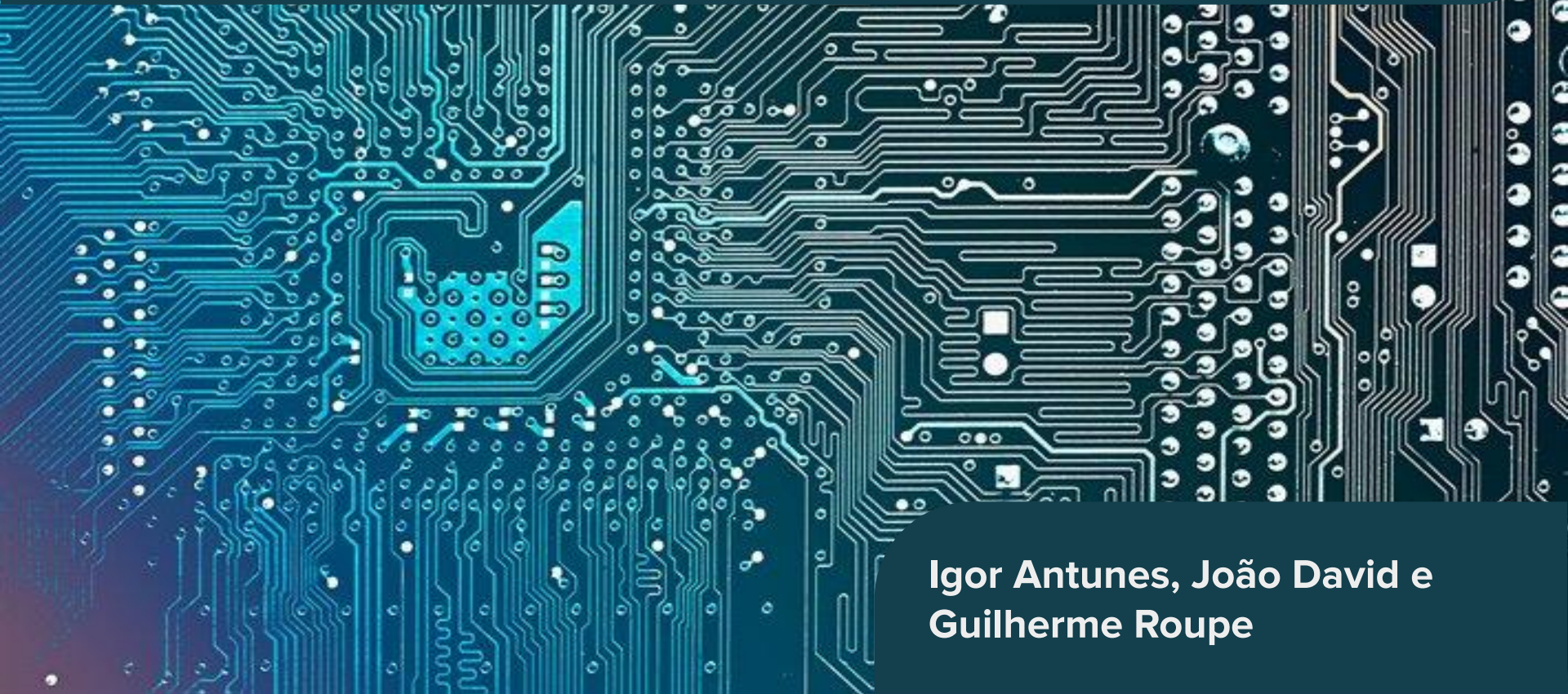
Circuitos Lógicos:

Máquina de estados, contador 4 Bits e 8 bits em VHDL



Igor Antunes, João David e Guilherme Roupe

Questão 2:

Implemente em VHDL um contador com de 4 bits crescente/descrescente, com o sentido controlado por um sinal UP_DN e cuja contagem pode ser suspensa pelo sinal de HOLD. O contador gera uma saída RCO toda vez que houver "overflow" ou "underflow" na contagem e deve ter ainda um sinal de CLR assíncrono para zerar a saída. Implemente um arquivo de testes (TESTBENCH) em VHDL para verificar o funcionamento do seu projeto. Simule com GHDL e GTKWAVE e apresente um relatório com os arquivos em VHDL e os resultados da simulação obtidos. Adicionalmente, apresente o código completo em VHDL de um contador de 8 bits utilizando PORT MAP e referenciando duas instâncias do contador de 4 bits já projetado.

```
15  library IEEE;
16  use IEEE.STD_LOGIC_1164.ALL;
17  use ieee.numeric_std.all;
18
19  entity contador_binario IS
20      port (CLK, CLR, UP_DN, HOLD : in std_logic;
21      Q : out std_logic_vector(3 downto 0);
22      RCO : out std_logic);
23  end contador_binario;
24
```

A arquitetura da entidade

```
25  architecture Behavior of contador_binario is
26
27      signal COUNTER : unsigned(3 downto 0) :="0000";
28
29      begin
30
31      process(CLK, CLR)
32      begin
33          if (CLR = '1') then
34              COUNTER <= "0000";
35          else
36              if(rising_edge(CLK)) then
37                  if (HOLD = '0')  then
38                      if (UP_DN = '0') then
39                          COUNTER <= COUNTER +1;
40                          if (COUNTER = "1111") then
41                              RCO <= '1';
42                          end if;
43                      else
44                          COUNTER <= COUNTER -1;
45                          if (COUNTER = "0000") then
46                              RCO <= '1';
47                          end if;
48                      end if;
49                  end if;
50              else
51                  if (CLK = '0') then
52                      RCO <= '0';
53                  end if;
54              end if;
55          end if;
56      end process;
57      Q <= std_logic_vector(COUNTER);
58
59  end architecture Behavior;
```

Vetor “unsigned”, da biblioteca numeric_std

Implementação do CLR

Ativação na subida do clock

Função HOLD

Aplicação do sinal Up_Down

Incremento do vetor unsigned

Funcionamento do RCO

Conversão para vetor lógico e aplicação à saída

Testbench: componente e port map

```
15  library IEEE;
16  use IEEE.STD_LOGIC_1164.ALL;
17  use ieee.numeric_std.all;
18
19  entity contador_binario_TB is
20  end entity contador_binario_TB;
21
22  architecture TB of contador_binario_TB is
23
24      signal CLK, CLR, UP_DN, HOLD, RCO : std_logic :='0';
25      signal Q : std_logic_vector(3 downto 0) :="0000";
26
27      component contador_binario
28          port (CLK, CLR, UP_DN, HOLD : in std_logic;
29          Q : out std_logic_vector(3 downto 0);
30          RCO : out std_logic);
31      end component;
32
33      begin
34
35      DUT : contador_binario port map (CLK => CLK,
36          CLR => CLR, UP_DN => UP_DN, HOLD => HOLD,
37          Q => Q, RCO => RCO);
38
```

Declaração de entidade do teste, sem portas

Entradas e saídas são declaradas como sinais na arquitetura

Declaração do componente com suas portas

PORT MAP

Os processos do testbench

```
39      clock : process
40      begin
41          CLK <= '0';
42          wait for 10 ns;
43          CLK <= '1';
44          wait for 10 ns;
45      end process;
46
47      inputCLR : process
48      begin
49          wait for 500 ns;
50          CLR <= '1';
51          wait for 20 ns;
52          CLR <= '0';
53      end process;
54
55      inputHOLD : process
56      begin
57          wait for 1000 ns;
58          HOLD <= '1';
59          wait for 200 ns;
60          HOLD <= '0';
61      end process;
62
63      inputUP_DN : process
64      begin
65          UP_DN <= '0';
66          wait for 1500 ns;
67          UP_DN <= '1';
68          wait for 1500 ns;
69      end process;
70
71  end architecture TB;
```

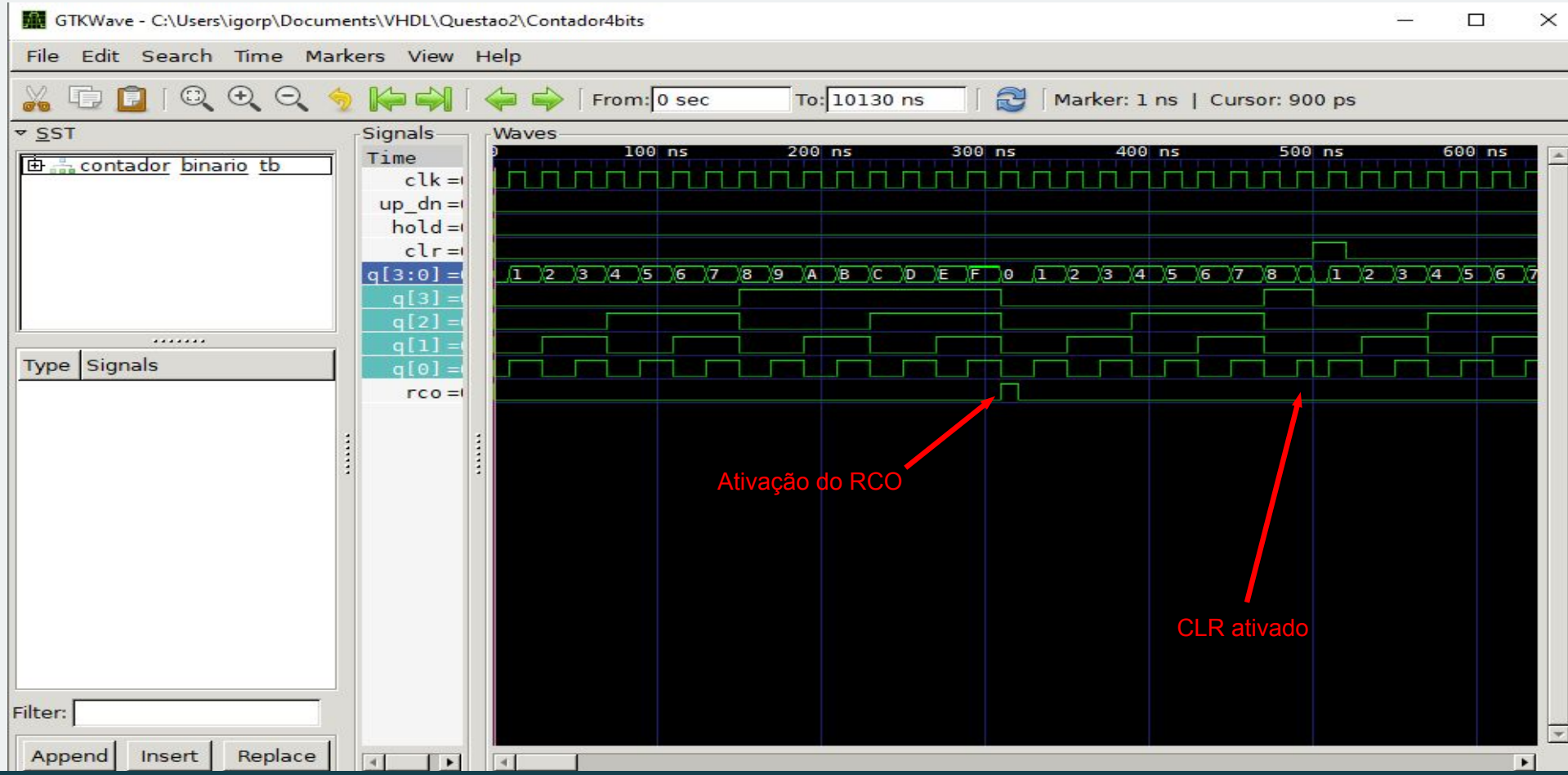
Operação do Clock

Simulação da ativação do Clear

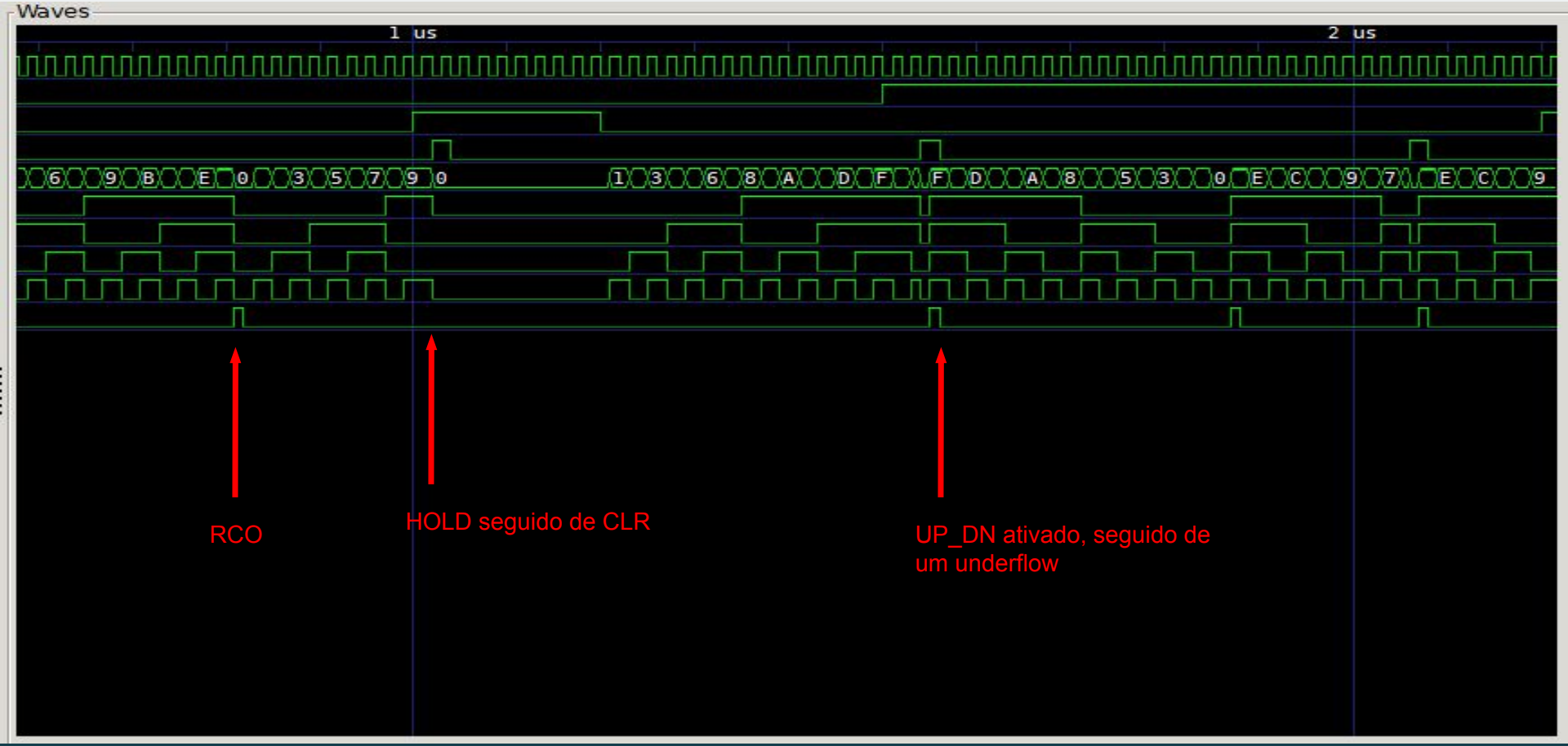
Simulação da ativação do Hold

Simulação da ativação do Up_Down

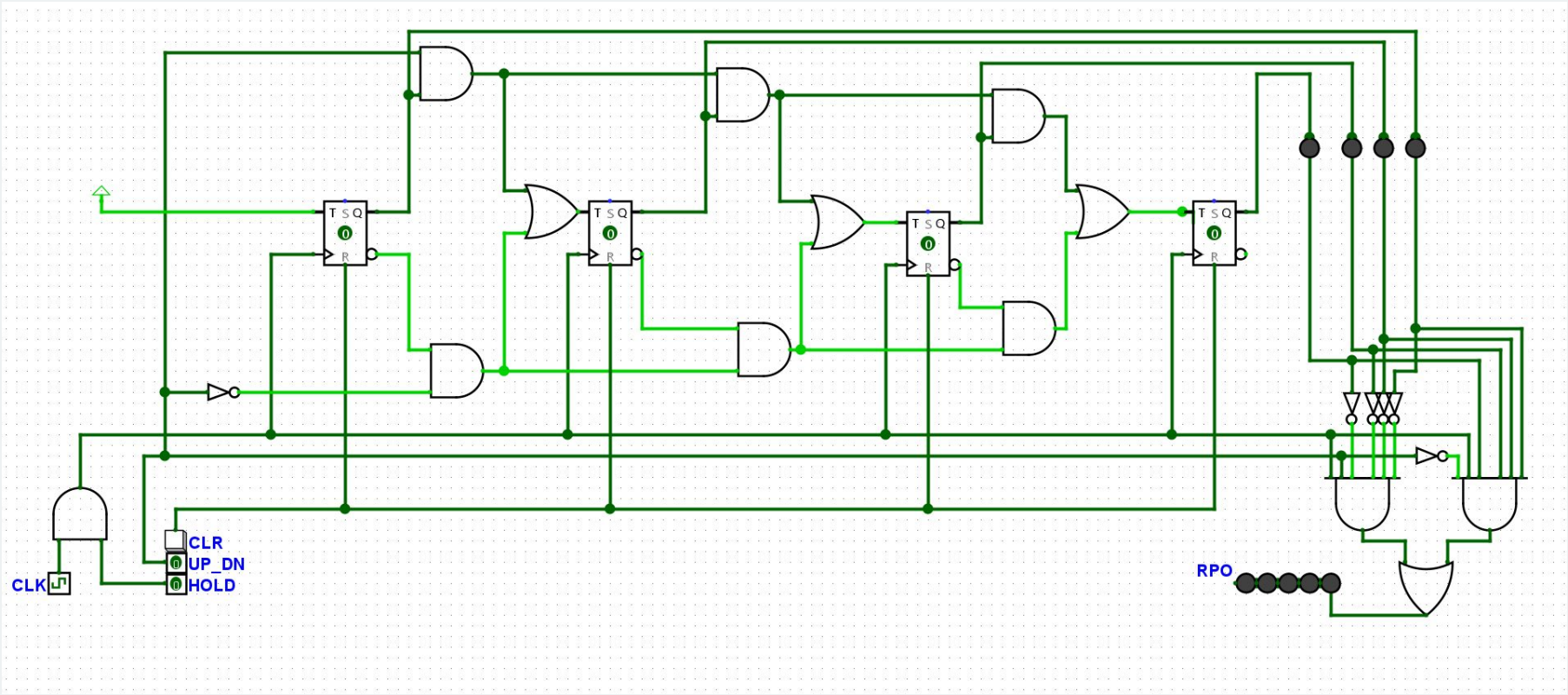
Simulação no GTKWAVE



Simulação no GTKWAVE



Construção no LogiSim-Evolution



Contador de 8 bits

```
16  library IEEE;
17  use IEEE.STD_LOGIC_1164.ALL;
18  use ieee.numeric_std.all;
19
20  entity contador_binarioV2 is
21      port (CLK, CLR, UP_DN, HOLD : in std_logic;
22      Q : out std_logic_vector(7 downto 0);
23      RCO : out std_logic);
24  end entity contador_binarioV2;
25
26  architecture Behavior of contador_binarioV2 is
27
28      component contador_binario
29          port (CLK, CLR, UP_DN, HOLD : in std_logic;
30          Q : out std_logic_vector(3 downto 0);
31          RCO : out std_logic);
32      end component;
33
34      signal RCO1, RCO2, CONNECT : std_logic :='0';
35      signal QTEMP : std_logic_vector(7 downto 0) :="00000000";
36
```

Nova entidade, com portas

Contador anterior como componente

Contador de 8 bits

```
37      begin
38
39      DUT1 : contador_binario port map (CLK => CLK,
40      CLR => CLR, UP_DN => UP_DN, HOLD => HOLD,
41      Q => QTEMP(3 downto 0), RCO => RCO1);
42
43      DUT2 : contador_binario port map (CLK => CONNECT,
44      CLR => CLR, UP_DN => UP_DN, HOLD => HOLD,
45      Q => QTEMP(7 downto 4), RCO => RCO2);
46
47
48      carry : process(CLK, QTEMP)
49      begin
50          if (UP_DN = '1') then
51              CONNECT <= QTEMP(3);
52          else
53              CONNECT <= not QTEMP(3);
54          end if;
55      end process;
56
57      Q <= QTEMP;
58      RCO <= RCO1 and RCO2;
59
60  end architecture Behavior ;
```

Dois port maps da mesma entidade

Conversão da saída de uma entidade no CLK da outra

RCO do novo contador

Testbench do contador de 8 bits

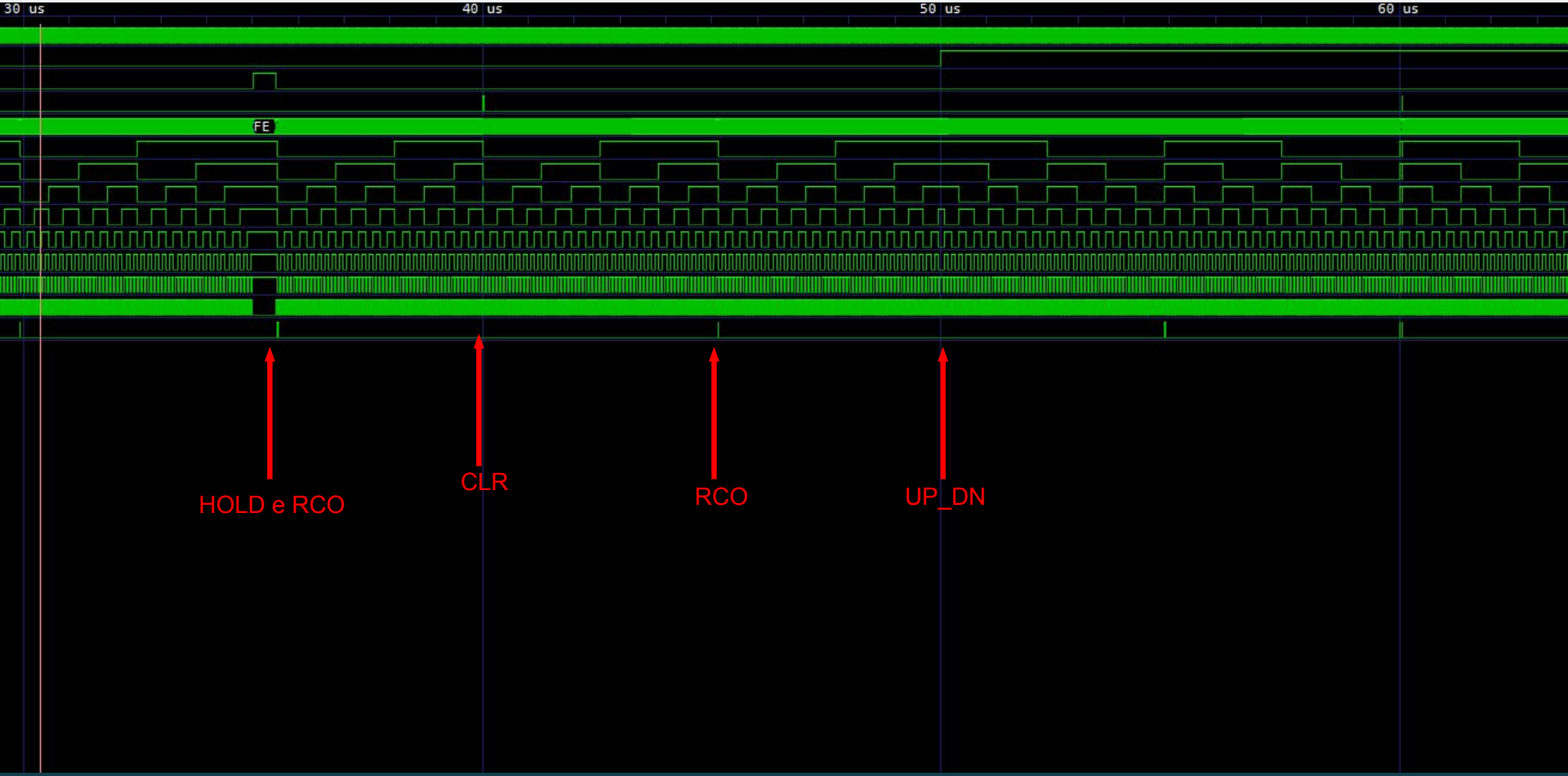
```
16  library IEEE;
17  use IEEE.STD_LOGIC_1164.ALL;
18  use ieee.numeric_std.all;
19
20  entity contador_binarioV2_TB is
21  end entity contador_binarioV2_TB;
22
23  architecture TB of contador_binarioV2_TB is
24
25      component contador_binarioV2
26          port (CLK, CLR, UP_DN, HOLD : in std_logic;
27          Q : out std_logic_vector(7 downto 0);
28          RCO : out std_logic);
29      end component;
30
31      signal CLK, CLR, UP_DN, HOLD, RCO : std_logic :='0';
32      signal Q :  std_logic_vector(7 downto 0) := "00000000";
33
34      begin
35
36      DUT3 : contador_binarioV2 port map (CLK => CLK,
37      CLR => CLR, UP_DN => UP_DN, HOLD => HOLD,
38      Q => Q, RCO => RCO);
39
40
```

```
41      clock : process
42      begin
43          CLK <= '0';
44          wait for 10 ns;
45          CLK <= '1';
46          wait for 10 ns;
47      end process;
48
49      inputCLR : process
50      begin
51          wait for 20000 ns;
52          CLR <= '1';
53          wait for 20 ns;
54          CLR <= '0';
55      end process;
56
57      inputHOLD : process
58      begin
59          wait for 35000 ns;
60          HOLD <= '1';
61          wait for 500 ns;
62          HOLD <= '0';
63      end process;
64
65      inputUP_DN : process
66      begin
67          UP_DN <= '0';
68          wait for 50000 ns;
69          UP_DN <= '1';
70          wait for 50000 ns;
71      end process;
72
73  end architecture TB;
```

tempos consideravelmente mais longos entre os inputs

Simulação no GTKWAVE

Simulação no GTKWAVE



Questão 1:

"Implemente em VHDL uma máquina de estados finitos, utilizando o modelo de Moore, que atua como um detector de sequências "1011". Seu projeto DEVE detectar sequências sobrepostas. Suponha que a entrada seja denominada X, a saída seja denominada Z e que um sinal de reset (RST) reinicia a máquina de uma forma assíncrona. A construção VHDL ENTITY é fornecida. Escreva a construção VHDL ARCHITECTURE correspondente para implementar a FSM. Implemente com flip-flops acionados na transição positiva. Implemente um arquivo de testes (TESTBENCH) em VHDL para verificar o funcionamento do seu projeto. Simule com GHDL e GTKWAVE e apresente um relatório com os arquivos em VHDL e os resultados da simulação obtidos."

```
16  library ieee;
17  use ieee.std_logic_1164.all;
18  use ieee.numeric_std.all;
19
20  entity detector is
21      port (X : in std_logic;
22      CLK : in std_logic;
23      RST : in std_logic;
24      Z : out std_logic;
25      STATE : out integer    --Adicionado para observaçao
26      );
27  end detector;
28
```

Biblioteca numeric_std para a utilização do tipo Integer

A arquitetura da entidade

```
29  architecture Behavior of detector is
30
31      signal STATE_TEMP : integer:=1;
32
33      begin
34
35      process(CLK, RST)
36      begin
37
38      if(RST = '1') then
39          STATE_TEMP <= 1;
40      else
41          if(rising_edge(CLK)) then
42              case STATE_TEMP is
43
44                  when 1 =>
45                      if(X = '1') then
46                          STATE_TEMP <= 2;
47                      else
48                          STATE_TEMP <= 1;
49                      end if;
50
51                  when 2 =>
52                      if(X = '1') then
53                          STATE_TEMP <= 2;
54                      else
55                          STATE_TEMP <= 3;
56                      end if;
57
58                  when 3 =>
59                      if(X = '1') then
60                          STATE_TEMP <= 4;
61                      else
62                          STATE_TEMP <= 1;
63                      end if;
64
```

Análise do estado atual e da entrada, atribuição do próximo estado

```
72                  when 5 =>
73                      if(X = '1') then
74                          STATE_TEMP <= 2;
75                      elsE
76                          STATE_TEMP <= 3;
77                      end if;
78
79                  when others =>
80                      STATE_TEMP <= 1;
81              end case;
82          end if;
83      end if;
84      end process;
85
86      process(STATE_TEMP)
87      begin
88          case STATE_TEMP is
89              when 5 =>
90              z <= '1';
91              when others =>
92              z <= '0';
93          end case;
94      end process;
95
96      STATE <= STATE_TEMP;  -- novamente, para observação
97
98  end architecture Behavior;
```

atribuição da saída

O Testbench

Análise do estado atual e da entrada, atribuição do próximo estado

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity detector_tb is
end entity detector_tb;

architecture TB of detector_tb is

    component detector
        port (X : in std_logic;
        CLK : in std_logic;
        RST : in std_logic;
        Z : out std_logic;
        STATE : out integer   --Adicionado para observaçao
        );
    end component;

    signal X, CLK, RST, Z: std_logic :='0';
    signal STATE : integer := 1;

    begin

    DUT : detector port map (X => X, CLK => CLK,
        RST => RST, Z => Z, STATE => STATE);

    clock : process
    begin
        CLK <= '1';
        wait for 10 ns;
        CLK <= '0';
        wait for 10 ns;
    end process;

```

Simulação de inputs aleatórios para X

```vhdl
    input_X : process
    begin
        wait for 100 ns;
        X <= '1';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '0';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
        X <= '1';
        wait for 20 ns;
    end process;

end architecture TB;
```

Simulação no GTKWAVE

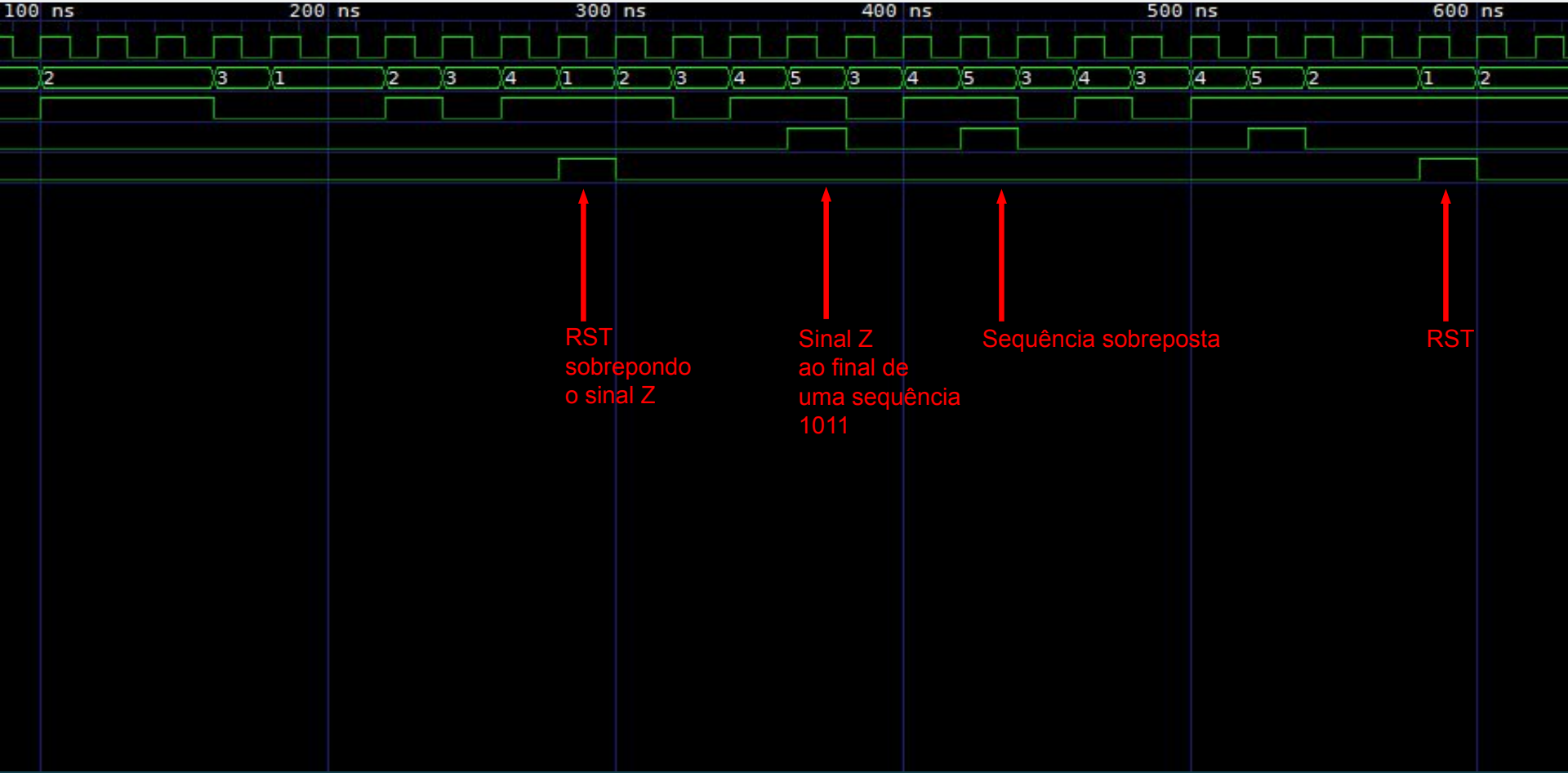
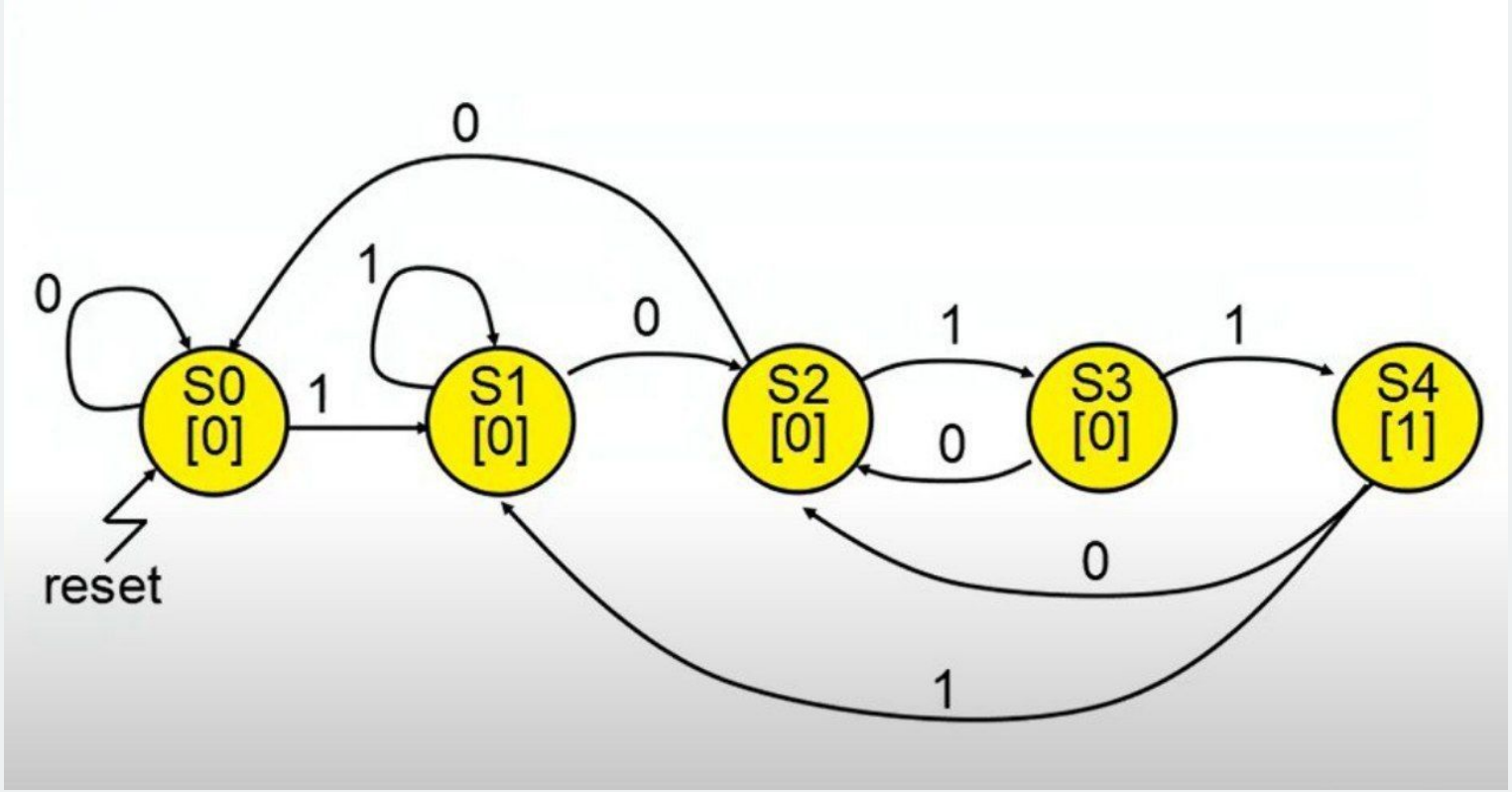


Diagrama de estados



Fim