = 01, Clock_out = 1
                   205 Clock_in = 1, Reset = 1, Sel = 01, Clock_out = 0
                   210 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
                   215 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
                   220 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
                   225 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
                   230 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
                   235 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
                   240 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
                   245 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
                   250 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
                   255 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
                   260 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
                   265 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
                   270 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
                   275 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
                   280 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
                   285 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
                   290 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
                   295 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
```



```
300 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
305 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
310 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
315 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
320 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
325 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
330 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
335 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
340 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
345 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
350 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
355 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 0
360 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 0
365 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
370 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
375 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
380 Clock_in = 0, Reset = 1, Sel = 10, Clock_out = 1
385 Clock_in = 1, Reset = 1, Sel = 10, Clock_out = 1
390 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
395 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
400 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
405 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
410 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
415 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
420 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
425 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
430 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
435 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
440 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
445 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
450 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
455 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
460 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
465 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
470 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
475 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
480 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 1
485 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
490 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
495 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
500 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
505 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
510 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
515 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
520 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
525 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
530 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
535 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
540 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
545 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
550 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
555 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 0
560 Clock_in = 0, Reset = 1, Sel = 11, Clock_out = 0
565 Clock_in = 1, Reset = 1, Sel = 11, Clock_out = 1
570 Clock_in = 0, Reset = 0, Sel = 11, Clock_out = 0
575 Clock_in = 1, Reset = 0, Sel = 11, Clock_out = 0
580 Clock_in = 0, Reset = 0, Sel = 11, Clock_out = 0
```

SINAIS NO GTKWAVE:


Signals
Waves
Time
Clock_in
Clock_out
Reset
Sel[1:0]
100 ns
200 ns
300 ns
400 ns
500 ns